*mathematics*

MDPI

*Article*

# General Image Manipulation Detection Using Feature Engineering and a Deep Feed-Forward Neural Network

**Sajjad Ahmed [1,†], Byungun Yoon [2,*,†], Sparsh Sharma [3,†], Saurabh Singh [4,†] and Saiful Islam [5]**

[1] School of Computer Science Engineering, VIT Bhopal University, Bhopal-Indore Highway, Kothrikalan, Sehore 466114, Madhya Pradesh, India; sajjadahmed@vitbhopal.ac.in
[2] Department of Industrial & System Engineering, Dongguk University, Seoul 04620, Republic of Korea
[3] Department of Computer Science Engineering, National Institute of Technology Srinagar, Srinagar 190001, Jammu and Kashmir, India; sparsh.sharma@nitsri.ac.in
[4] Department of AI and Big Data, Woosong University, Seoul 34606, Republic of Korea; singh.saurabh@wsu.ac.kr
[5] Zakir Husain College of Engineering and Technology, Aligarh Muslim University, Aligarh 202002, Uttar Pradesh, India; saifulislam@zhcet.ac.in
* Correspondence: posman3@dongguk.edu; Tel.: +82-2-2260-8659
† All authors contributed equally to this work.

**Abstract:** Within digital forensics, a notable emphasis is placed on the detection of the application of fundamental image-editing operators, including but not limited to median filters, average filters, contrast enhancement, resampling, and various other operations closely associated with these techniques. When conducting a historical analysis of an image that has potentially undergone various modifications in the past, it is a logical initial approach to search for alterations made by fundamental operators. This paper presents the development of a deep-learning-based system designed for the purpose of detecting fundamental manipulation operations. The research involved training a multilayer perceptron using a feature set of 36 dimensions derived from the gray-level co-occurrence matrix, gray-level run-length matrix, and normalized streak area. The system detected median filtering, mean filtering, the introduction of additive white Gaussian noise, and the application of JPEG compression in digital Images. Our system, which utilizes a multilayer perceptron trained with a 36-feature set, achieved an accuracy of 99.46% and outperformed state-of-the-art deep-learning-based solutions, which achieved an accuracy of 97.89%.

**Keywords:** digital image forensics; multilayer perceptron; general-purpose image manipulation detection; operator detection; neural network; texture features

**MSC:** 68T07; 68U10



**Citation:** Ahmed, S.; Yoon, B.; Sharma, S.; Singh, S.; Islam, S. General Image Manipulation Detection Using Feature Engineering and a Deep Feed-Forward Neural Network. *Mathematics* **2023**, *11*, 4537. https://doi.org/10.3390/math11214537

Academic Editors: Hongang Qi, Yan Liu and Jun Miao

Received: 3 October 2023
Revised: 23 October 2023
Accepted: 30 October 2023
Published: 3 November 2023



## 1. Introduction

The creation of multimedia content such as digital images and videos for several platforms is comparatively easy in the current environment. Multimedia security challenges have multiplied significantly as a result of the general availability of computing gear and software, as well as the ease with which digital information can be created and altered. Determining the authenticity of digital content that originates from an illegitimate or unknown source could be challenging. Such digital content must first have its validity confirmed before consumption. An important form of digital contents are digital images [1]. A digital image is a representation of a two-dimensional visual scene, object, or subject in electronic form. It is a collection of individual picture elements or "pixels", each of which is a tiny square or dot that contains gray-value or color information. These pixels are organized in a grid, with each pixel having a specific color or gray value, which can be displayed on a screen or printed on paper. Even though digital images are a crucial sort of digital content, it is quite challenging to confirm their legitimacy [2,3].

The authenticity of digital photos is becoming more questioned with the introduction of modern image processing tools and the ease with which information is shared and altered. As a result, there is a growing preference for blind image forensic techniques.

Digital image forensics is a specialized field within digital forensics that focuses on the analysis and authentication of digital images to determine their origin and integrity, as well as the presence of any alterations or forgeries. It involves using various techniques and tools to examine digital images for signs of manipulation, tampering, or other forms of digital deception. Digital image forensics experts employ methods such as metadata analysis, image compression analysis, noise patterns, and error-level analysis to uncover inconsistencies and anomalies in images. This discipline is crucial in a world where digital images play a significant role in both legal and non-legal contexts, ensuring the credibility and trustworthiness of visual information in domains like criminal investigations, journalism, and the verification of digital evidence. Forensic examiners benefit from being able to observe the background of how much a digital image has been processed.

Digital image forensics aims to restore trust in digital image forensics (DIF). DIF confirms an image's legitimacy using image-editing fingerprints, and no prior knowledge-based techniques, such as watermarking, are needed [4].

General-purpose image manipulation operations involve the application of sets of operations that do not change the semantics or meaning of the images. Rather, they are used to remove traces left by other operations, making the detection of certain operators difficult. Various modifications, including median filtering, resampling, JPEG compression, and contrast enhancement, are on the list of general-purpose image manipulation and must be detected as part of the digital image forensics process [5,6].

In digital image forensics, inherent characteristic signatures left behind by image-editing methods are used to detect image changes, and the same process is applied for detection of general-purpose image manipulation operations performed on images. Detecting general-purpose image manipulation operators is a reasonable initial step in investigating the processing history of an image that may have gone through several transformations. Image forgers make use of these fundamental operators, such as median filters, which are intrinsically nonlinear. This allows them to remove any traces of evidence that may have been left behind by linear operations carried out on the images. Furthermore, in the fields of watermarking and steganography, the image's history is also important [4,7]. The research literature offers a variety of techniques for detecting fundamental operators applied to digital images. In most cases, these methods build techniques to detect basic operators on an individual basis. In contrast, comparatively little effort is put into the design of procedures that are effective in the detection of numerous operators.

The main contributions of our work are summarized as follows:

- We present a method of undertaking image classification for the purpose of image forensics by utilizing an existing body of domain knowledge called feature engineering.
- We developed a 36-dimension feature vector based on texture features for general-purpose image manipulation detection.
- We designed a system in which we replaced a CNN-based solution with an MLP-based solution. The MLP-based solution was found to perform better than the state-of-the-art methods.
- Furthermore, we propose GIMP-FCN, a multilayer perceptron (MLP) consisting of fully connected layers followed by activation layers that accept texture-based features for further learning from features and ultimately performs general-purpose image manipulation detection.
- The performance of our approach is superior to that of the most recent and cutting-edge method.
- Our work shows that a multilayer perceptron in combination with feature engineering can be employed for digital image forensics.

## 2. Related Work

Recent efforts by professionals have been directed toward the development of image forensic tools that are suitable for use in a variety of contexts and can determine whether or not an image has been processed and in what way the processing took place. The tools that were first developed for steganography have been repurposed so that they can be used to perform image forensics for general applications.

Researchers have employed deep learning methods to solve a large number of problems in various fields. Deep learning has also found use in the field of digital image forensics, where researchers are working to solve challenges connected to the detection of image tampering. In particular, the field of basic operator forensics makes use of convolutional neural networks (CNNs).

A steganalytic model-based universal forensics technique was introduced in [8]. The use of universal steganalytic features allows for a variety of image processing processes to be described as steganography and detected using these features. The findings of experiments reveal that all of the examined steganalyzers function well, in addition to showing that certain steganalytic approaches, such as the spatially rich model (SRM) [9] and LBP [10]-based methods perform significantly better than specialized forensic procedures. A detector of image manipulation that can be put to a variety of different uses was developed by Fan et al. [11]. The Gaussian mixture model (GMM) properties of small image patches are utilized by this detector in order to train itself on image-altering fingerprints. After collecting these fingerprints, one can determine whether or not a image has been altered.

A general approach for detecting basic operator manipulation was provided by Bayer et al. in [12]. The authors trained a CNN to automatically extract features from images after suppressing image contents by restricting a new convolutional layer called the constrained convolutional layer to only learn prediction error filters. This allowed them to achieve their goal. This method allowed the authors to accurately identify four different types of image-editing procedures, including median filtering and resampling, AWGN image corruption, and Gaussian filtering.

Mazumdar et al. [13] provided a general-purpose forensic technique based on a Siamese CNN. A Siamese neural network evaluates image similarity. Untrained for the detection of AWGN and gamma correction, the model's ability to recognize these two operations is an intriguing finding.

By studying image modification traces, researchers developed algorithms to detect targeted editing. This strategy has led to successful forensic algorithms, yet an issue persisted, i.e., the creation of individual forensic image detectors is difficult and time-consuming. Forensic analysts need access to general-purpose forensic algorithms that are in a position to recognize a wide variety of image manipulations. Bayar and Stemm [14] proposed a novel approach that can be used in forensic investigations in general and makes use of convolutional neural networks (CNNs) as the primary tool. The developed model is also available for transfer-learning-based image forensics.

