Computers, Materials & Continua
DOI: 10.32604/cmc.2022.022748
Article

Tech Science Press

An Improved Optimized Model for Invisible Backdoor Attack Creation Using Steganography

Daniyal M. Alghazzawi[1], Osama Bassam J. Rabie[1], Surbhi Bhatia[2] and Syed Hamid Hasan[1,*]

[1]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia
[2]Department of Information Systems, College of Computer Sciences and Information Technology, King Faisal University, Saudi Arabia
*Corresponding Author: Syed Hamid Hasan. Email: shhasan@kau.edu.sa
Received: 17 August 2021; Accepted: 14 December 2021

Abstract: The Deep Neural Networks (DNN) training process is widely affected by backdoor attacks. The backdoor attack is excellent at concealing its identity in the DNN by performing well on regular samples and displaying malicious behavior with data poisoning triggers. The state-of-art backdoor attacks mainly follow a certain assumption that the trigger is sample-agnostic and different poisoned samples use the same trigger. To overcome this problem, in this work we are creating a backdoor attack to check their strength to withstand complex defense strategies, and in order to achieve this objective, we are developing an improved Convolutional Neural Network (ICNN) model optimized using a Gradient-based Optimization (GBO)(ICNN-GBO) algorithm. In the ICNN-GBO model, we are injecting the triggers via a steganography and regularization technique. We are generating triggers using a single-pixel, irregular shape, and different sizes. The performance of the proposed methodology is evaluated using different performance metrics such as Attack success rate, stealthiness, pollution index, anomaly index, entropy index, and functionality. When the CNN-GBO model is trained with the poisoned dataset, it will map the malicious code to the target label. The proposed scheme's effectiveness is verified by the experiments conducted on both the benchmark datasets namely CIDAR-10 and MSCELEB 1M dataset. The results demonstrate that the proposed methodology offers significant defense against the conventional backdoor attack detection frameworks such as STRIP and Neutral cleanse.

Keywords: Convolutional neural network; gradient-based optimization; steganography; backdoor attack; and regularization attack

1 Introduction

Digital communication is the most preferred and promising way of communication made in this smart world. There are various Digital Media available to transmit information from one person to

another via varying types of channels. Moreover, this also increases security issues due to the possibility of copying and transmitting the data illegally [1]. Besides, illegal hackers can transmit the information without any loss and quality. This becomes a daunting issue to the authentication, security, and copyright contents. Hence it is necessary to protect the information during communication. Following this many works have been carried out to safeguard the information. One of the most widely used protection techniques is Cryptography which utilizes some secrecy methods to encrypt and decrypt the information [2–4]. The sender uses ciphertext to protect the messages however this can be easily noticeable. Hence the researchers used several techniques to maintain the secrecy. The secrecy of the information can be sustained by using steganography [5]. The hidden secret data is then extracted by the authorized user at the destination [6].

To conceal all types of digital contents (text, video, image, audio, etc.,) modern image processing techniques use steganography [7]. Using steganography, the last bit of each image is kept hidden and the modification of the last bit of the pixel value would not make any changes in visual perception [8,9].

To embed data using steganography many techniques have been used and all follow their own numerical methods. Therefore, to make an attack is an arduous process. Most of the works conducted are based on the secret key of sizes 8 bit, 16-bit, and 20 bits. Due to the advancement of the technology, those provide a low-security scheme for the applications, and therefore the secret key size of more than 64 bits has been used nowadays. The training of CNN based approach faces some daunting issues due to the Backdoor attacks. This attack sometimes poisons the samples of training images visibly or invisibly. Moreover, this attack can be carried out by following some specified patterns in the poisoned image and replace them with respective pre-determined target labels. Based on this, attackers make invisible backdoors to the trained model and the attacked model behaved naturally. Hence it is arduous to predict the attack by the human. Since the attack is the best form of defense. In this paper, we are presenting an ICNN-GBO model to conduct a backdoor attack via an image towards a specified target. The major contributions in this work are presented as follows:

- The GBO algorithm provides a dual-level optimization to the ICNN architecture by minimizing the attackers' loss function and enhancing the success rate of the backdoor attack.
- Two types of triggers are created in this work and in the first trigger, a steganography-based technique is used to alter the Least Significant Bit (LSB) of the image to enhance its invisibility and the regularization-based trigger is created using an optimization process.
- The effectiveness of the proposed methodology is evaluated with two baseline datasets in terms of different performance metrics such as stealthiness, functionality, attack success rate, anomaly index, and entropy index.

The remainder of the work is organized as follows. Section 2 presents the literature review and the preliminaries involved in this work is presented in Section 3. Section 4 provides the proposed methodology in detail and Section 6 demonstrates the extensive experiments conducted using baseline datasets to evaluate the effectiveness of the proposed technique. Section 6 concludes this article.

