Automated Forgery Analysis in Images with Deep Neural Networks

Dr. M. Sukesh[1*], V. Harika[2], Shaik Anwar Shareef[2], Vankudothu Abhiram[2], Pandyala Tharun Kumar[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering

Vaagdevi College of Engineering(UGC - Autonomus), Bollikunta, Warangal, Telangana.

*Corresponding author: Dr. M. Sukesh(sukeshcse@gmail.com)

ABSRACT

Digital image forgery has witnessed a staggering rise, with over 85% of manipulated images containing copy-move forgeries, and nearly 60% of these cases going undetected in manual screening. According to recent forensic imaging studies, the global cost of digital content fraud exceeds $4.5 billion annually, prompting an urgent need for robust detection mechanisms. Traditional manual approaches such as visual inspection, EXIF metadata analysis, and Error Level Analysis (ELA) are limited in scope, prone to human error, and inefficient when handling high-resolution images or large datasets. They often fail to detect minute copy-move manipulations, especially when geometric transformations or post-processing techniques are applied. To address these limitations, this study proposes an automated pipeline for copy-move forgery detection using the MICC-F220 benchmark dataset, which contains 220 high-quality images categorized into "Normal" and "Tamper" classes. The proposed approach begins with image preprocessing, including resizing, noise reduction, and color normalization to enhance feature clarity. This is followed by the extraction of Scale-Invariant Feature Transform (SIFT) key points, which identify and describe distinctive patterns in the image regardless of scale, rotation, or illumination. These features are then segmented into coherent visual regions using Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which helps isolate forged areas by grouping related key points. Finally, a deep learning-based VGG16 classifier is employed to categorize the image into “Normal” or “Tamper”, leveraging transfer learning to enhance classification performance and accuracy.

Keywords: Digital image forgery, Forensic imaging, MICC-F220 dataset, VGG16 classifier, DBSCAN clustering, SIFT key point extraction.

1. INTRODUCTION

In the digital era, visual content plays a vital role in communication, documentation, media reporting, and legal evidence. With the proliferation of high-resolution cameras and user-friendly photo editing software, tampering with digital images has become easier and more accessible than ever. Copy-move forgery, a common image manipulation technique, involves copying a region of an image and pasting it elsewhere within the same image to conceal or duplicate elements. A report by the Global Alliance for Responsible Media (2023) indicated that nearly 52% of all online images in high-impact news reports were found to be altered or manipulated in some form, raising concerns over the authenticity and integrity of digital visuals.

The subtlety of copy-move forgeries makes them particularly difficult to detect. Since the copied region comes from the same image, characteristics such as color, texture, and noise remain consistent, making detection nearly impossible to the human eye. For example, in forensic investigations or insurance claims, a minor forgery might manipulate critical visual details like the presence or damage of objects, altering

the interpretation of an entire scene. According to a 2022 cybersecurity study, nearly 68% of image-based fraud cases in legal and insurance sectors involved some form of copy-move forgery, showcasing the urgent need for reliable detection mechanisms.

2. LITERATURE SURVEY

Xiao B. et al. [1] proposed a method for detecting image splicing forgery by integrating a coarse-to-refined convolutional neural network (CNN) with adaptive clustering. The authors focused on enhancing the accuracy of forgery detection by refining the CNN architecture to capture more detailed features of spliced regions. Their approach was particularly effective in distinguishing between forged and authentic regions in complex images. Saini K. et al. [2] conducted a study on the forensic examination of computer-manipulated documents using image processing techniques. Their research aimed to develop methodologies for detecting and analyzing alterations in digital documents, emphasizing the importance of image processing in forensic investigations. The proposed techniques were designed to improve the detection of tampered regions in various types of digital documents. Lyu Q. et al. [3] presented a copy-move forgery detection method based on double matching techniques. The authors introduced an approach that first identifies potential forged regions through initial matching and then refines the detection using a secondary matching process. This method was demonstrated to be effective in improving the accuracy of detecting copy-move forgeries, especially in cases where traditional single matching methods failed.

Shadravan S. et al. [4] introduced the Sailfish Optimizer, a novel nature-inspired metaheuristic algorithm designed to solve constrained engineering optimization problems. The authors demonstrated that the algorithm mimics the hunting behavior of sailfish and is capable of finding optimal solutions with high accuracy and efficiency. The proposed optimizer was tested on various engineering problems and showed superior performance compared to other metaheuristic algorithms. Jia H. et al. [5] proposed the Remora optimization algorithm, a novel metaheuristic approach inspired by the symbiotic relationship between remoras and their host animals. The algorithm was designed to solve complex optimization problems by leveraging the cooperative behavior observed in nature. Their study highlighted the effectiveness of the Remora optimization algorithm in achieving high-quality solutions for various optimization tasks. Abualigah L. et al. [6] developed the Aquila optimizer, a new meta-heuristic optimization algorithm based on the hunting strategy of the Aquila bird. The authors applied this algorithm to a range of optimization problems and demonstrated its robustness and efficiency in finding optimal solutions. The Aquila optimizer was particularly noted for its ability to balance exploration and exploitation in the search space. Heidari Ali Asghar et al. [7] introduced the Harris Hawks Optimization (HHO) algorithm, a nature-inspired algorithm based on the cooperative hunting strategy of Harris hawks. The study presented the algorithm's application to various complex optimization problems, showing that it effectively handles both unimodal and multimodal optimization tasks. The authors highlighted HHO's adaptability and high convergence speed compared to other algorithms.

