STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–24.
Published online in International Academic Press (www.IAPress.org)

An Efficient Approach to Detect a Copy-Move Forgery in Videos

Mahmoud Atef [1,*], Mohamed S. Farag [2], Mohammed Abdel Razek [2], Gaber Hassan [3]

[1] *Department of Computer Science, Faculty of Information Technology and Computer Science, Sinai University, Al-Arish 45511, Egypt*
[2] *Department of Mathematics, Faculty of Science, Al-Azhar University, Cairo, Nasr City, Egypt*
[3] *Department of Computer Science, Faculty of computers and information, Arish University, Al-Arish 45511, Egypt*

Abstract In recent years, the advancement in editing technologies, especially multimedia, has increased the use of image and video modification. However, object-based video tampering, such as including or excluding objects in the video frames, poses a significant challenge for video authentication. Video forgery detection is critical to maintaining media integrity since multimedia content is commonly used across numerous fields. Regarding different types of manipulations in the domain of videos, copy-move forgery is typical and complex at the same time. This study introduces a relatively efficient DCNN model to detect forged copy-move videos. The proposed method can be described as follows: first, the video is divided into frames, then a convolutional neural network model is employed to traverse each frame to establish its features. We then use the described features to train a new CNN model, making it possible for us to determine whether a particular frame is real or fake. Also, the structure incorporates batch normalization to enable easy layer weight initialization, ease in training at a higher learning rate, high accuracy, non-overfitting, and process stability. We also conducted extensive practical experiments on a massive video dataset, including original and manipulated content. We use specific performance metrics like accuracy, precision, recall, F1-score, and the Matthews Correlation Coefficient to assess the performance of the suggested model. The recommended method demonstrated superior performance compared to all previously proposed methods on the GRIP, VTD, and SULFA datasets. The model's accuracy was 95.60%, 96.70%, and 100%, with the shortest time of 25.10 sec, 27.35 sec, and 20.22 sec, respectively.

Keywords Convolutional Neural Network, Deep Learning, Video Forgery Detection, Copy-Move Forgery, Regularization.

DOI: 10.19139/soic-2310-5070-2677

1. Introduction

Recently, more video material has been sent over the Internet. One reason is that the technology for editing videos is getting better. We can quickly and accurately edit complicated and advanced videos using commercially available software tools and open-source libraries. You do not need to be a professional to edit videos. Because these video editing tools and devices are so standard, changing video content to hide things, events, or surveillance footage is getting easier without leaving any proof. Because of this, it is happening more often that videos that have been changed are passed off as real. Copy-move is a popular but difficult method for creating manufactured videos. This method involves copying and pasting video segments into other areas, resulting in fabricated or misleading content. These developments make it difficult to determine if video content is accurate and honest upon viewing. For that, acquiring trustworthy detection techniques and technologies is crucial for proving the reliability and authenticity of video content [1, 2].

D'Amiano et al. [3] described a dense-field method with a patch match for video copy-moves detection. Using the Patch Match algorithm, it splits the video frames into overlapping patches and then discovers patches that

*Correspondence to: Mahmoud Atef (Email: mahmoud.atef@su.edu.eg). Department of Computer Science, Faculty of Information Technology and Computer Science, Sinai University, Al-Arish 45511, Egypt.

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 202x International Academic Press

are similar to the input patch. It also provides a dense field that states the motion of each patch between frames. Therefore, the dense fields of the different frames can help check and identify copy-move forgeries in the video with the help of this algorithm. The paper's authors also noted that for several benchmark real datasets, the proposed approach correctly identifies the faults and locates them with very high accuracy compared to similar approaches. The suggested approach in the study shows that the newly developed framework performs well in detecting and pinpointing areas of copy-move forgeries for various video applications.

In light of the challenging issues that may hamper video copy-move forgery detection, Zhong et al. [4] proposed a technique called VCMFD. Applying the most effective SIFT (Scale-Invariant Feature Transform) framework, they extract the necessary information in all cases of video copy-move modification. Besides, they give a practical approach – the keypoint-label matching (FKLM), based on the label and keypoint clustering for high-dimensional data that increases matching accuracy. The experimental findings show that the suggested method's detection accuracy is very high, the false positive rates are minimal, and the F1 scores are significantly enhanced. The GRIP 2.0 dataset and a combination of the SULFA 2.0 and REWIND datasets demonstrated improvements of at least 16% and between 0-8%, respectively, compared to the alternatives. In addition, it is faster than existing methods, thus making it efficient for use in the analytical process.

Several motion residual and object-tracking approaches have been used by researchers to come up with cases of altered video motion. In the case of individual frames, Oliaei and Azghani [5] described a technique for detecting motion. This method follows the movement of objects within the sequence of frames in the video through the mean-shift technique once it filters out the regions with motion residuals through the analysis of comparable things. Once it focuses on detecting objects and following them across frames, it recognizes temporal and spatial changes within sequences. These techniques match motion sequences depending on object similarity, using simulations on datasets like SULFA and GRIP.

Mohiuddin et al. [6] conducted a recent research study to develop an comprehensive framework to detect copy-move forgery in video. The suggested method suggests that patches from video frames can be efficiently learned using the sliding window technique, and many CNN models were trained with these patches employing various hyperparameters to extract diverse features. Concerning the incorporation of a patch into the videos, models determine the outcome through a weighted voting process. They also present an additional dataset called Vision for Visibility Dataset (VIVID), which contains several videos featuring copy-move forgeries. In the same year, Jasim, and Atia [7] presents an evolutionary learning algorithm for designing convolutional neural networks for deepfake detection. The algorithm uses residual and dense networks, generating different CNN structures. The method works on different datasets without preprocessing and outperforms state-of-the-art models by 1% in accuracy, indicating intuitive design as a new direction for better generalization.

Kim et al. [8] synthesized a novel object-based frame identification network consisting of reciprocally overlapping motion residuals for higher accuracy in video frame identification. This method uses a deep neural network to improve temporal variations and overlapping windows. Additionally, the process employs a non-symmetric network topology to test and train a single primary convolutional neural network efficiently. This involves optimizing learning by using two networks with identical architectures during training. They recommend and assess two testing strategies for two- and three-class frame identifications compared to existing techniques. The analysis of the experimental results supports the usage of the proposed solution in terms of providing consistent identification outcomes in both cases.

Rodriguez-Ortega et al. [9] addressed a fast-growing threat in which high-quality fake photographs are spread through social networks and media. They briefed about the criticality and necessity of a better mechanism underlying the recognition system that can easily identify such fake news. The copy-move technique, a common technique in picture and video editing, emphasizes the duplication of specific sections of an image to alter its content. Conventional image processing methods rely on hand pattern recognition to find duplications, therefore restricting their capacity to scale for extensive classification applications. Deep learning techniques, on the other hand, have shown improved outcomes and possibilities to overcome these difficulties. Still, they often deal with generalizing difficulties that mostly rely on the volume and diversity of training data as well as the choice of appropriate hyperparameters. Together with their training and inference times, the evaluation compares these models using several criteria.

Singh and Singh, k. [10] conducted a systematic review of the identification of copy-move forgery in digital videos with an emphasis on the passive approach. Given that people are employing digital videos to capture life's occurrences and sharing those occurrences on social media, the security of such videos is an essential issue. This paper starts by explaining copy-move forgery and its categories and goes further to provide a view of the existing passive strategies. Last, it overviews these techniques in terms of features, datasets, parameters, forgery types, and the limitations in the table format. The survey also defines the parameters of the evaluation, the datasets, and the detection tools before going over future trends in the field and the issues arising from the use of deep learning for automatic detection.

Ulutas et al. [11] proposed another video forgery detection system where Local Difference Binary (LDB) features have been used with the help of deep learning models. Initially, the approach selects LDB features to extract local texture details from the video frames; subsequently, it constructs a binary code for each block. The substance of the real and fake video frames is then translated to binary numbers and, next, classified with the CNN deep neural network. Their approach strengthens the system by applying data augmentation methods and getting more training samples. The suggested method performs well and yields a satisfactory detection accuracy percentage. The offered research is a promising contribution to identifying videos forged using deep learning and adequate LDB features, which would be valuable in various fields.

Further, Kumar et al. [12] proposed a robotic forgery detection system to detect the frame-insertion forgery in videos. Ranging their approach using deep features identifies any extreme shift that may lead to forgery. This method computes correlation coefficient distance across frames and identifies important features with the parallel convolution neural network. This method computes correlation coefficient distance across frames and identifies important features with the parallel convolution neural network. This calculation helps to detect disassociation between frames, which is the primary indicator of video tampering. Common databases such as VIFFD and SULFA confirm the applicability of this approach. On the VIFFD dataset, the method is 86.5% accurate; on the SULFA dataset, it is 92% accurate; it additionally detects inserted frames that can recreate original videos.

El-Shafaiet et al. [13] made a detailed examination of the extent of jeopardy that advanced online processing software has brought in the area of video' authenticity. It has developed a new kind of video tempering software that is capable of twisting the truth and sharing fake messages, especially on a platform that specializes in uploading and sharing videos and images like YouTube, Facebook, WhatsApp, Twitter, Instagram, and many others. When it comes to the distinction between works, it is necessary to indicate that genuine and manipulated videos have a significant impact on the field of journalism, law enforcement activity, surveillance, and medical imaging, but various methods have been developed to mislead them. Further, using deep learning, the study provided advanced theories of development involving the use of perceptual analysis recurrent, deep convolutional, adaptive analysis, as well as the use of adaptive detectors, all integrated for development.

Recently, many authors have been using convolutional neural networks (CNNs) in various applications due to their outstanding ability in pattern recognition and classification. Among these applications are disease detection [14-16], emotion detection [17, 18], and others.

In this study, we propose an innovative approach employing deep learning to identify object-based forged content in video frames. Thus, developing detection algorithms and related technologies is important, as they constitute the foundation for creating trust and validity in video content across diverse applications.

Hence, our work aims to ensure the authenticity of objects that appear within the frames of a video. Given a pre-trained CNN, the most important features from each frame are extracted, then modelled, and the learned model is then used to predict whether a given frame is real or manipulated. After each convolutional layer, we add batch normalization and pass the data through the max-pooling layer. The two regularisation techniques provided, L1 and L2, assist the model in enhancing the steadiness and profitability resulting from batch normalizing during the training as well as eliminating overfitting [19, 20]. The objective of this work is to design and implement a new CNN model with batch normalization in the described framework in order to enhance the capabilities of the model to learn discriminative features for copy-move forgery detection. We also conducted extensive practical experiments on a massive dataset of videos, which included both original and manipulated content. We use specific performance metrics like precision, recall, accuracy, F1-score, and the Matthews Correlation Coefficient to assess the performance of the suggested model. Overall, the proposed model better detects copy-move video forgery

than the previous methods. The utilization of this model remains broad, and some applications include crime investigations and the verification of multimedia content, enhancing the credibility of multimedia content in today's world. This study presents the primary contributions as follows:

• Deep convolutional neural network (DCNN) technique is proposed for identifying copy-move object-based forgeries in videos.