2 Literature Survey

The backdoor attack is a trendy and most discussed research area nowadays. This type of attack is the most threatening problem while training the deep learning approaches. The training data are triggered by using backdoor attacks and provide inaccurate detection. There are two types of backdoor attacks (i) visible attack, (ii) invisible attack. Li et al. [10] presented novel covert and scattered triggers

to attack the backdoors. The scattered triggers can easily make fool of both the DNN and humans from the inspection. The triggers are embedded into the DNN with the inclusion of the first model Badnets via a steganography approach. Secondly, the Trojan attacks can be made with the augmented regularization terms which create the triggers. The performances of the attacks are analyzed by the Attack success rate and functionality. Using the definition of perceptual adversarial similarity score (PASS) and learned perceptual image patch similarity (LPIPS) the authors measure the invisibility of human perception. However, the attacks against the robust models are difficult to make.

Li et al. [11] proposed sample-specific invisible additive noises (SSIAN) as backdoor triggers in DNN based image steganography. The triggers were produced by encoding an attack-specified string into benign images via an encoder and decoder network. Li et al. [12] delineated a deep learning-based reverse engineering approach to accomplish the highly effective backdoor attack. The attack is made by injecting malicious payload hence this method is known as neural payload injection (NPI) based backdoor attack. The triggered rate of NPI is almost 93.4% with a 2 ms latency overhead.

The invisible backdoor attack was introduced by Xue et al. [13]. They embedded the backdoor attacks along with facial attributes and the method is known as Backdoor Hidden in Facial features (BHFF). Masks are generated with facial features such as eyebrows and beard and then the backdoors are injected into it and thus provide visual stealthiness. In order to make the backdoors look natural, they introduced the Backdoor Hidden in Facial features naturally (BHFFN) approach which encloses the artificial intelligence and achieved better visual stealthiness and Hash similarity score of about 98.20%.

Barni et al. [14] proposed a novel backdoor attack that triggers only the samples of the target class and without the inclusion of label poisoning. Thus this method ensures the possibility of backdoor attacks without label poisoning. Nguyen et al. [15] presented a novel warping-based trigger. This can be achieved by the presented novel training method known as noise mode. Henceforth the attack is possible in trained networks and the presented attack is a type of visible backdoor attack.

Tian et al. [16] stated a novel poisoning MorphNet attack method to trigger the data in the cloud platform. By using a clean-label backdoor attack, some of the poisoned samples are injected into the training set.

The Sophos identifies a new malicious web page for a time interval of 14 s. Based on the Ponemons 2018 state of endpoint security risk report [17], more than 60% of IT experts conveyed that the frequency of attacks has gone very high from the past 12 months. 52% of participants say that not every attack can be stopped. The antivirus software is only capable of blocking 43% of attacks and 64% of the participants informed that their organizations have faced more than one endpoint attack that results in a data breach. Hence, these problems can be solved only when we aim from the attacker's perspective. In this paper, a novel ICNN-GBO model is proposed to conduct an attack that is concealed within an image and embattled towards a specific target by conducting an attack against a specific target. The attack can be only evident to the specified target for which it was designed. Researchers have focused their attention on stealthy attacks rather than direct attacks because these attacks are hard to identify. In stealthy attacks, the attacker waits for the victim to visit the malicious site with the help of the internet. The attack created is mainly designed to breach the integrity of the system but also provide normal functionality to the users.

The backdoor attack [10,11] is mainly related to the adversarial attack and data poisoning. The backdoor attack avails the higher learning ability of the DNN towards the non-robust features. In the training stage, both the data poisoning and backdoor attacks possess certain similarities. Both attacks control the DNN by injecting poisoning samples. However, the goals of both attacks differ in

different perspectives. In this work, initially, an image steganography technique is used to inject the trigger in a Bit Level Space (BLS), and in the next phase, a regularization technique is used to inject the trigger and make it undetectable by humans. The performance of the proposed model is evaluated using different backdoor attack performance metrics and their invisibility to humans and intrusion detection software.

In the threat model [18], we are developing a backdoor attack for the incorrect image classification task. The backdoor attackers can only poison the training data and cannot alter the inference data since they don't have any additional information regarding the additional settings such as network structure or training schedule.

3 Backdoor Attack Formation

An image agnostic poisoning dataset $\mathrm{E}^{m} = \mathrm{E}^{m}_{train} \cap \mathrm{E}^{m}_{val}$ is created for the trigger m with a one-to-one mapping $o' = f(o, m)$ where o' is marked as the target label k. The poisoning training dataset E^{m}_{train} is used to retrain the ICNN-GBO model b and the E^{m}_{val} dataset is used to estimate the success rate of the backdoor attack. The function f is utilized to apply the trigger in the input which results in the poisoning data point o'.