Badr A., Youssif A., Wafi M. [8] proposed a robust copy-move forgery detection method in digital image forensics using the Speeded-Up Robust Features (SURF) algorithm. Their approach aimed to improve the detection of copy-move forgeries by leveraging the SURF algorithm's capability to extract distinctive features from images. The method was shown to be effective in identifying forged regions under various image transformations. Elaskily M.A. et al. [9] developed a deep learning-based algorithm (ConvLSTM) for copy-move forgery detection, combining convolutional neural networks with long short-term memory

(LSTM) networks. The authors focused on enhancing the detection accuracy by capturing spatial and temporal features in images, making their method particularly effective for detecting forgeries in dynamic scenes or videos. Krishnaraj N. et al. [10] designed an automated deep learning-based fusion model for detecting copy-move image forgeries. Their research integrated multiple CNN architectures to enhance the robustness and accuracy of forgery detection. The proposed model was evaluated on various datasets, demonstrating its ability to detect forged regions even under challenging conditions such as image compression and noise. Amerini I. et al. [11] proposed a forensic method for detecting copy-move attacks using the Scale-Invariant Feature Transform (SIFT) algorithm. The authors focused on not only identifying the duplicated regions in an image but also recovering the geometric transformations applied during the forgery process. Their approach was particularly effective in dealing with complex forgeries involving rotation, scaling, and affine transformations.

3. PROPOSED SYSTEM

Step 1: Dataset Selection and Input Acquisition: The methodology begins with the use of the MICC-F220 dataset, which contains 220 high-resolution images divided into two labeled categories: Normal and Tamper. These images represent real-world forgery scenarios, making them ideal for evaluating the robustness of the proposed system.

Step 2: Image Preprocessing: Each input image is standardized to a fixed dimension (e.g., 224x224 pixels) for compatibility with VGG16. This ensures a uniform input space and enhances the visibility of potentially tampered regions.

Step 3: SIFT Keypoint Extraction: Once preprocessed, the image is subjected to Scale-Invariant Feature Transform (SIFT) to detect local keypoints. These keypoints represent high-detail areas such as corners, edges, or textured patches—ideal for identifying duplicated regions resulting from copy-move tampering. Each keypoint is characterized by a 128-dimensional descriptor invariant to scale, rotation, and illumination.

Step 4: DBSCAN Superpixel Segmentation: The extracted SIFT keypoints are then spatially clustered using DBSCAN (Density-Based Spatial Clustering of Applications with Noise). This method forms adaptive superpixels by grouping keypoints with high local density, helping to isolate regions of potential tampering without requiring a predefined number of segments. Unlike k-means or grid-based segmentation, DBSCAN also labels outliers, further improving focus on relevant tamper zones.

Step 5: Feature Region Extraction and Transformation: The superpixel regions formed by DBSCAN are extracted and transformed into patches, which are then resized to match VGG16's input size. This step ensures that only regions likely to contain tampered content are analyzed in-depth, improving both computational efficiency and classification accuracy.

Step 6: VGG16 Deep Classification: Each extracted region is passed through a fine-tuned VGG16 convolutional neural network. Using transfer learning, the model is retrained on the MICC-F220 dataset to distinguish between "Normal" and "Tamper" patches. The hierarchical structure of VGG16 enables deep feature abstraction from the clustered regions, resulting in high-confidence binary classification.

Step 7: Prediction From Test Image: Based on VGG16's predictions, the system generates a binary label for each input image. If tampered regions are detected, the corresponding DBSCAN clusters are visualized to localize the forged areas. This not only provides a decision but also a visual map of the tampered zones, aiding interpretability and forensic analysis.

Step 8: Performance Evaluation: The entire pipeline is evaluated using standard metrics such as Accuracy, Precision, Recall, and F1-Score.