• This approach significantly outperforms the previous CNN models by applying a method known as batch normalization, thus improving the efficiency even further. L1 and L2 regularization were applied to further improve performance and prevent overfitting.

• Batch normalization contributes to input data standardization throughout the training process, minimizing internal covariate shift. This uniformity can result in better and more accurate learning, as well as training and accuracy in validation data. Batch normalization manages to make the rate of convergence of the training process significantly faster in order to reach the peak of efficiency.

• We employed three datasets to ensure the accuracy of our research: SULFA, Video Tampering Dataset VTD, and GRIP are the most widely used of these, as reported [21–24].

• We experimented with the three benchmark datasets described above and did a comparative to examine the efficiency difference of the suggested model to the previously established models [3], [5], [12], [25-27], [34], [37] and [38]. Consequently, the comparison identified the fact that the suggested technique had a better performance than the other models under consideration.

• Our proposed model is quite accurate and requires minimal parameters as compared to the recently published techniques.

The other parts of this study have been organized as follows: Section 2 provides the description of the traditional CNN model, a batch normalization concept, and a detailed explanation of the proposed method. Section 3 presents an exhaustive analysis of the results, a comparison with other related studies using the aforementioned criteria, and a final discussion in Section 4. Section 5 includes an overview of the outcomes and conclusions of the research.

2. Research Method

2.1. The Structure of CNN Model

The conventional CNN can be divided into two main components, as depicted in Figure 2 below. The layers of convolution and pooling are used in the first part for feature extraction. The next part is the classifier part, and in most of the deep models, it may involve a fully connected layer, which a linear layer can replace. These layers use the features derived from other earlier convolutional and pooling layers to sort the input image into various classes. However, a new trend in CNN architecture is replacing fully connected layers with several global average pooling layers. In this approach, the features obtained from the last convolutional layer are subjected to a worldwide average pooling step instead of flattening them and passing them through one or more fully connected layers. This process produces one value for each feature map by summing all the features in the last convolutional layer [28, 29, 37]. Generally, CNN models usually consist of a sequence of layers, each performing special tasks in the overall feature extraction and classification process. The basic framework of a CNN model can be characterized, per [11, 28, 29] as follows: Input layer: is the first layer of the network, which simply sends the image or data that has been inputted to the successive layers. Convolutional layers: A sequence of convolutional layers that contain learnable filters moving over the input image and performing convolution on it. The number of filters within the layers may rise with the network's depth, thus allowing the model to learn more complex characteristics. This lets the model identify ever more complicated features. Activation function: An activation function is applied after each convolutional layer, introducing non-linearity and allowing the network to perform linear transformations. Examples include sigmoid, tanh, ReLU, leaky ReLU, and softmax. Each function is different and can be used for different problems and designs. Standard functions like ReLU and sigmoid provide probabilities for events on a scale from 0 to 1, with ReLU returning 1 for positive values and 0 for negative values, as shown in Figure 1.

• ReLU function: is a simple piecewise function. It gives a result of 1 if the input value is positive and 0 if the value is not positive. This can be seen in equation (1):

$$ReLU(x) = Max(0, 1) \tag{1}$$

• Sigmoid function: A mathematical function known as the sigmoid function provides a value between 0 and 1. Equation (2) suggests that this number represents your likelihood of belonging to a specific group:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

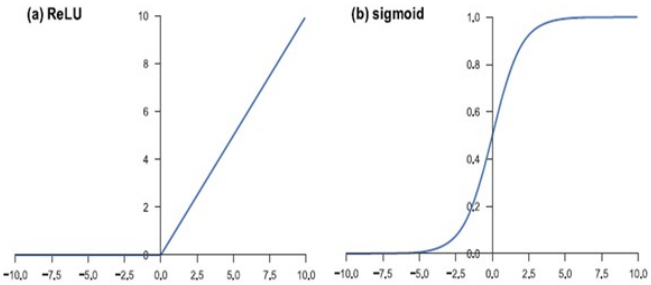

Figure 1. The sigmoid and ReLU activation functions.

• Pooling layers: These layers keep essential characteristics while reducing the dimensionality of feature maps. These layers result in fewer parameters to learn and a reduction in network computation. One frequent method is max pooling, which uses each window's maximum value.

• Fully connected layers: After the convolution and pooling layers extract and reduce the features, a few fully connected layers process this information to produce the final outputs, like the likelihood of each class in classification tasks. As previously noted, every fully connected layer will be followed by a nonlinear function such as ReLU.

• Output layer: Responsible for producing the final model output, this layer generates a probability distribution over the potential output classes.

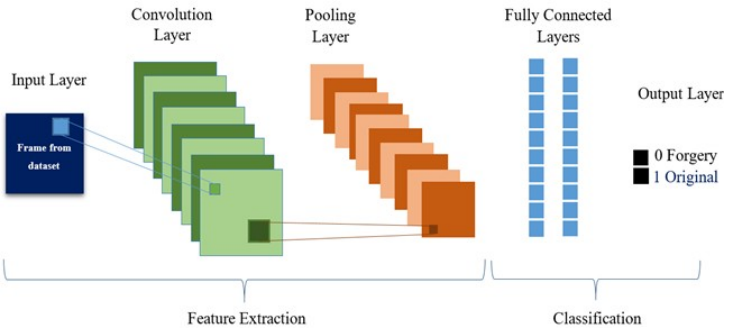

Figure 2. The structure of CNNs model

2.2. The Concept of Batch Normalization

Deep learning uses batch normalization to ensure the normalization of each layer's inputs. Its goal is to stabilize the network's distribution and speed up training by reducing the internal covariate shift [19].

2.2.1. The mechanism of batch normalization process:

- Firstly, the mean (average) and standard deviation of all feature activations in each mini-batch will be computing during training.
- The normalization step arranging data in a database to remove redundancy and enhance efficiency. Procedure: Normalize the activations by dividing them by the mean and standard deviation computed in step 1.
- Scaling and Shifting: Introduce trainable parameters (beta and gamma) for each feature that allow the network to select the most suitable scaling and shifting for normalized activations.
- Learning: Back propagation modifies the other network parameters and the learnable parameters (beta and gamma) during the training process. The mathematical representation of the batch normalization formula is below:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu^{(k)}}{\sqrt{\sigma^{2(k)} + \varepsilon}} \tag{3}$$

where the input to the layer is represented by$x^{(k)}$, with $\mu^{(k)}$ representing the mean of activations, $\sigma^{2(k)}$ as the variance, ε for numerical stability and prevent division by zero, $\hat{x}^{(k)}$ is the normalized input, and k represents the layer. After normalization, we scale and shift the normalized input to obtain the batch normalization output.

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)} \tag{4}$$

where $\gamma^{(k)}$ and $\beta^{(k)}$ represent the scale and shift learnable parameters for each feature in the layer.

2.2.2. The The Advantages of Batch Normalization: Batch normalization improves the training process's stability by reducing the influence of internal covariate shifts. This leads to more steep gradients and less oscillations which in turn leads to a faster convergence, thus a faster training. This makes it possible to reduce the usage of other regularization methods like the dropout. This makes it possible to employ high learning rates, which keeps the network from straying and becoming unstable. The generalization performance is enhanced because inputs of every layer are standardized.

2.3. The Proposed Model

Along the lines of the deep convolutional neural networks, the proposed model aims to identify copy-move video frauds. The suggested model applies batch normalizing to all the layers, which include a convolutional layer, a batch normalization layer, and a dropout layer, followed by a max pooling layer. It has three main components, including pre-processing, convolutional, and classification, as depicted in Figure 3 below. The model is compiled using the Adam optimizer, and the default learning rate 0.001 is typically used, a batch size of 32 is used in the training process, and the convolution layers in the model use a kernel size of (3, 3) with valid padding and a stride of (1, 1), which reduces the spatial dimensions with each convolution. We can summarize the model's architecture as follows:

1. Pre-processing: In this Step, the video input is acquired and divided into frames, which are then used as input for the subsequent steps as images.

2. Convolutional Part: The convolution part of the process is the algorithm responsible for the feature extraction of the input frames. It has 13 convolutional layers with filter sizes of 16, 32, 64, 128, and 256 in augment as the number of convolutional layers gets deeper. For the purpose of improving the stability and the speed of the training and also affecting activation ReLU functions, we made use of batch normalization across each of the layers. Here is an outline of the model:

- First Block (Layers 1-2): This first block of the model takes an input frame or image with the size of (224, 224, 3) and feeds it through a series of steps. This block represents the initial block of the given model. The first Conv2D layer convolves the input with a kernel of 3 x 3 and produces 16 feature maps. The batch normalization technique then normalizes these feature maps to increase stability and be faster when training. Next, one more Conv2D layer is added for feature extraction to enable the model to learn better features.

Finally, we rescale the feature maps by using the MaxPooling2D layer, which has a pool size of (2, 2), decreasing the size of the feature maps by half but retaining only the most significant features. This layer takes an input of size (220, 220, 16) from the previous layer and yields an output of size (110, 110, 16).

- Second Block (Layers 3-4): is similarly has the same pattern as the previous block. The input shapes (110, 110, 16) are taken from the previous. The first Conve2D layer applies 64 filters of size 3 x 3 to the input images, using the ReLU activation function to extract the important features. The batch normalization technique then normalizes these feature maps. After that, we applied the second Conv2D layer to extract features further. In the end, we downsample the feature maps by employing a MaxPooling2D layer, which has a pool size of (2, 2); this layer takes an input of size (106, 106, 32) from the previous layer and produces an output of size (53, 53, 32).

- Third Block (Layers 5-7): is consists of 3 convolutional layers, which are encompassed with 2 batch normalization layers of the same size and a single Maxpooling2D layer at the final stage. This block processes an input frame or image with dimensions (53, 53, 32) by putting it through a series of procedures. The first Conv2D layer applies a 3x3 kernel to the input, generating 64 feature maps to extract the important features. The batch normalization technique then normalizes these feature maps. Then, the second Conv2D layer applies a 3x3 kernel to the input, generating 64 feature maps to extract the important features. The batch normalization technique then normalizes these feature maps. After that, we applied the third Conv2D layer to extract features further. In the end, we downsample the feature maps by employing a MaxPooling2D layer, which has a pool size of (2, 2); this layer takes an input of size (46, 46, 64) from the previous layer and produces an output of size (23, 23, 64).

- Fourth Block (Layers 8-10): It is similar to the pattern of the third block with the input shapes (23, 23, 64). The first Conve2D layer, second Conve2D layer, and third Conve2D layer are applied with 128 feature maps with dimension size 2x2. The batch normalization layer is performed only after the first and second convolutional layers. In the end, a MaxPooling2D layer with a pool size of (2, 2) reduces the feature maps, takes the input shape size (16, 16, 128), and produces an output shape (8, 8, 128).