The GBO algorithm takes the backdoor attack problem as a dual-level optimization problem which is framed in Eq. (1). The initial optimization ($\min_{loss}$) optimizes the attacker's loss function and improves the success rate of the backdoor attack without deteriorating the normal data accuracy. The ICNN-GBO model's retraining optimization is the second objective, in which the model learns the backdoor attack via the poisoning training data.

$$\min_{loss}(\mathrm{E}^{m}_{train}, \mathrm{E}^{m}_{val}, b') = \sum_{x=1}^{n} l(o_x, c_x, b^*) + \sum_{y=1}^{p} l(o_y, k, b^*) \tag{1}$$

$$such\ that\ b^* \in \arg\min_{b} loss(E_{train} \cup (f(o, t), k), b), \tag{2}$$

The untainted validations set size is represented as m, c_x is the training sample, and E^{m}_{train}, E^{m}_{val} is the actual dataset. The poisoning validation set is taken as p. The term $f(o, t)$ becomes an image agnostic when the trigger pattern t is applied to any image o. The loss function $l(o_x, c_x, b^*), (o_x, c_x) \in E$ can be both an appropriate loss or cross-entropy function. The term $l(o_y, k, b^*)$ strengthens the CNN-GBO model b^* to accurately identify both the trigger pattern (t) and target label output (k).

The first objective satisfies the normal user functionality whereas the second objective satisfies the success rate of the attacker in poisoning the data. For the parameters b^* of the backdoor attack classifier, the objective function is based on $f(o, t)$. This implies that the attacker is capable of inserting only a minimal amount of poisoning samples in the training set. In this way, the attacker solves the optimization problem by injecting a set of poisoned data points in the training data.

4 Proposed CNN-GBO Model for Backdoor Attack Execution

In conventional backdoor attacks, the f mapping mainly implies the function that adds the trigger directly to the images. The trigger pattern comes in different shapes and sizes. In our first backdoor attack, we are using steganography to enhance the invisibility by altering the Least Significant bit (LSB) of the image to inject the trigger via poisoning the training dataset. In the regularization-based backdoor attack framework, the triggers are generated via an optimization process and not in any artificial manner. To make the size and shape of the trigger patterns we are using L_p norm

regularization. This process is very similar to the perturbations used in the adversarial examples. The trigger generated by the L_p norm can modify specific neurons. The steganography-based attack is conducted when the adversary chooses a predefined trigger. If the adversary doesn't hesitate to use any size or shape of a trigger, it selects the L_p norm regularization attack. In steganography-based attacks, the triggers are manually created and hidden in the cover images. In the regularization-based attacks, three types of L_p norm are utilized using the values 0, 2, and ∞. In this way, the distribution is scattered and the visibility of the target is reduced. The overall architecture of the proposed framework is presented in Fig. 1.