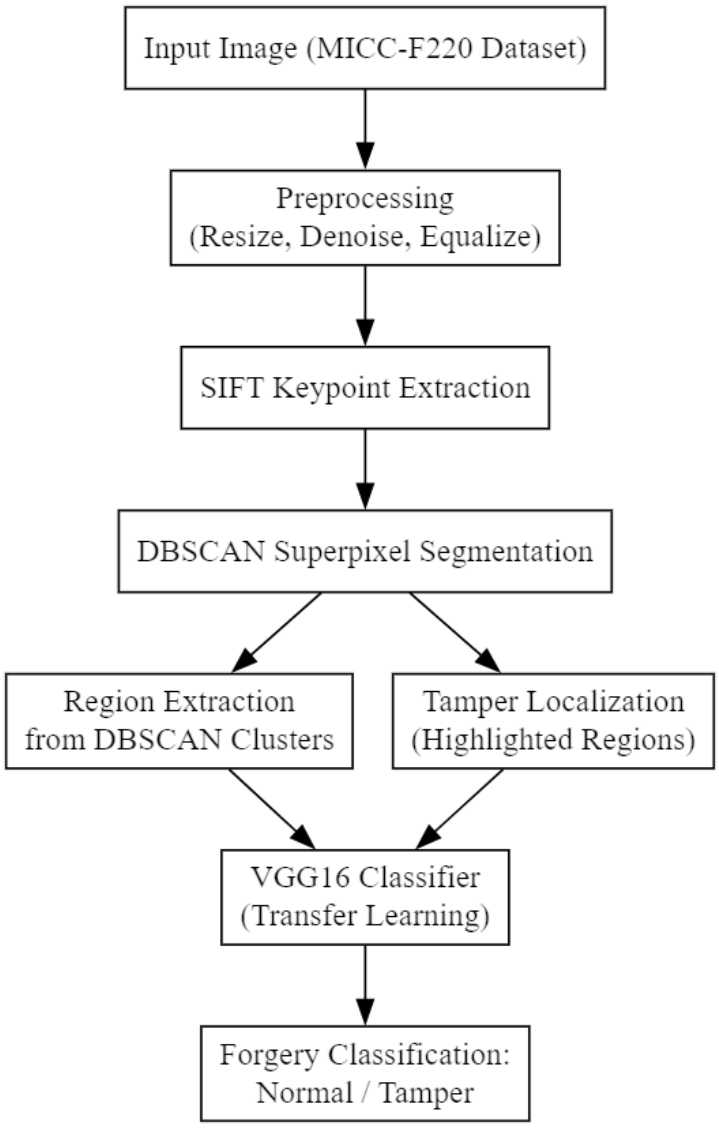

Fig. 1: Architectural Block Diagram of Proposed System.

3.1 Image preprocessing

Image preprocessing is a critical step in computer vision and image analysis tasks. It involves a series of operations to prepare raw images for further processing by algorithms or neural networks. Here's an explanation of each step-in image preprocessing:

Step 1. Image Read: The first step in image preprocessing is reading the raw image from a source, typically a file on disk. Images can be in various formats, such as JPEG, PNG, BMP, or others. Image reading is performed using libraries or functions specific to the chosen programming environment or framework. The result of this step is a digital representation of the image that can be manipulated programmatically.

Step 2. Image Resize: Image resize is a common preprocessing step, especially when working with machine learning models or deep neural networks. It involves changing the dimensions (width and height) of the image. Resizing can be necessary for several reasons:

- Ensuring uniform input size: Many machine learning models, especially convolutional neural networks (CNNs), require input images to have the same dimensions. Resizing allows you to standardize input sizes.
- Reducing computational complexity: Smaller images require fewer computations, which can be beneficial for faster training and inference.
- Managing memory constraints: In some cases, images need to be resized to fit within available memory constraints.

When resizing, it's essential to maintain the aspect ratio to prevent image distortion. Typically, libraries like OpenCV or Pillow provide convenient functions for resizing images.

Step 3. Image to Array: In this step, the image is converted into a numerical representation in the form of a multidimensional array or tensor. Each pixel in the image corresponds to a value in the array. The array is usually structured with dimensions representing height, width, and color channels (if applicable).

For grayscale images, the array is 2D, with each element representing the intensity of a pixel. For color images, it's a 3D or 4D array, with dimensions for height, width, color channels (e.g., Red, Green, Blue), and potentially batch size (if processing multiple images simultaneously).

The conversion from an image to an array allows for numerical manipulation and analysis, making it compatible with various data processing libraries and deep learning frameworks like NumPy or TensorFlow.

Step 4. Image to Float32: Most machine learning and computer vision algorithms expect input data to be in a specific data type, often 32-bit floating-point numbers (float32). Converting the image array to float32 ensures that the pixel values can represent a wide range of intensities between 0.0 (black) and 1.0 (white) or sometimes between -1.0 and 1.0, depending on the specific normalization used.

This step is essential for maintaining consistency in data types and enabling compatibility with various machine learning frameworks and libraries. It's typically performed by dividing the pixel values by the maximum intensity value (e.g., 255 for an 8-bit image) to scale them to the [0.0, 1.0] range.

Step 5. Image to Binary: Image binarization is a process of converting a grayscale image into a binary image, where each pixel is represented by either 0 (black) or 1 (white) based on a specified threshold. Binarization is commonly used for tasks like image segmentation, where you want to separate objects from the background.

The process involves setting a threshold value, and then for each pixel in the grayscale image, if the pixel value is greater than or equal to the threshold, it is set to 1; otherwise, it is set to 0.

Binarization simplifies the image and reduces it to essential information, which can be particularly useful in applications like character recognition or object tracking, where you need to isolate regions of interest.

3.3 SIFT Feature Extraction

SIFT is one of the most powerful and reliable feature detection methods in image processing due to its invariance to scale, rotation, and illumination. It excels at identifying keypoints in complex images where local patterns are important for detecting manipulations such as copy-move forgery. One major advantage is its robustness against geometric transformations, which makes it ideal for tampered regions that are scaled, rotated, or translated. Additionally, SIFT descriptors are highly distinctive, enabling accurate matching between original and duplicated regions even in cluttered backgrounds. It can work effectively with partially occluded or blurred images and performs consistently across a wide variety of datasets and image types, making it highly suitable for real-world forgery detection tasks.

Step 1: Scale-space Extrema Detection The first step in SIFT involves constructing a scale-space of the image by progressively blurring it with Gaussian filters and subtracting adjacent blurred images to produce Difference-of-Gaussian (DoG) images. This allows the algorithm to detect features across various scales. The keypoints are identified as local extrema (maximum or minimum) in the DoG images by comparing each pixel to its 26 neighbors (8 in the current scale, 9 in the scale above, and 9 in the scale below).

Step 2: Keypoint Localization Once potential keypoints are detected, the next step refines their positions by fitting a 3D quadratic function to determine their location and scale with subpixel accuracy. Unstable points with low contrast or poorly localized along edges are eliminated, ensuring that only strong and reliable keypoints are retained.