The authors of [15] proposed a densely connected CNN for general-purpose image forensics based on isotropic constraints and taking into account antiforensic attacks. By reducing the image content information, the isotropic convolutional layer functions as a high-pass filter to highlight artifacts of image processing operations.

The CNN proposed by Yang et al. [16] includes a magnified layer that is part of the preprocessing process. In order to obtain an adaptive average pooling function from global average pooling that is able to accommodate any size of input pictures, the input images are enlarged using the nearest-neighbor interpolation algorithm in the magnified layer, then input into the CNN model for classification. This strategy was put to the test using six widely used image processing operations.

Rana et al. [17] designed a CNN called a multiscale residual deep convolutional neural network (MSRD-CNN) for image manipulation detection. In the first step of the procedure, which is called the preprocessing stage, an adaptive multiscale residual module is utilized to first extract the prediction error or noise features. Then, high-level image-

tampering features are retrieved from the collected noise features using a feature extraction network with several feature extraction blocks (FEBs). After that, the resulting feature map is presented to the fully connected dense layer for classification purposes. Although the MSRD-CNN achieves good results, it is very complex, consisting of around 76 layers, of which 26 are convolutional layers, 19 are batch normalization layers and 17 are 'ReLU' activation layers.

Ensemble learning is a machine learning technique that harnesses the power of diversity to enhance predictive accuracy and robustness. It involves the combination of the outputs of multiple individual models to create a more reliable and high-performing meta model. The key idea behind ensemble learning is that by aggregating the wisdom of several models, we can reduce the risk of overfitting and capture complex patterns in the data; this method was previously use in [18–21] for image forensics.

Table 1 summarizes the state-of-the-art methods for general image manipulation techniques, and Table 2 provides a summary of operators studied in some of the important studies in the literature.

**Table 1.** A summary of the methods used for general image manipulation detection.

| Method | Year | Reference | Network |
|---|---|---|---|
| Qui et al. | 2014 | [8] | Spatially rich model |
| Fan et al. | 2015 | [11] | Gaussian mixture model (GMM) |
| Mazumdar et al. | 2018 | [13] | Siamese CNN |
| Bayar et al. | 2018 | [14] | Constrained CNN |
| Rana et al. | 2022 | [17] | MSRD-CNN |

**Table 2.** A List of operators studied in different works in the literature works in the literature. Abbreviations: MF, median filtering; GB, Gaussian blurring; AWGN, additive white Gaussian noise (AWGN); RS, resampling; JPEG, JPEG compression; GC, gamma correction; UM, unsharp masking; AF, antiforensics.

| Method | OR | MF | GB | RS | GC | JPEG | AWGN | USM | AF |
|---|---|---|---|---|---|---|---|---|---|
| Qui et al. [8] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | × |
| Fan et al. [11] | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | × |
| Mazumdar et al. [13] | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | × |
| Bayar et al. [14] | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | × | × |
| Rana et al. [17] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ |

Because deep learning methods that extract features directly from images necessitate knowledge of topology, training methods, and other factors, there is no universally accepted theory that can be used to select appropriate deep learning tools. Training can be quite expensive due to the complexities of deep learning models. This applies to both the time and effort required to explore and select optimal deep learning model parameters, as well as the quantity of processing required [22]. Domain specialists have a significant edge over deep learning algorithms when using approaches like feature engineering, which require significantly less effort from the researcher. Furthermore, unlike deep learning systems, understanding the relationship between inputs and outputs is significantly simpler. The primary benefit of deep-learning-based solutions, on the other hand, is that no particular feature needs to be created for the problem. Without human involvement, the deep neural network extracts the desirable features [23–25].

A multilayer perceptron (MLP) is characterized an input layer, an output layer, and one or multiple optional hidden layers. An MLP is an example of a feed-forward artificial neural network that is made up of many perceptrons. The very first layer is known as the input layer and is responsible for feeding input data into the system. The last layer, known as the output layer, is responsible for making predictions based on the information that has been provided. In addition, there may be any number of other hidden layers in

the space in between these two levels. Every node that makes up this network is referred to as a neuron, and it uses nonlinear activation functions. During the forward pass, the signal is sent from the input layer to the output layer by way of the hidden layers. Through the use of the universal approximation theorem, George Cybanko [26] showed that a feed-forward network with a restricted number of neurons, a single hidden layer, and a nonlinear activation function can approximate training objects with a low error rate.

Recently, the use of popular neural network structures has been questioned, and multilayer perceptron (MLP)-based solutions with performance similar to that of deep-neural-network-based solutions have been proposed [24]. In [27], the authors investigated the possible performance of a neural network devoid of convolutions and offered suggestions on how the performance of fully connected networks might be improved. For the purpose of image classification, the authors of [28] introduced ResMLP, an architecture fully comprising multilayer perceptrons. In [29], researches replaced the attention layer with a simple feed-forward network. In [30], a gMLP, i.e., MLPs with gating, was designed to challenge vision transformer (ViT)-based solutions, inferring that gMLP performs better in comparison with bidirectional encoder representations from transformers popularly know as BERT models.

Shi et al. [31] compared deep learning development tools for the FCN-5 fully connected neural network, a five-layer fully connected neural network, and FCN-8, an eight-layer neural network, with CNN-based solutions AlexNet and ResNet-50 [32] for a variety of hardware and software tool combinations. The results indicate that the FCN-based solutions performed comparably to the CNN-based solutions. Zhao et al. [33] compared CNN-, transformer-, and MLP-based solutions and discovered that these three network architectures are comparable in terms of the accuracy–complexity tradeoff. We drew inspiration from the works cited above to perform image forensics tasks using an MLP.

The proposed work blends image processing domain expertise with a deep learning methodology. We developed a solution that detects image modification by basic operators by integrating existing domain knowledge in digital image forensics and image staganalysis with an MLP with nonlinearity in the form of activation layers after each fully connected layer. In this work, a feature vector based on texture characteristics is retrieved from an image. The texture characteristics are derived from the gray-level run-length matrix (GLRLM), the gray-level co-occurrence matrix (GLCM) matrices, and a normalized streak area (nsa) feature inspired by the percentage streak area (psa) developed in [34]. The gray-level co-occurrence matrix is used to create the first 22 features, the gray-level run-length matrix is used to derive the next 11 features, and the normalized streak area is used to derive the last 3 features.

Next, we developed a deep neural network that can discern between original images and images that are the result of the application of a range of image-editing processes. This network has fully connected layers and activation layers placed at strategic positions throughout its structure. In the end, we used a very large number of optimization parameters to compare the deep neural network against itself in order to optimize its performance. We performed many experiments in order to obtain a solid understanding of how well the designed system would perform. Furthermore, the results demonstrate that the proposed method can effectively differentiate between unfiltered and basic editing operators such as median filters, mean filters, additive white Gaussian noise, and JPEG compression when compared to the benchmark research [14] and the state-of-the-art method [17].

The remainder of this paper is laid out as follows. In Section 3, we provide details about the proposed features of the neural network, and in Section 4, we provide details about the experimental setup. Results are reported and discussed in Section 5; finally, Section 6 contains a discussion of future work. The proposed work combines domain knowledge of image processing with a deep learning approach. First, we designed a feature vector, then trained a deep neural network for classification.

## 3. The Proposed Method

In this section, we describe the proposed feature vector and the deep fully connected network developed for general-purpose image operation detection. The texture-based features are described in Section 3.1, and the proposed deep fully feed-forward neural network is described in Section 3.2, as shown in Figure 1. Figure 2 shows how training of the system is performed. The GLCM-, GLRLM-, and streak-area-based feature extractors extract features from the training images. These features are then used to train the proposed model depicted in Figure 1. The trained model is then fed features extracted from testing images, and the image manipulator operator is detected.

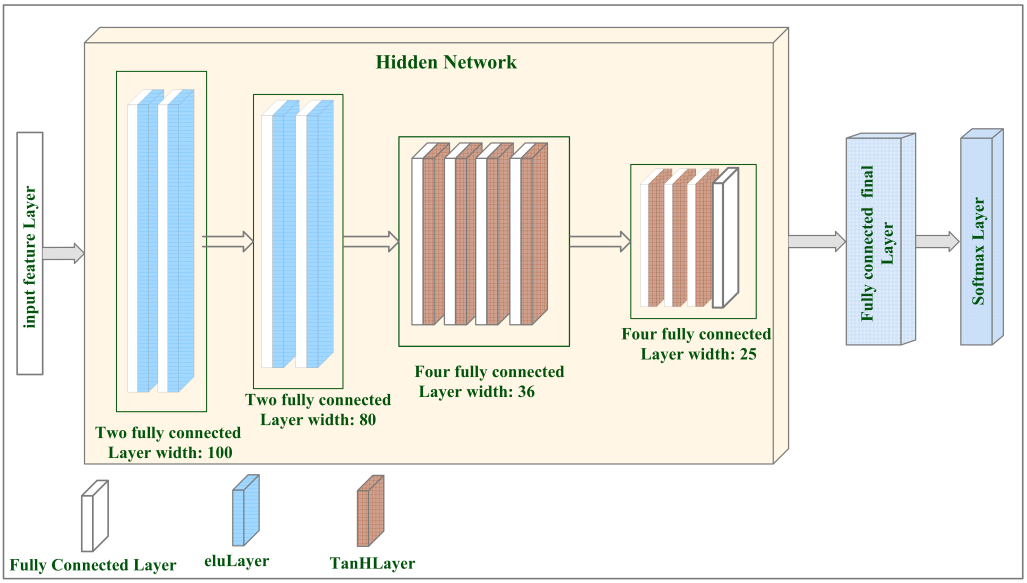



**Figure 1.** Neural network architecture: The feature vector that was extracted from the dataset images is accepted by the input feature layer. The elu activation layer comes after the previous two layers, which are each fully connected layers with a width of 100. The elu layer comes after two layers, each with a width of 80. A tanh activation layer is then followed by a group of four fully connected layers, each with a width of 36. Then, a set of four layers with a width of 25 follows, each followed by a tanh activation layer, except for the fourth-last layer. The total number of classes determines the final fully connected layer width. The classification process is then completed using a softmax layer.