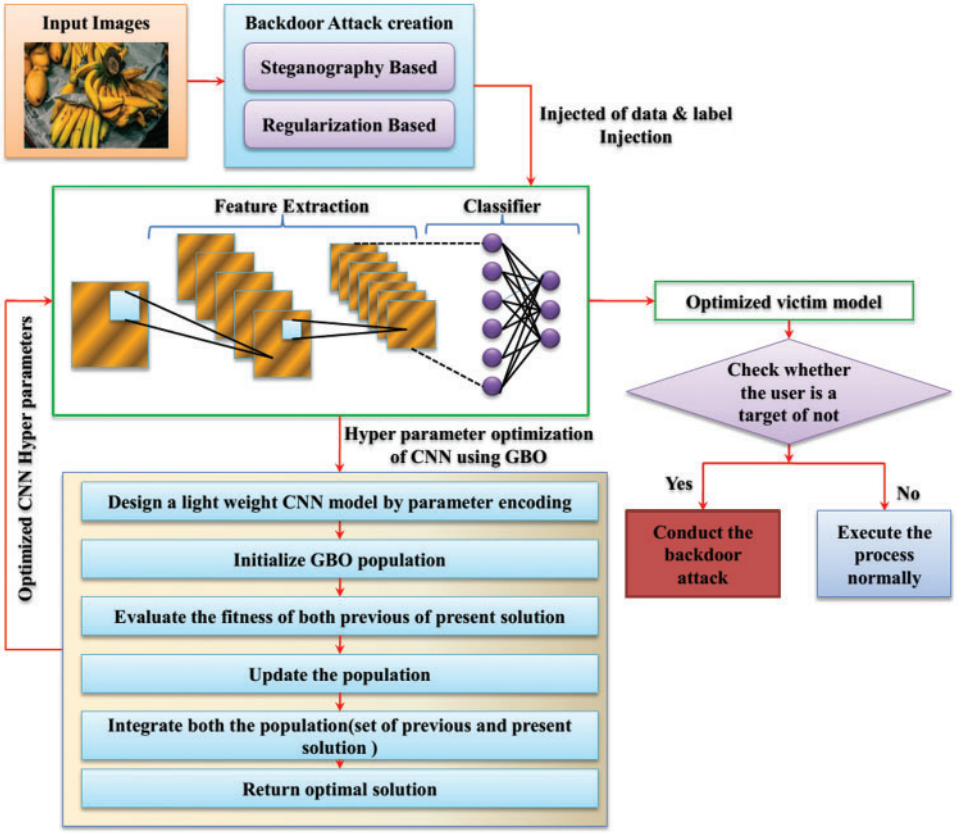

Figure 1: Overall architecture of the proposed framework

4.1 Generating Triggers Using Steganography

The Least Significant Bit is altered in the image to inject the trigger into the images. It is the most easier way to hide the trigger data without any noticeable differences. The LSB of the image is replaced with the data that needs to be hidden. The trigger and cover image are converted into binary in this step. The ASCII code of each character present in the text trigger is transformed into an 8-bit binary string. The LSB of each pixel is altered using the trigger. If the binary secret IMAGE size exceeds A*B*C (channel, weight, and height). In this case, the next right most LSB of the cover image is altered from the initial step to the sequential process to alter every pixel with trigger bits. If the size of the binary trigger is larger than the cover image, the binary trigger length is iterated several times. Hence the trigger size has an increased effect on the attack's success rate.

The DNN finds it hard to identify the trigger when the length of the text is small enough. The trigger features can be easily captured when the size of the trigger is large. Hence a balance needs to be achieved between the invisibility and attack success rate. We are using the text as a trigger to supply sufficient information to inject into the cover image. Tab. 1 provides the trade of achieved between the attack success rate and trigger. When the size of the trigger is increased a lot of bits are altered in the cover images which minimizes the invisibility. However, the increase in the size of the trigger is increased, the DNN easily captures the bit level features improving the attack success rate. The increase in trigger size also reduces the number of epochs to retrain the attack model to learn the bit-level features. For a trigger size of 200, we need a total of 300 epochs to converge. For a trigger size of 600, the model converges with just 11 epochs. Hence the backdoor attack can be easily injected via steganography into the DNN models with the help of the larger triggers.

Table 1: Interrelationship between the attack success rate and invisibility

Trigger size	Invisibility	Attack success rate (%)
200	0.99995	74.6%
300	0.99992	93.6%
400	0.99989	96.8%
500	0.99985	98.6%
600	0.99975	98.8%

A baseline model b is built initially as the target model and the poisoning dataset is created using the LSB algorithm. The source class images which is used as a sample for cover images should be entirely different from the target class. For trigger is injected into the cover images from the source class for poisoning the dataset. Except for the target class, this class is one class of the original training set. The selected images are given a specific target label k. At the end of this step, we get a poisoning image set $(o', k) \in E^{m}_{kpt}$. In the next step, we integrate the actual clean training set with the poisoning dataset to form our new training dataset.

The bit-level features are encoded into the DNN model by retraining the baseline model b via the novel dataset created. After we have created the new backdoor model $b*$, the two validation sets can be built from the initial validation set E_{val}. The E_{val} is mainly used to compute the functionality performance metric. To design a poisoning dataset E^{m}_{val}, we poison every image present in the source class in the actual validation dataset. With the help of the E^{m}_{val}, we are measuring the attack success rate.

4.2 Regularization Based Trigger Generation

Most of the Trojan Backdoor attacks are employed by using triggers that are created. The triggers that are created are more perceived by human eyes than the triggers that are used in BadNets. Our proposed approach follows the optimization process to create the trigger β^{*} by utilizing arbitrary Gaussian noise β_0. While performing the optimization process, the noise is adjusted so that the neuron activations are amplified as $G(\beta)[I]$ by deteriorating the $\mathrm{L_p}$ norm of the noise. Here I is used to denote the positions that are to be amplified. The optimal noise ψ can be accomplished with the achievement of $\mathrm{L_p}$ norm by the optimization process. The threshold $\mathrm{L_p}$ norm is very small and hence it is arduous to predict the noise by humans. Even though the threshold value is set to halt the optimization process,

it also generates the trade-off between the invisibility of the attack and the Attacks' Success rate. If the threshold value is high, it will activate the anchor neuron effectively and it can be easily perceived by the human eyes. Meanwhile, a smaller threshold value means it is arduous to inject the Backdoor into the CNN. For the L_2 attack, we have set as the value ranges from 1 to 10 and for the L_0 attack the value must be 1 to 5. For L_∞ the value ranges from 0.11 to 0.15. The rest of the optimization utilizes this obtained noise as the backdoor trigger. It can be expressed as,

$$\arg\min_{\beta} \psi ||G(\beta)[I] - c * G(\beta_0)[I]||_2 + \eta ||\beta||_p \tag{3}$$