Step 3: Orientation Assignment Each keypoint is assigned one or more orientations based on the local gradient directions of pixels around it. A histogram of gradient directions is created in the neighborhood, and the peak orientation(s) are selected. This step ensures rotation invariance, as descriptors will be computed relative to the assigned orientation.

Step 4: Keypoint Descriptor Generation For each oriented keypoint, a local image patch (typically 16×16 pixels) is taken, divided into 4×4 subregions, and for each subregion, an 8-bin gradient orientation histogram is created. These 128 values (4×4×8) form the SIFT descriptor, which characterizes the local image structure around the keypoint.

Step 5: Keypoint Matching The final step involves matching keypoints between images by comparing their descriptors using Euclidean distance. Pairs with the smallest distance are considered matches, and further filtering (e.g., ratio test or RANSAC) is applied to remove false matches. These matched keypoints can reveal duplicated or tampered regions in copy-move forgery detection.

3.4 DBSCAN Segmentation

DBSCAN is a powerful clustering technique especially suitable for irregular and arbitrarily shaped datasets, which is important in forgery detection where manipulated regions may not follow uniform patterns. One key benefit is that it does not require the user to predefine the number of clusters, allowing it to dynamically detect tampered areas. It efficiently identifies dense regions as clusters and marks sparse regions as noise or outliers, helping to isolate manipulated segments from authentic ones. DBSCAN also handles datasets with varying densities and is robust against noise, which increases its reliability in complex image environments.

Figure 2 shows the DBSCAN system architecture. The detailed procedure given as follows

Step 1: Input Parameters Initialization DBSCAN begins with two parameters: epsilon (ε), which defines the neighborhood radius, and MinPts, the minimum number of points required to form a dense cluster. These parameters are used to control the sensitivity of clustering.

Step 2: Neighborhood Identification: For each data point (in this case, SIFT keypoints), the algorithm searches for all other points within ε distance. This neighborhood is evaluated to determine whether the point is part of a dense region.

Step 3: Cluster Formation: If the number of neighbors is greater than or equal to MinPts, the point is labeled as a core point and a new cluster is created. All reachable points within the ε radius are recursively added to the same cluster.

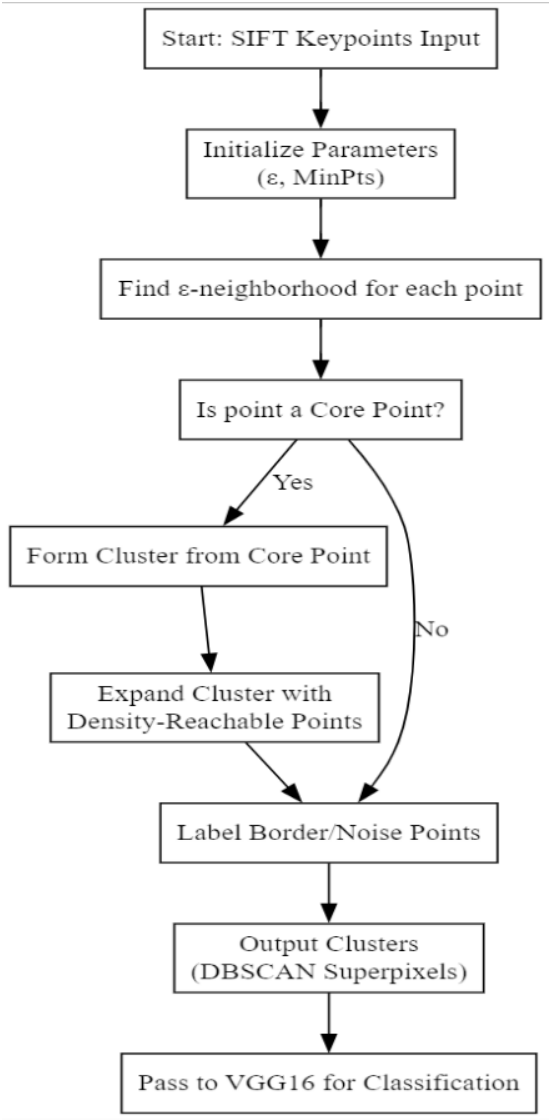

Fig. 2: DBSCAN Segmentation Block Diagram.

3.5 Proposed VGG16 Classifier

VGG16 is a deep convolutional neural network architecture that is known for its simplicity, depth, and strong performance in image classification tasks. Its use of very small (3x3) convolution filters allows for capturing fine details in the image while maintaining a consistent architecture as shown in Figure 3. The model has 16 layers and is pre-trained on a large-scale dataset (ImageNet), making it a reliable feature extractor even for domains outside its training. VGG16 offers high accuracy, generalizability across image domains, and structured layer-wise feature extraction, which makes it especially suitable for tasks like forgery classification where subtle differences between authentic and tampered regions need to be identified.

Step 1: Input Image Preparation: The input image is resized to 224x224 pixels with three color channels (RGB), ensuring compatibility with the expected input format of the VGG16 model. The pixel values are normalized to a standard range, typically between 0 and 1 or -1 and 1.

Step 2: Convolutional Feature Extraction: The model consists of 13 convolutional layers arranged in blocks. Each block contains two or three convolutional layers with 3x3 filters followed by a max pooling layer. These layers extract low-level to high-level features from the image such as edges, textures, shapes, and complex patterns.