- Fifth Block (Layers 11-13): It similarly has the same pattern as the fourth block with the input shapes (8, 8, 128). The first Conve2D layer, second Conve2D layer, and third Conve2D layer are applied with 256 feature maps with dimension size 2x2. The batch normalization layer is performed only after the first and second convolutional layers. In the end, a MaxPooling2D layer with a pool size of (2, 2) reduces the feature maps, takes the input shape size (2, 2, 256), and produces an output shape (1, 1, 256).

3. Classification Part: After incorporating convolutional groups, a flattening layer converts 3D feature maps to 1D. The flattening technique has to be applied in order to prepare the data to be fed into the remaining fully connected layers of the network. Then, a dense module with 256 nodes and a ReLU activation function is used to develop a deep layer to enhance and transform features extracted/selected from the input matrix. In order to avoid overfitting, L1 and L2 are used as a form of regularization of the Inside of this dense layer. Finally, in a bid to reduce cases of overfitting, we incorporate a dropout layer, which we set at 0. 5, which zeros out at random half of the input units during training. We then follow with a discriminator-like dense layer with 128 units and ReLU activation, which will aid in the extraction of even more features. Last is the output layer, which has two nodes that use the sigmoid activation function to enable binary classification; this is the method used to decide if input data is in one class.

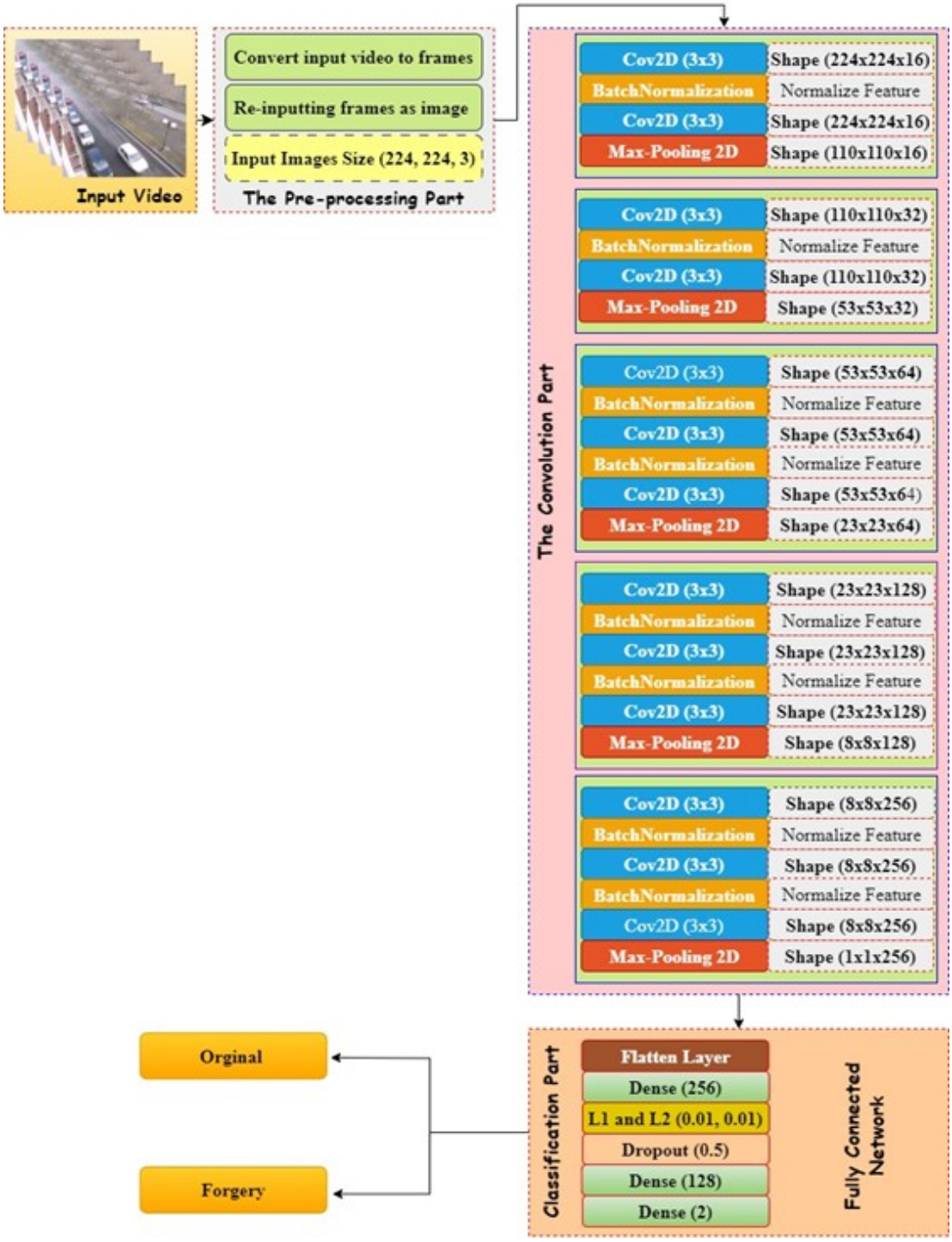

Figure 3. The structure of proposed model

For clarity, the pseudocode given in Algorithm 1 shows the sequential processes of the suggested model.

Algorithm 1: Algorithm of Copy-Move Forgery Detection System

$INPUT : DatasetD = \{(x_i, y_i)\}_{i=1}^{n} where x_i \in R^{H\times W\times C}, y_i \in \{0, 1\}$
$OUTPUT : TrainedmodelM, Performancemetrics\phi = \{Prec, Rec, F1, Acc\}$

1: Procedure COPY_MOVE_DETECTION
2: // Data Preprocessing Phase
3 : $D' \leftarrow \varnothing$
4 : $foreach(x, y) \in Ddo$
5 : $\mathrm{x}' \leftarrow \mathrm{RESIZE(x}, 224 \times 224 \times 3)$
6 : $x' \leftarrow \mathrm{NORMALIZE}\,(x', 1/255)$
7 : $ifx \in Dtrainthen$
8 : $\mathrm{x}' \leftarrow AUGMENT\,(\mathrm{x}', \{shear, zoom, flip))$
9: end if
10 : $D' \leftarrow D' \cup \{(x', y)\}$
11: end for
12: //Model Architecture Construction
13 : $\mathrm{M} \leftarrow SEQUENTIAL_MODEL()$
14 : for $i = 1$ to 5 do
15 : $\mathrm{M} \leftarrow \mathrm{M} + CONV2DO + BATCHNORMO + CONV2DO + MAXPOOLO$
16 : $ifi \geq 3then$
17 : $\mathrm{M} \leftarrow \mathrm{M} + DROPOUT(0.5)$
18: end if
19: end for
20 : $\mathrm{M} \leftarrow \mathrm{M} + FLATTEN()$
21 : $\mathrm{M} \leftarrow \mathrm{M} + \mathrm{DENSE}\,(256,\ \mathrm{L1\_L2\_}\ REG) + \mathrm{DROPOUT}(0.5)$
22 : $\mathrm{M} \leftarrow \mathrm{M} + \mathrm{DENSE}(128) + \mathrm{DROPOUT}(0.5)$
23 : $\mathrm{M} \leftarrow \mathrm{M} + DENSE(2, SIGMOID)$
24: // Model Compilation and Training
25 : $\mathrm{COMPILE}\,(\mathrm{M}\,, lossBCE, optimizer = \mathrm{ADAM}\left(\mathrm{r} = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999\right))$
26 : $\mathrm{M} \leftarrow TRAIN(\mathrm{M}, \mathrm{D}'train, \mathrm{D}'val, epochs = 30, batch_size = 32)$
27 : $//ModelEvalution$
28 : $\hat{\mathrm{y}} \leftarrow PREDICT(M, D'test)$
29 : $\Phi \leftarrow \mathrm{EVALUATE\_METRICS(ytest},\hat{y})$
30: return M, Φ
31: end procedure
32 : $procedureEVALUATE\_METRICS(y_{true}, y_{pred})$
33 : $CM \leftarrow CONFUSION\_METRIX(y_{true}, y_{pred})$
34: Prec $\leftarrow T_p/(T_p + F_p)$
35: Rec $\leftarrow T_p/(T_p + F_N)$
36 : $F1 \leftarrow 2 \times (Prec \times Rec)/(Prec + Rec)$
37: Acc$\leftarrow (T_p + T_N)/(T_p + T_N + F_p + T_N)$
38 : $returnPrec, Rec, F1, Acc$
39: end procedure

3. Experiments and Results

This section provides the results of our tampered video detection model, emphasizing its performance across multiple measures. The individual frames of video were analyzed to ensure comprehensive detection of alterations.

The assessment measures employed, along with detailed comparisons to other models, are explained below to demonstrate the effectiveness of our technique.

3.1. Experimental setup

In this study, we used the Google Colab environment to train a DCNN model to identify video frame forgeries. We used forged and authentic video frames that had been preprocessed to enhance performance and train the suggested model. Table 1 describes the model setup, dataset configuration, and training environment.

Table 1. Experimental Setup of The Proposed Model

Components	Configuration
Input Size	All frames resized to (224,224,3)
Platform	Google Colab with free GPU access (Limited)
Hardware	GPU: NVIDIA Tesla T4 (16 GB VRAM) CPU: 2-core Intel Xeon @ 2.30GHz RAM: 12 GB
Software	Python 3.10 and TensorFlow 2.13 Keras (via tf.keras) NumPy, Scikit-learn, Matplotlib, Seaborn, PyWavelets
Used Dataset	Divided into train 50%, Validation 25%, and Testing 25%
Data Augmentation	Zoom (0.2), horizontal flipping, shear (0.2), and rescaling (Applied to the training set in the model)
Model	Conv2D (×13,16→256 filters) Batch Normalization (×8), Max-pooling2D (×5) Dense (256, 128), Dropout (0.5), Output (sigmoid, 2) Regularization (L1=0.01 and L2=0.01)
Training	Up to 35 epochs only, Batch size of 32 Adam Optimizer (LR=0.001, b1=0.9, b2=0.999) Callbacks (EarlyStopping, ModelCheckpoint, CSVLogger)

3.2. Datasets

We tested our study's accuracy using three different benchmark datasets: the Video Tampering Dataset (VTD), the GRIP Forged Videos Dataset, and the Surrey University Library for Forensic Analysis (SULFA) Dataset, which are publicly available [5]–[8], [37], [39]–[41].

