*I.J. Image, Graphics and Signal Processing*, **2026, 1, 51-69**
Published Online on February 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2026.01.04

# Enhancing Image Forgery Detection through Dataset Balancing and a Fine-Tuned ResNet50: Focus on Copy-Move and Splicing

**Jayanti Rout**
P.G. Department of Computer Science, Fakir Mohan University, Balasore, 756019, Odisha, India
E-mail: routjayanti2k19@ieee.org
ORCID iD: https://orcid.org/0000-0001-7597-131X

**Minati Mishra***
P.G. Department of Computer Science, Fakir Mohan University, Balasore, 756019, Odisha, India
Email: minatiminu@gmail.com
ORCID iD: https://orcid.org/0000-0002-6784-4449
*Corresponding Author

**Ram Chandra Barik**
Department of CSE, C. V. Raman Global University, Bhubaneswar, 752054, Odisha, India
Email: ram.chandra@cgu-odisha.ac.in
ORCID iD: https://orcid.org/0000-0002-2803-5868

Received: 18 May, 2025; Revised: 12 September, 2025; Accepted: 08 January, 2026; Published: 08 February, 2026

**Abstract:** In last two decades, due to expansion of usage of multimedia especially images and also the trendy image editing tools, a huge amount of altered and forged images has been generated and circulated in social media and world wide web. Forged images are a threat to individuals, organizations, and society in terms of revenue, goodwill, etc. Verifying the authenticity of an image against possible forgery manually is prone to bias and is not feasible. Machine Learning (ML) models have their pros and cons in detecting image forgeries. Deep learning (DL)-based approaches have shown significant potential in the identification of tampered images due to their inherent feature extraction approaches, model configurations, and limitations of a well-distributed, unbiased public dataset. In this work, a fine-tuned pre-trained ResNet50 model has been proposed to detect tampered images. It focuses on detection of copy-move and splicing forgeries. In addition, a single-point crossover and mutation of Genetic Algorithms (GA) are used to address the class imbalance in the CASIA v2 dataset effectively. An extensive evaluation using simulation-based experiments shows that the proposed approach achieves promising and consistent performance with test accuracy and AUC of 0.9202 and 0.9741. A consistent result for both classes along with low computational complexities suggest the effectiveness of the proposed approach over other balancing strategies. The scalable design of the approach improves the reliability of image forgery detection.

**Index Terms:** Image Forgery, Copy-Move, Splicing, Deep Learning, CNN

## 1. Introduction

Images have become a primary medium of communication for various platforms. It includes social media, news, legal, and documentation. The ability of an image to convey information quickly and efficiently has made it an important aspect [1]. In addition, they are also used as an important evidence in forensic examinations, legal proceedings, etc., [2, 3, 4]. However, the abundance of an image editing tools [5, 6, 7] has made their manipulation easy. It allows individuals, even newbies to manipulate images seamlessly. Such tools often leave no trace of manipulations or falsification. These manipulations can be for defamation, support fake narratives, etc., [8, 9, 10].

Manipulation of an image refers to the alterations within it to enhance or change its contents. It is used in photography, advertising, creative design, etc. It can be categorized into two categories based on the fact whether the change in semantic content of image: Content Preservation Manipulations (CPM) and Content Altering Manipulations

(CAM) [11]. Any intentional manipulations made to the images that are made with the motive of tricking viewers is called image tampering. It is widely used to spread misinformation, identity fraud, etc.

### *1.1. Content-Preserving Manipulations*

CPM do not change the semantic meaning of the image but may enhance or adjust its appearance. They are often used for image enhancement, restoration, or compression without significantly modifying the original content. Various CPM are described in Table 1.

Table 1. Different types of content-preserving manipulations

| Type | Description | Examples |
|---|---|---|
| Enhancement | Adjusting brightness, contrast, and colors. | Histogram Equalization, Gamma Correction |
| Filtering | Applying noise reduction, sharpening, blurring. | Gaussian Blur, Median Filtering |
| Compression | Reducing file size while maintaining quality. | JPEG, WebP, PNG Compression |
| Resizing | Adjusting image dimensions without altering content. | Aspect Ratio Change, Cropping |
| Rotation | Reorienting the image without modifying objects. | 90°/180° Rotations, Flipping |
| Watermarking | Embedding marks for ownership verification. | Visible and Invisible Watermarking |
| Color Transformation | Adjusting color spaces or grayscale conversion. | RGB to HSV, Grayscale |

### *1.2. Content-Altering Manipulations*

CAM change the semantic content of an image, often leading to forgery, deepfake generation, or tampering. They can be used for both malicious intent and artistic creation. Various CAM are discussed in Table 2. Among various image tamperings or forgeries, Copy-Move Forgery (CMF) and splicing forgeries are the most common and deceptive [12].

i. CMF: It involves duplicating a section of an image and pasting it elsewhere within the same image. It is usually used to hide or replicate objects, as depicted in the first row of Figure 1.
ii. Splicing forgery: It signifies combining portions or sections from various images to create a single image that appears as a single authentic piece. It is depicted in the second row of Figure 1.

Table 2. Different types of content-altering manipulations

| Type | Description | Examples |
|---|---|---|
| Splicing | Combining parts of multiple images. | Face Swapping, Deepfake Synthesis |
| Copy-Move | Cloning and pasting a region within the same image. | Object Removal, Background Filling |
| Inpainting | Filling missing parts of an image automatically. | AI-based Content Filling |
| Morphing | Gradual transformation between images. | Face Morphing in Deepfakes |
| Removal / Additions | Altering objects to change the scene meaning. | Fake News Images, Object Removal |
| Style Transfer | Applying artistic effects from one image to another. | Neural Style Transfer |
| Deepfake Manipulation | AI-based generation of fake realistic images. | Fake Faces, AI-Generated Celebrities |

Detecting these forgeries is challenging because the alterations often preserve color tones, textures, and other intrinsic image properties, making hand-crafted forensic techniques, such as block-based analysis, keypoint-based methods, Color Filter Array (CFA) analysis, and chromatic aberration analysis, inefficient [13, 14].

Machine Learning (ML)-based approaches were introduced subsequently to overcome the limitations of traditional manual techniques. These methods involve extracting the unique features of an image to train classifiers. Different widely used classifiers include Support Vector Machines (SVM), Random Forests (RF), and k-Nearest Neighbors (k-NN). This approach can identify real and tampered images. Although ML models improved detection accuracy, their effectiveness depended on manually extracted features. As a result, their ability to classify unseen forgeries is limited. As a result, the demand for automated and efficient forgery detection models increased. This led to the development of Deep Learning (DL) models.

DL are widely used over ML-based approaches. Their use of Convolutional Neural Networks (CNN) and other advanced architectures provides the ability to automatically learn features from images [15, 12, 16]. They limit the need for manual extraction of features. As a result, the model becomes adaptable to different forgeries with higher detection performance. It also makes them feasible to train on large datasets. However, there is a scarcity of large labeled datasets in practice. In addition, existing ones are imbalanced in nature based on the class. Training models on such datasets increase their bias towards the abundant class. Its mitigation is an oversight in various studies.

This study aims to enhance image forgery detection by fine-tuning the pre-trained ResNet50 model and addressing dataset imbalance using a Genetic Algorithm (GA)-based single-point crossover and mutation operations. The proposed approach is assessed through multiple evaluation metrics such as accuracy, precision, recall, and F1-score, highlighting its effectiveness in identifying both CMF and splicing image forgeries. In addition, the impact of the balance of the dataset on model performance is analyzed, ensuring an optimized trade-off between detection accuracy and

computational efficiency. The results highlight the importance of combining dataset balancing techniques with DL models to develop a more robust and effective forgery detection framework.

The remainder of this paper is structured as follows: Section 2 reviews various existing work in the similar domain. Section 3 discusses different preliminary concepts to provide a foundation of the proposed approach. Section 4 describes the proposed methodology in detail. Section 5 discusses the experimental setup and analyzes the results. Section 6 concludes the work and outlines potential directions for future research.

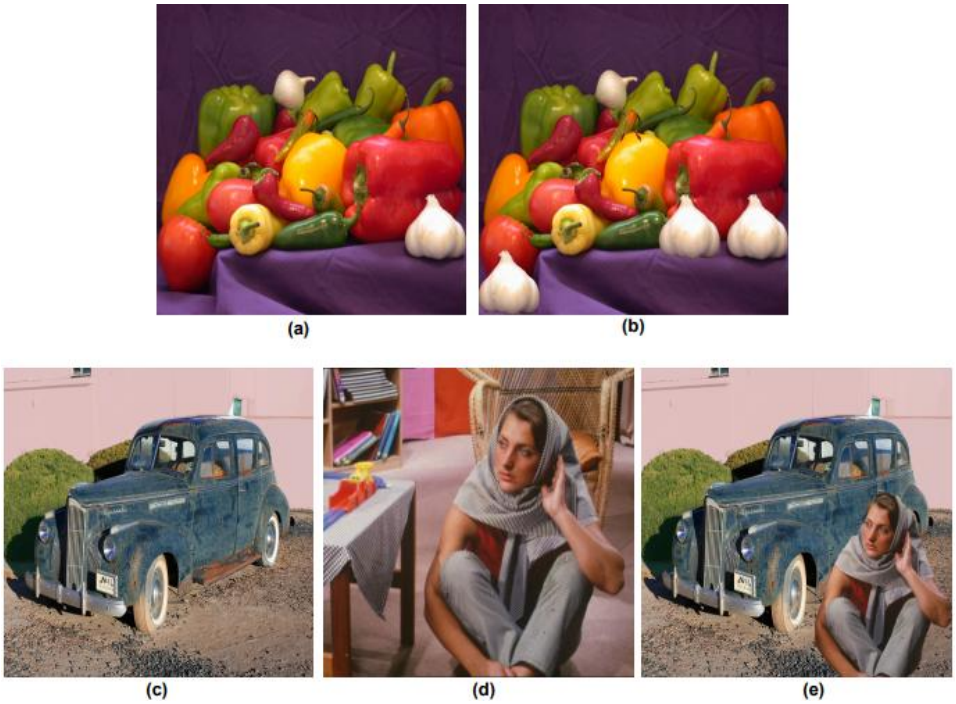



Fig. 1. Demonstration of copy-move and splicing: (a) Genuine image, (b) Copy-move image of (a), (c) Genuine image 1, (d) Genuine image 2, (e) Image splicing of (c) and (d) [12].

## 2. Related Works

Below, we discuss few papers from the existing literatures that share similar research interests.