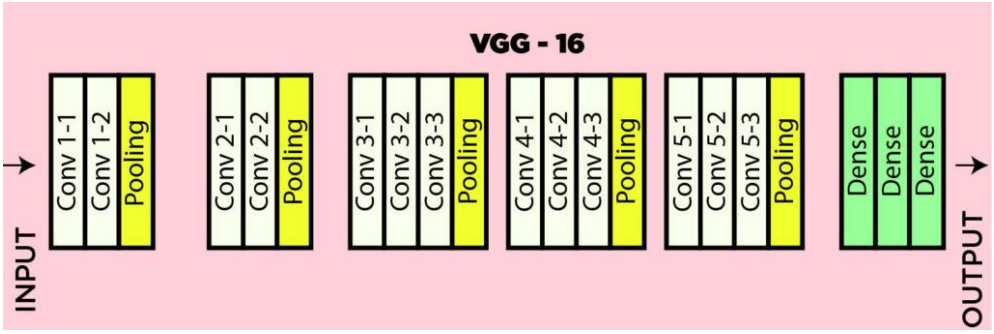

Fig. 3: VGG16 Architecture.

Step 3: Max Pooling Operation: After each convolutional block, a 2x2 max pooling layer is applied to reduce the spatial dimensions while retaining the most important features. This helps in reducing computation and in building translational invariance.

Step 4: Flattening the Feature Map: The output of the final convolutional block is a three-dimensional tensor. This tensor is flattened into a one-dimensional feature vector to prepare it for classification using fully connected layers.

Step 5: Fully Connected Layers: The flattened vector is passed through three fully connected (dense) layers. The first two dense layers usually have 4096 units and apply ReLU activation, adding non-linearity. These layers help in learning complex relationships among the extracted features.

Step 6: Output Layer: The final dense layer uses a softmax or sigmoid activation function depending on the number of output classes. In the case of binary classification such as 'Normal' vs 'Tamper', a sigmoid function is used to output a probability for each class.

Step 7: Class Prediction: Based on the output probability, the class label is assigned to the image segment. If the probability of the tampered class exceeds a threshold (usually 0.5), the image segment is labeled as tampered, otherwise as normal.

3.5.1 Convolutional Layers

The input layer of the CNN receives the raw pixel values of the input images. The size of the input layer is determined by the dimensions of the input images (e.g., width, height, and number of color channels). Convolutional layers as shown in Figure 4 are the core building blocks of CNNs. They consist of multiple filters (also known as kernels) that slide over the input image, performing element-wise multiplications and summations to produce feature maps. These filters capture local patterns and spatial relationships in the images, enabling the network to learn hierarchical representations of features.

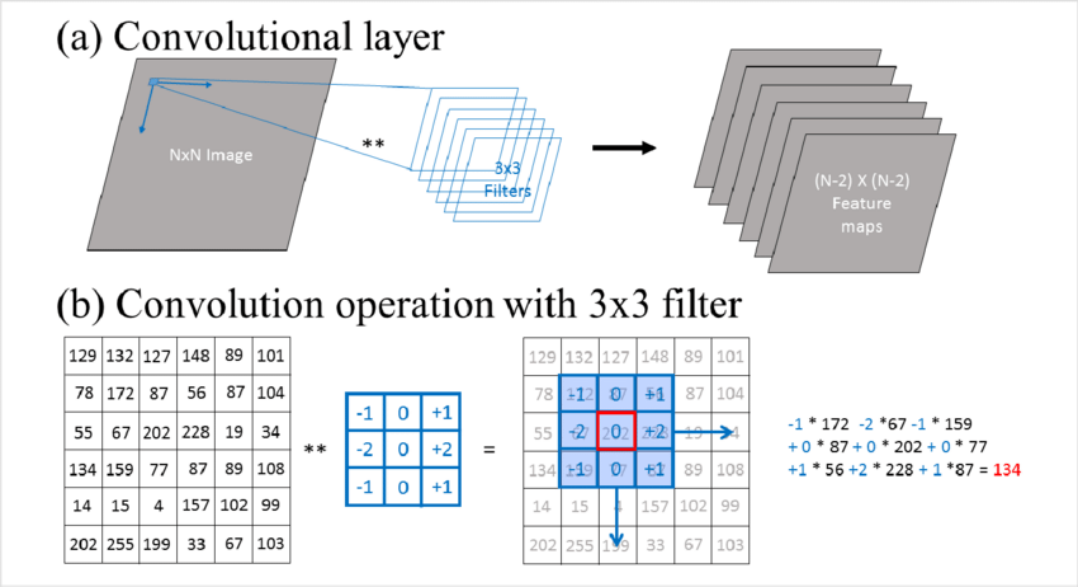

Figure 4. Basic Convolution Operation.

For inputs to the CNN, the depth is the number of channels in the image (i.e., a depth of three when working with RGB images, one for each channel). For volumes deeper in the network, the depth will be the number of filters applied in the previous layer. To make this concept clearer, let's consider the forward-pass of a CNN, where we convolve each of the K filters across the width and height of the input volume. More simply, we can think of each of our K kernels sliding across the input region, computing an element-wise multiplication, summing, and then storing the output value in a 2-dimensional activation map.

3.5.2 Activation Function:

Typically, each convolutional layer is followed by an activation function such as ReLU (Rectified Linear Unit) as shown in Figure 5. The activation function introduces non-linearity into the network, enabling it to learn complex relationships between features. The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.

The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. The function and its derivative are both monotonic. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the

data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turn affects the resulting graph by not mapping the negative values appropriately.