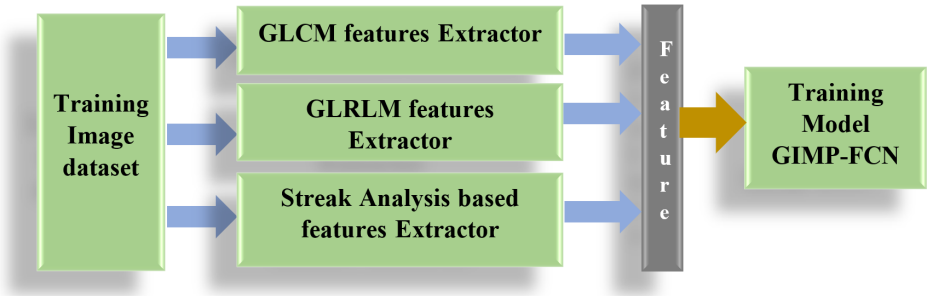



**Figure 2.** Training of the proposed deep fully connected neural network.

### *3.1. Proposed Features*

We designed the feature set for our purposes based on texture-based features. The Maximum Relevance Minimum Redundancy (MRMR) algorithm was applied to the texture features, and the 36 top-ranked features were selected. A total of 22 features that use GLCM for feature extraction were selected, as described in Section 3.1.1, with 11 GLRLM-based features selected, as described in Section 3.1.2. We also developed three streak-area analysis-based novel features, inspired by [34], as described in detail in Section 3.1.3. The final

feature ($f$) was generated by concatenating the three sets of features extracted from the GLCM, GLRLM, and normalized streak area of the images.

$$fv_{final} = [f_{GLCM}, f_{GLRLM}, f_{nsa}] \tag{1}$$

3.1.1. GLCM-Based Features

The gray-level co-occurrence matrix (GLCM) was first suggested by Haralick [35] in 1979 for the purpose of interpreting satellite images. It is one of the most researched and often used generic methodologies for texture analysis, and it has recently attracted the attention of a number of research organizations. Second-order statistics are taken into consideration in the GLCM. This technique studies pairs of pixels that are in certain spatial relationships to one another. There are certain advantages to the GLCM technique, but there are some disadvantages as well, such as the high dimensionality of the matrix. For this reason, a collection of features is often retrieved from the GLCM matrix for use in a variety of image processing applications. For our study, 14 GLCM-based features were taken from the original work of Haralick et al. [35], with 4 features from Soh et al. [36] and 4 features from Clausi et al. [37]. Let $p(i,j)$ be the $(i,j)^{th}$ entry in a normalized GLCM. The feature set of 22, *F1–F22*, features calculated from the GLCM is outlines as follows:

1. Energy, (angular second moment):

$$F1 = \sum_{i=1}^{Ng-1} \sum_{j-0}^{Ng-1} p(i,j)^2 \tag{2}$$

2. Contrast:

$$F2 = \sum_{n=0}^{Ng-1} n^2 \left\{ \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} p(i,j), |i-j| = n \right\} \tag{3}$$

3. Correlation:

$$F3 = \frac{\sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{4}$$

where $\mu_x$, $\mu_y$, $\sigma_x$, and $\sigma_y$ are the mean and standard deviations of $p_x$ and $p_y$, respectively.

4. Sum of squares (variance):

$$F4 = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} (i-\mu)^2 p(i,j) \tag{5}$$

5. Inverse difference moment:

$$F5 = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} \frac{1}{1+(i-j)^2} p(i,j) \tag{6}$$

6. Sum average:

$$F6 = \sum_{i=2}^{2Ng} i p_{x+y}(i) \tag{7}$$

7. Sum variance:

$$F7 = \sum_{i=2}^{2Ng} (1-f8)^2 p_{x+y}(i) \tag{8}$$

8. Sum entropy:

$$F8 = -\sum_{i=2}^{2Ng} p_{x+y}(i) log(p_{x+y}(i)) \tag{9}$$

9. Entropy:

$$F9 = -\sum_{i}^{Ng}\sum_{j}^{Ng} p(i,j) log(p(i,j)) \tag{10}$$

10. Difference variance:

$$F10 = \text{variance of } p_{x-y} \tag{11}$$

11. Difference entropy:

$$F11 = -\sum_{i=0}^{Ng-1} p_{x-y}(i) log(p_{x-y}(i)) \tag{12}$$

12. Information measure of correlation 1:

$$F12 = \frac{HXY - HXY1}{max\{HX, HY\}} \tag{13}$$

13. Information measure of correlation 2:

$$F13 = (1 - exp[-2.0(HXY2 - HXY)])^{1/2} \tag{14}$$

where HY and HY are the entropies of $p_x$ and $p_y$, respectively, and

$$HXY1 = -\sum_{i}\sum_{j} p(i,j) log(p_x(i) p_y(j))$$

$$HXY2 = -\sum_{i}\sum_{j} p_x(i) p_y(j) log(p_x(i) p_y(j))$$

14. Maximal correlation coefficient (MCC):

$$F14 = (\text{ Second-Largest Eigenvalue of Q})^{1/2} \tag{15}$$

where

$$Q(i,j) = \sum_{k} \frac{p(i,k) p(j,k)}{p_x(i) p_y(k)}$$

15. Homogeneity/inverse difference moment:

$$F15 = \sum_{i}\sum_{j} \frac{1}{1 + (i-j)^2} p(i,j) \tag{16}$$

16. Autocorrelation:

$$F16 = \sum_{i}\sum_{j} (ij) p(i,j) \tag{17}$$

17. Dis-similarity:

$$F17 = \sum_{i}\sum_{j} |i-j| p(i,j) \tag{18}$$

18. Cluster shade:

$$F18 = \sum_{i}\sum_{j} (i + j - \mu_x - \mu_y)^3 p(i,j) \tag{19}$$

19. Cluster prominence:

$$F19 = \sum_i \sum_j (i + j - \mu_x - \mu_y)^4 p(i,j) \tag{20}$$

20. Maximum probability:

$$F20 = MAX(i,j)\{p(i,j)\} \tag{21}$$

21. Inverse difference normalized (INN):

$$F21 = \sum \frac{C_i j}{1 + |i - j|} \tag{22}$$

22. Inverse difference moment normalized (IDN):

$$F22 = \sum \frac{C(i,j)}{1 + |i - j|^2} \tag{23}$$

where $C(i,j) = \frac{P(i,j)}{\sum_{i,j=1}^{Ng} P(i,j)}$.

3.1.2. GLRLM-Based Features

Galloway [38] first presented the gray-level run-length matrix (GLRLM)-based technique, which is a statistical approach to texture analysis. There is a vast collection of features based on the GLRLM that have been proposed in the research literature. These features are based on the properties of the gray-level runs that are present in the image. Coarse textures include several adjacent pixels with the same gray level, which is the basis for the GLRLM concept. Fine textures, on the other hand, are defined by a few pixels that are next to each other and have the same gray level. The GLRLM technique has been used in a variety of applications, as described in [39].

The 11-feature set denoted by $f_{glrlm}$ was selected for image forensics purposes, with the features extracted as described in [40] and as defined in the equations below.

A run-length matrix, denoted by the symbol $p$, where $p(i,j)$, is the number of runs that contain pixels with a of gray level $i$ and a run length of $j$. A run-length matrix with dimensions of MxNcan then be used to extract a variety of other attributes with respect to the texture. The first five *f1–f5* characteristics of run-length statistics derived by Galloway [41], *f6–f7* were proposed by Chu et al. [42] to extract more gray-level information from the matrix. According to the concept of a joint statistical measure of gray level and run length, Dasarathy and Holder [43] presented another four feature extraction functions: *f8* through *f11*.