Ali et al. [17] proposed a lightweight DL model to detect forgeries in digital images using CNN and image compression techniques. The model focuses primarily on CMF and splicing attacks on the image. The proposed technique works on the principle of finding the spike in the irregularities of the Discrete Cosine Transform (DCT) coefficients between the input image and the recompressed version of it. When an image contains forgery, the forged portion compresses differently from the rest because of the difference between the original image and the forged portion. This emphasizes the forgery component, which is used to train a CNN-based model to detect image forgery. The proposed technique performed better in terms of various parameters, including the time to process an image, tested using the CASIA v2 dataset.

In [18], researchers focusing on image forgery detection used Transfer Learning (TL) to acquire and transfer the weights of the pre-trained model You Only Look Once (YOLO) CNN to the developed method based on RESNET50V2. The use of transfer learning minimizes the training time and computational resources required to train a new model on a large dataset for which the existing model was already trained. The proposed model focuses mainly on dealing with image splicing attacks. The experimentation was performed using CASIA v1 and CASIA v2 datasets. Cross-validation was also performed, which reduced the loss and increased the accuracy on the other hand.

Khalil et al. proposes a custom DL model and technique using TL to detect forgery in digital images by calculating the compressed quality of the forged area and then generating a featured image for the training of the pre-trained model [19]. The CASIA v2 dataset was used to train and evaluate the model. A comparative analysis among eight pre-trained models was performed, and MobileNetV2 was found to outperform others with the least number of features and a accuracy of 94.6%.

The researchers in [20] proposed a DL model to detect forgeries in digital images using a dual-branch CNN. One of the primary issues with CNN, as discussed by researchers, is the inability to locate the necessary features to detect tampering directly. To address the aforementioned issue, the researchers presented a dual-branch CNN that utilizes ELA and noise residuals from the Spatial Rich Model (SRM) to detect image tampering. The model was trained on the CASIA v2 dataset and a validation accuracy of 98.55% was achieved.

Zeng et al. proposed a Dual-Pathway Multiscale Network (DPMSN) to detect forged images using extracted edge information of the spliced portion [21]. The multiscale feature fusion module integrates information from two pathways, using low-prediction results for accurate localization. Four datasets were used in the study, CASIA v1, NIST16, IMD2020, and COLUMBIA. To validate the efficacy of the proposed model, the model was compared with four different DL models, using various performance measures. The highest AUC of 98.92% and F1-score of 83.31% was observed with the NIST2016 dataset, among all four.

In [22], a dual-branch model is proposed that combines noise characteristics and RGB features using a hierarchical ConvNext module to detect both shallowfake and deepfake manipulations. The model incorporates edge supervision loss for precise localization and was tested on datasets including CASIA, COVERAGE, COLUMBIA, and NIST16, achieving an AUC score of 99%.

Uliyan proposed a DL model, specifically addressing CMF and various other transformations, including rotation, blurring, scaling, etc., of images [23]. With the use of the Simple Linear Iterative Clustering (SLIC) algorithm and Speeded Up Robust Features (SURF) for image segmentation and feature extraction, respectively, the Generative Adversarial Network (GAN) helps to distinguish real and fake images. This methodology also helps reduce false positive rates (FPR). The experiment was carried out using the CoMoFoD and MICC F200 datasets.

Arivazhagan et al. proposed a custom CNN model using transfer learning, focusing on addressing CMF detection [24]. Various pre-trained models, including Alexnet, VGG16, and Mobilenet V2, were utilized and fine-tuned to detect CMFs in images. The proposed architecture was evaluated on diverse datasets, including CoMoFoD, MICC-F220, MICC-F600, and MICC-F2000, and achieved a true positive rate (TPR) of 100%.

The researchers in [25] proposed a tamper image detection model that combines dual-branch deep models and reinforcement learning approaches. The model performs classification as well as localization of the tampering. The approach is evaluated using the seven datasets, including CASIA V2. Similarly, the researchers in [26] proposed MDL-Net that focuses on classification as well as localization of tampering in facial images. It uses a dual-stream feature learning strategy for it. The primary focus of the approach is to improve the cross-dataset performance. The model is trained and evaluated on multiple datasets to evaluate its efficacy.

From the above studies, it can be observed that existing studies focus on development of tamper detection models but doesn't focus on balancing the datasets. There is also a significant gap in the discussion of the class-wise results of the experiments. It shows the potential bias towards a specific class of the dataset. The proposed GA-based data balancing approach can alleviate this issue.

## 3. Preliminary Concepts

This section provides a brief explanation of the fundamental concepts of this investigation.

### *3.1. Genetic Algorithm*

GA is a type of optimization techniques which was introduced by J. Holland in the 1970s [27, 28, 29, 30]. It is based on the concept of biological evolution. It represents the problem space as a population of candidate solutions, called individuals. It is an iterative process where GA aims to find the "fittest" individual in the population. It does so by using various evolutionary principles on the population.

The workflow of GA is presented in Figure 2. The process begins with randomly generated individuals. Each one represents a potential solution to the problem. After multiple generations, the populations evolve and the quality of the solution increases. The fitness is evaluated using a fitness function. The function quantifies how well a solution performs in addressing the problem. The population is refined and converges to high-quality solution at the end.

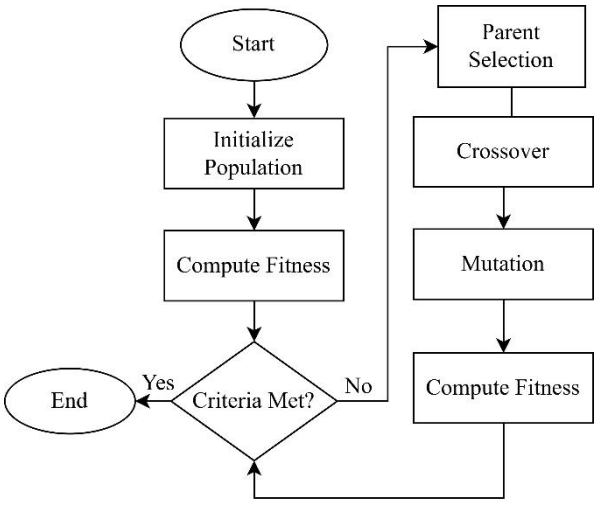



Fig. 2. Flowchart of the working of GA.

Three genetic operators are used for every generation of GA: selection, crossover, and mutation. These operations are performed on the individuals within the population with specific probabilities. These operations ensure the transition to optimal solutions [31]. GAs is effective for problems with multi-modal objective functions due to the use of multi-point search strategy. In addition, they are also suitable for discrete search spaces.

The search space is represented by strings called chromosomes. It represents the candidate solutions for a particular problem. Each chromosome is evaluated using fitness function to quantify its fitness. A population consists of a group of chromosomes along with their respective fitness values. Algorithm 1 discusses the process of searching the candidate solutions to identify the best one.

**Algorithm 1** Genetic Algorithm for Optimization

**Initialize:** Generate an initial population of *n* individuals (potential solutions).
**while** stopping criteria are not met **do**
**Evaluate Fitness:** Compute the fitness score *f(x)* for each individual *x*.
**Form New Generation:** Repeat until a new population is formed:
    **Selection:** Choose parent solutions probabilistically based on fitness scores.
    **Crossover:** Generate offspring by recombining selected parents.
    **Mutation:** Introduce small, random alterations in offspring.
    **Survivor Selection:** Add offspring to the new population.
**Update Population:** Replace the current generation with the newly formed one.
**end while**
**Return:** The best-performing solution from the final population.

### *3.1.1. Selection Operation*

The selection process identifies which individuals in the present population will be chosen as parents to produce offspring. The fitness of each individual serves as an important criterion for the selection and identification of the most promising candidates. Several techniques exist to choose the fittest chromosomes, such as roulette wheel selection, Boltzmann selection, tournament selection, steady-state selection, elitism selection, etc.

### *3.1.2. Crossover Operations*

In GA, new candidate solutions are formed through mutation and crossover processes. The crossover operation creates an offspring by merging genetic material from two parent chromosomes. This process uses a predefined crossover mask that dictates how the bits of each parent contribute to the offspring. The three main crossover techniques commonly used are single-point crossover, two-point crossover, and uniform crossover. In this work, we have used the single-point crossover method for class balancing.

In the proposed approach, a single crossover point is randomly selected along the length of the chromosome. The genetic material before this point is inherited from one parent, whereas the rest comes from the second parent. The crossover mask comprises an initial segment of n ones, followed by zeros. The bits corresponding to 1s in the mask are taken from the first parent, while bits corresponding to 0s in the mask are taken from the second parent. As illustrated in Figure 3, if the crossover occurs at position 5, the mask 11111000000 dictates that the first five bits come from the first parent, while the remaining bits come from the second parent.

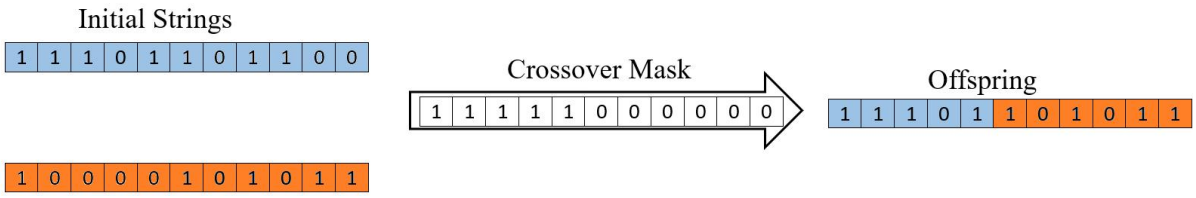



Fig. 3. Single-Point Crossover Operation

### *3.1.3. Mutation Operations*

It is a key genetic operator that creates diversity by making small random modifications to an individual's genetic sequence. Unlike crossover, which combines genetic material from two parents, it operates on a single individual. It alters one or more bits to maintain genetic variability in the population. A commonly used mutation technique is point mutation. Here, a randomly chosen bit within a chromosome is flipped (from 0 to 1 or vice versa). Mutation is usually applied after crossover. It prevents premature convergence and ensure exploration of new solution spaces. Figure 4 shows this process.

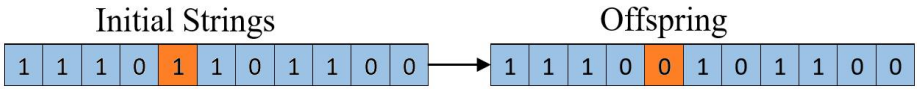



Fig. 4. Point Mutation Operation.