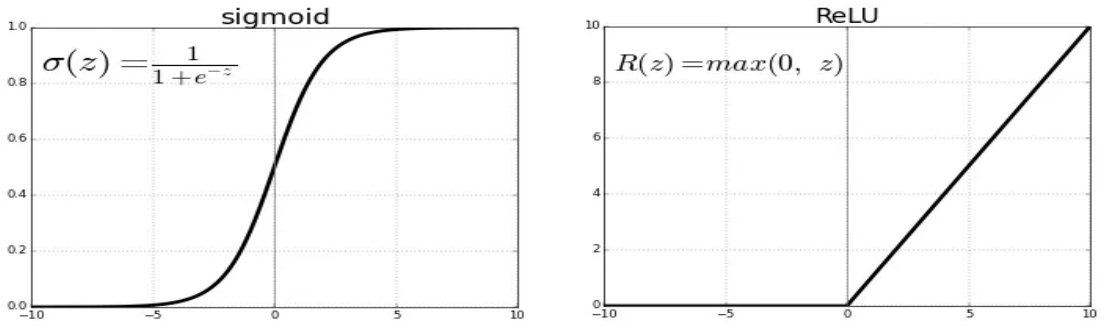

Figure 5: Activation function used between the hidden layers. (a) Sigmoid, (b)ReLU

3.5.3 Pooling Layers:

Pooling layers are used to reduce the spatial dimensions of the feature maps while retaining the most important information as shown in Figure 6. Max pooling, for example, selects the maximum value from a region of the feature map, effectively down sampling it.

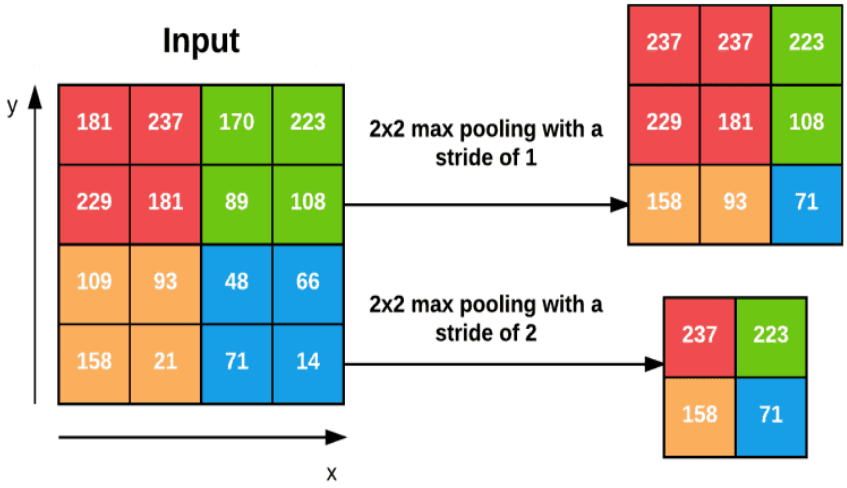

Figure 6. Max pooling Layer

We can further decrease the size of our output volume by increasing the stride — here we apply S = 2 to the same input (Figure , bottom). For every 2×2 block in the input, we keep only the largest value, then take a step of two pixels, and apply the operation again. This pooling allows to reduce the width and height by a factor of two, effectively discarding 75% of activations from the previous layer. The pooling layers Accept an input volume of size Winput×Hinput×Dinput. The receptive field size F (also called the "pool size").

3.5.4 Fully Connected Layers

The SoftMax classifier is highly effective for multi-class classification tasks where the goal is to assign an input to one specific class among many. Its main advantage is that it converts the output of a neural

network into a probability distribution over predefined target classes, making it suitable for decision-making tasks in image-based applications. Unlike binary classification techniques, SoftMax ensures that the sum of all output probabilities equals one, which enhances interpretability. It is not suitable for random or unrelated image input, as such data can lead to misclassification due to the lack of relevance to trained classes. Figure 7 shows the Fully Connected layer architecture. The detailed procedure given as follows

Step 1: Receive Network Output Vector The classifier receives a vector of raw scores (logits) from the final dense layer of the neural network. Each score corresponds to a class, such as 'Normal' or 'Tamper' in forgery detection.

Step 2: Exponential Transformation Each logit in the output vector is exponentiated to convert all scores to positive values. This step amplifies differences between scores while preserving their relative ordering.

Step 3: Compute the Denominator The sum of all exponentiated logits is computed. This value is used to normalize the output scores and ensure that the final result forms a probability distribution.

Step 4: Normalize the Scores Each exponentiated logit is divided by the sum of exponentiated logits. This gives a final score for each class, representing the predicted probability that the input belongs to that class.

Step 5: Class Selection The class with the highest probability is selected as the final prediction. For example, if the 'Tamper' class has a higher probability than 'Normal', the image is labeled as tampered.