- VTD Dataset: The VTD is made up of 52 videos, which are synthesized with the purpose of containing copy-move forgery artifacts that are to be detected. It holds the related original videos (26) and the manipulated videos (26) on it also were shown. Thus, the provided dataset enables the assessment of forgery detection algorithms with special attention to the identification of videos with tampered frames and the localization of forged videos in relation to their authentic copies.
- GRIP Dataset: A large number of videos are available in the GRIP dataset containing ninety videos, which makes it really beneficial for video copy move forgery detection research. It includes 45 tampered videos, 45 corresponding original videos they used in the experiments besides. Forgery detection can be assessed with the help of the GRIP dataset, which allows to estimate how accurately manipulated videos are distinguished from the originals when using the corresponding systems.
- SULFA Dataset: In order to assess the classification performance, the University of Surrey has provided its realistic dataset termed as SULFA where some of the videos are genuine while others have been forged. The provided collection of 150 videos from three cameras is located in the link and is freely available on the website of the university. The videos are low quality, 320X240 pixels, 10 seconds in duration and they are at

a frame rate of 30 frames/second. Survey 1 comprises of 80 videos, and among these are manipulated videos and normal videos. It is important for assessing and comparing forgery detection algorithms notably in cases where fake videos have to be detected in respect to their real forms.

These datasets consist of a set of videos that includes a lot of variants of forgeries. It is seen from the figure that the frames that are used in datasets are similar to the ones presented in Figure 4. The website related to each data set contains more and related information. The most crucial characteristics of these datasets are described in Table 2.

Table 2. The VTD, GRIP, and SULFA dataset details

Datasets	Videos		No. of Frames					
			Training		Validation		Testing	
VTD	Tampered	Original	Tamp.	Orig.	Tamp.	Orig.	Tamp.	Orig.
	26	26	190	176	95	88	95	87
GRIP	Tampered	Original	Tamp.	Orig.	Tamp.	Orig.	Tamp.	Orig.
	45	45	166	152	84	76	83	76
SULFA	Tampered	Original	Tamp.	Orig.	Tamp.	Orig.	Tamp.	Orig.
	40	40	140	150	70	75	70	74

Figure 4. Samples of Some Frames of Used Datasets (VTD, GRIP, and SULFA)

3.3. Performance Metrics

Performance metrics are significant for evaluating the effectiveness and accuracy of deep learning algorithms and other machine learning models. They provide numerical measures that are useful in assessing how well a model applies its predictions to the resolve of a particular problem. According to [30-33], [37] metrics for performance are determined based on the problem's characteristics, the data's type, and the expected outcome.

- Accuracy: The accuracy metric checks how accurate the model is by finding the percentage of correct predictions, both positive (TP) and negative (TN), out of all the predictions. The formula for this is given in terms of %, which can be found in Eq. (5).

$$Accuracy = \frac{\mathrm{T_P} + \mathrm{T_N}}{\mathrm{T_P} + \mathrm{F_P} + \mathrm{T_N} + \mathrm{F_N}} \times 100\% \tag{5}$$

- Matthews Correlation Coefficient (MCC): In binary classification problems, the MCC is a quantitative metric that ranges from -1 to 1, evaluating true positives, true negatives, false positives, and false negatives, with 1 indicating perfect predictions, 0 indicating randomness, and -1 indicating full mismatch, represented as a percentage in Eq. (6).

$$\mathrm{MCC} = \frac{\mathrm{T_P} \times \mathrm{T_N} - \mathrm{F_P} \times \mathrm{F_N}}{\sqrt{(\mathrm{T_P} + \mathrm{F_P})\,(\mathrm{T_P} + \mathrm{F_N})\,(\mathrm{T_N} + \mathrm{F_P})\,(\mathrm{T_N} + \mathrm{F_N})}} \times 100\% \tag{6}$$

- Precision: Precision is the percentage of model predictions that are correct, calculated using Eq. (7) in terms of %, focusing on true positives compared to total positive predictions [28].

$$Precision = \frac{T_P}{T_P + F_P} \times 100\% \tag{7}$$

- Recall: Recall, sometimes referred to as sensitivity or true positive rate, evaluates the proportion of accurate predictions obtained among all the actual positive samples. It estimates the model's ability to accurately detect any relevant instances. In terms of percentage, Eq. (8) expresses the recall equation:

$$Recall = \frac{T_P}{T_P + F_N} \times 100\% \tag{8}$$

- F1-score: The F-score, or F-measure, is a predictive performance measure derived from the test's precision and recall. The F1-score can be calculated using either Eq. (9) or Eq. (10) expressed as a percentage.

$$\text{F1\_score} = \frac{2 \times (\text{ Precision } \times \text{ Recall })}{\text{Precision } + \text{ Recall}} \tag{9}$$

$$\text{F1\_score} = \frac{2 \times \mathrm{T_P}}{2 \times \mathrm{T_P} + \mathrm{F_P} + \mathrm{F_N}} \times 100\% \tag{10}$$

In that context, the number of true positives is noted by T_P, where the number of true negatives is T_N while, F_P stands for the number of false positives and the number of false negatives is written as F_N. Measures of efficiency are crucial components to any models of machine learning as they determine the strengths and weaknesses of the generated predictive algorithms. They assist in model selection and the enhancement and optimization to obtain superior results in real-life situations. To read the situation, it is necessary that the right measures are chosen depending on the requirements and nature of the context [28].

3.4. Empirical Results over the GRIP, VTD, and SULFA Datasets

As discussed above, we evaluated the suggested approach on three benchmark datasets: the GRIP, VTD, and SULFA datasets. As mentioned earlier, the confusion matrices shown in Figure 5 can be used to examine the performance evaluation results. Table 3 summarizes the results obtained from the confusion matrices for the generated data resources, namely the GRIP dataset, the VTD dataset, and the SULFA dataset.

From the results highlighted in Table 4 below, the proposed technique provided better results on the GRIP, VTD, and SULFA. The accuracy was impressive and, in some cases, even reached 100%, though in other cases, it was still high at 95%. The way it differentiates between modified and original videos can easily be understood with the help of the presented model. The MCC also remained in the 91 to 100 percent range, which exhibited the usefulness of the model's classification. Moreover, on the SULFA dataset, the defined model obtained an accuracy level of up to 100, which shows potential for achieving targeted manipulation. The recall scores varied between 0.95 and 1, which proves that the model worked exceptionally well and indicated manipulative conduct with high levels of accuracy.

Moreover, it set up high F1 scores varying from 95.81% to 100%, demonstrating that the algorithm can detect object-based manipulation more effectively. The model's efficiency is illustrated by the consumption analysis of a model of approximately 20.22-25.56 seconds. Therefore, the published results demonstrate the ability of the suggested approach to detect object-based tampering within movies, which can potentially be useful in several fields, ranging from video analysis to security.

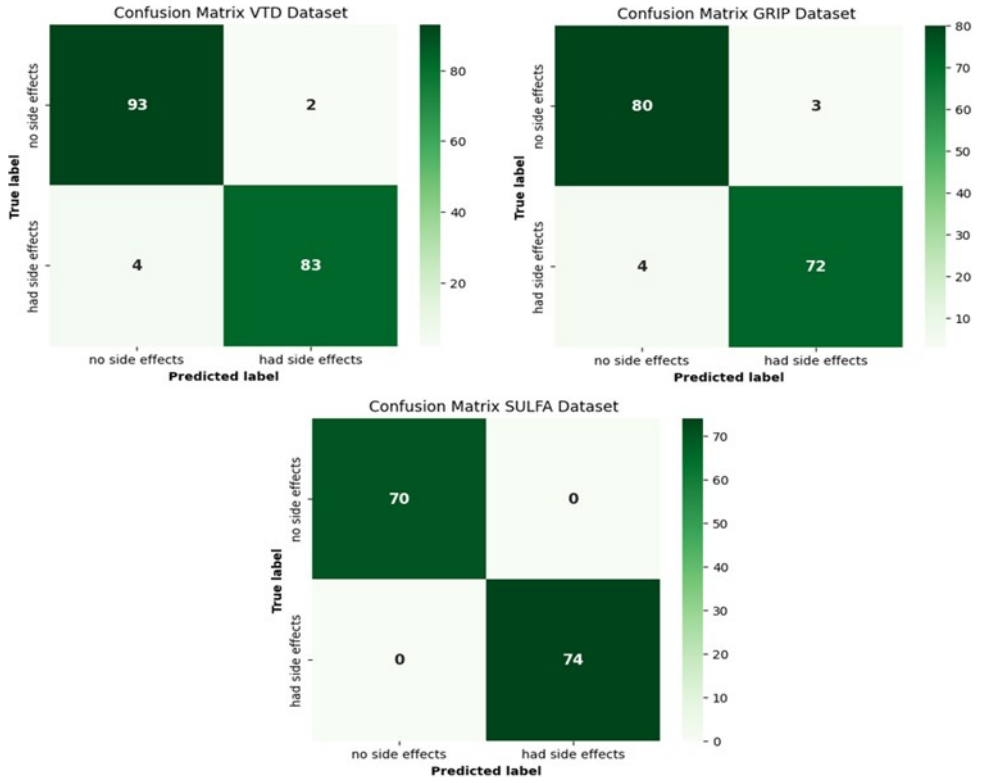

Figure 5. The Confusion Matrix Output for GRIP, VTD, and SULFA Datasets

Table 3. Confusion Matrices of the Proposed Model Concerning GRIP, VTD, and SULFA

Dataset	Classes	+	-	Total
GRIP	+	80	3	83
	-	4	72	76
	Total	84	75	159
VTD	+	93	2	95
	-	4	83	87
	Total	97	85	182
SULFA	+	70	0	70
	-	0	74	74
	Total	70	74	144

Table 4. Results of the proposed model concerning GRIP, VTD, and SULFA datasets.

Metrics	GRIP	VTD	SULFA
Accuracy (%)	95.60	96.70	100
MCC (%)	91.18	93.41	100
Precision (%)	96.39	97.89	100
Recall (%)	95.24	95.88	100
F1 Score (%)	95.81	96.88	100
Time (seconds)	25.10	27.35	20.22

Thus, to compare our results, we have computed MCC, recall, precision, and F1-Score for our proposed model while comparing with other previously released models with respect to the GRIP dataset, as stated in Table 5. In particular, for the GRIP dataset, the proposed approach appeared to be significantly better, with F1-score, MCC, precision, and recall equal to 95.81%, 91.18%, 96.39%, and 95.24%, respectively. It produced a higher result than [3, 4, 25, 26, 27, 34] by 74.60%, 85%, 7.60%, 12.50%, 85%, and 44.20% in terms of F1-Score, respectively. Recently, Eltoukhy et al. [37], and Oliaei & Azghani [5] got slightly better F1 scores by 8737%, and 89%, respectively, compared to the approach of others mentioned above, while H. Alfraihi et al. [36] portrayed a superior F1 score value of 95.26%. Additionally, when looking at the MCC evaluation, our results were better than the previous studies in [3, 25, 26, 34, 36, 37], which had scores of 72.60%, 2.20%, 15.70%, 37.10%, and 72.90% respectively, while we achieved a score of 91.18%.

Table 5. A Comparative Analysis of Recall, Precision, F1-score, and MCC on the GRIP dataset.