The effectiveness of a GA is determined by a fitness function that evaluates the quality of candidate solutions. In classification tasks, this function typically assesses the accuracy of a model in a given dataset. Additional factors, such as model complexity and generalizability, can also influence the fitness score. In general, when a bit-string represents a complex decision-making process, the fitness function quantifies its overall performance rather than assessing individual components in isolation.

### *3.2. Convolutional Neural Networks*

CNN, a type of DL model, is highly effective to process visual data. Commonly applied to tasks including image classification, object recognition, and segmentation, as they use the spatial hierarchies inherent in image data. A CNN architecture consists of three primary layers: convolutional layers, pooling layers, and fully connected layers. Figure 5 illustrates a basic CNN model used for fake image classification [32].

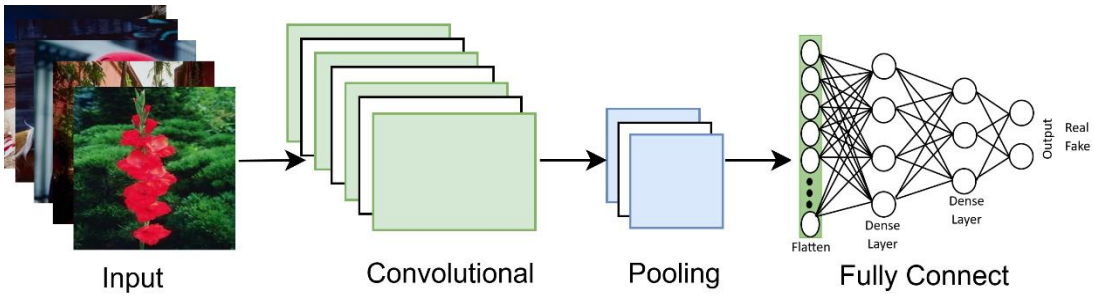



Fig. 5. A basic CNN architecture for fake image classification.

#### *3.2.1. Convolutional Layer*

The convolutional layer serves as the foundation for a CNN. Using a collection of trainable filters (kernels) in the input image, it captures spatial features such as edges, textures, and structural patterns. Mathematically, the convolution operation between an input image F and a filter W is expressed as:

$$(F \circledast W)(u,v) = \sum_{p=-a}^{a} \sum_{q=-b}^{b} F(u+p, v+q) W(p,q) \tag{1}$$

where F(u, v) denotes the intensity value of the input image at coordinates (u, v), W(p, q) represents the weights of the convolutional kernel, a and b correspond to the kernel size parameters.

To support the model in understanding complex and non-linear dynamics, an activation function commonly the Rectified Linear Unit (ReLU) is applied to the result of the convolutional layer. The function is mathematically expressed
as:

$$\emptyset(\mathrm{x}) = \begin{cases} x, \text{if } \mathrm{x} > 0 \\ 0, \text{otherwise} \end{cases} \tag{2}$$

In this formulation, ϕ(x) yields the input value when it is greater than zero; for non-positive values, it returns zero. This type of activation introduces a non-linear transformation within the neural network, enabling it to capture and approximate complex data patterns and relationships.

#### *3.2.2. Pooling Layer*

Pooling layers perform downsampling to reduce the spatial dimensions of the feature maps, improving computational efficiency while making the network more robust to slight transformations in input images. One of the most commonly used techniques is max pooling, which retains only the highest pixel value in each window.

#### *3.2.3. Fully Connected Layer*

Once features are extracted through convolution and pooling operations, they are flattened and passed to fully connected layers. These layers function similarly to conventional neural networks and perform classification based on the learned representations.

In multi-class classification problems, the softmax activation function is commonly employed in the final layer to estimate the probability mapping over categories across multiple classes. It is defined as:

$$\hat{y}k = \frac{\exp(a_k)}{\sum_{l=1}^{K}(\exp a_l)} \quad (3)$$

where $\hat{y}k$ denotes the predicted probability of the input belonging to class k, $a_k$ is the input to the activation function (i.e., the logit) for class k, and K is the total number of classes.

The sigmoid activation function is utilized in binary classification tasks (e.g., to determine the authenticity or manipulation of an image) which converts the output into a probability between 0 and 1. It is given by:

$$\hat{y} = \frac{1}{1+e^{-s}} \quad (4)$$

where $\hat{y}$ is the predicted probability of the positive class and s is the logit corresponding to the input.

## 4. Methodology

This section discusses the proposed approach and its implementation strategy in detail. The proposed approach uses a dual-branch DL model to classify fake images. It uses spatial RGB features and noise-based forensic features. The model combines a fine-tuned ResNet50 model for semantic representation. It also uses a noise residual branch based on Spatial Rich Model (STM) filters. It improves sensitivity to manipulation artifacts. The overall workflow of the model is presented in Figure 6.

### *4.1. Data Preparation (Dataset and Transformations)*

The CASIA v2 dataset is used in the study. It contains 7,491 authentic (Au) images and 5,123 tampered (Tp) images. The tampered images include 3,274 copy-move (Tp S) and 1,849 splicing (Tp D) images. In addition, 5,123 ground-truth images are also present that are of different dimensions. It focuses on CMF and splicing forgeries. The images vary in resolution and content. It reflects realistic forgery scenarios. RGB images are used as input. Various preprocessing steps are conducted on the image before it is provided as input to the proposed model. After preprocessing, the images are converted to a fixed size of 224 × 224 × 3. The 224 represents both height and width. In addition, 3 represents RGB color channels. The dataset is divided into training, testing, and validation sets in a ratio of 70:15:15. To balance the training set, GA has been used. As discussed in the previous section, singe-point crossover and mutation are used to generate tampered splicing and CMFs, respectively. They are represented in Algorithms 2–4. Samples of copy-move and splicing forgeries are presented in Figure 7.

**Algorithm 2** Parent Selection from Training Set

**Require:** $X_{train}$: List of training image file paths
**Require:** $y_{train}$: List of corresponding labels (0 = Authentic, 1 = Tampered)
**Ensure:** $P_D$: Parent pool for splicing (Tp D type images)
**Ensure:** $P_S$: Parent pool for copy-move (Tp S type images)
**procedure** SELECTPARENTS($X_{train}$, $y_{train}$)
    $P_D \leftarrow \emptyset$ ▷ Initialize splicing parent pool
    $P_S \leftarrow \emptyset$ ▷ Initialize copy-move parent pool

    **for** ($f_i$, $y_i$) ∈ *zip*($X_{train}$, $y_{train}$) **do**
        **if** $y_i = 1$ **then** ▷ Only tampered images qualify as parents
            *filename* ← ExtractBasename($f_i$)

            **if** *filename* starts with "Tp D" **then**
                $P_D \leftarrow P_D \cup \{f_i\}$ ▷ Add to splicing pool
            **else if** *filename* starts with "Tp S" **then**
                $P_S \leftarrow P_S \cup \{f_i\}$ ▷ Add to copy-move pool
            **end if**
    **end for**
    **return** $P_D$, $P_S$
**end procedure**

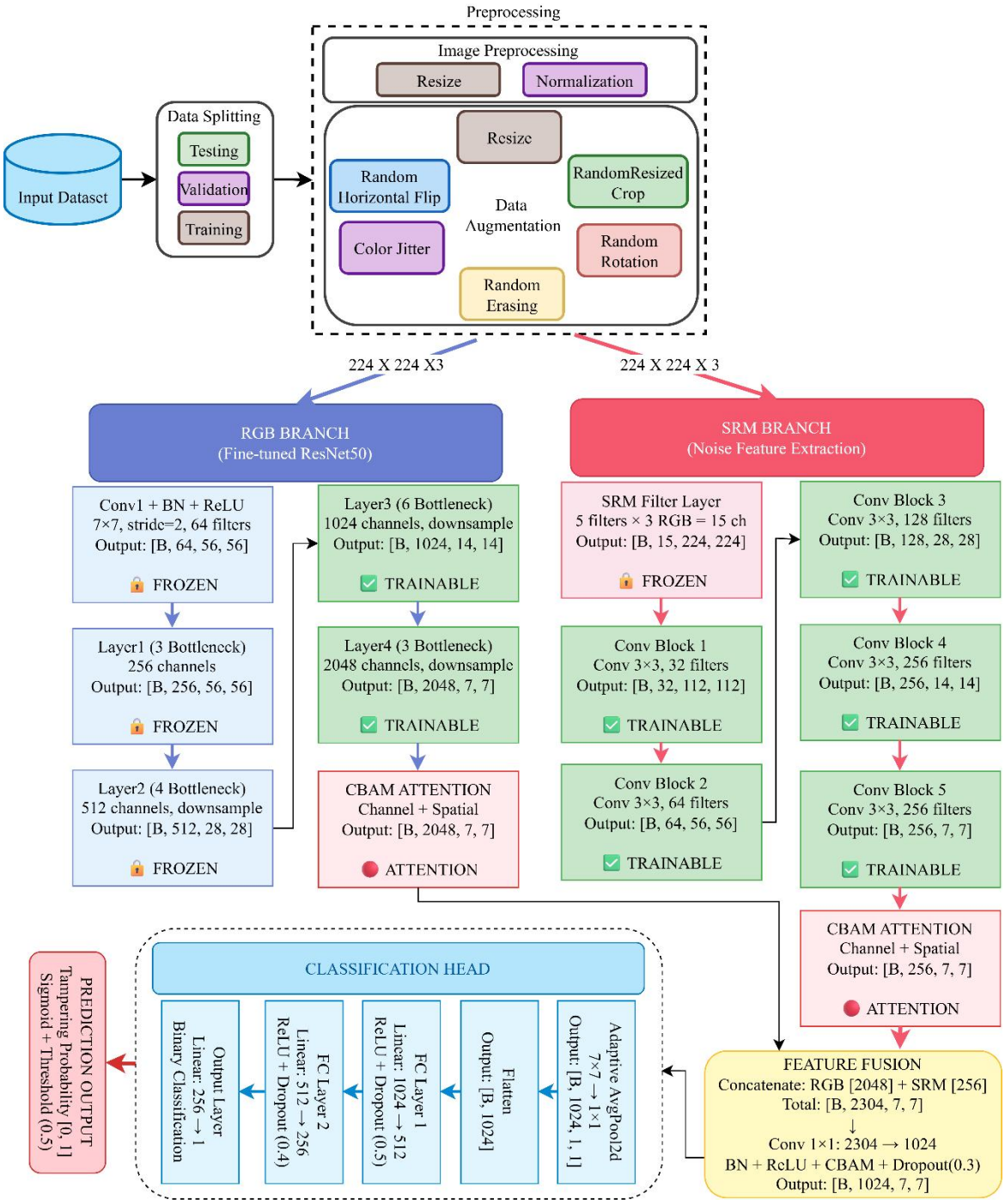
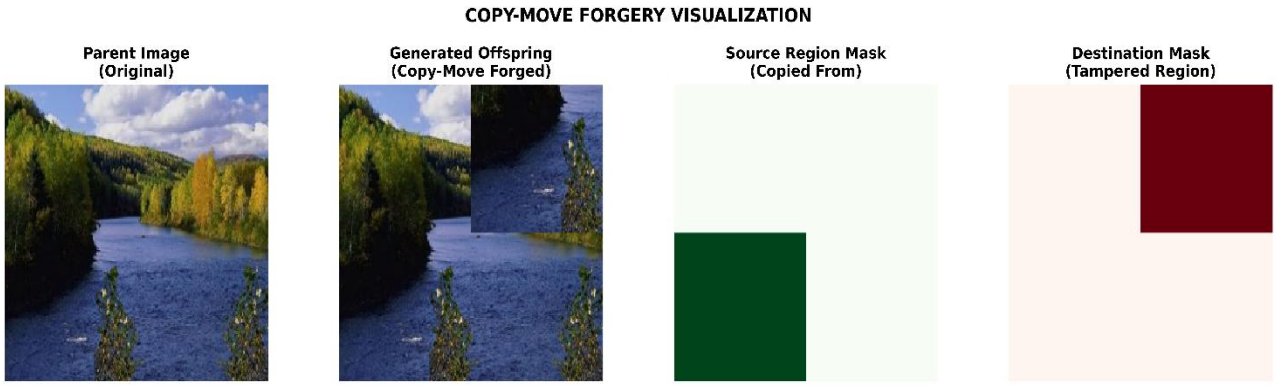