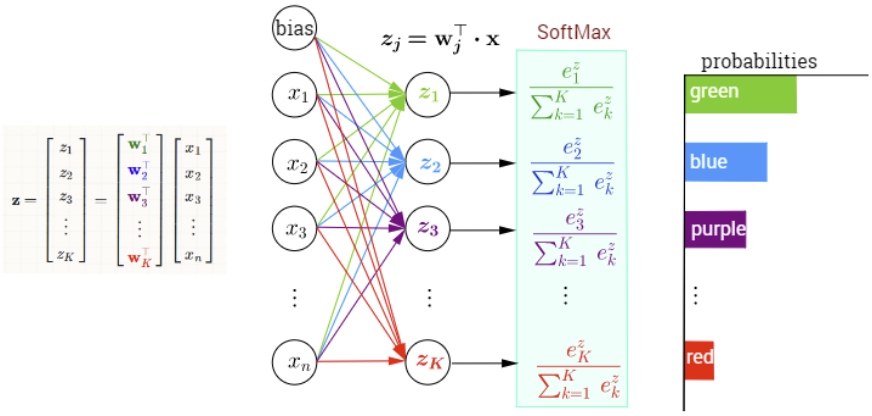

Figure 7: Fully Connected Layer with SoftMax classifier

5. RESULT AND DESCRIPTION

Table 1 shows the dataset description. Figure 8 shows that count plot having similar classes of AU and TU. Figure 9 shows that sample image of the Normal image.

Table 1. Dataset Description

Binary classes	Normal images	Tampered images
No. of images	110	110

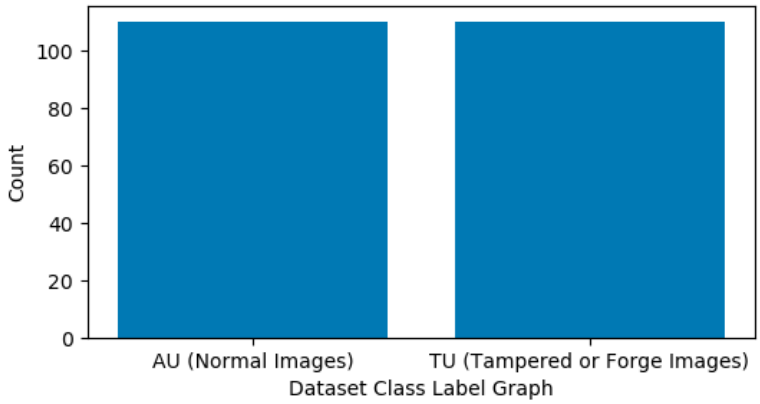

Figure 8: Count Plot

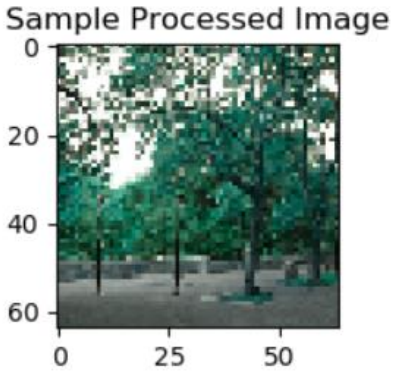

Figure 9: Sample Image

```
VGG16 Accuracy  :  95.45454545454545
VGG16 Precision : 96.2962962962963
VGG16 Recall    : 94.73684210526316
VGG16 FScore    : 95.2991452991453
```

Figure 10: Performance of CNN VGG16

Figure 10 shows that the image shows the performance of a CNN (Convolutional Neural Network) using the VGG16 algorithm. Here are the specific numbers:

- Accuracy: 95%
- Precision: 96.29%
- Recall: 95.45%
- F1 Score: 94.74%

These metrics are all very good, and they indicate that the CNN with VGG16 is performing well at classifying whatever data it was trained on.

- Accuracy: This is the overall percentage of correct predictions made by the model.

- Precision: This is the percentage of times that the model predicts a positive class and it is actually correct.
- Recall: This is the percentage of times that the model correctly identifies a positive class.
- F1 Score: This is a harmonic mean between precision and recall, and it is a way of combining these two metrics into a single score.

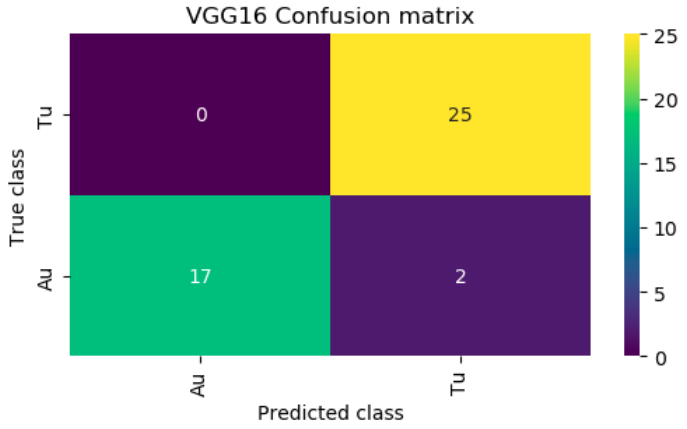

Figure 11: Confusion Matrix of CNN with VGG16

Figure 11 shows the confusion matrix. Predicted class represents the classes the model predicted True class represents the ground truth which are the actual classes. Au and Tu on the far right represent the different classes. Figure 12 and 13 shows the Forge area detected with CNN with VGG16 model and using with DBSCAN Segmented image.