The final 11-dimensional feature vector used in our study is described mathematically in [40] as follows:

1. Short-run emphasis (SRE):

$$f1 = \frac{1}{n_r} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{p(i,j)}{j^2} \tag{24}$$

2. Long-run emphasis (LRE):

$$f2 = \frac{1}{n_r} \sum_{i=1}^{M} \sum_{j=1}^{N} p(i,j).j^2 \tag{25}$$

3. Gray-Level non-uniformity (GLN):

$$f3 = \frac{1}{n_r}\sum_{i=1}^{M}\left(\sum_{j=1}^{N} p(i,j)\right)^2 \tag{26}$$

4. Run-length non-uniformity (RLN):

$$f4 = \frac{1}{n_r}\sum_{j=1}^{N}\left(\sum_{i=1}^{M} p(i,j)\right)^2 \tag{27}$$

5. Run percentage (RP):

$$f5 = \frac{n_r}{n_p} \tag{28}$$

6. Low gray-level run emphasis (LGRE):

$$f6 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)}{i^2} \tag{29}$$

7. High gray-level run emphasis (HGRE):

$$f7 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i,j).i^2 \tag{30}$$

8. Short-run, low-gray-level run emphasis (SRLGE):

$$f8 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j)}{i^2.j^2} \tag{31}$$

9. Short-run, high-gray-level run emphasis (SRHGE):

$$f9 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j).i^2}{j^2} \tag{32}$$

10. Long-run, low-gray-level run emphasis (LRLGE):

$$f10 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{p(i,j).j^2}{i^2} \tag{33}$$

11. Long-run, high-gray-level run emphasis:

$$f11 = \frac{1}{n_r}\sum_{i=1}^{M}\sum_{j=1}^{N} p(i,j).j^2.i^2 \tag{34}$$

where $n_r$ is the total number of streaks, and $n_p$ is the number of pixels in the image.

#### 3.1.3. Normalized Streak-Area-Based Feature

A streak is a series of pixels with the same or almost the same intensity value that appear consecutively. If there is a run length in the picture of exactly n consecutive pixels that all have the same intensity values, then the image contains a streak of length n. The amount of streaking that is present in an image after it has been median-filtered is noticeably different from the amount of streaking that was present in the image before

it was median-filtered. The streaking effect was quantified in [34] and further improved in [44] for differentiation between median-filtered and unfiltered images. The last three features investigated in this work are inspired by [44].

Let $I$ be a digital image in gray-scale mode with dimensions of $MxN$. The total number of pixels in the image is $A = MxN$.

Let $\overrightarrow{\zeta}(j)$ represent the number of horizontal streaks with a pixel length of $j$ that are present in the images ($I$) when measured from left to right. $\overrightarrow{\zeta}(I)$ is the sum of pixels involved in the row-wise streaks in the image ($I$) and can be written as $\overrightarrow{\zeta}(I) = \sum_{j=2}^{N}\left(j * \overrightarrow{\zeta}(j)\right)$. For image $I$, the normalized streak area measured from left to right ($\overrightarrow{nsa}(I)$) is expressed as

$$\overrightarrow{\eta}(I) = \frac{\overrightarrow{\zeta}(I)}{A} \tag{35}$$

In a similar manner, the normalized column-wise streak is expressed as

$$\downarrow \eta(I) = \frac{\downarrow \zeta(I)}{A} \tag{36}$$

Similarly, the normalized diagonal streak is expressed as

$$\searrow \eta(I) = \frac{\searrow \zeta(I)}{A} \tag{37}$$

Finally, a three-dimensional feature vector is extracted by applying Equations (35)–(37) as follows:

$$f_{nsa} = \left[\overrightarrow{\eta}(I), \downarrow \eta(I), \searrow \eta(I)\right] \tag{38}$$

### *3.2. Neural Network Architecture*

We developed a deep feed-forward network using fully connected layers with an activation function at appropriate places, with a final fully connected layer with an output size of two for binary classification and five for multiclass classification and a final softmax layer for the classification task. The input layer takes input feature vector; in our case, the size of the input layer was configured to accept 36 features. The input is preprocessed by applying z-score normalization as shown to improve the performance of machine learning methods [45]. Z-score normalization refers to the process of adjusting each value in a dataset such that the mean of all of the values is equal to zero and the standard deviation is equal to one. In a mathematical sense, the z-score modification of data is applied to each and every feature vector. For data with $\mu$ as a mean and $\sigma$ as the standard deviation, after z-score normalization of the input feature ($fv$), the output feature ($zfv$) is expressed as:

$$zfv = \frac{fv - \mu}{\sigma} \tag{39}$$

The first fully connected layer of the neural network is connected to the network input, and each layer after that is fully connected to the layer before it. Following the multiplication of the input by a weight matrix in each fully connected layer, a bias vector is added. After each fully connected layer, an activation layer is applied. No activation layer is used before the final fully connected layer. Subsequently, the softmax activation function produces the classification scores.

We designed our neural network by considering several parameters such as the number of fully connected layers from 1 to 100; the activation function was searched among 'relulayer', 'tanhlayer', 'sigmoidlayer', 'swishlayer', 'elulayer', 'gelulayer', and 'none'. A detailed survey of different activation-layer functions can be found in [46]. The width of each fully connected layer was searched from 10 through 300 .

The three different initial layer weights were adopted from [47–49] . Initial layer biases were searched from 'zero' and 'one' . The maximum number of training iterations was

kept as 8000, and loss tolerance was kept at $10^8$. The learning rate was optimized in the range of $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$. Every network was trained for over 80 epochs. It is the responsibility of optimization algorithms or techniques to reduce losses and offer the most accurate outcomes possible. We used the adaptive moment estimation (ADAM) optimization solver for our problem. The summary of parameters for designing MLP is provided in Table 3. The optimized neural network is shown in Figure 1.

**Table 3.** Search space for the design of an MLP.

| Parameter | Search Space |
|---|---|
| No. of fully connected layers | 1 to 100 |
| Width of fully connected layers | 10 to 300 |
| Activation layers | { 'Mixed-layers', 'relulayer', 'tanhlayer', 'sigmoidlayer', 'swishlayer', 'elulayer', 'gelulayer', 'none' } |
| Learning rate | $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ |
| Layer biases | { 'zeros' and 'ones' } |

The final optimized neural network contained 12 fully connected layers with different input sizes. The 'elu' activation layer was the best choice for the activation function for the first four layers; for the next twelve fully connected layers, the 'tanh' layer was used as the activation function, with initially orthogonal weights [49] for fully connected layers, the number of initial layer biases set to zero, and the optimal learning rate found to be 0.0001. The processed features can than be used for general-purpose image manipulation detection by applying an appropriate classification layer, as show in Figure 1.

## 4. Experimental Setup

In order to test the performance of the proposed method in the identification of various image processing activities, we performed an extensive set of experiments. Standard image datasets UCID [50], BOSSBass [51], RAISE [52], and the Dresden image dataset (DID) [53] were used to generate various training and testing sets for various experiments.

$$DS_{orig} = \{UCID, RAISE, BOSSBass, DID\} \tag{40}$$

A frequently used dataset called UCID [50] (Uncompressed Color Image Datasets) contains 1338 colored images with resolutions of $512 \times 384$ and $384 \times 512$. Images can be used as a base to create testing and training datasets for the benchmarking of detectors on uncompressed image datasets, from which additional processed datasets can be generated. The main feature of UCID is that images are in their uncompressed state. The UCID dataset, which consists of images in the TIFF format, was created initially for content-based image retrieval (CBIR). It is now used by a very wide range of image-based algorithms and is one of the primary datasets on which researchers test operator detectors.

Released in May 2011, the BOSS base 1.1 [51] dataset (Break Our Stenographic System) consists of 10,000 uncompressed $512 \times 512$-resolution images from the BOSS competition that were taken by seven different cameras. The images in the dataset were produced from color, full-resolution RAW images. The BOSS dataset has also been updated in the past. With CNN-based techniques, the BOSSbase dataset is more widely used.

The Dresden Image Dataset was initially created for camera-based digital forensic methods. It is made up of over 14,000 photos taken with roughly 73 different cameras. Images from many different scenarios can be found in the dataset.

1388 images with dimensions of $512 \times 386$ from UCID, $10,000$ images with dimensions of $512 \times 512$ from the BOSSbase, 1448 images of varying dimensions from DID, and 4000 images from the RAISE dataset were used to set up a total count of $16,836$ images. A total of $16,000$ images of varying sizes were thus selected to construct $DS_{orig}$. The original image set was then used as a base for the generation of various training and testing datasets.

All images in $DS_{orig}$ were cropped to extract multiple image patches with dimensions of 256 × 256 to create $DS_{orig256}$. The large images in datasets such as DID were cropped from the center, and multiple non-overlapped images with dimensions of 256 × 256 were extracted for dataset generation, with small image datasets such as UCID and BOSSbase contributing one or two image patches.