Fig. 6. Architecture of the proposed fine-tune ResNet50 for the fake image classification model.



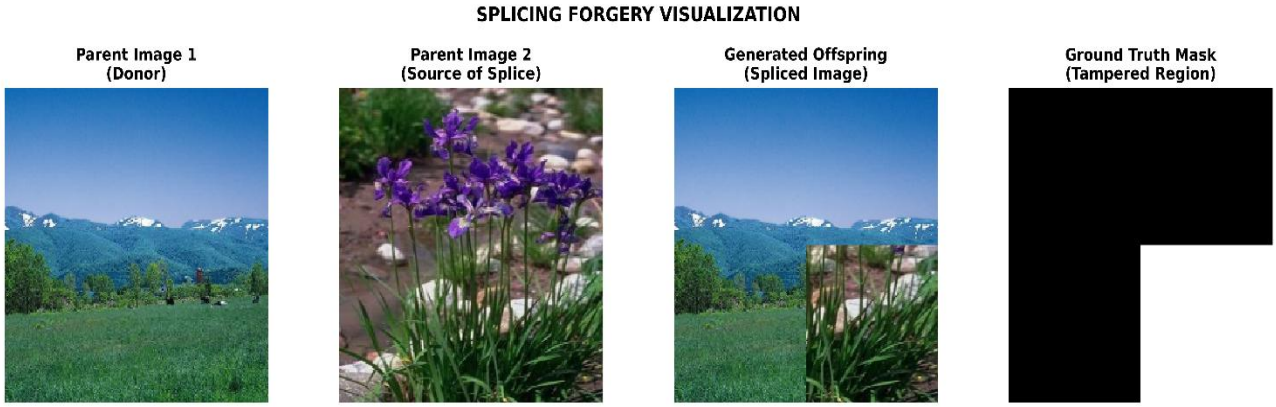



Fig. 7. Samples of copy-move and splicing tampering using GA.

**Algorithm 3 Single-Point Crossover Operation for Splicing Forgery**

**Require:** $I_1$: First parent image (RGB matrix)
**Require:** $I_2$: Second parent image (RGB matrix)
**Require:** S: Target image size (H, W) = (224, 224)
**Ensure:** O: Offspring image with splicing forgery
**procedure** *SINGLEPOINTCROSSOVER*($I_1$, $I_2$, S)
$I_1 \leftarrow$ Resize($I_1$, $S$) ▷ Step 1: Normalize parent dimensions
$I_2 \leftarrow$ Resize($I_2$, $S$)
$(H, W, C) \leftarrow$ Shape($I_1$) ▷ Step 2: Calculate block boundaries
$h_{mid} \leftarrow H/2$
$w_{mid} \leftarrow W/2$ ▷ Step 3: Decompose Parent 1 into quadrant blocks (genes)
$B_1^{(0)} \leftarrow I_1[0 : h_{mid}, 0 : w_{mid}]$
Top-left
$B_1^{(1)} \leftarrow I_1[0 : h_{mid}, w_{mid} : W]$ Top-right
$B_1^{(2)} \leftarrow I_1[h_{mid} : H, 0 : w_{mid}]$
Bottom-left
$B_1^{(3)} \leftarrow I_1[h_{mid} : H, w_{mid} : W]$ Bottom-right ▷ Step 4: Decompose Parent 2 into quadrant blocks (genes)
$B_2^{(0)} \leftarrow I_2[0 : h_{mid}, 0 : w_{mid}]$
$B_2^{(1)} \leftarrow I_2[0 : h_{mid}, w_{mid} : W]$
$B_2^{(2)} \leftarrow I_2[h_{mid} : H, 0 : w_{mid}]$
$B_2^{(3)} \leftarrow I_2[h_{mid} : H, w_{mid} : W]$
$k \leftarrow$ RandomInteger(0, 3) ▷ Step 5: Select random crossover point
▷ Step 6: Perform single-point crossover
$B_O \leftarrow$ Copy($B_1$) Copy all blocks from Parent 1
$B_O^{(k)} \leftarrow B_2^{(k)}$ Replace block k with Parent 2's block ▷ Step 7: Reconstruct offspring image
$top \leftarrow$ HConcat($B_O^{(0)}$, $B_O^{(1)}$)
$bottom \leftarrow$ HConcat($B_O^{(2)}$, $B_O^{(3)}$)
$O \leftarrow$ VConcat($top$, $bottom$)
**end procedure return** $O$

**Algorithm 4 Mutation Operation for CMF**

**Require:** I: Parent image (RGB matrix)
**Require:** S: Target image size (H, W) = (224, 224)
**Ensure:** O: Offspring image with CMF
**procedure** *MUTATION(*I, S*)*
$I \leftarrow$ Resize($I$, $S$) ▷ Step 1: Normalize parent dimensions
($H$, $W$, $C$) $\leftarrow$ Shape($I$) ▷ Step 2: Calculate block boundaries
$h_{mid} \leftarrow \lfloor H/2 \rfloor$
$w\ mid \leftarrow \lfloor W/2 \rfloor$ ▷ Step 3: Decompose into quadrant blocks (genes)
$B^{(0)} \leftarrow I[0 : h_{mid}, 0 : w_{mid}]$ ▷ Top-left
$B^{(1)} \leftarrow I[0 : h_{mid}, w_{mid} : W]$ ▷ Top-right
$B^{(2)} \leftarrow I[h_{mid}: H, 0 : w_{mid}]$ ▷ Bottom-left
$B^{(3)} \leftarrow I[h_{mid} : H, w_{mid} : W]$ ▷ Bottom-right
▷ Step 4: Select mutation point
($s$, $d$) $\leftarrow$ RandomSample($\{0, 1, 2, 3\}$, $k = 2$) ▷ s = source, d = destination
▷ Step 5: Perform mutation (internal block duplication)
$B^{(d)} \leftarrow$ DeepCopy($B^{(s)}$) ▷ Copy source block to destination
▷ Step 6: Reconstruct offspring image
$top \leftarrow$ HConcat($B^{(0)}$, $B^{(1)}$)
$bottom \leftarrow$ HConcat($B^{(2)}$, $B^{(3)}$)
$O \leftarrow$ VConcat($top$, $bottom$)
**end procedure return** $O$

In order to improve the generalization and reduce overfitting, various transformations are used during the training. It includes:

- Random resizing and cropping
- Horizontal flipping
- Small random rotations
- Color jittering
- Normalization using ImageNet mean and standard deviation

Only resizing, center cropping, and normalization are used in the validation and testing sets. It ensures consistent evaluations.

To capture the forensic traces that arise during image manipulation, a noise-based branch is used. It integrates a noise-based branch that processes SRM-filtered images. SRM filters are applied to all the images to suppress semantic content. It is used to improve the high-frequency residuals that are introduced due to tampering. These maps act as an additional input to the RBG stream. It provides the flexibility to jointly learn the semantic and forensic details.

### *4.2. Model Architecture*

The proposed model integrates a dual-branch design. It consists of:

#### *4.2.1. RGB Branch: Fine-tuned ResNet50*

The first stage is based on ResNet50. It is a widely used deep CNN model that is built using the principles of residual learning. It was designed to alleviate the vanishing gradient problem, The input RGB image passes through

- A 7 × 7 convolution with stride 2 and padding 3
- Batch normalization and ReLU activation
- A 3 × 3 max-pooling layer

It is followed by four residual stages (Conv2 x to Conv5 x). It progressively extracts hierarchical features. The first residual block, conv2 x, receives an input of size 56 × 56. After max pooling and uses 64 filters for three residual bottleneck blocks with a stride of 1. It preserves spatial resolution. The final convolutional output is combined using Global Average Pooling (GAP) to generate compact feature representations. The weights of pre-trained ImageNets are used for initialization. The early layers are frozen to retain the generic visual features. On the other hand, the deeper layers are fine-tuned to adapt to the forgery detection.

#### *4.2.2. Noise Branch: SRM-based Feature Stream*

The second branch receives SRM-filtered images as input. This branch consists of shallow convolutional layers. These features are later combined with the RGB features to form a single feature integration. This combination passes through fully connected layers with various regularization. The final layer provides a single output. A sigmoid activation function is used to generate the final class probability.

### *4.3. Training Strategy*

To ensure the reproducibility of the approach, fixed random seeds are used for all the libraries. In addition, CUDA deterministic settings are used to reduce nondeterminism during training. Binary Cross Entropy with Logits Loss (BCE-WithLogitsLoss) is used to classify the image. It provides adaptive design of learning strategies. A OneCycle Learning Rate scheduler is used to dynamically adjust the learning rate during training. It allows the model to explore a wide range of ML. It helps to adapt the model to a suitable learning rate. To ensure stability and generalization, the following techniques are used:

- Dropout layers are used in the classifier to reduce co-adaptation
- Gradient clipping is used to prevent exploding gradients

The model is trained for 100 epochs. It also uses early stopping based on validation loss to prevent overfitting. The best-performing model is saved for final evaluation.