Methods	Recall %	Precision %	F1-score %	MCC %
Li, H., et al. [25]	-	-	7.60	2.50
Bappy, J. H., et al. [26]	-	-	12.50	15.70
Jin, X., et al. [34]	-	-	44.20	37.10
D'Amiano, L., et al. [3]	-	-	74.60	72.60
Zhong, J. L. et al. [4]	-	-	85	-
Zhong, et al. [27]	-	-	85	-
Eltoukhy, M.M., et al. [37]	-	-	87.7	72.9
Oliaei and Azghani [5]	-	-	89	-
H. Alfraihi, et al. [36]	95.26	95.17	95.13	90.43
Proposed Method (GRIP dataset)	95.24	96.39	95.81	91.18

In Table 6, using the VTD dataset, it can be observed that the proposed method used in the detection of copy-move forgery realized a high F1-score of 96.88%, with an MCC of 93.41%, while the recall noted was at 95.88%. The previous approaches [3, 25, 26, 34] have proved to be worse in performance. These methods give F1-score values of 1.20% to 26.90% and MCC values of 1.70% to 26.90%. More recently, Eltoukhy et al. [37] obtained significantly better F1 and MCC scores of 83.4% and 63.8%, respectively, compared to the above-mentioned approaches. H. Alfraihi et al. [36] attained the highest F1-score, achieving a score of 92.83%, and the highest MCC, achieving a score of 86.16%, in contrast to the others. Thus, the specified data may confirm the applicability and effectiveness of the proposed technique. If applied to the VTD dataset, algorithms of that kind provide a significantly improved method for forgery detection compared to prior methods, as they allow for excellent balance between precision and recall.

Table 6. A Comparative Analysis of Recall, Precision, F1-score, and MCC over the VTD Dataset.

Methods	Recall %	Precision %	F1-score %	MCC %
Bappy, J. H., et al. [26]	-	-	1.20	3.00
Li, H., et al. [25]	-	-	2.50	26.90
D'Amiano, L., et al. [3]	-	-	2.90	1.70
Jin, X., et al. [34]	-	-	1.90	2.00
Eltoukhy, M.M., et al. [37]	-	-	83.4	63.8
H. Alfraihi, et al. [36]	92.67	93.50	92.83	86.16
Proposed Method (VTD dataset)	95.88	97.89	96.88	93.41

Table 7. A comparative Analysis of Recall, Precision, F1-score, and MCC over the SULFA Dataset.

Methods	Recall %	Precision %	F1-score %	MCC %
D'Amiano, L., et al. [3]	-	-	2.50	3.70
Bappy, J. H., et al. [26]	-	-	3.50	3.50
Li, H., et al. [25]	-	-	5.40	3.10
Jin, X., et al. [34]	-	-	8.80	4.50
Zhong, J. L. et al. [4]	-	-	84	-
Zhong et al. [27]	-	-	85	-
Oliaei and Azghani [5]	-	-	89	-
Eltoukhy, M.M., et al. [37]	-	-	86.9	74
Proposed Method (SULFA dataset)	100	100	100	100

Table 7 provides a head-to-head comparison of the methods discussed in this work for copy-move forgery detection on the SULFA dataset. Comparing all the methods, it is evident that there are major differences in the performance indicators. The proposed method is also noteworthy because all the estimated indicators of recall, precision, F1-score, and MCC are equal to 100%, which means that the method works without errors and does not produce false positives. On the other hand, other methods proposed by [9, 10, 11] yield significantly lesser F1 scores, which have an average value of 2. 50% to 5%. Based on these outcomes, it can be noted that these strategies do not possess satisfaction in terms of identifying forged cases effectively and efficiently. Jin et al. [34] proved to show improvement with the F1-score increased to 8.80% and MCC of 4.50% of them did not even achieve the level of the suggested approach. In the extraction, the two- works of Zhong et al. [4, 27] reported more accurate results with high F1 scores, around 85 and 84%, respectively. At the same time, Oliaei and Azghani [5] recently obtained a higher F1 score of about 89% than other approaches. Additionally, Eltoukhy et al. [37] obtained significantly better F1 and MCC scores of 86.9% and 74%, respectively. Thus, based on the result derived in the proposed work presented here to detect counterfeit products in the SULFA dataset, it is evident that the suggested method provides better accuracy and reliability than the existing method, establishing a new benchmark in this field.

To evaluate the acquired results in combination with other methods of estimating time and accuracy. The results of our suggested technique are compared to those of recent works [3, 4, 25, 26, 27, 34, 36, 37] in Tables 8, 9, and 10. It is observed that our method outperforms the others in terms of both accuracy and computational time. Table 8 shows that the proposed model performs exceptionally well on the GRIP dataset. With a processing time of 25.10 seconds, it achieves an accuracy of 95.60%, outperforming [3, 26, 34], which reported accuracies of 73.60%, 43.20%, and 29.40%, respectively, with processing times of 21.30, 45.80, and 42.40 seconds. Meanwhile, the [25] model achieved an acceptable accuracy with a score of 74.90% but with a higher processing time of 69.60 seconds. In contrast, [4] achieved a superior accuracy of 83.33% without reporting the processing time required. In comparison to the others, H. Alfraihi et al. [36] and Eltoukhy et al. [37] achieved the highest accuracy, with scores of 95.26% and 86.16%, respectively. Additionally, Eltoukhy et al. [37] achieved the lowest processing time of 9.60 seconds in the testing phase.

In addition, the results of our proposed method and other related methods [3, 4, 25, 26, 27, 34, 36, 37] for the second dataset, the VTD dataset, are shown in Table 9. The proposed model achieved outstanding accuracy, with an accuracy of 96.70% and a processing time of only 27.35 seconds. On the other hand, [3] and [26] demonstrated abysmal accuracy levels, specifically 19.00% and 15.60%, respectively, along with higher processing times of 75.80 seconds and 68.70 seconds. The accuracy of [34] increases to 64.30% with a processing time of 46.60 seconds, while [25] obtains an accuracy of 29.00% with a maximum processing time of 91.30 seconds. Performance metrics are not reported in Zhong et al. [4]. In contrast, Alfraihi et al. [36] and Eltoukhy et al. [37] achieved the highest accuracy, recording 92.67% and 81.87%, respectively. In contrast, Alfraihi et al. [36] and Eltoukhy et al. [37] achieved the highest accuracy, recording 92.67% and 81.87%, respectively. Eltoukhy et al. [37] also achieved the lowest processing time, 13.7 seconds, in the testing phase.

Finally, to compare the proposed model with previous works on the SULFA dataset and evaluate its performance using commonly used metrics, Table 10 shows that the proposed model outperforms all the previous methods aimed at the dataset in achieving ideal values that are equal to 100% for all the metrics demonstrated in the table,

which implies that the proposed method is capable of detecting all objects in the dataset and does not result in any false detection. At the same time, the accuracy scores of [3, 25, 26] collapsed to just 24.20%, 52.60%, and 33.70%, respectively, with different processing times. On the other hand, [38], [34], [37], [4], and [12] achieved superior performance with an accuracy of approximately 89.70%, 83.30%, 85.42%, 88.90%, and 92%, respectively, and [37] achieved the lowest processing time of 9.60 seconds in the testing phase. Moreover, the SULFA dataset's excellent performance of the suggested method demonstrates its higher accuracy and reliability compared to prior methods. The accuracy results shown in Tables 7, 8, and 9 are illustrated schematically in Figure 6 below.

Table 8. Performance Comparison of Various Methods on the GRIP Dataset.

Methods	Accuracy %	Time (second)
Bappy, J. H., et al. [26]	29.40	45.80
Jin, X., et al. [34]	43.20	42.40
D'Amiano, L., et al. [3]	73.60	21.30
Li, H., et al. [25]	74.90	69.60
Zhong, J. L. et al. [4]	83.33	-
H. Alfraihi, et al. [36]	95.26	13.15
Eltoukhy, M.M., et al. [37]	86.16	11.4
Proposed method (GRIP dataset)	95.60	25.10

Table 9. Performance Comparison of Various Methods on the VTD Dataset

Methods	Accuracy %	Time (second)
Bappy, J. H., et al. [26]	15.60	68.70
D'Amiano, L., et al. [3]	19.00	75.80
Li, H., et al. [25]	29.00	91.30
Jin, X., et al. [34]	64.30	46.60
Zhong, J. L. et al. [4]	-	-
H. Alfraihi, et al. [36]	92.67	10.2
Eltoukhy, M.M., et al. [37]	81.87	13.7
Proposed method (GRIP dataset)	96.70	27.35

Table 10. Performance Comparison of Various Methods on the SULFA Dataset

Methods	Accuracy %	Time (second)
D'Amiano, L., et al. [3]	24.20	44.20
Bappy, J. H., et al. [26]	33.70	62.20
Li, H., et al. [25]	52.60	75.10
Jin, X., et al. [34]	83.30	40.4
Zhong, J. L. et al. [4]	88.90	-
Kumar, V., et al. [12]	92.00	-
Eltoukhy, M.M., et al. [37]	85.42	9.6
Pandey, R, et al. [38]	89.7	-
Proposed method (SULFA dataset)	100	20.22

3.5. Cross Dataset Evaluation

The cross-dataset experiments' results indicate that the proposed model has a generalization capability across different datasets. When training on dataset SULFA and testing on GRIP, the accuracy reached 92.14%, which is

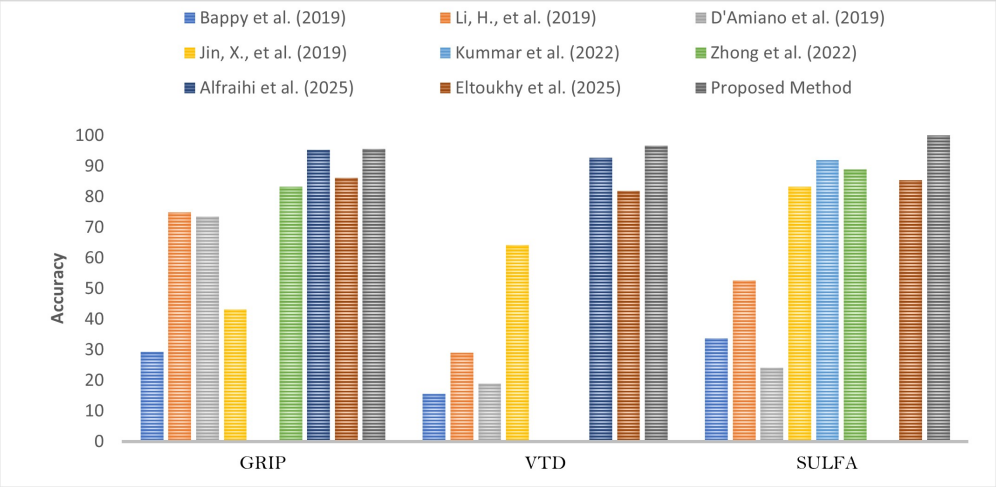

Figure 6. Performance Comparison of Various Methods on the GRIP, VTD, and SULFA Datasets in terms of Accuracy

Table 11. Performance Evaluation on Cross-Dataset in Different Scenarios

Cross-Dataset		2*Recall %	2*Precision %	2*F1-score %	2*Accuracy %
Train	Test				
SULFA	GRIP	94.89	90.54	92.67	92.14
SULFA	VTD	92.46	100	96.46	95.76
GRIP	SULFA	92.26	97.86	94.97	94.99
GRIP+VTD	SULFA	94.79	97.50	96.13	96.20

slightly lower than GRIP's original accuracy 95.6%. This decrease can be attributed to the distribution shift between the datasets, such as differences in resolution, compression artifacts, and visual characteristics of the forgeries. Also, training on SULFA and testing on VTD yielded an accuracy of 95.76%, while VTD's original accuracy was 96.7%. This decrease in accuracy is predicted, given the differences in scene content and manipulation traces among datasets.