Gray-scale conversion was performed on all colored images as per Rec.ITU-R BT.601-7 [54], grouping together a weighted average of the red(R), green(G), and blue(B) components as follows:

$$grayvalue = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{41}$$

*Dataset Generation*

The $DS_{orig256}$ was used to generate datasets for this study. To construct datasets for individual operations such as the median-filtered image dataset ($DSmf_w$), window sizes of w, $= \{3, 5, 7\}$ were employed to filter the $DS_{orig256}$ images, generating three different datasets ($DSmf_3$,$DSmf_5$,$DSmf_7$). Similarly, the additive white Gaussian noise (AWGN), denoted by the $DSAWGN_\sigma$ dataset, was created by setting $\sigma = \{0.1, 0.6, 1.2, 1.8\}$. The JPEG-compressed dataset ($DSJPEG_{QF}$) was created by compressing $DS_{orig256}$ with a JPEG compression quality factor of $QF = \{30, 50, 70, 90\}$. A mean filter datasets ($DSMeanF_w$) was created by mean filtering each image in $DS_{orig256}$ using a filter window with dimensions of $w = \{3 \times 3, 5 \times 5, 7 \times 7\}$. The Figure 3 shows images in various datasets generated for the study. The Figure 3a shows image from original gray scale image dataset. The Figure 3b shows the same image median filtered with filter window size of 3 × 3. The Figure 3c shows the image compressed with JPEG compression. The Figure 3d shows the image with added AWGN noise. The Figure 3e shows the images mean filtered image. Table 4 summarizes the parameters used for dataset generation.

(**a**) (**b**) (**c**) (**d**) (**e**)

**Figure 3.** Sample images from the dataset used in this study: (**a**) original image; (**b**) median-filtered image; (**c**) JPEG-compressed image; (**d**) AWGN-added image; (**e**) mean-filtered image.

**Table 4.** Dataset generation parameters for experimentation.

| Editing Operation | Parameter |
|---|---|
| Median filtering (MF) | mw = 3, 5, 7 |
| Mean filter (MnF) | mnw = 3 × 3, 5 × 5, 7 × 7 |
| Additive white Gaussian noise (AWGN) | $\sigma$ = 0.1, 0.6, 1.2, 1.8 |
| JPEG compression | QF = 30, 50, 70, 90 |

Finally, results are reported in terms of parameters defined in Equations (42)–(49) as follows:

Accuracy is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{42}$$

Recall is defined as

$$Recall = \frac{TP}{TP + FN}. \tag{43}$$

Specificity is defined as

$$Specificity = \frac{TN}{FP + TN}. \tag{44}$$

Precision is defined as

$$Precision = \frac{TP}{TP + FP}. \tag{45}$$

The false-positive rate (FPR) is defined as

$$FPR = \frac{FP}{FP + TN}. \tag{46}$$

The F1 score is defined as

$$F1score = \frac{2 * TP}{2 * TP + FP + FN} \tag{47}$$

The Error, miss classification error, is defined as

$$Error = \frac{FP + FN}{FP + FN + TP + TN}. \tag{48}$$

The Matthews correlation coefficient (MCC) is defined as

$$MCC = \frac{(TP * TN - FP * FN)}{\sqrt{((TP + FP)(TP + FN)(TN + FP)(TN + FN))}}. \tag{49}$$

The Matthews correlation coefficient (MCC) can have values between $-1$ and 1, with $-1$ being the lowest and 1 being the highest. A value of $-1$ means that the predicted classes and the actual classes are completely different. A value of 0 means that the guessing was totally random, and a value of 1 means that the predicted classes and the actual classes are exactly the same. The MCC is a more reliable statistical rate that only yields a high score if the prediction was correct in all four of the confusion matrix categories [55].
In the equations shown above, the notation TP represents for the number of true positives, TN refers to the number of true negatives, FP stands for the number of false positives, and FN stands for the number of false negatives.

We compared our work with two very significant works in the state of the art: those reported by Bayers [14] and Rana [17]. Bayers work was implemented with network was trained as described by the author . Rana's [17] work was also implemented and simulated; for a better comparison, we used the same dataset described in Equation (40).

We implemented the experiments using Matlab 2021 [56] on a system with an Intel Core -i7 and a Nvidia GeForce GTX 1080 GPU graphics processing unit (GPU) with 8 GB of dedicated memory and 16 GB of RAM . The deep learning toolbox [57] was employed to design the networks, and the Experiment Manager app [58] was used to manage the experiments and for thorough testing of the models.

## 5. Results and Discussion

The state-of-the-art methods are based on deep learning approaches and are dominated by one particular type of deep learning model called convolutional neural networks (CNNs). Tables 1 and 2 provide a summary of such methods. The problem with any deep-learning-based solution is that it takes a large amount of data to outperform other solutions. Because such methods require knowledge of the topology, training method, and other characteristics, there is no universally accepted theory that can be applied to the selection of appropriate tools for deep learning . Therefore, it is challenging for those with lower levels of deep neural network design knowledge to adapt and design neural network models. In addition, due to the complexity of the data models, training may be extremely computationally expensive. This is true in terms of both the time and effort required to research and make an acceptable selection of various deep learning model parameters, as well as the quantity of computation that is necessary. When compared to deep learning methods, the application of techniques such as feature engineering, which require significantly less effort on the part of the user, gives domain specialists a significant advantage in their ability to find

answers faster. When compared with systems based on deep learning, it is much simpler to analyze and comprehend the relationship between the inputs and outputs of the system. In our proposed method, we combined the best of deep learning and feature engineering methods. We developed a way to detect general image alteration operations, using the domain knowledge gained from working in the field of image forensics to create a solution using deep learning for automatic extraction of classification information from the features extracted from the images. We applied domain knowledge in the field of image forensics to engineer features and developed a neural network to detect general image manipulation operations.

The Figure 4 shows the strategy for testing of the proposed model for single and multiple manipulation operation detection.

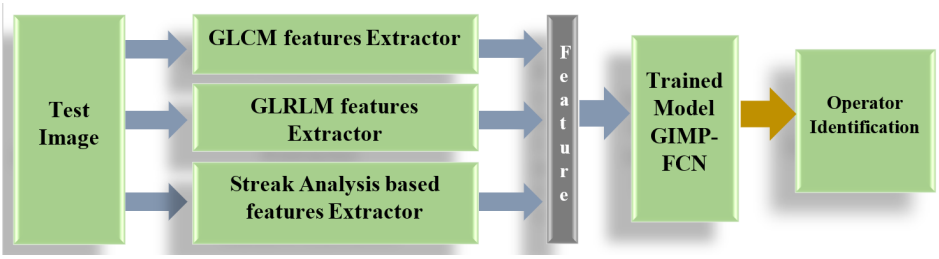



**Figure 4.** Implementation of the operator manipulation identifier using the proposed deep fully connected neural network.

### *5.1. Single-Manipulation Detection*

The detection of the use of a single operator is very important. We trained our model for detection classification, whether an image is original or modified using, one of the operators investigated in this study. Single-manipulation detection involves the binary classification of original images and images tampered with by the application an operator. We trained our model for binary classification for the detection of whether an image is original or modified using one of the operators investigated in this study. To detect single-operator modification, we created datasets from original image patches and operator datasets. All binary classifications were performed with original images vs. operator datasets, i.e., { $DSmf_3$, $DSmf_5$, $MeanF_3$, $MeanF_5$, $AWGN_{1.2}$, $AWGN_{1.8}$, $RS_{1.4}$, $RS_{1.4}$, $JPEG_{70}$, $JPEG_{80}$, and $JPEG_{90}$ }. Next, the image features described in Section 3.1 were extracted from the corresponding image datasets. Finally, training, validation, and testing sets were generated for each original vs. operation binary classification, with 70% of data used for training, 10% for validation, and the last 20% used for testing purposes. The deep neural network described in Section 3.2 and Figure 1 was configured by setting up a last fully connected layer with a width of two and a 'softmax' layer for binary classification. The input layer was a feature input layer that accepts a feature vector with a length of 36. The hidden network contained 12 fully connected layers, each followed by an activation layer. The first four fully connected layers were followed by an ' elu' activation layer, and the final seven fully connected layers were followed by a 'tanh' activation layer. The 12th fully connected layer was not followed by any activation layer. Each network was trained for 80 epochs. The initial layer weight was adopted from Xavier [47], with initial layer biases set to 'zero' and the optimal learning rate value found to be 0.0001. Figure 1 shows the proposed neural network architecture. The Adam optimization solver, a popular extension of stochastic gradient descent, was used to train our deep neural network model. Table 5 shows the results obtained using the proposed method. The first column of Table 6 describes the accuracy obtained when binary classification of original vs. median-filtered images with a filter size of $3 \times 3$ was performed. For comparison, the transfer learning models provided by Bayers were utilized. The results are presented in the form of a confusion matrix, as well as crucial assessment criteria for machine learning algorithms. Our combined approach of feature engineering and the application of a deep neural network design outperformed the approach proposed by Rana [17] for single-operation detection, as evident from Table 5.