Here, the scale factor is denoted as c and the pre-trained model h can be activated by neurons on the input noise β. ψ, η are the parameters used to weight and also estimates the losses in our method that is loss function. The threshold value of the input noise β can be increased by performing scaling neuron activations. However, the reduction of threshold value leads to arduous to make neuron activation. Hence to overcome these issues we are focus to make the scaling of neuron activation in a particular location in order to access the target values easily. By revising the gradient value of the input noise β will be varied and makes the L_p increase continuously. Secondly, the optimization is used to make the triggers not to predicted by reducing the L_p-norm. To analyze the local optimum, we have adopted the Coordinate Greedy (iterative improvement) in the second attack.

Henceforth, the first term of the loss function can be optimized along with the small value of η and performed neuron activation until the threshold value and beyond. The second term of Eq. (3) is called a regularization term and can be predetermined along with a small value of weight i.e., $\eta = 1$.

4.2.1 Evaluating the Anchor Locations

The next issue is that to set the neuron location I in the networks to be amplified. To achieve the neuron activation, we first want to select the penultimate layer as the target layer. After that, the location of the anchor is selected in the targeted layer. The penultimate layer is in the shape of $[b\iota, N]$ and can be used for multiclass classification. Here, $b\iota$ shows the size of the batch, and N is the hidden units in the penultimate layer. Following this is a fully linked layer with a weight matrix of shape $[N, N_\iota]$ and N_ι is the number of class labels used. The classification output of each class is obtained by utilizing the softmax layer after the fully connected layer. Hence we exploit ResNet-18 as the network architecture and it uses the ReLU activation function to activate the neuron of each residual block. Therefore, with the support of the estimated weight of the fully connected layer, ω the location of the anchor can be evaluated.

$$\log its[t] = \frac{1}{N} \sum G_p * \beta\omega[:, t] + b[t] \tag{4}$$

The activation factor of the penultimate layer is given as G_p; the target label is indicated as t; The t^{th} column vector of the fully connected weight ω can be given as $\omega[:, t]$. Mostly to achieve the efficacy of selecting the anchor locations the descending order of $\omega[:, t]$ will be selected.

4.2.2 Performing Optimization Based on K2 Regularization

The scaling of activation of the anchor locations is performed after the completion of selecting the anchor location. It is conducted via the objective that has been determined in Eq. (3). The performance can be analyzed by deeming the three types of threshold (L_p-norm) regularization i.e., L_2, L_0, and L_∞. The first regularization begins with arbitrary Gaussian noise β_0 and ends with local optimal perturbation β^*.

4.2.3 Performing Optimization Based on K0 Regularization

The optimization based on L_0 regularization on Eq. (3) faces two issues such as (i) selecting the locations that can be utilized for optimization (ii) selecting the number of locations in the image that are to be optimized. The former can be overcome by employing Saliency Map [19]. This map records the priority of each location from the input image and thus circumvents the problem. The later one produces the tradeoff between the efficacy of training the trigger and invisibility and most promptly it ended with the result of predictable by the human eyes.

The Saliency Map can be constructed by using the iterative algorithm and at the end of each iteration, the inactive pixels are identified. Then the pixels are fixed by using the Saliency map which means the values of the fixed pixels remain constant throughout the process. The number of fixed pixels increases with the number of iterations and halt the process when we acquire the required locations for optimization. Also, the detection of a minimal subset of pixels is made to create an optimal trigger by modifying their values. The steps involved in the iterative optimization algorithm are shown in Algorithm 1.

Algorithm 1: Pseudocode for the Saliency Map Generation

Initialize the input Gaussian β_0, Saliency Map Mask {1}, Let the shape be $\omega \times H$,
Target activation value $\iota = c * G_p(\beta_0)[anchor]$
The minimal pixels number is defined as T
The trigger L_2 can be generated with T_0.
Output: Saliency Map mask, Optimal pattern β^*
Begin
For each iteration i do
$\beta = \beta_0$
For $(0 < j < T_0)$ do
$f(\beta) = \iota - G_p(\beta)[anchor]$
$\beta = \beta - lr * mask * \Delta_{\beta} f(\beta)$
End
$\alpha = \beta - \beta_0$
$a = \Delta_{\beta} f(\beta)$
$j = \arg\min_j \alpha_j . a_j$
Mask [j] = 0
$\beta_0 = Bin(\beta) clipping\ the\ value\ to\ [0, 255]$
If $i > (\omega * H - T)$ then break;
end
$\beta^* = \beta$
Return β^*, mask
end

The estimation of loss function f is carried for each iteration among the activation value $G_p(\beta_0)[anchor]$ and its scale target value ι. The gradient obtained from the loss function be considered as α. The mask of the input β can be updated by using the Saliency Map mask which results in $\beta = \beta - lr * mask * \Delta_{\beta} f(\beta)$. The gradient objective function can be estimated as $a = \Delta_{\beta} f(\beta)$. The fixed pixels are given as $j = \arg\min_j \alpha_j . a_j$ from which the i is removed.