Likewise, the accuracy reached 95% while training on the dataset GRIP and testing on the dataset SULFA. This study indicates that datasets GRIP and SULFA share more features than dataset VTD, which promotes enhanced cross-dataset performance.

Additionally, with an accuracy of almost 96%, the best cross-dataset performance is obtained when testing on SULFA and training on a combination of GRIP and VTD. This enhancement results from the training data's greater diversity, which reduces overfitting to dataset-specific patterns and enables the model to learn an additional representation of forgery artifacts.

Table 11 presents a detailed summary of the results for Recall, Precision, and F1 score throughout the cross-dataset, and Figure 7 illustrates the confusion matrices corresponding to each cross-dataset experiment detailed in Table 11.

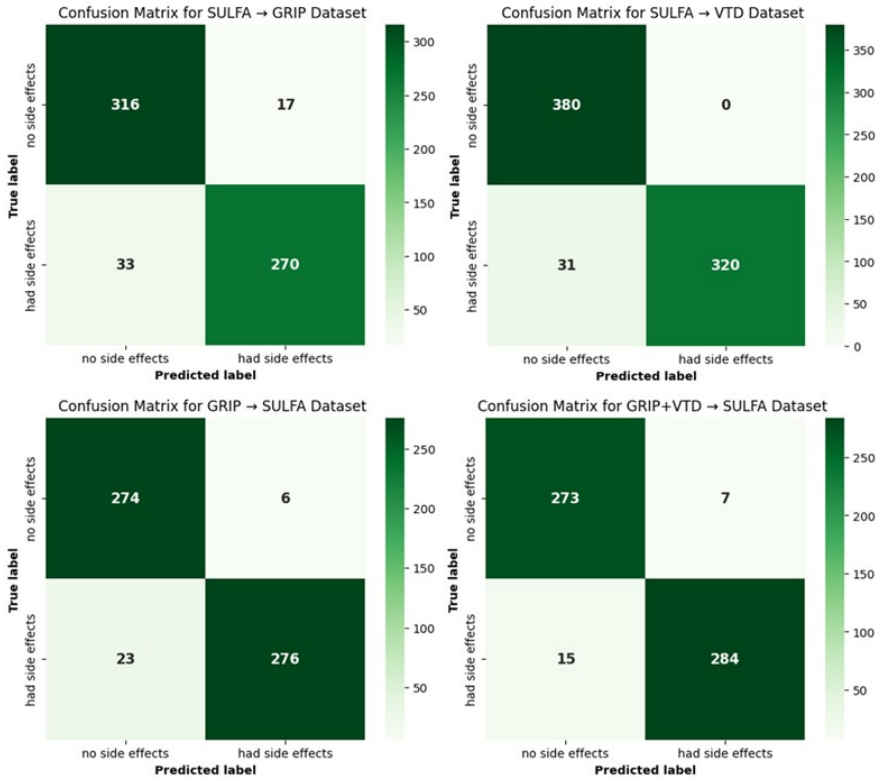

Figure 7. Confusion Matrix for Cross-Dataset in Different Scenarios

4. Discussion

The proposed technique outperformed on SULFA, GRIP, and VTD, with impressive accuracy levels of up to 100%. It differentiated between modified and original videos, with MCC ranging from 91 to 100%. The model achieved up to 100% accuracy on the SULFA dataset, with recall scores ranging from 0.95 to 1. It also received high F1 scores, demonstrating its effectiveness in detecting object-based manipulation.

We have compared the proposed model for detecting counterfeit products in the SULFA dataset with other models on the GRIP and VTD datasets. The results indicate that the proposed approach surpasses the previous methods in terms of recall, precision, F1-Score, and MCC. Compared to the previous methods, the proposed approach yielded superior results for the GRIP dataset by 95.81%, 91.18%, 96.39%, and 95.24%, respectively. The VTD dataset had a high F1 score of 96.68%, an MCC of 93.41%, and a recall of 95.88%. The proposed approach surpasses the previously mentioned methods in terms of computing time and accuracy. On the GRIP dataset, the proposed model achieved an accuracy of 95.60%, surpassing the previous methods by 73.60%, 43.20%, and 29.40%. In comparison to the VTD dataset, the proposed model was very accurate (96.7%) and only took 27.35 seconds to process. In comparison to previous works on the SULFA dataset, the proposed model outperforms all the previous methods in achieving ideal values equal to 100% for all metrics demonstrated in Table 7. This implies that the proposed method is capable of detecting all objects in the dataset without any false detection. However, the accuracy scores of the previous methods collapsed with just a value of 24.20%, 52.60%, and 33.70%, respectively.

4.1. Ablation Study on Hyperparameters using SULFA Dataset

To justify the design choice of our deep CNN architecture, we conducted an ablation study by varying key hyperparameters, including the number of convolutional layers, the learning rate, and the kernel size. Table 12

summarizes the effect of these parameters on the model's classification performance using the SULFA dataset, and Appendix B has the confusion matrices for all configuration patterns.

Table 12. Ablation Study with Different Configuration Patterns

Different Configuration Patterns	TPR	FPR	FNR	TNR	Param.	FLOPs (x10^6)	Accuracy (%)
	Based on best weights (%)						
8 Conv Layers, LR= **0.001**, Kernel size 3×3 [36]	100	0	27.03	72.97	360K∼	1,053	86.11
10 Conv Layers, LR= **0.001**, Kernel size 3×3	97.14	2.86	5.41	94.59	546K∼	1,128	95.83
13 Conv Layers, LR= **0.01**, Kernel size 3×3	95.71	4.29	6.76	93.24	2M∼	1,321	94.44
12 Conv Layers, LR= **0.001**, Kernel size 5×5	98.57	1.43	5.71	94.59	1.4M∼	1,211	96.53
13 Conv Layers, LR= **0.001**, Kernel size 3×3 (Ours)	100	0	0	100	2M∼	1,325	100

Based on these results, the optimal configuration (13-conv layer depth, 3×3 kernel size, and Adam optimizer with a learning rate of 0.001) satisfies both the objectives of model correctness and stability. However, performance dropped when the network depth was reduced or the kernel size was increased, possibly because of a lack of ability to extract features or because local spatial details were lost. Similarly, higher or lower learning rates resulted in suboptimal convergence behavior. Furthermore, we compared our architecture with a simpler, shallower CNN model comprising only 8 convolutional layers with Global Average Pooling instead of a Flatten layer (with fewer parameters) [37], as shown in Table 10. The simplified model achieved an accuracy of only 85.42, which supports the necessity of the proposed architecture's depth and complexity.

Furthermore, the floating-point operations (FLOPs) escalated from 1053×10^6 in the baseline model to 1325×10^6 in the final model, indicating the increased complexity for more features extracted. Memory consumption increased, with the most complex model requiring 2180 MB (approximately) compared to the baseline's 1388 MB (approximately). The suggested DCNN model, which exhibits a fair trade-off between complexity and detection performance, requires roughly 1.32×10^9 FLOPs per forward pass, which is greater than the other configuration models but has the best accuracy.

4.2. Computational and Efficiency Analysis

We examined the computational resource requirements using various metrics to assess the model's accuracy and speed using the SULFA dataset. These included the following: number of parameters, floating-point operations per second (FLOPS), memory consumption, inference time per frame, and classification accuracy. Further, we tested the proposed model in both CPU and GPU execution settings, comparing it to a lightweight baseline model (MobileNet-v2).

As shown in Table 13, our model maintains a moderate parameter count of 2M, lower than the lightweight model's (2.4M). For deeper feature extraction, the FLOPs of our proposed model (1322×10^6) are larger than those of MobileNet-v2 (600×10^6), and the significant improvement in categorization accuracy, from 54.17% to 100%, justifies this increase.

Regarding resource usage, our model requires 2504 MB on CPU and 109 MB on GPU, which remains feasible for modern hardware. The inference time remains competitive, with 150 ms/frame on CPU and 62.84 ms/frame on GPU. This significantly improves performance while showing a little delay compared to the lightweight model (113.76 ms/frame CPU, 61.72 ms/frame GPU).

The results indicate that while the lightweight Model is more resource-efficient, it suffers a noteworthy drop in accuracy. In contrast, our proposed Model achieves a +45.83% accuracy improvement with only a moderate

increase in computational cost, making it a practical choice for more applications where detection reliability is critical. Consequently, the suggested model provides a good trade-off between detection accuracy and computing cost, which qualifies it for real-world implementation.

Table 13. Comparative analysis with lightweight MobilNet-v2.

Model	Usage Interface	Parameters	FLOPs ($\times 10^6$)	Memory Used (MB)	Inference Time (ms/frame)	Accuracy (%)
MobileNet-v2	CPU	2.4M$\sim$	600$\sim$	1347.95	113.76ms	54.17
	GPU	2.4M$\sim$	600$\sim$	12.94	61.72ms	54.17
Proposed	CPU	2M$\sim$	1,322$\sim$	2504.38	150ms	100
	GPU	2M$\sim$	1,322$\sim$	109.23	62.84 ms	100

4.3. Restrictions and Ethical Issues

Although the proposed model demonstrates remarkable efficacy in forgery detection, we underscore the importance of using it only as an ancillary instrument, especially in evaluating forensic evidence and related matters. Even with its accuracy, this method can identify original content as counterfeit or fail to detect forgery, leading to inaccurate results. This may result in erroneous allegations and spurious and misleading evidence fabrication.

Further, the model may be subjected to adversarial attempts to avoid detection. To improve flexibility and generalization, data augmentation and adversarial training on a huge dataset will be the primary approaches used in future research, alongside significant development in this model. By combining the wavelet transform with DCNN, the proposed framework can enhance the detection of forgery even in the presence of attacks such as Geometric attacks (rotation, scaling, etc.) and Additive noise attacks (Gaussian, etc.).

We aim to promote the responsible and trustworthy use of AI-based video forgery detection systems by addressing these limitations and ethical concerns.