**Table 5.** Testing accuracy for single-operator detection.

| Method | MF | | Mean Filter | | AWGN | | RS | | JPEG Compression | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $3 \times 3$ | $5 \times 5$ | $3 \times 3$ | $5 \times 5$ | $\sigma = 1.2$ | $\sigma = 1.8$ | **1.4** | **1.8** | **QF = 30** | **QF = 70** | **QF = 90** |
| **Proposed** | 99.93 | 99.95 | 99.96 | 99.96 | 99.99 | 99.99 | 99.68 | 99.67 | 99.56 | 99.85 | 99.91 |
| **Bayers [14]** | 99.65 | 99.78 | 99.85 | 99.85 | 99.84 | 99.86 | 99.38 | 99.26 | 99.42 | 99.45 | 99.52 |
| **Rana [17]** | 99.15 | 99.18 | 99.45 | 99.40 | 99.14 | 99.26 | 99.28 | 99.06 | 98.70 | 98.52 | 99.10 |

**Table 6.** Image datasets used in the study.

| Dataset | Reference | Number of Images | Download Link |
|---|---|---|---|
| UCID | [50] | 1388 | https://nas.minelab.tw:5001/fsdownload/lXGvXOg5g/UCID20Database/ (accessed on 01 November 2023) |
| BOSSBase | [51] | 10,000 | https://dde.binghamton.edu/download/ImageDB/BOSSbase_1.01.zip (accessed on 01 November 2023) |
| DID | [53] | 1448 | http://forensics.inf.tu-dresden.de/ddimgdb/ (accessed on 01 November 2023) |

### *5.2. Multiple-Manipulation Detection*

Finding the precise technique that was used to change an image is a key challenge in the field of forensics because it might be difficult to determine which procedure was used. When it comes to operator application, this can be a difficult task due to the fact that several other procedures leave behind forensic traces that are quite similar to one another and can make it difficult to differentiate between them. Multiple operators must be identified by the trained model in order for multiple-manipulation detection to be effective. Such a multiple-manipulation detector is capable of identifying various operators. To accomplish this, we used a multiclass classification to enable our model to distinguish between the original images, median-filtered images, mean-filtered images, JPEG-compressed images, and images with AWGN added.

In order to verify the efficacy of our technique for general-purpose image operation detection, we produced a large dataset consisting of 16,000 images of varying sizes from our original dataset, as denoted by $DS_{orig}$. Only one or two patches were taken from smaller images, such as those produced by UCID and BOSSbase, which were then clipped from the image's central region. For large image datasets such as Dresden and RAISE, images with dimensions of $8 - 10$ $256 \times 256$ were cropped relative to the image's center. The number of images cropped from a single image depended on the size of the image. The name given to this dataset was $DS_{orig256}$. The entire $DS_{orig256}$ dataset, which comprises 30,000 randomly selected image patches measuring $256 \times 256$ pixels, was then subjected to processing in order to generate datasets using the five distinct manipulations detailed in Table 4and Section 4, resulting in a total of 150,000 image patches being produced. To generate the training datasets, a total of 105,000 patches were randomly extracted from the operation datasets. A total of 15,000 image patches were kept for validation of the model. The number of image patches selected from each operation dataset was maintained at the same level. Similarly, the test dataset required a total of 30,000 images, with 6000 images taken from each of the operation datasets. Finally, the training dataset for multiclass classification was fed to a deep feed-forward neural network that was customized for multiclass classification by changing the final fully connected layer to a width of 5. The Bayers technique and that proposed by Rana [17] were used to compare our results obtained for single- and multiple-operation detection. For texture-based feature extraction using the proposed method, we first extracted features from dataset images; then, the extracted features were fed to he proposed fully connected network for further processing.

The results of multiclass classification are presented in the form of a confusion matrix in Tables 7 and 8. This matrix demonstrates that the proposed method performs more

effectively than the Bayers method, as well as the Rana method, for each class. The state-of-the-art benchmark approaches proposed by Bayer [14] and Rana [17] were surpassed by our method in its ability to differentiate between original, median-filtered, and mean-filtered images, as well as between photos with AWGN noise added and JPEG-compressed images, as evident from Table 9 in term of the reported statistics; for example, MCC and kappa and are more reliable parameters than simple accuracy.

The results obtained using our method are presented in term of the most commonly evaluated classification metrics in Table 9. Comparatively, Bayer's technique obtained an accuracy of 97.89%, whereas our proposed solution reached 99.46 % accuracy as a macro average. The proposed method performed well in terms of the following evaluation parameters: Accuracy, Error, Recall, Specificity, Precision; False-positive rate, F1 score, kappa, and Matthews correlation coefficient (MCC). Both the kappa and Matthews correlation coefficient provided encouraging results as compared to [14,17].

**Table 7.** Confusion matrix for Proposed method and Bayers method for operator detection.

| | | **Predicted Class** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Proposed Method** | | | | | **Bayer [14]** | | | | |
| | | **Orig** | **MF** | **MnF** | **AWGN** | **JPEG** | **Orig** | **MF** | **MnF** | **AWGN** | **JPEG** |
| **True Class** | **orig** | **99.43** | 0.18 | 0.15 | 0.08 | 0.15 | **98.15** | 0.42 | 0.78 | 0.37 | 0.28 |
| | **MF** | 0.05 | **99.67** | 0.12 | 0.03 | 0.13 | 0.35 | **98.67** | 0.58 | 0.25 | 0.15 |
| | **MnF** | 0.05 | 0.27 | **99.42** | 0.05 | 0.22 | 0.17 | 0.45 | **97.7** | 0.65 | 1.03 |
| | **AWGN** | 0.03 | 0.05 | 0.15 | **99.58** | 0.18 | 0.32 | 0.45 | 0.57 | **97.85** | 0.82 |
| | **JPEG** | 0.12 | 0.08 | 0.27 | 0.35 | **99.18** | 0.48 | 0.48 | 0.85 | 1.1 | **98.05** |

**Table 8.** Confusion matrix for Proposed method and Ranas method for operator detection

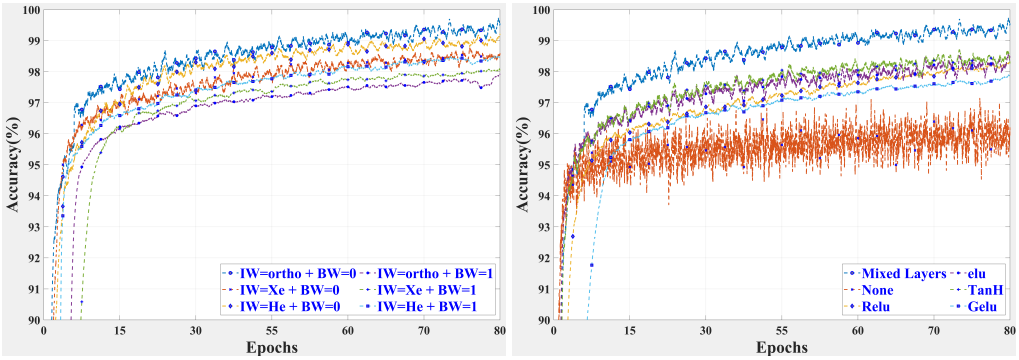
| | | Predicted Class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Proposed Method** | | | | | **MSRD-CNN [17]** | | | | |
| | | **Orig** | **MF** | **MnF** | **AWGN** | **JPEG** | **Orig** | **MF** | **MnF** | **AWGN** | **JPEG** |
| **True Class** | **orig** | **99.43** | 0.18 | 0.15 | 0.08 | 0.15 | **97.00** | 0.42 | 0.92 | 0.42 | 1.25 |
| | **MF** | 0.05 | **99.67** | 0.12 | 0.03 | 0.13 | 0.48 | **97.38** | 0.78 | 0.22 | 1.13 |
| | **MnF** | 0.05 | 0.27 | **99.42** | 0.05 | 0.22 | 0.57 | 1.02 | **96.75** | 0.32 | 1.35 |
| | **AWGN** | 0.03 | 0.05 | 0.15 | **99.58** | 0.18 | 0.83 | 0.38 | 0.35 | **97.68** | 0.75 |
| | **JPEG** | 0.12 | 0.08 | 0.27 | 0.35 | **99.18** | 0.82 | 0.90 | 0.98 | 0.72 | **96.58** |

**Table 9.** Multiclass classification results for general image manipulation detection.

| | **Proposed Method** | **Bayer [14]** | **MSRD-CNN [17]** |
|---|---|---|---|
| **Accuracy (%)** | 99.46 | 97.89 | 97.08 |
| **Error (%)** | 00.54 | 02.11 | 02.92 |
| **Recall (%)** | 99.46 | 97.90 | 97.08 |
| **Specificity (%)** | 99.86 | 99.47 | 99.27 |
| **Precision (%)** | 99.46 | 97.90 | 97.08 |
| **FPR (%)** | 00.14 | 00.53 | 0.73 |
| **F1 score (%)** | 99.46 | 97.89 | 97.08 |
| **MCC** | 0.9932 | 0.9737 | 0.9635 |
| **Cohen's Kappa(%)** | 98.30 | 93.42 | 90.88 |

One of the most important hyperparameter decisions in deep learning systems affecting both convergence times and model performance is the choice of initial component values for the optimization of deep neural networks. We experimented with various weight initialization algorithms and combinations of activation layers. The weight initialization algorithms proposed by Xavier [47] and He [48], as well as orthogonal [49] and bias initialization of 'zero' and 'one' were studied. Figure 5a shows the training accuracy when

the proposed model was employed for general-purpose image manipulation detection for different numbers of epochs.