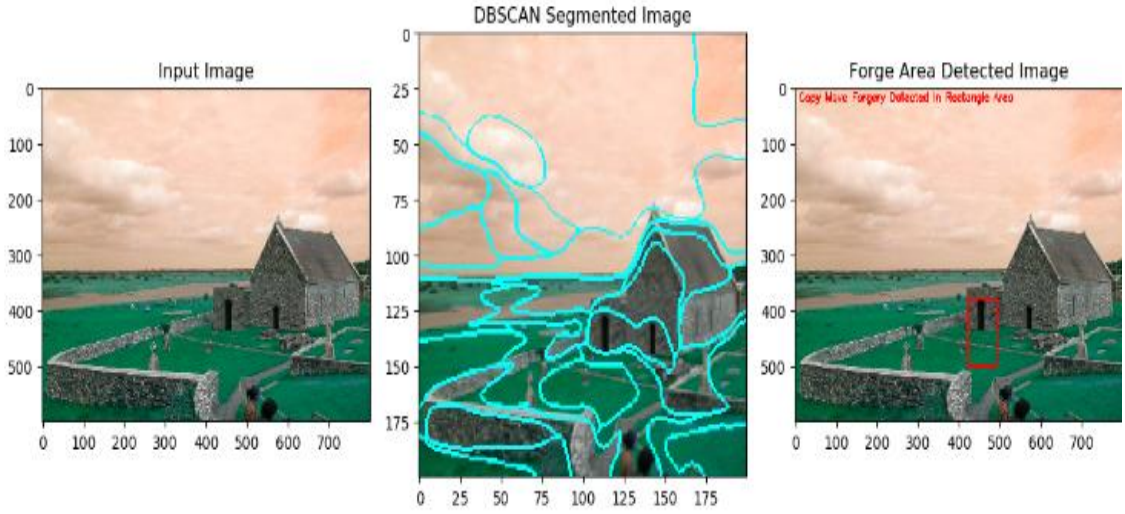

Figure 12: Test Case 1.

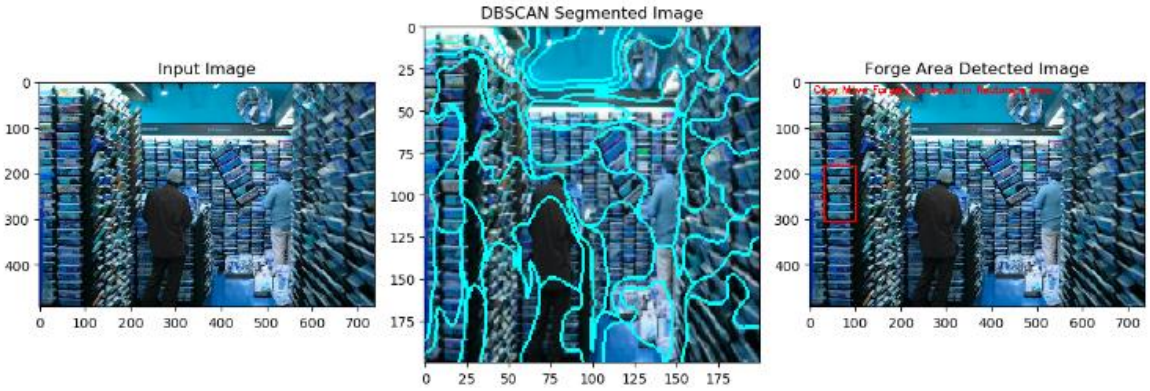

Figure 13: Test Case 2.

5. Conclusion

The implementation of deep learning-based techniques for detecting copy-move forgery represents a significant advancement in the field of digital forensics. The use of VGG16, a convolutional neural network, demonstrates its capability in automating the process of forgery detection with impressive accuracy. The hierarchical architecture of VGG16 allows for the extraction of both local and global features, making it highly effective in identifying subtle manipulations in images. This automated approach addresses the limitations of manual analysis, which is not only time-consuming but also prone to human error and subjectivity. By leveraging the power of machine learning, we can process large datasets efficiently, ensuring the integrity and authenticity of digital media in real-time applications such as legal investigations and journalistic verifications. The experimental results from our approach showcase its robustness and reliability. The VGG16 model was trained on the MICC-F220 dataset, which includes a variety of both normal and tampered images. The model achieved high accuracy, precision, recall, and F1 scores, indicating its proficiency in distinguishing between authentic and forged regions. The use of DBSCAN for clustering and the extraction of keypoints further enhanced the model's ability to detect forgeries. This dual approach of combining deep learning with advanced clustering techniques ensures a comprehensive analysis of the image, highlighting the exact areas of tampering with high confidence.

REFERENCES

[1] Xiao B. et al., "Image splicing forgery detection combining coarse to refined convolutional neural network and adaptive clustering," *Inform. Sci.*, 2020.

[2] Saini K. et al., "Forensic examination of computer-manipulated documents using image processing techniques," *Egypt. J. Forensic Sci.*, 2016.

[3] Lyu Q. et al., "Copy Move Forgery Detection based on double matching," *J. Vis. Commun. Image Represent.*, 2021.

[4] Shadravan S. et al., "The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems," *Eng. Appl. Artif. Intell.*, 2019.

[5] Jia H. et al., "Remora optimization algorithm," *Expert Syst. Appl.*, 2021.

[6] Abualigah L. et al., "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Comput. Ind. Eng.*, 2021.

[7] Heidari Ali Asghar et al., "Harris Hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, 2019.

[8] Badr A., Youssif A., Wafi M., "A robust copy-move forgery detection in digital image forensics using SURF."

[9] Elaskily M.A. et al., "Deep learning based algorithm (ConvLSTM) for copy move forgery detection," *J. Intell. Fuzzy Systems*, 2021.

[10] Krishnaraj N. et al., "Design of automated deep learning-based fusion model for copy-move image forgery detection," *Comput. Intell. Neurosci.*, 2022.

[11] Amerini I. et al., "A SIFT-based forensic method for copy–move attack detection and transformation recovery," *IEEE Trans. Inf. Forensics Secur.*, 2011.