5. Conclusion

This paper proposes a new architecture of deep convolutional neural network (DCNN) for copy-move video forgery detection with our assurance of high classification accuracy and the least possible processing time. A crucial component of this model is batch normalization along with L1 and L2 regularization, which may be used to initialize a layer output, accelerate training, boost the learning rate, significantly improve accuracy, reduce overfitting, and improve operation stability. According to the compared approaches [3, 4, 5, 12, 25, 26, 27, 34, 36, 37, 38], the proposed model achieved superior and exceptional results, demonstrating a high level of accuracy and an F1-score of 100% in both of the cases. In addition, with regard to the various methodologies, the MCC achieved a perfect score of 100%, potentially surpassing the other approaches. Hence, while using and examining our recommended SULFA dataset, the proposed approach is highly effective and proficient in detecting object-oriented manipulation. At the same time, in the experiment carried out over the VTD dataset, our suggested approach yielded superior performance to every other approach. It achieved 96.70% accuracy, as indicated by the F1 score of 96.88, along with an MCC of 93.41%. In conclusion, the results above prove that the method we pointed out is more accurate regarding accuracy, F1-Score, and MCC than the other methods. Again, higher accuracy levels have supported the proposed strategy over all the techniques used on the GRIP dataset. As such, it got an accuracy of 95.60%, an F1-Score of 95.81%, and an MCC of 91. 18%.

Furthermore, our method achieved accurate performances in all the datasets and yielded the shortest average testing time of 21-28 seconds. Hence, we provide a textual description of our suggested method that has been established to be effective, therefore concentrating more on the SULFA, VTD, and GRIP data sets that support our algorithm's viability in video analysis and security.

Future work will focus on detecting forgery in an extensive dataset by using a hybrid model that combines the proposed model with the wavelet transform to enhance the extraction of features from input images or videos.

Addressing this complex strategy would improve the effectiveness of forgery detection methods to handle the intricacies of collective attacks. By combining the wavelet transform with DCNN, the proposed framework can enhance the detection of forgery even in the presence of attacks such as Geometric attacks (rotation, scaling, etc.) and Additive noise attacks (Gaussian, etc.).

Appendix A: Mathematical Modeling in Batch Normalization

A.1 Mathematical Background and Practical Implication

Batch Normalization (BN) is a regularization and optimization technique that addresses the problem of internal covariate shift - i.e., the change in the distribution of inputs to each layer during training. BN stabilizes learning and enables the use of higher learning rates by normalizing the output of each layer to have zero mean and unit variance.

Given an input feature map for a mini-batch $X = x_1, x_2, x_3, \ldots, x_n$, BN transforms it using the following step
- Step 1: compute batch statistics (Mean and Variance)

$$\begin{array}{ll} \text{Batch Mean} & \mu_B = \frac{1}{n}\sum_{j=1}^{n} x_j \\ \text{Batch Variance} & \sigma_B^2 = \frac{1}{n}\sum_{j=1}^{n}\left(x_j - \mu_B\right)^2 \end{array}$$

- Step 2: Normalize the inputs (Normalize the Features)

$$\widehat{x}_j = \frac{(x_j - \mu_B)}{\sqrt{\sigma_B^2 + c}}$$

we use a small constant value ε (e.g., 10^{-5}) to prevent division by zero
- Step 3: Apply Learnable Scale and Shift Parameters (α and β)

$$y_i = \alpha\widehat{x}_J + \beta$$

where α scale and β shift are trainable parameters learned during training the model. These parameters allow the model to retain representation flexibility after normalization, enabling it to adjust the output distribution if needed.

A.2 Numerical Example of Batch Normalization

To explain how Batch Normalization (BN) works behind a convolutional layer, suppose the following simplified scenario:

Suppose that after applying a Conv2D layer with a single filter on a small grayscale image, the resulting feature map (tiny feature map) is:

$$X = \begin{bmatrix} 12 & 13 & 4 \\ 10 & 9 & 16 \\ 14 & 5 & 7 \end{bmatrix}$$

Apply Batch Normalization
- Compute the mean:

$$\mu = \frac{1}{n}\sum_{j=1}^{n} x_j = \frac{12+13+4+10+9+16+14+5+7}{9} = 10$$

- Compute the variance:

$$\sigma^2 = \frac{1}{n}\sum_{j=1}^{n}(x_j-\mu)^2 = \frac{1}{9}\left[\begin{array}{c}(12-10)^2+(13-10)^2+(4-10)^2+(10-10)^2+(9-10)^2\\ +(16-10)^2+(14-10)^2+(5-10)^2+(7-10)^2\end{array}\right]$$

$$\sigma^2 = \frac{1}{n}\sum_{j=1}^{n}(x_j-\mu)^2 = \frac{1}{9}[4+9+36+0+1+36+16+25+9] = 15.11$$

- Normalize the values: Assume that the constant $\epsilon = 10^{-5}$

$$\hat{X} = \frac{(x_{i,j}-10)}{\sqrt{15.11+0.00001}} = \frac{(x_{i,j}-10)}{3.887}$$

$$\widehat{X} = \begin{bmatrix} \frac{12-10}{3.887} & \frac{13-10}{3.887} & \frac{4-10}{3.887} \\ \frac{10-10}{3.887} & \frac{9-10}{3.887} & \frac{16-10}{3.887} \\ \frac{14-10}{3.887} & \frac{5-10}{3.887} & \frac{7-10}{3.887} \end{bmatrix} = \begin{bmatrix} 0.515 & 0.772 & -1.545 \\ 0 & -0.257 & 1.545 \\ 1.029 & -1.286 & -0.772 \end{bmatrix} = \text{Output after BN}$$

We note that Batch Normalization centers and scales the feature map values. As a result, the values become normally distributed with a mean of 0 and a standard deviation of 1, which leads to faster convergence and more stable deep learning training.

- After that, apply learnable parameters α and β : the model learns the optimal value of the two parameters α and β during the training. For illustration, we assume that $\alpha = 2$, and $\beta = 1$

$$y_i = \alpha\hat{x_j} + \beta = 2\hat{x_j} + 1$$

$$y_i = \begin{bmatrix} 2.03 & 2.544 & -2.09 \\ 1 & 0.486 & 4.09 \\ 3.058 & -1.572 & -0.544 \end{bmatrix} = \text{Final output}$$

Batch Normalization's learnable parameters, α (scaling) and β (shift), enable the network to modify the normalized output, preserving model expressiveness while providing training stability. This allows the network to once again represent complex feature distributions.

A.3 Feature Maps Before and After Batch Normalization

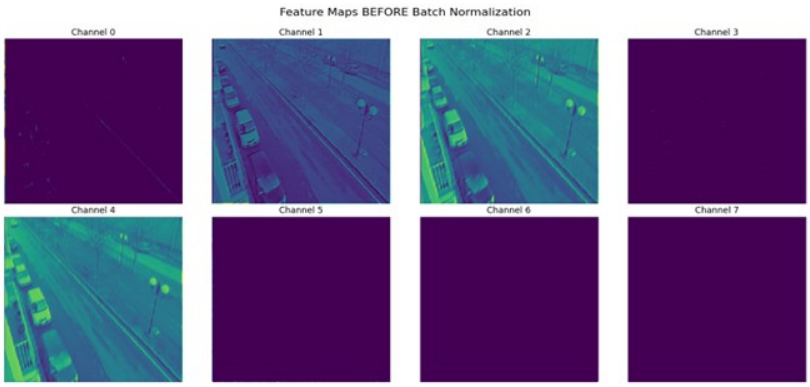

Figure 8. Feature Maps before Batch-Normalization from first Conv-2D Layer

To demonstrate the impact of Batch Normalization on intermediate features, we selected a representative frame from our dataset and showed its feature maps subsequent to the initial convolutional layer and the ensuing Batch Normalization layer.

- Before applying Batch Normalization (BN): feature maps may display inconsistent brightness, noise, or imbalanced activation values.

- After applying Batch Normalization (BN): feature maps demonstrate improved consistency, reduced noise, and more pronounced patterns.

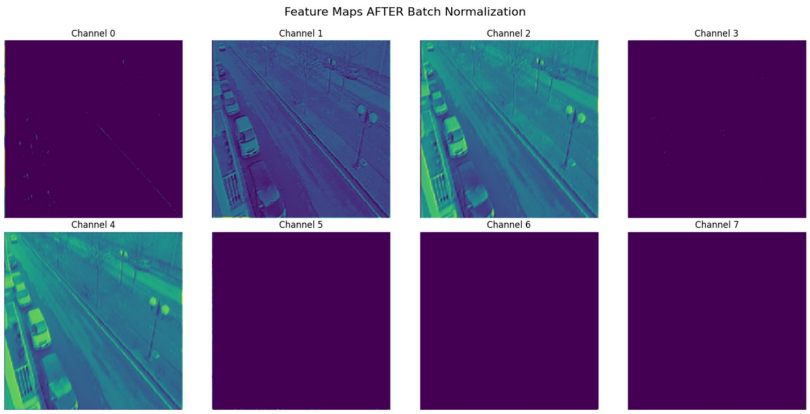

Figure 9. Feature Maps after Batch-Normalization from first Conv-2D Layer

Figures 8 and 9 display the obtained feature map for the first 8 channels (out of 16) of the first layer of the proposed framework, both before and after Batch Normalization.

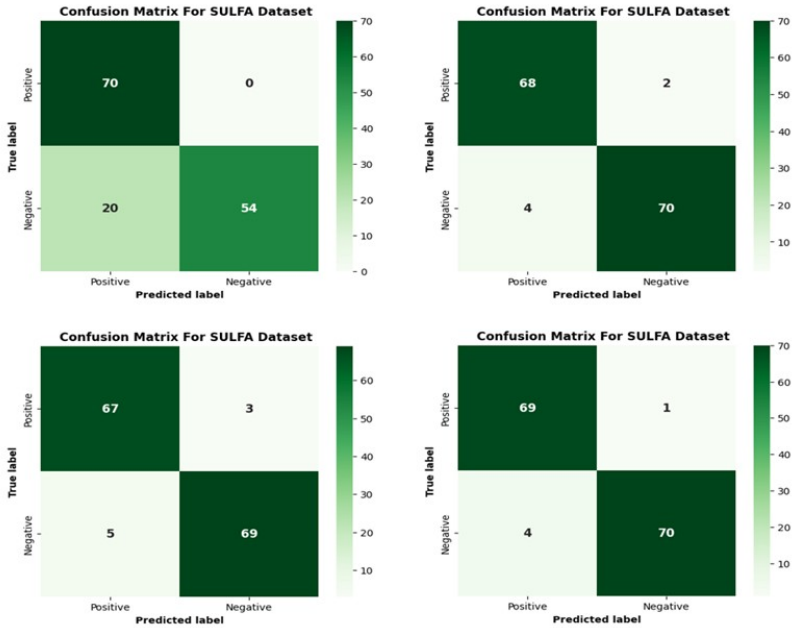

Figure 10. Confusion Matrix for SULFA dataset in Different Configuration Patterns with varying kernel size, learning rate, and conv layers

Appendix B: Confusion Matrices of Different Configuration Patterns

To further illustrate the performance of each setting, we have added Confusion Matrices for all configurations in this part, showing the classification behaviour in detail, as shown below in Figure 10.