(**a**) (**b**)

**Figure 5.** Experimentation with weight, bias and activation layers. (**a**) Accuracy for various weight and bias combinations; (**b**) Accuracy for various activation layers.

The system performed best for with orthogonal weight and bias set to 'zero' as compared to other combinations of weight initialization and bias initialization schemes, as summarized in Table 10. Among the various combinations, the accuracy of six combinations of weight initialization and bias initialization methods are reported. The results show that the combination of orthogonal weights and 'zero' bias performed better than other tested combinations. We also tested narrow normal bias initialization, but the results were not satisfactory.

**Table 10.** Testing accuracy for various combinations of layer weight and bias initialization methods.

| Weight Initialization | Bias Weight | Accuracy |
|---|---|---|
| He | 'Zero' | 98.38 |
| Xe | 'Zero | 98.44 |
| Orthogonal | 'Zero | 99.46 |
| He | 'One' | 98.23 |
| Xe | 'One' | 97.79 |
| Orthogonal | 'One' | 98.01 |

All these experiments were conducted by keeping the number and width of each fully connected layer fixed, with activation methods that were a mix of 'elu' and 'tanh'.

We also experimented with various combinations of activation methods, i.e., 'ReLU', 'tanh', 'elu', ' gelu', 'swish', 'leaky ReLU', 'none', as well as a mixture of activation layers placed at different positions. Every component of the input is subjected to a threshold operation when a 'ReLU' layer is present. This operation resets any value less than zero to zero. The tangent hyperbolic 'tanh' method was applied to the layer inputs by an activation layer using the 'tanh' function. When fed positive inputs, an ' elu' activation layer carries out the identity operation, whereas when fed negative inputs, it carries out an exponential nonlinearity operation. A 'leaky ReLU' layer carries out a threshold operation that includes multiplication of any input value that is smaller than zero by a constant scalar. Another type of activation method is called a swish activation layer, which uses the swish function on the inputs. Gated linear units (gelus) are the component-wise product of two linear projections, one of which is passed via a sigmoid function beforehand. Activation methods are discussed in detail in [46]. Table 11 summarizes the obtained results, and Figure 5b shows the testing accuracy for different epochs for activation functions employed in between fully connected layers. We can clearly see that the mixture of activation layers in which the first four layers were followed by an 'elu' layer and next nine fully connected layers were followed by a 'tanh' layer performed better than network architectures in which only one activation function was used throughout the network structure. The use of no

activation layer performed poorly as compared to the other layers. The 'ReLU', 'elu', 'tanh' and 'gelu' configurations produced similar results, but the mixed combination performed exceptionally well. The 'swish' and 'leaky ReLU' configurations are not reported, as their results were not satisfactory. We experimented with different mixes of layers for our network architecture and found that the best mix of activation layers was that with four 'elu' and nine 'tanh' activation layers, as shown in Figure 1 and Table 11.

**Table 11.** Testing accuracy for various activation functions.

| Activation Function | Accuracy (%) |
|---|---|
| None | 96.33 |
| ReLU | 98.08 |
| elu | 98.38 |
| tanh | 98.45 |
| gelu | 98.00 |
| 4 elu + 9 tanh | 99.46 |

The limitations of the work are two stage development. As compared to CNN designing where we supply images directly and feature extraction is done by the CNN model, our method works by first performing feature selection and then an MLP is designed for the problem-solving. MLP has a smaller search space as compared to CNN that has an infinite search space.

## 6. Conclusions and Future Work

We developed a method for general-purpose image alteration detection by applying the feature engineering methodology and combined it with a deep neural network design strategy. First, we established a set of features to achieve image modification detection . Next, we designed a deep neural network that utilizes fully connected layers with activation layers at appropriate points to differentiate between original images and a variety of image alteration procedures. Finally, we optimized the performance of the deep neural network by comparing it to itself using a very large number of optimization parameters.

In order to get a good idea of how well the planned system would function, we conducted a number of tests. The findings of the studies show that the proposed system, which employs a multilayer perceptron (MLP) trained with a 36-feature set, attained an accuracy of 99.46%. This method outperforms current deep-learning-based solutions, which achieved an accuracy of 97.89%.

In the future, it will be necessary to include a large number of operators in research on operation detection, and it will also be necessary to implement a more thorough approach for feature engineering that takes into account the extraction of larger feature sets from images for operator detection. Another dimension in which this work can be extended is experimentation with recent innovations in deep learning models.

In comparison to state-of-the-art methods, the real-world implementation of this work is anticipated to be faster because it requires less computation and has fewer parameters. In the future, we will perform a detailed time and space complexity analysis of the system proposed above.

**Author Contributions:** Conceptualization, S.A. and S.S. (Sparsh Sharma); methodology, S.S. (Sparsh Sharma); software, S.S. (Saurabh Singh); validation, S.I., B.Y. and S.A.; formal analysis, S.A.; investigation, S.S. (Sparsh Sharma); resources, S.S. (Saurabh Singh); data curation, S.I.; writing—original draft preparation, S.A.; writing—review and editing, B.Y.; visualization, S.I.; supervision, S.S. (Sparsh Sharma); project administration, S.S. (Sparsh Sharma); funding acquisition, B.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea [under Grant NRF-2021R1I1A2045721]. The work was also supported by the Woosong University Academic Research Fund in 2023.

**Data Availability Statement:** All datasets utilized in this study are publicly available.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| CNN | Convolutional Neural Network |
| MCC | Mathews Correlation Coefficient |
| MF | Median Filter |
| MLP | Multi Layer perceptron |
| MnF | Mean Filter |
| GLCM | Gray Level Concurrence Matrix |
| GLRLM | Gray Level Run Length Matrix |
| GB | Gaussian Blurring |
| AWGN | Additive White Gaussian Noise |
| RS | Resampling |

## References

1. Piva, A. An Overview on Image Forensics. *ISRN Signal Process.* **2013**, *2013*, 496701. [CrossRef]
2. Stamm, M.C.; Wu, M.; Liu, K.J.R. Information Forensics: An Overview of the First Decade. *IEEE Access* **2013**, *1*, 167–200. [CrossRef]
3. Qureshi, M.A.; Deriche, M. A bibliography of pixel-based blind image forgery detection techniques. *Signal Process. Image Commun.* **2015**, *39*, 46–74. [CrossRef]
4. Farid, H. Digital doctoring: How to tell the real from the fake. *Significance* **2006**, *3*, 162–166. [CrossRef]
5. Kujur, A.; Raza, Z.; Khan, A.A.; Wechtaisong, C. Data Complexity Based Evaluation of the Model Dependence of Brain MRI Images for Classification of Brain Tumor and Alzheimer's Disease. *IEEE Access* **2022**, *10*, 112117–112133. [CrossRef]
6. Khan, A.A.; Madendran, R.K.; Thirunavukkarasu, U.; Faheem, M. D2PAM: Epileptic seizures prediction using adversarial deep dual patch attention mechanism. *CAAI Trans. Intell. Technol.* **2023**, *8*, 755–769. [CrossRef]
7. Zhu, B.B.; Swanson, M.D.; Tewfik, A.H. When seeing isn't believing [multimedia authentication technologies]. *IEEE Signal Process. Mag.* **2004**, *21*, 40–49. [CrossRef]
8. Qiu, X.; Li, H.; Luo, W.; Huang, J. A Universal Image Forensic Strategy Based on Steganalytic Model. In Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security, New York, NY, USA, 11–13 June 2014; MMSec '14, pp. 165–170. [CrossRef]
9. Fridrich, J.; Kodovsky, J. Rich models for steganalysis of digital images,. *IEEE Trans. Inf. Forensics Secur.* **2011**, *7*, 868–882. [CrossRef]
10. Shi, Y.Q.; Sutthiwan, P.; Chen, L. Textural Features for Steganalysis. In *Proceedings of the Information Hiding*; Kirchner, M., Ghosal, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 63–77.
11. Fan, W.; Wang, K.; Cayre, F. General-purpose image forensics using patch likelihood under image statistical models. In Proceedings of the 2015 IEEE International Workshop on Information Forensics and Security (WIFS), Rome, Italy, 16–19 November 2015; pp. 1–6. [CrossRef]
12. Bayar, B.; Stamm, M.C. A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. In Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, New York, NY, USA, 20–22 June 2016; MMSec '16, pp. 5–10. [CrossRef]
13. Mazumdar, A.; Singh, J.; Tomar, Y.S.; Bora, P.K. Universal image manipulation detection using deep siamese convolutional neural network. *arXiv* **2018**, arXiv:1808.06323.
14. Bayar, B.; Stamm, M.C. Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2691–2706. [CrossRef]
15. Chen, Y.; Kang, X.; Shi, Y.Q.; Wang, Z.J. A multi-purpose image forensic method using densely connected convolutional neural networks. *J. Real-Time Image Process.* **2019**, *16*, 725–740. [CrossRef]
16. Yang, L.; Yang, P.; Ni, R.; Zhao, Y. Xception-Based General Forensic Method on Small-Size Images. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*; Pan, J.S., Li, J., Tsai, P.W., Jain, L.C., Eds.; Springer: Singapore, 2020; pp. 361–369.
17. Rana, K.; Singh, G.; Goyal, P. MSRD-CNN: Multi-Scale Residual Deep CNN for General-Purpose Image Manipulation Detection. *IEEE Access* **2022**, *10*, 41267–41275. [CrossRef]
18. Mehta, R.; Kumar, K.; Alhudhaif, A.; Alenezi, F.; Polat, K. An ensemble learning approach for resampling forgery detection using Markov process. *Appl. Soft Comput.* **2023**, *147*, 110734. [CrossRef]
19. Singh, D.; Jain, T.; Gupta, N.; Tolani, B.; Seeja, K.R. Fake Image Detection Using Ensemble Learning. In *Proceedings on International Conference on Data Analytics and Computing*; Yadav, A., Gupta, G., Rana, P., Kim, J.H., Eds.; Springer: Singapore, 2023; pp. 383–393.