4.2.4 Optimization with L_∞

The L_∞norm is a more invisible distance metric when compared with L_0. Here in the L_∞ attack, we replace the term with L_2 term in the objective function. The normalization penalty is derived as shown below:

$$\arg\min_{\beta} \theta||Z(\beta)[\varphi] - s * (\beta_0)[\varphi]||_2 + \kappa||\beta||_\infty \tag{5}$$

Since the term $||\beta||_\infty$ penalizes the largest entry when evaluated with gradient descent often poor results are obtained and to overcome this problem an iterative search is conducted in the L_∞ search space. In Eq. (5), the L_∞ normalization cost is modified when the penalty of any pixels increases the initial value of τ and decreased slowly in each round. The initial value of τ is set as 1. The value of L_∞ is optimally changed as shown in Eq. (6).

$$\arg\min_{\beta} \theta||Z(\beta)[\varphi] - s * (\beta_0)[\varphi]||_2 + \kappa \sum_{j\in\rho} ||[(\beta_j - \tau)^+]|| \tag{6}$$

The position set of the input sample is represented as ρ in Eq. (6) and $a^+ = \max(a, 0)$. In our every experiment the value of β_j is less than τ. In this way, we can directly search the search space to find an optimal value for τ in the real number space. Here the value τ represents the L_∞ normalization of our trigger and every pixel in our trigger is set at a value less than 1.

4.2.5 Universal Backdoor Attack

After the last trigger (β^*) is generated, the poisoning image is constructed by injecting the trigger into the image randomly taken from the original training data set E^m_{train}. The dataset is assigned a sampling ratio η and the target label k is constructed by the attacker. This attack is mainly termed as universal because we are constructing a poisoned image o` by selecting a random image o without giving any importance to the original labels. After the input images are poisoned using the above process, we have a poisoning image set $(o', k) \in E^m_{kpt}$. To form a novel training set ($\mathrm{E}_{train} \cup E_m$) we integrate both the actual and poisoning training set together. The η value is an element of (0, 0.1] to manage the amount of pollution in the poisoning training dataset (E^m_{kpt}) to the overall training set. The new training set created can be exploited to retrain the b^* classifier from the original ICNN-GBO model. The ICNN-GBO model gives high performance after retraining it with a pollution rate (η) of 0.05 for a total number of 5 epochs before convergence. The attack performance is evaluated using two validation sets.

4.3 Improved CNN Classifier Model Implementation

In this proposed work, the ICNN [20] is used to conduct a background attack on the target. During the training phase, the malicious image is injected along with the target label. The output of the network is computed using an error function (softmax). The network parameters are modified by comparing the output with the error rate and the appropriate response obtained. Using the Stochastic Gradient Descent, every parameter is modified to minimize the network error.

4.3.1 Convolution Layer

This layer is used by the CNN layer to convolve the input image. The advantages offered by the convolutional layer are: the parameters are minimized via the weight distribution strategy, the adjacent pixels are trained using the local links, and stability is created when there is any alteration in the position of the object.

4.3.2 Hybrid Pooling Technique

To minimize the network parameters, the pooling layer is placed next to the convolutional layer. In the improved CNN, we are using a hybrid pooling technique. The pooling technique delivers an output P_i for a pooling region A_i, where $i = 1, 2, 3, \ldots, I$. The activation sequence in A_i is represented as $\{s_1, s_2, \ldots, s_{|A_j|}\}$ and the total number of activations is represented using $|A_j|$. By retrieving the outputs of every pooling region, the pooling feature map $\{P = \{P_1, P_2, \ldots, P_I\}\}$ can be derived. For each feature map present in the convolutional layer in the training phase, the average pooling is applied with a probability value ρ and the maximum pooling is applied with a probability $\rho - 1$. This process is applied to every region. When a pooling process is applied for the convolutional feature map, we get the following pooling process:

$$p = \begin{cases} P_{avg} & \rho \\ P_{\max} & \rho - 1 \end{cases} \tag{7}$$

$$P_{avg} = \{P_1^{avg}, \ldots, P_i^{avg}\} \tag{8}$$

$$P_i^{avg} = \left(\frac{1}{|A_i|}\right) \sum_{i \in A_i} s_1 \tag{9}$$

$$P_{\max} = \{P_1^{\max}, \ldots, P_i^{\max}\} \tag{10}$$

$$P_i^{\max} = \max_{i \in R_i} s_i \tag{11}$$

In the testing phase, for each pooling region output obtained and the expected value for each pooling region P is computed using the equation below:

$$P = P_{HYB} \equiv \rho P_{avg} + (1 - \rho) P_{\max} \tag{12}$$

where P_{HYB} represents the hybrid pooling and Fig. 2 demonstrates the hybrid pooling process with a stride and filter size of 2. The existing stochastic pooling algorithms [21,22] were mainly used to select a pooling technique using the probability value. The hybrid pooling technique differs from these techniques in such a way that it can improve the generalization capability using different feature extraction techniques and also averages them in each stage. In each convolutional layer, the pooling process is managed using the parameter ρ. For an instance, if the value of ρ is 0, then the average pooling operation is executer and if the value of ρ is 0, then the max-pooling process is executed. The hybrid pooling process is implemented when the value is $0 < \rho < 1$. To enhance the generalization capability in both the training and testing stages, hybrid pooling is applied using different variants.

For images of 16×16 pixels and 32×32 pixels, three convolutional and two-hybrid pooling layers are implemented. Whereas for the images of 64×64 pixels, 4 convolutions and three hybrid pooling layers are deployed. The stride is increased to enhance the speed of the ICNN process by reducing the stride value. The ICNN input is a color image with a fixed size and the Rectified Linear Unit (ReLU) is deployed in the last layer. The batch normalization layers are added in between the convolutional and ReLU layer to speed up the ICNN training process and also minimize the sensitivity associated with network initialization. Tab. 2 provides the configuration of an image with a size of 32×32 pixels.

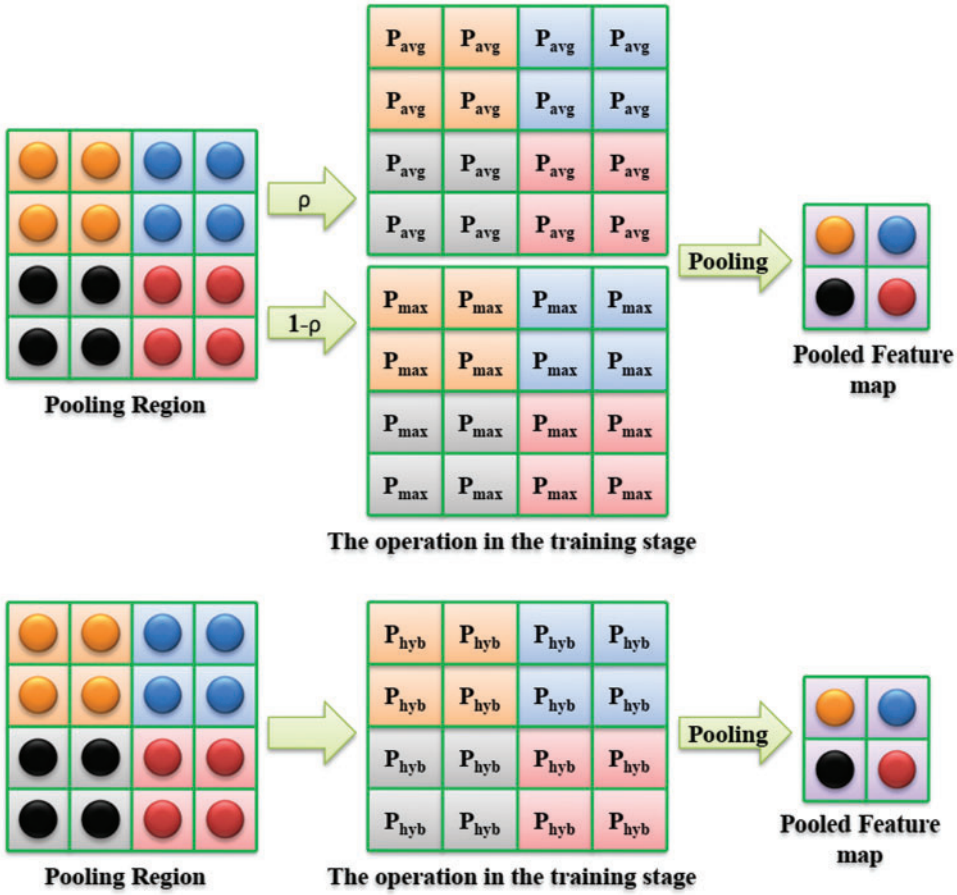

Figure 2: Working of the hybrid pooling methodology

Table 2: ICNN configuration for an image size of 32 × 32

Layer	Name of the layer	Number of filters	Filter size	Filter dimension	Stride	Depth of data	Size of receptive field	Stride of receptive field
0	Input	-	-	-	-	1	3	-
1	Convolution	80	3	3	1	80	4	1
2	Batch normalization	-	1	-	1	80	12	1
3	Hybrid pooling	-	2	-	4	80	12	4
4	Convolution	90	3	80	1	90	16	4
5	Batch normalization	-	1	-	1	90	32	4
6	Hybrid pooling	-	2	-	4	90	32	16
7	Convolution	1000	2	90	1	1000	32	16
8	Batch normalization	-	1	-	1	1000	32	16