REFERENCES

1. D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, et al., "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," Electronics, vol. 10, no. 20, p. 2470, 2021, doi: 10.3390/electronics10202470.
2. M. A. Alsmirat, R. A. Al-Hussien, W.A.T Al-Sarayrah, Y. Jararweh, and M. Etier, "Digital video forensics: A comprehensive survey. International Journal of Advanced Intelligence Paradigms, vol. 15, no. 4, p. 437-456, 2020, doi: 10.1504/IJAIP.2020.106040.
3. L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, "A PatchMatch-Based Dense-Field Algorithm for Video Copy–Move Detection and Localization," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 3, pp. 669-682, 2019, doi: 10.1109/TCSVT.2018.2804768
4. J. L. Zhong, Y. F. Gan, C. M. Vong, J. X. Yang, J. H. Zhao, et al., "Effective and efficient pixel-level detection for diverse video copy-move forgery types," Pattern Recognition, Vol. 122, p. 108286, 2022, doi: 10.1016/j.patcog.2021.108286.
5. H. Oliaei, and M. Azghani, "Video motion forgery detection using motion residual and object tracking," Multimedia Tools and Applications, vol. 83, no. 5, pp.12651-12668, 2024, doi: 10.1007/s11042-023-15763-6
6. S. Mohiuddin, S. Malakar, and R. Sarkar, "An ensemble approach to detect copy-move forgery in videos," Multimedia Tools and Applications, Springer, vol. 82, no. 3, pp. 3453-3474, 2023, doi: 10.1007/s11042-023-14554-3
7. R. M. Jasim, and T. S. Atia, "An evolutionary-convolutional neural network for fake image detection," Indonesian Journal of Electrical Engineering and Computer Science, vol. 29, no. 3, pp.1657-1667, 2023, doi: 10.11591/ijeecs.v29.i3.pp1657-1667
8. T. H. Kim, C. W. Park, and I. K. Eom, "Frame Identification of Object-Based Video Tampering Using Symmetrically Overlapped Motion Residual," Symmetry, vol. 14, no. 2, p. 364, 2022, doi: 10.3390/sym14020364
9. Y. Rodriguez-Ortega, D. M. Ballesteros, and D. Renza, "Copy-Move Forgery Detection (CMFD) Using Deep Learning for Image and Video Forensics," Journal of Imaging, vol. 7, no. 3, p. 59, 2021, doi: 10.3390/jimaging7030059
10. G. Singh, and K. Singh, "Copy-Move Video Forgery Detection Techniques: A Systematic Survey with Comparisons, Challenges and Future Directions," Wireless Personal Communications, springer, vol. pp. 1863-1913, 2024, doi: 10.1007/s11277-024-10996-6
11. G. Ulutas, B. Ustubıoglu, M. Ulutas, and V. Nabıyev, "Video forgery detection method based on local difference binary. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, vol. 26, no. 5, pp. 983-992, 2020.
12. V. Kumar, M. Gaur, and V. Kansal, "Deep feature-based forgery detection in video using parallel convolutional neural network: VFID-Net," Multimedia Tools and Applications, vol. 81, no. 29, pp. 42223-42240, 2022, doi: 10.1007/s11042-021-11448-0
13. W. El-Shafai, M. A. Fouda, E.S.M. El-Rabaie, and N. A. El-Salam, "A comprehensive taxonomy on multimedia video forgery detection techniques: challenges and novel trends," Multimedia Tools and Applications, vol. 83, no. 2, pp. 4241-4307, 2024 doi: 10.1007/s11042-023-15609-1
14. P. Kumar, M. Rakhimzhanova, S. Rawat, A. Orynbek, and V. Kamra, "Deep learning based COVID and pneumonia detection using chest X-ray," Indonesian Journal of Electrical Engineering and Computer Science, vol. 34, no. 3, pp.1944-1952, 2024, doi: 10.11591/ijeecs.v34.i3.pp1944-1952
15. K.B. handraiah, and N.K. Bhoganna, "An optimal model for detection of lung cancer using convolutional neural network," Indonesian Journal of Electrical Engineering and Computer Science, vol. 34, no. 1, pp.134-143, 2024, doi: 10.11591/ijeecs.v34.i1.pp134-143
16. M. Kalita, L.B. Mahanta, A.K. Das, and M. Nath, "A new deep learning model with interface for fine needle aspiration cytology image-based breast cancer detection," Indonesian Journal of Electrical Engineering and Computer Science, vol. 34, no. 3, pp.1739-1752, 2024, doi: 10.11591/ijeecs.v34.i3.pp1739-1752
17. A.K. Kumar, and S. Kumar, "Enhancing emotion detection with synergistic combination of word embedding and convolutional neural networks," Indonesian Journal of Electrical Engineering and Computer Science, vol. 35, no. 3, pp.1739-1752, 2024, doi: 10.11591/ijeecs.v35.i3.pp1933-1941
18. M. Sam'an, Safuan, and M. Munarif, "Convolutional neural network hyperparameters for face emotion recognition using genetic algorithm," Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no.1, pp.442-449, 2024, doi: 10.11591/ijeecs.v33.i1.pp442-449
19. C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," Multimedia Tools and Applications, vol. 79, no. 19, pp.12777–12815, 2020, doi: 10.1007/s11042-019-08453-9
20. F. Z. Mehrjardi, A. M. Latif, M. S. Zarchi, and R. Sheikhpour, "A survey on deep learning-based image forgery detection," Pattern Recognition, Vol. 144, p.109778, 2023, doi: 10.1016/j.patcog.2023.109778.
21. Akbari, S. Al Maadeed, O. Elharrouss, F. Khelifi, and A. Lawgaly, "A New Dataset for forged smartphone videos detection: description and analysis," IEEE Access, Vol. 11, pp.70387-70395, 2023. doi: 10.1109/ACCESS.2023.3267743
22. G. Qadir, S. Yahaya, and A. T. Ho, "Surrey University Library for Forensic Analysis (SULFA) of video content," In: Proceedings of the IET Conference on Image Processing, pp 1–6, 2012. http://sulfa.cs.surrey.ac.uk/
23. B. Tokas, V. R. Jakkinapalli, and N. Singla, "Video Forgery Detection and Localization with Deep Learning Using W-NET Architecture," in Computational Intelligence. Lecture Notes in Electrical Engineering, vol 968, pp. 31-38, 2023, Springer, Singapore. Doi: 10.1007/978-981-19-7346-8_3
24. O. I. Al-Sanjary, A. A. Ahmed, and G. Sulong, "Development of a video tampering dataset for forensic investigation," Forensic science international, vol. 266, pp. 565-572, 2016, doi: 10.1016/j.forsciint.2016.07.013
25. H. Li, J. Huang, "Localization of Deep Inpainting Using High-Pass Fully Convolutional Network," IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), pp. 8300-8309, doi: 10.1109/ICCV.2019.00839.

26. J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury, "Hybrid LSTM and Encoder–Decoder Architecture for Detection of Image Forgeries," In IEEE Transactions on Image Processing, vol. 28, no. 7, pp. 3286-3300, 2019, doi: 10.1109/TIP.2019.2895466.
27. J. L. Zhong, C. M. Pun, and Y. F. Gan, "Dense moment feature index and best match algorithms for video copy-move forgery detection," Information Sciences, vol. 537, 184-202, 2020, doi: 10.1016/j.ins.2020.05.134
28. L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," Journal of big Data, Springer, vol. 8, no. 53, 2021, doi: 10.1186/s40537-021-00444-8
29. A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artificial intelligence review, Springer, Vol. 53, pp. 5455–5516, 2020, doi: 10.1007/s10462-020-09825-6
30. A. Hegazi, A. Taha, and M. M. Selim, "An improved copy-move forgery detection based on density based clustering and guaranteed outlier removal," Journal of King Saud University - Computer and Information Sciences, vol. 33, no. 9, pp.1055–1063, 2021, doi: 10.1016/j.jksuci.2019.07.007
31. G. Tahaoglu, G. Ulutas, B. Ustubioglu, and V.V. Nabiyev, "Improved copy move forgery detection method via L*a*b* color space and enhanced localization technique," Multimedia Tools and Applications, vol. 80, pp.23419–23456, 2021, doi: 10.1007/s11042-020-10241-9
32. D.K. Shukla, A. Bansal, and P. Singh, "A survey on digital image forensic methods based on blind forgery detection," Multimedia Tools and Applications, Springer, vol. 83, pp. 67871–67902, 2024, doi: 10.1007/s11042-023-18090-y
33. M. Kalita, L. B. Mahanta, A.K. Das, and M. Nath, "A new deep learning model with interface for fine needle aspiration cytology image-based breast cancer detection," Indonesian Journal of Electrical Engineering and Computer Science, vol. 34, no. 3, pp.1739-1752, 2024, doi: 10.11591/ijeecs.v34.i3.pp1739-1752
34. X. Jin, Z. He, Y. Wang, J. Yu, and J. Xu, "Towards general object-based video forgery detection via dual-stream networks and depth information embedding," Multimedia Tools and Applications, 81(25), pp.35733–35749, 2022, doi: 10.1007/s11042-021-11126
35. K. B. Chandraiah, and N. K. Bhoganna, "An optimal model for detection of lung cancer using convolutional neural network," Indonesian Journal of Electrical Engineering and Computer Science, vol. 34, no. 1, pp.134-143, 2024, doi: 10.11591/ijeecs.v34.i1.pp134-1434
36. H. Alfraihi, M.S.A. Alzaidi, H. Alqahtani, et al. A multi-model feature fusion-based transfer learning with heuristic search for copy-move video forgery detection. Scientific Reports, 15, 4738 (2025). https://doi.org/10.1038/s41598-025-88592-2
37. M. M. Eltoukhy, F. S. Alsubaei, A. M. Mortda, K. M. Hosny, "An efficient convolution neural network method for copy-move video forgery detection". Alexandria Engineering Journal, 110, 429-437, 2025. https://doi.org/10.1016/j.aej.2024.10.030
38. R. Pandey & A.K.S. Kushwaha. Video Forgery Detection Using Multi-scale Feature Extraction with ResNet50. In International Conference on Artificial Intelligence and Smart Energy (pp. 165-171). Cham: Springer Nature Switzerland, 2025.
39. VTD (Video Tampering Dataset): [Available Online: https://www.youtube.com/channel/UCZuuu-iyZvPptbIUHT9tMrA, Accessed on February 2024]:
40. GRIP (Forged Videos Dataset): [Available Online: http://www.grip.unina.it/download/prog/, Accessed on February 2024] SULFA (Surrey University Library for Forensic Analysis): [Available online: http://sulfa.cs.surrey.ac.uk/, accessed on July 2023]
41. SULFA (Surrey University Library for Forensic Analysis): [Available online: http://sulfa.cs.surrey.ac.uk/, accessed on July 2023]