### *4.4. Model Evaluation*

The trained model is evaluated using the held-out test set. Predictions are generated by applying a sigmoid function and thresholding at 0.5. Performance is assessed using multiple metrics, including accuracy, confusion matrix, precision, recall, F1-score, and ROC-AUC. The model is loaded from the best checkpoint using (best model.pth). It ensures the most optimal state. The predictions of the model are compared against ground truth labels.

Table 3. Summary of Evaluation Metrics

| Metric | Definition | Ideal Value | Range | Key Insights | Best Suited For |
|---|---|---|---|---|---|
| Accuracy | The fraction of correctly classified instances | 1 | [0,1] | A higher value indicates better overall classification but may be misleading for imbalanced datasets. | Balanced datasets |
| Precision | Measures the correctness of positive predictions | 1 | [0,1] | A high precision score means fewer false positives, which is crucial when false alarms are costly. | Imbalanced datasets where false positives should be minimized |
| Recall | Assesses the ability to identify actual positive cases | 1 | [0,1] | A high recall value implies fewer false negatives, making it important when missing positive cases is undesirable. | Imbalanced datasets where false negatives should be minimized |
| F1-Score | The harmonic mean of precision and recall | 1 | [0,1] | Useful when both precision and recall are critical, ensuring a balanced trade-off. | Imbalanced datasets |
| AUC-ROC | Assesses how effectively the model differentiates among the various categories | 1 | [0,1] | A higher AUC indicates better class separation, making it a robust performance measure. | Suitable for both balanced and imbalanced datasets |

### *4.5. Evaluation Metrics*

Different performance measures are employed to comprehensively assess the proposed model, as summarized in Table 3. In addition to overall accuracy, particular emphasis is placed on precision and recall due to the critical nature of correctly identifying forged images. Visual tools such as confusion matrices and ROC curves are also utilized to analyze classification behaviour.

## 5. Results and Discussion

All experiments were carried out in Kaggle, using the P100 GPU to take advantage of its high computational power for training and inference of the DL model. The model was implemented in TensorFlow and Keras, taking advantage of essential libraries such as OpenCV, Scikit-learn, and Matplotlib for efficient data processing and analysis.

The model was trained with a batch size of 32 and training was carried out over 100 epochs. The dataset was divided into training, validation, and test sets with no overlap between them.

### *5.1. Proposed Model Results*

The proposed approach uses the OneCycle learning rate strategy for optimization. It allows the learning rate to increase gradually during the initial phase of training. A controlled decay is carried out towards later epochs. The trends in the learning rate are presented in Figure 8. The model reaches an effective learning region within the first few epochs and maintains a stable learning in the middle phase. It fine-tunes the parameters during the final phase through small decays. This adjustment helps the model in avoiding local minima. The results can also be observed in the performance of the model.

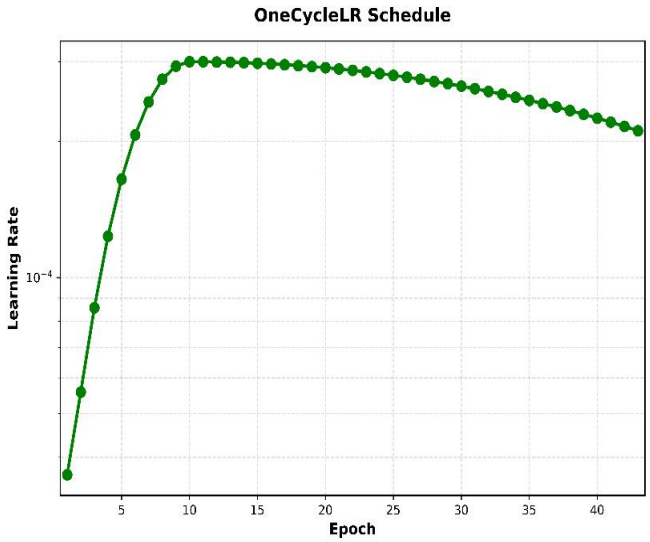



Fig. 8. Trends in learning rate throughout Training.

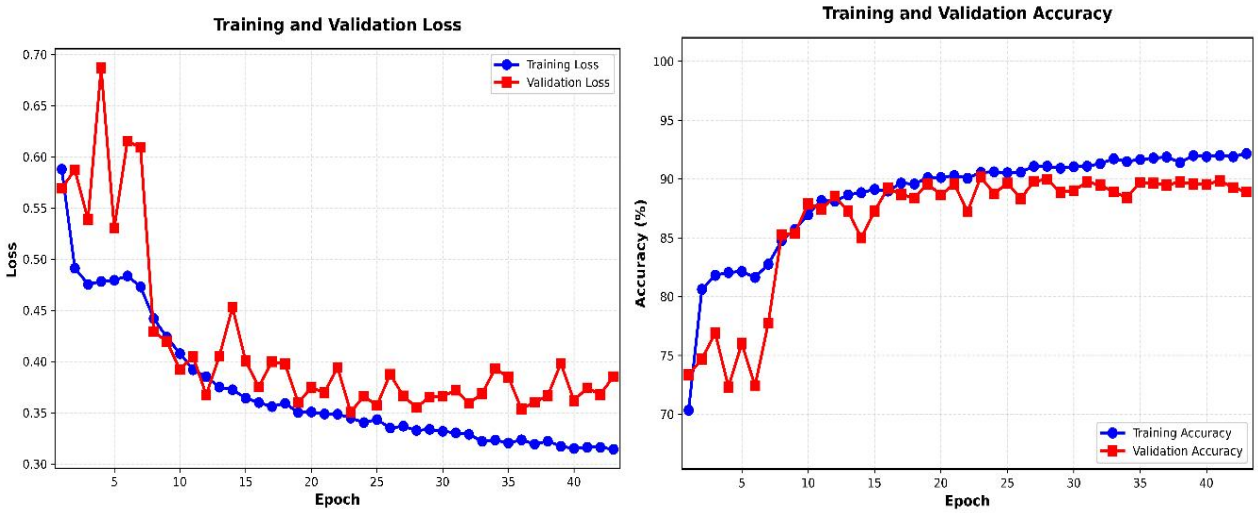



Fig. 9. Trends in Training and Validation Loss and Accuracy.

The trends in accuracy and loss of the model are presented in Figure 9. A smooth and consistent learning behavior is observed. The loss and accuracy follow opposite trends for both training and validation. The close alignment between both curves suggests that the model does not suffer from overfitting. Minor fluctuations in validation loss were observed. It is due to the batch-level variations and the effects of regularization. The training accuracy reaches ~92% by the end of the training. Similarly, the validation accuracy reaches around 89–90%. An AUC of 97.41% has been achieved, as mentioned in Figure 10.

The confusion matrix for the test result is presented in Figure 11. The low number of misclassifications indicates strong identification of fake samples. Slightly increase in false positives than in false negatives suggests that the model is more conservative. It prefers to flag ambiguous samples as fake than to allow forged samples to pass. It is a necessary behavior in security-sensitive applications, where missing a forged instance is more dangerous than incorrectly flagging a real one.

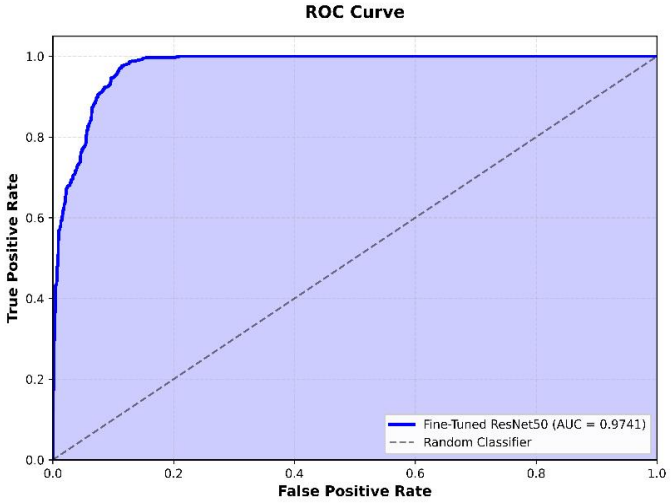



Fig. 10. Visualization of ROC Curve.

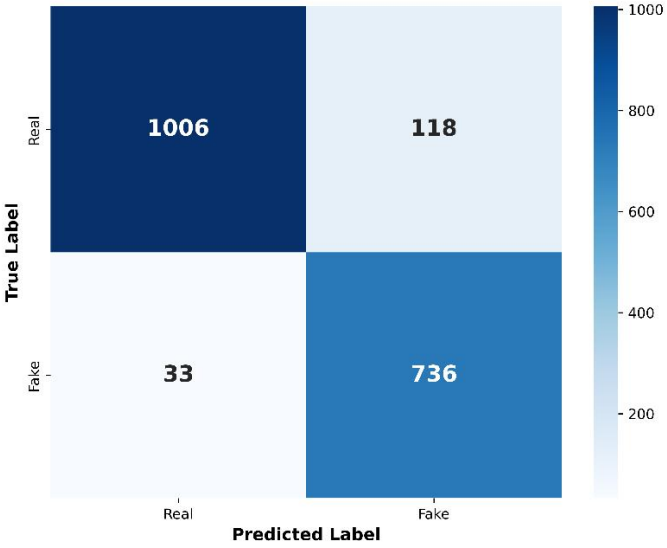



Fig. 11. Confusion Matrix of the proposed approach.