20. Yeganeh, A.; Pourpanah, F.; Shadman, A. An ANN-based ensemble model for change point estimation in control charts. *Appl. Soft Comput.* **2021**, *110*, 107604. [CrossRef]
21. Weeraddana, D.; Khoa, N.L.D.; Mahdavi, N. Machine learning based novel ensemble learning framework for electricity operational forecasting. *Electr. Power Syst. Res.* **2021**, *201*, 107477. [CrossRef]
22. Li, X.; Zhang, G.; Huang, H.H.; Wang, Z.; Zheng, W. Performance Analysis of GPU-Based Convolutional Neural Networks. In Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP), Philadelphia, PA, USA, 16–19 August 2016; pp. 67–76. [CrossRef]
23. Marcus, G. Deep learning: A critical appraisal. *arXiv* **2018**, arXiv:1801.00631.
24. Amerini, I.; Anagnostopoulos, A.; Maiano, L.; Celsi, L.R. Deep Learning for Multimedia Forensics. *Found. Trends Comput. Graph. Vis.* **2021**, *12*, 309–457. [CrossRef]
25. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 1–74. [CrossRef]
26. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
27. Lin, Z.; Memisevic, R.; Konda, K. How far can we go without convolution: Improving fully-connected networks, *arXiv* **2015**, arXiv:1511.02580.
28. Touvron, H.; Bojanowski, P.; Caron, M.; Cord, M.; El-Nouby, A.; Grave, E.; Izacard, G.; Joulin, A.; Synnaeve, G.; Verbeek, J.; et al. ResMLP: Feedforward Networks for Image Classification with Data-Efficient Training. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 5314–5321. [CrossRef] [PubMed]
29. Melas-Kyriazi, L. Do You Even Need Attention? A Stack of Feed-Forward Layers Does Surprisingly Well on ImageNet. *arXiv* **2021**, arXiv:2105.02723.
30. Liu, H.; Dai, Z.; So, D.; Le, Q.V. Pay Attention to MLPs. In *Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 9204–9215.
31. Shi, S.; Wang, Q.; Xu, P.; Chu, X. Benchmarking State-of-the-Art Deep Learning Software Tools. In Proceedings of the 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, 16–18 November 2016; pp. 99–104. [CrossRef]
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
33. Zhao, Y.; Wang, G.; Tang, C.; Luo, C.; Zeng, W.; Zha, Z.J. A Battle of Network Structures: An Empirical Study of CNN, Transformer, and MLP. *arXiv* **2021**, arXiv:2108.13002.
34. Ahmed, S.; Islam, S. Median filter detection through streak area analysis. *Digit. Investig.* **2018**, *26*, 100–106. [CrossRef]
35. Haralick, R.M.; Shanmugam, K.; Dinstein, I. Textural Features for Image Classification. *IEEE Trans. Syst. Man Cybern.* **1973**, *SMC-3*, 610–621. [CrossRef]
36. Soh, L.K.; Tsatsoulis, C. Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *IEEE Trans. Geosci. Remote Sens.* **1999**, *37*, 780–795. [CrossRef]
37. Clausi, D.A. An analysis of co-occurrence texture statistics as a function of grey level quantization. *Can. J. Remote Sens.* **2002**, *28*, 45–62. [CrossRef]
38. Galloway, M.M. Texture analysis using gray level run lengths. *Comput. Graph. Image Process.* **1975**, *4*, 172–179. [CrossRef]
39. Castellano, G.; Bonilha, L.; Li, L.; Cendes, F. Texture analysis of medical images. *Clin. Radiol.* **2004**, *59*, 1061–1069. [CrossRef]
40. Tang, X. Texture information in run-length matrices. *IEEE Trans. Image Process.* **1998**, *7*, 1602–1609. [CrossRef]
41. Gallagher, N.; Wise, G. A theoretical analysis of the properties of median filters. *IEEE Trans. Acoust. Speech Signal Process.* **1981**, *29*, 1136–1141. [CrossRef]
42. Chu, A.; Sehgal, C.M.; Greenleaf, J.F. Use of gray value distribution of run lengths for texture analysis. *Pattern Recognit. Lett.* **1990**, *11*, 415–419. [CrossRef]
43. Dasarathy, B.V.; Holder, E.B. Image characterizations based on joint gray level—Run length distributions. *Pattern Recognit. Lett.* **1991**, *12*, 497–502. [CrossRef]
44. Ahmed, S.; Islam, S. Median filtering detection using improved percentage Streak Area. In Proceedings of the Virtual International Research Conference on IoT, Cloud and Data Science, Online, 23–24 April 2021; p. 11.
45. Fei, N.; Gao, Y.; Lu, Z.; Xiang, T. Z-Score Normalization, Hubness, and Few-Shot Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 142–151.
46. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. In Proceedings of the second International Conference on Computational Sciences and Technology, Jamshoro, Pakistan, 17–19 December 2020; pp. 124–133.
47. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; Volume 9, pp. 249–256.
48. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.

49. Saxe, A.M.; McClelland, J.L.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv* **2013**, arXiv:1312.6120.
50. Schaefer, G.; Stich, M. UCID: An uncompressed color image database. In *Electronic Imaging 2004*; SPIE: San Jose, CA, USA, 2003; pp. 472–480. [CrossRef]
51. Bas, P.; Filler, T.; Pevný, T. Break Our Steganographic System": The Ins and Outs of Organizing BOSS. In *Information Hiding*; Filler, T., Pevný, T., Craver, S., Ker, A., Eds.; Berlin/Heidelberg, Germany, 2011; pp. 59–70, ISBN 978-3-642-24177-2. [CrossRef]
52. Dang-Nguyen, D.T.; Pasquini, C.; Conotter, V.; Boato, G. RAISE: A Raw Images Dataset for Digital Image Forensics. In Proceedings of the 6th ACM Multimedia Systems Conference, MMSys 15, New York, NY, USA, 18–20 March 2015; pp. 219–224. [CrossRef]
53. Gloe, T.; Böhme, R. The dresden image database for benchmarking digital image forensics. *J. Digit. Forensic Pract.* **2010**, *3*, 150–159. [CrossRef]
54. Union, I.T. green BT.601: Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios. Status : In force (Main). 2011. Available online: https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en (accessed on 8 March 2011).
55. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [CrossRef]
56. The MathWorks, Inc. MATLAB Version: 9.13.0 (R2021a). 2021. Available online: https://in.mathworks.com/products/new_products/release2021a.html (accessed on 1 November 2023).
57. The MathWorks, Inc. Deep-learning Toolbox: 9.4 (R2021a). 2021. Available online: https://in.mathworks.com/solutions/deep-learning.html (accessed on 1 November 2023).
58. The MathWorks, Inc. Experiment Application (R2021a). 2021. Available online: https://in.mathworks.com/help/deeplearning/manage-experiments (accessed on 1 November 2023).