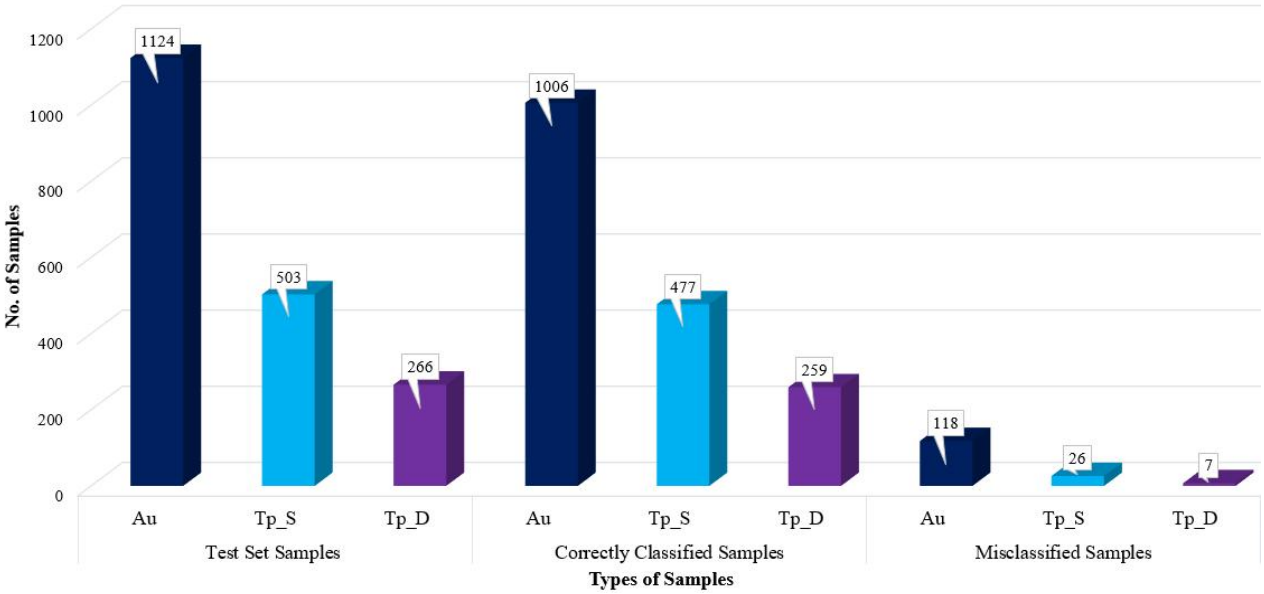
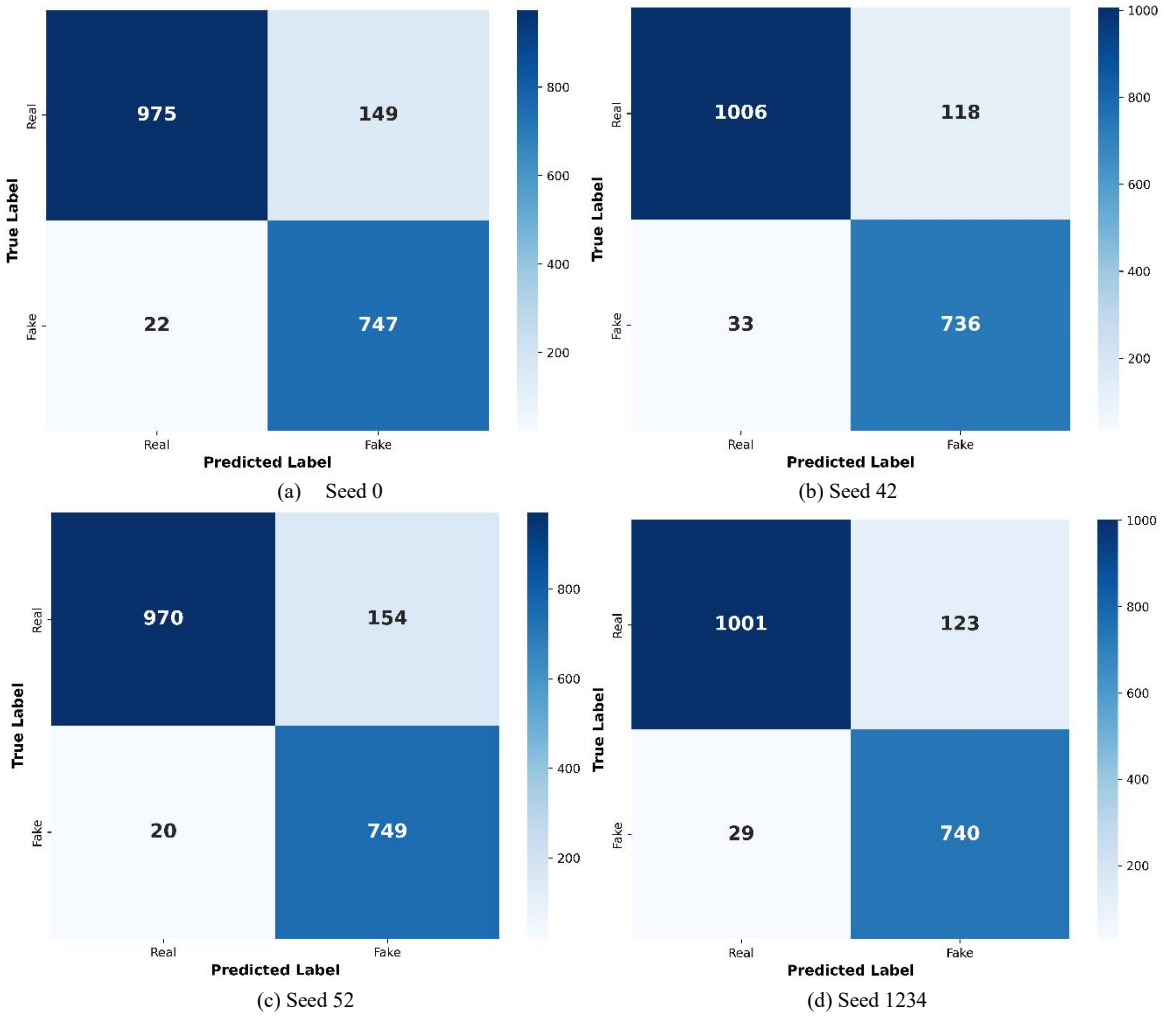



Fig. 12. Analysis of class-wise test results.

The effectiveness of the proposed approach is further validated by the performance in the test set. It is presented in Figure 12. Among all classes, Au contains the largest number of test samples and also yields the majority of misclassifications. Out of entire misclassifications, 78% of the total misclassified samples belong to the Au class. It achieves an accuracy of 89.5%. However, seven out of eight samples are correctly classified. This error is due to class dominance rather than weak performance. On the other hand, Tp S and Tp D achieve a relatively high accuracy of 94.8% and 97.4%, respectively. This suggests that the model learns the discriminative features effectively.

### *5.2. Statistical Reproducibility Analysis*

The proposed approach has been evaluated using different seeds to evaluate the reproducibility of the approach. These seeds are used to control the stochastic components of the training and evaluation process. It is responsible for initialization of network weights, shuffling of training data, randomness in the data augmentation process, etc. Testing the approach using different seeds also ensures that the results are not due to chance or favorable seeds. Various quantitative metrics are used to evaluate the approach. In addition, the variations are calculated using mean and standard deviation. Four random seeds are used for the evaluation. All of them are tested under similar environments for fair evaluation. Performance metrics are presented in Table 4. The accuracy varies within a small range of 90.81%–92.02%. A deviation of 1.21% has been achieved. It indicates the robustness to random initialization and data shuffle. Similarly, the AUC-ROC remains consistently high between 96.55–97.52%. A variation of <1% is achieved. It presents the high discriminative power of the model. In terms of class, precision and recall remained mostly stable. A slightly high variation was observed for the precision of the fake class. This suggests a slight difficulty in the detection of forged samples. On the other hand, the F1-scores remained stable with a small fluctuation of <1.3%. The results are further supported by the confusion matrices presented in Figure 13.




(a) Seed 0 (b) Seed 42

(c) Seed 52 (d) Seed 1234

Fig. 13. Confusion Matrices for Different Random Seeds.

The computational performance of all seeds was observed to remain consistent, as shown in Table 5. The average inference time varies within a small window of 0.87–0.97 ms. It helped to achieve a high throughput of 121.25–157.34 images per second. Although a small decrease in throughput was observed due to a longer test time, overall metrics remained mostly stable. It corresponds to the real-time performance-ready design of the model. Training time and other computational metrics are presented in Table 5. Training time shows greater variability, i.e., ranging from 42.48–58.81 minutes. It is mostly due to the early stopping behavior of the model. On the other hand, the average epoch time was found almost constant. This indicates that the variation does not affect the computational cost and the overall process is stable.

Table 4. Classification Performance for Different Random Seeds

| Metric | Class | Seed 0 | Seed 1234 | Seed 52 | Seed 42 | Mean ± Std |
|---|---|---|---|---|---|---|
| Precision | Real | 0.9779 | 0.9718 | 0.9798 | 0.9682 | 0.9744 ± 0.0053 |
| | Fake | 0.8337 | 0.8575 | 0.8295 | 0.8618 | 0.8456 ± 0.0156 |
| Recall | Real | 0.8674 | 0.8906 | 0.8630 | 0.8950 | 0.8790 ± 0.0155 |
| | Fake | 0.9714 | 0.9623 | 0.9740 | 0.9571 | 0.9662 ± 0.0077 |
| F1-Score | Real | 0.9194 | 0.9294 | 0.9177 | 0.9302 | 0.9242 ± 0.0060 |
| | Fake | 0.8973 | 0.9069 | 0.8959 | 0.9070 | 0.9018 ± 0.0056 |
| Accuracy | – | 0.9097 | 0.9197 | 0.9081 | 0.9202 | 0.9144 ± 0.0064 |
| AUC-ROC | – | 0.9655 | 0.9752 | 0.9672 | 0.9741 | 0.9705 ± 0.0049 |

Table 5. Computational Performance for Different Random Seeds

| Seed | Total Training Time (min) | Avg Epoch Time (s) | Total Test Time (s) | Avg Inference Time (ms) | Throughput (img/s) |
|---|---|---|---|---|---|
| 0 | 58.81 | 65.22 | 12.05 | 0.87 | 157.04 |
| 1234 | 50.02 | 65.11 | 12.03 | 0.89 | 157.34 |
| 52 | 42.48 | 65.17 | 12.07 | 0.90 | 156.80 |
| 42 | 47.46 | 66.03 | 15.61 | 0.97 | 121.25 |

Table 6. Model complexity and training characteristics for different random seeds

| Seed | Total Parameters | Trainable Parameters | Frozen Parameters | Best Epoch | Best Val Accuracy (%) |
|---|---|---|---|---|---|
| 0 | 28,175,210 | 26,729,157 | 1,446,053 | 34 | 90.43 |
| 1234 | 28,175,210 | 26,729,157 | 1,446,053 | 26 | 90.33 |
| 52 | 28,175,210 | 26,729,157 | 1,446,053 | 19 | 90.86 |
| 42 | 28,175,210 | 26,729,157 | 1,446,053 | 23 | 90.17 |

The complexity of the model was also observed to be stable in nature. It is presented in Table 6. All models exhibited 28.17 million parameters. Out of which 94.9% are trainable in nature. It is independent of changes in the network structure or parameterization. The higher number of trainable parameters than non-trainable ones indicate the high learning capability of the model. The small fluctuation of 0.7% in validation accuracy indicates a consistent generalization of the model. The best epoch for the models was observed to vary. It shows that the change in seeds can affect the convergence trajectory due to initialization and order of the data.

### *5.3. Performance analysis using different data-balancing method*

In order to evaluate the comparative efficacy of the approach, multiple data-balancing strategies are investigated and evaluated at different levels. They are as follows:

- Loss-level: It modifies the learning objective to penalize misclassification of minority samples more intensely.
- Data-level: It alters the data distribution by augmentation or oversampling.
- Hybrid augmentation strategies: It combines data transformation with weighted loss.
- A proposed GA-based balancing approach: It optimizes the sampling process using evolutionary search.

All approaches are evaluated in a similar environment for a fair evaluation.

The classification and computational performance of the models are presented in Tables 7 and 8. Table 9 presents the details of the complexity of the model. All approaches exhibited similar parameters that suggest structural similarity and no potential bias in strength. The baseline model is trained on the imbalanced dataset and achieves an accuracy of 92.2%. However, it achieves weaker precision for fake images. This indicates bias towards real samples. On the other hand, the use of class-weighted loss improves the recall of fake classes by ∼1.4% and slightly increases the overall accuracy. It was found that pure data-level augmentation performs poorly. It reduces accuracy by ∼11%. This suggests that only augmentation fails to preserve the unique features of forgeries. SMOTE oversampling was found to achieve an improved accuracy of 9% over standard augmentation approaches. However, it still underperforms the baseline methods. This indicates the limitations of synthetic sample generation to capture patterns of forgeries.

Table 7. Classification performance under different data-balancing techniques

| Category | Method | Precision (Real) | Precision (Fake) | Recall (Real) | Recall (Fake) | F1 (Real) | F1 (Fake) | Accuracy | AUC-ROC |
|---|---|---|---|---|---|---|---|---|---|
| Baseline | Imbalanced CASIA v2 | 0.9755 | 0.8565 | 0.8888 | 0.9701 | 0.9310 | 0.9098 | 0.9218 | 0.9702 |
| Loss-level | Class-weighted loss | 0.9881 | 0.8554 | 0.8861 | 0.9844 | 0.9343 | 0.9154 | 0.9260 | 0.9701 |
| Data-level | Standard augmentation | 0.9139 | 0.7077 | 0.7464 | 0.8973 | 0.8217 | 0.7913 | 0.8077 | 0.8501 |
| Oversampling | SMOTE (k=5) | 0.8441 | 0.7324 | 0.8043 | 0.7828 | 0.8237 | 0.7568 | 0.7956 | 0.8741 |
| Augmentation | MixUp + Weighted BCE | 0.9721 | 0.8675 | 0.8995 | 0.9623 | 0.9344 | 0.9125 | 0.9250 | 0.9751 |
| Augmentation | CutMix + Weighted BCE | 0.8728 | 0.7509 | 0.8123 | 0.8270 | 0.8415 | 0.7871 | 0.8183 | 0.8731 |
| Proposed | GA-based balancing | 0.9682 | 0.8618 | 0.8950 | 0.9571 | 0.9302 | 0.9070 | 0.9202 | 0.9741 |

Table 8. Computational performance for different data-balancing scenarios

| Category | Total Training Time (min) | Avg Epoch Time (s) | Total Test Time (s) | Avg Inference Time (ms) | Throughput (img/s) |
|---|---|---|---|---|---|
| Baseline | 58.94 | 55.97 | 12.03 | 0.89 | 157.40 |
| Loss-level | 50.42 | 54.12 | 12.02 | 0.88 | 158.10 |
| Data-level | 34.13 | 53.06 | 12.00 | 0.86 | 160.25 |
| Oversampling | 42.41 | 65.12 | 12.01 | 0.89 | 157.55 |
| Augmentation | 53.79 | 57.47 | 12.05 | 0.93 | 157.11 |
| Augmentation | 42.55 | 56.61 | 12.04 | 0.87 | 157.18 |
| Proposed | 47.46 | 66.03 | 15.61 | 0.97 | 121.25 |

Table 9. Model complexity and training characteristics for different data-balancing scenarios

| Category | Total Parameters | Trainable Parameters | Frozen Parameters | Best Epoch | Best Val Accuracy (%) |
|---|---|---|---|---|---|
| Baseline | 28,175,210 | 26,729,157 | 1,446,053 | 43 | 90.12 |
| Loss-level | 28,175,210 | 26,729,157 | 1,446,053 | 54 | 90.27 |
| Data-level | 28,175,210 | 26,729,157 | 1,446,053 | 11 | 78.33 |
| Oversampling | 28,175,210 | 26,729,157 | 1,446,053 | 19 | 79.7 |
| Augmentation | 28,175,210 | 26,729,157 | 1,446,053 | 36 | 90.27 |
| Augmentation | 28,175,210 | 26,729,157 | 1,446,053 | 25 | 79.55 |
| Proposed | 28,175,210 | 26,729,157 | 1,446,053 | 23 | 90.17 |

Hybrid approaches were observed to perform better. MixUp with weighted loss achieves the highest AUC of ~97.5%. It improves the recall for the fake class by 2% than the baseline approaches. It also maintains the highest accuracy. This shows that the combination of controlled augmentation with loss reweighting can be more effective than either standalone one. On the other hand, CutMix was observed to perform weaker. This suggests that mixing aggressive regions can distort important patterns. The proposed GA-based approach achieves a strong balance between classes. The F1-scores are above 90% for both classes. In addition, AUC is close to the best-performing model. It converges faster than the baseline as well as the hybrid approaches. The decrease in training time suggests a trade-off in terms of computational efficiency and performance. This makes it a competitive and suitable alternative for environments where speed and efficiency matter.

### *5.4. Comparison with SOTA models*

Table 10 presents a comparison analysis of the proposed approach with various other models. Various performance measures are used to quantify the approach. It can be observed that, even though a few studies have impressive performance measures, there is a lack of use of the balance dataset. Various studies were found to use the imbalanced dataset that can hinder the performance of the model. It can result in a biased model since major datasets have more authentic images than the forged ones. The researchers in [18] proposed using Adobe Photoshop to balance the dataset, but it is not feasible, requires manual labor, and is not scalable. Our fine-tuned ResNet50 model was trained with the original images of the dataset as well as synthetically generated images to enhance performance. The proposed GA-based approach achieved an accuracy and AUC value of 92.02% and 97.41%, respectively. The measures achieved are almost comparable to other models that did not balance the dataset.

Table 10. Comparison of existing works with the proposed method

| SL | Ref. | Technique(s) | Dataset(s) | Dataset Balanced | Accuracy | Precision | Recall | F1-score | AUC |
|---|---|---|---|---|---|---|---|---|---|
| 1 | [17] | CNN + image re-compression | CASIA_v2 | No | 92.13 | 0.85 | 0.98 | 91.08 | – |
| 2 | [18] | TL + YOLO CNN + ResNet50V2 | CASIA_v1 | Yes (Manual via Adobe Photoshop) | 81 | – | – | – | – |
| | | | CASIA_v2 | | 99.30 | – | – | – | – |
| 3 | [19] | TL + compression difference + fine-tuned pretrained model | CASIA_v2 | No | 94.69 | 94.21 | 94.74 | 0.94 | 0.95 |
| 4 | [20] | Dual-branch CNN + ELA + SRM noise residuals | CASIA_v2 | No | 98.55 | 98.71 | 99.26 | 98.98 | – |
| 5 | [21] | Dual-pathway multiscale network with feature fusion | CASIA_v1 | No | – | – | – | 65.59 | 93.12 |
| | | | NIST16 | | – | – | – | 83.31 | 98.92 |
| | | | IMD2020 | | – | – | – | 29.07 | 77.49 |
| | | | COLUMBIA | | – | – | – | 43.76 | 75.48 |
| 6 | [22] | Hierarchical ConvNeXt module | CASIA | No | – | – | – | 0.8956 | 0.9542 |
| | | | COVERAGE | | – | – | – | 0.8826 | 0.8826 |
| | | | COLUMBIA | | – | – | – | 0.6075 | 0.9725 |
| | | | NIST16 | | – | – | – | 0.9542 | 0.9979 |
| 7 | [23] | SLIC + SURF + GAN | CoMoFoD | No | – | 95.51 | 93.21 | 94.34 | – |
| | | | MICC-F200 | | – | 94.12 | 93.12 | 93.62 | – |
| 8 | [24] | Custom CNN with transfer learning on pretrained models | CoMoFoD | No | 99.20 | 0.9915 | 0.9919 | 0.984 | – |
| | | | MICC-F220 | | 97.72 | 0.8974 | 0.9459 | 0.915 | – |
| | | | MICC-F600 | | 85.83 | 0.653 | 0.79 | 0.719 | – |
| | | | MICC-F2000 | | 96.00 | 0.8974 | 0.9459 | 0.915 | – |
| 9 | Proposed | Fine-tuned ResNet50 | CASIA_v2 | Balanced using GA | 92.02 | – | – | – | 97.41 |

## 6. Conclusion

In this study, we fine-tuned the pre-trained ResNet50 model to detect CMF and splicing forgeries in digital images. Given the significant impact of dataset imbalance on the DL models, we employed a single-point crossover operation from genetic algorithms to balance the CASIA v2 dataset. Our experimental results demonstrate that balancing the dataset improves the performance of ResNet50, achieving an accuracy of 92.02% on the test set. These findings highlight the effectiveness of combining dataset preprocessing with DL models for image forgery detection. Although our study shows promising results, there are several areas for further improvement. Future research could explore the impact of more advanced data augmentation and synthetic image generation techniques to further enhance the diversity of datasets. Additionally, integrating attention mechanisms or transformer-based architectures, such as Vision Transformers (ViTs), may improve feature extraction and classification performance. Investigating multi-modal approaches by incorporating metadata and frequency-domain analysis for more robust forgery detection could be another notable approach. Finally, evaluating the generalizability of the model in larger and more diverse datasets may help ensure its effectiveness in real world scenarios.

## Acknowledgment

The authors gratefully acknowledge the support of the CyberSecurity Center of Excellence (CoE) at C. V. Raman Global University. The CoE provided the necessary facilities and technical support used for experimentation throughout the study.

## Conflict of Interest

The authors declare no conflict of interest.

## References

[1] Rune Pettersson. *Visual Information*. Educational Technology Publications, Englewood Cliffs, NJ, 1993.
[2] Jennifer L Mnookin. The image of truth: Photographic evidence and the power of analogy. *Yale JL & Human*., 10:1, 1998.

[3] Jiebo Luo, Matthew Boutell, and Christopher Brown. Pictures are not taken in a vacuum-an overview of exploiting context for semantic scene content understanding. *IEEE Signal Processing Magazine*, 23(2):101–114, 2006, doi: https://doi.org/10.1109/MSP.2006.1598086.
[4] Peter Burke. Eyewitnessing: *The Uses of Images as Historical Evidence*. Cornell University Press, Ithaca, NY, 2001.
[5] Adobe Inc. Adobe photoshop free trial download, 2025. Accessed: 2025-02-13.
[6] Picsart Inc. Picsart photo editor, 2025. Accessed: 2025-02-13.
[7] The GIMP Team. Gimp - gnu image manipulation program, 2024. Accessed: 2024-10-22.
[8] Maria D Molina, S Shyam Sundar, Thai Le, and Dongwon Lee. "Fake News" Is Not Simply False Information: A Concept Explication and Taxonomy of Online Content. *American behavioral scientist*, 65(2):180–212, 2021, doi: https://doi.org/10.1177/0002764219878224.
[9] Esma A¨ımeur, Sabrine Amri, and Gilles Brassard. Fake news, disinformation and misinformation in social media: a review. *Social Network Analysis and Mining*, 13(1):30, 2023, doi: https://doi.org/10.1007/s13278-023-01028-5.
[10] Jinna Lv, Yuan Gao, Li Li, Lei Shi, and Siyu Li. Multi-modal fake news detection: A comprehensive survey on deep learning technology, advances, and challenges. *Journal of King Saud University Computer and Information Sciences*, 37(9):306, 2025, doi: https://doi.org/10.1007/s44443-025-00317-7.
[11] Santoshini Panda and Minati Mishra. Passive techniques of digital image forgery detection: Developments and challenges. In Kiran Sarma, Bhanu Pratap Singh, and V. Chandra Sagar, editors, *Advances in Electronics, Communication and Computing*: ETAEERE-2016, pages 281–290. Springer, Singapore, 2017, doi: https://doi.org/10.1007/978-981-10-4765-7_29.
[12] Shishir Kumar Raj, Bhabani Shankar Mohanty, Jayanti Rout, Ashutosh Soni, and Surendra Kumar Nanda. An automated deep learning model for detecting image forgeries. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pages 1–6. IEEE, 2024, doi: https://doi.org/10.1109/ICEECT61758.2024.10738982.
[13] Giselle Batista, Jos´e Welligton, Marcela Alves, Warley Junior, and Hugo Kuribayashi. Recent Advances Digital Image Forgery Detection: A Systematic Mapping Study. *IEEE Access*, pages 1–1, 2026, doi: https://doi.org/ 10.1109/ACCESS.2026.3657232.
[14] Jayanti Rout, Sangram Pati, and Minati Mishra. A robust watermark-based model for image content integrity verification and tampering detection. *Informatica*, 49(28), 2025, doi: https://doi.org/10.31449/inf.v49i28.9176.
[15] Subir Biswas, Md Ariful Islam, Jayanti Rout, Minati Mishra, and Debendra Muduli. Passive image forgery detection through ensemble learning with ela. In *2024 International Conference on Computer, Electronics, Electrical Engineering & their Applications (IC2E3),* pages 1–6. IEEE, 2024, doi: https://doi.org/10.1109/IC2E362166.2024.10827508.
[16] Jayanti Rout, Mrutyunjaya Sathua, Ashutosh Soni, and Surendra Kumar Nanda. Detection of genai-generated faces using a customized residual network model. In *2025 IEEE Silchar Subsection Conference (SILCON),* pages 1–6. IEEE, 2025, doi: https://doi.org/10.1109/SILCON67893.2025.11327204.
[17] Syed Sadaf Ali, Iyyakutti Iyappan Ganapathi, Ngoc-Son Vu, Syed Danish Ali, Neetesh Saxena, and Naoufel Werghi. Image forgery detection using deep learning by recompressing images. *Electronics,* 11(3):403, 2022, doi: https://doi.org/10.3390/electronics11030403.
[18] Emad Ul Haq Qazi, Tanveer Zia, and Abdulrazaq Almorjan. Deep learning-based digital image forgery detection system. *Applied Sciences,* 12(6):2851, 2022, doi: https://doi.org/10.3390/app12062851.
[19] Ashgan H Khalil, Atef Z Ghalwash, Hala Abdel-Galil Elsayed, Gouda I Salama, and Haitham A Ghalwash. Enhancing digital image forgery detection using transfer learning. *IEEE Access,* 11:91583–91594, 2023, doi: https://doi.org/10.1109/ACCESS.2023.3307357.
[20] Sunen Chakraborty, Kingshuk Chatterjee, and Paramita Dey. Detection of image tampering using deep learning, error levels and noise residuals. *Neural Processing Letters*, 56(2):112, 2024, https://doi.org/10.1007/s11063-024-11448-9.
[21] Nianyin Zeng, Peishu Wu, Yuqing Zhang, Han Li, Jingfeng Mao, and Zidong Wang. DPMSN: A Dual-Pathway Multiscale Network for Image Forgery Detection. *IEEE Transactions on Industrial Informatics*, 2024, doi: https://doi.org/10.1109/TII.2024.3359454.
[22] Deepak Dagar and Dinesh Kumar Vishwakarma. A noise and edge extraction-based dual-branch method for shallowfake and deepfake localization. *Signal, Image and Video Processing*, 19(3):198, 2025, doi: https://doi.org/10.1007/s11760-024-03775-0.
[23] Uliyan Diaa. A deep learning model to inspect image forgery on SURF keypoints of SLIC segmented regions. *Engineering, Technology & Applied Science Research*, 14(1):12549–12555, 2024, doi: https://doi.org/10.48084/etasr.6622.
[24] S Arivazhagan, Newlin Shebiah Russel, M Saranyaa, CNN-based approach for robust detection of copy-move forgery in images. *Inteligencia Artificial*, 27(73):80–91, 2024, doi: https://doi.org/10.4114/intartif.vol27iss73pp80-91.
[25] Yahya Al-Nabhani, Amirrudin Kamsin, Mohamad Nizam, and Khalil Al Ruqeishi. REFORGE: A Robust Ensemble for Image Forgery Detection and Localization in Social Network Images. *IEEE Access*, 14:8773–8788, 2026, doi: https://doi.org/10.1109/ACCESS.2026.3653134.
[26] Huasong Yi, Qian Jiang, Hongyue Huang, Li Tang, Shaowen Yao, and Xin Jin. MDL-Net: Multi-task Learning Network for Face Forgery Detection and Localization Using Dual-Stream Feature Extraction and Reconstruction. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, pages 1–1, 2026, doi: https://doi.org/10.1109/TBIOM.2026.3656922.
[27] John H Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.
[28] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer, Berlin, Heidelberg, 2008.
[29] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998.
[30] Lingaraj Haldurai, T Madhubala, and R Rajalakshmi. A study on genetic algorithm and its applications. *Int. J. Comput. Sci. Eng*, 4(10):139–143, 2016.
[31] David E Goldberg. Genetic algorithm in search, optimization and machine learning, addison. *Wesley Publishing Company, R eading*, MA, 1(98):9, 1989.
[32] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458,* 2015.

## Authors' Profiles

**Jayanti Rout** received her Master of Philosophy (M.Phil.) degree in Computer Science from Fakir Mohan University, Balasore, Odisha, India, in 2022. She is currently pursuing the Doctor of Philosophy (Ph.D.) degree in the P.G. Department of Computer Science at Fakir Mohan University, Balasore, Odisha, India. Her major field of study includes computer vision and digital image processing. She is actively involved in research focusing on digital multimedia content authentication, image forgery detection, and secure computer vision systems using machine learning, deep learning, and cryptographic techniques. Her broader research interests include computer vision, digital image processing, pattern recognition, artificial intelligence, machine learning, and cryptography. She has authored and co-authored 26 research publications, including 4 journal articles and 22 conference papers, published in reputed national and international venues. Ms. Rout is a Student Member of the IEEE. She has received the Best Paper Award at two IEEE international conferences in recognition of her research contributions.

**Minati Mishra** is an Associate Professor in the P.G. Department of Computer Science at Fakir Mohan University, Balasore, Odisha, serving since 2004. She brings over two decades of teaching and research experience across various government and private institutions in Odisha. As an academician, she has taught diverse subjects including Cryptography and Security, Computer Networks, Discrete Mathematics, Computer Graphics, Operating Systems, Theory of Computations, Computer Architecture, Digital Image Processing, Database Management Systems, Data Structures, Algorithm Design and Analysis, and Management Information Systems. Her current research focuses on Digital Image Processing, Multimedia Forensics, Steganography and Steganalysis, Image and Audio Encryption, and Cryptography \& Security. She is an active member of several professional organizations including CRSI, CSI, OITS, and UACEE.

**Ram Chandra Barik** received his PhD Degree from the Department of Computer Science and Engineering at the National Institute of Technology (NIT), Durgapur, India in October 2021. Currently, he is working as an Associate Professor in the Department of Computer Science and Engineering at C. V. Raman Global University. Previously he has total 15 years of teaching Experience in Engineering Colleges including 2 years at Veer Surendra Sai Universities of Technology, Burla. Dr. Barik has 2 years of Industry experience at Accenture Services Pvt. Ltd. Bangalore and awarded with Certificate of Excellence in June 2007 product release for his outstanding contribution. Till date he has published 59 research publications among which 26 international journals indexed in SCIE, ESCI and Scopus, 33 conferences, and 12 patents to his credit. He served as a reviewer of many high impact factor SCIE journal including IET Image processing, Expert System with Applications, ACM transaction on Security and privacy, IEEE Access, Multimedia Tools and Applications, Journal of Super Computing, Evolutionary Intelligence, etc. His research interests include Image Processing, Computer Vision, NLP & Speech Processing, AI/ML/DL, IoT Network Optimization, Image Encryption and Blockchain. He is committed to advancing research and innovation in these interdisciplinary areas.

**How to cite this paper:** Jayanti Rout, Minati Mishra, Ram Chandra Barik, "Enhancing Image Forgery Detection through Dataset Balancing and a Fine-Tuned ResNet50: Focus on Copy-Move and Splicing", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.18, No.1, pp. 51-69, 2026. DOI:10.5815/ijigsp.2026.01.04