(Continued)

Table 2: Continued

Layer	Name of the layer	Number of filters	Filter size	Filter dimension	Stride	Depth of data	Size of receptive field	Stride of receptive field
9	ReLU	-	1	-	1	1000	32	16
10	Fully connected	15	1	1000	1	15	32	16
11	Softmax loss function	-	1	-	1	1	32	16

Using the mean (μ_X) and variance value (σ_X) computed, the batch normalization function normalizes the inputs (a_i) for a mini-batch and for each input channel. The process can be described using the equation below:

$$a_i = \frac{a_i - \mu_x}{\sqrt{\sigma_X^2 + \eta}} \tag{13}$$

In Eq. (13), the variable η is used to enhance the numerical stability when the mini-batch variance value is very minimal. The layers that use batch normalization do not consider the mean and unit variance values whose input is zero as an optimal value. In this scenario, the batch normalization layer shifts and scales the activations as shown below:

$$o_i = \sigma a_i + o \tag{14}$$

During the network training, the offset (o) and scale factor (σ) are set as learnable parameters and are updated only during the training. The network training process is optimized using batch normalization. In this way, we can accommodate a higher learning rate which enhances the speed of the network during training. However, initialization of the weights can be complex and for this process, we are using the GBO algorithm. The GBO algorithm is mainly used to minimize the impact of the huge number of ICNN parameters while training. Because an increased number of parameters and layers complicate the training process which negatively influences the accuracy of the proposed methodology. The GBO algorithm mainly finds the optimal number of hyperparameters to tune the improved CNN model and offers increased accuracy to identify the target accordingly and conduct the attack.

4.4 Gradient-Based Optimization (GBO)

The adopted GBO [23] is exploited to solve the optimization problem in the attack creation to attack the image file of transmission. It follows several rules and is explained in the following sections.

4.4.1 Gradient Search Rule (GSR)

Mostly the GBO is used to prevent from falling for local optima. One of the feature movements of direction (DM) of GSR is used to acquire the valuable speed of convergence. Based on these GSR and DM parameters, the location of attackers can be updated to the vector position as follows:

$$A_i^j = a_i^j - randi \times \vartheta_1 \times \frac{2\Delta a \times a_i^j}{(a_{worst} - a_{best} + \varepsilon)} + rand \times \vartheta_2 \times (a_{best} - a_i^j) \tag{15}$$

$$\vartheta_1 = 2 \times rand \times \alpha - \alpha \tag{16}$$

$$\alpha = \left| \sigma \times \sin\left(\frac{3\Pi}{2} + \sin\left(\sigma \times \frac{3\Pi}{2} \right) \right) \right| \tag{17}$$

$$\sigma = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \times \left(1 - \left(\frac{j}{J} \right)^3 \right)^2 \tag{18}$$

The parameter j denotes the number of iterations; $\sigma_{\min}$, and $\sigma_{\max}$ parameters are predefined values with 0.32 and 1.34 respectively. J is the total number of iterations to be conduct and the parameter ε ranges from 0 to 0.1. *randi* is the i^{th} random number and the ϑ_2 is determined as,

$$\vartheta_2 = 2 \times rand \times \alpha - \alpha \tag{19}$$

$$\Delta a = rand(1 : M) \times |S| \tag{20}$$

$$S = \frac{(a_{best} - a^j_{r1}) + \zeta}{2} \tag{21}$$

$$\zeta = 2 \times rand \times \left| \frac{a^j_{r1} + a^j_{r2} + a^j_{r3} + a^j_{r4}}{4} - a^j_m \right| \tag{22}$$

In Eq. (20), $rand(1 : M)$ is the random number along with M dimensionality. *r1, r2, r3,* and *r4* are the selected random numbers from the (1:M) where $r1 \neq r2 \neq r3 \neq r4 \neq m$. The step size of the attackers can be determined with the aid of S parameters which will be evaluated from the a_{best} and a^j_{r1}. Henceforth, the optimized new vector $A2^j_m$ can be evaluated with the inclusion of a_{best} and a^j_{r1}.

$$A2^j_m = a_{best} - randi \times \vartheta_1 \times \frac{2\Delta a \times a^j_m}{(bx^j_m - by^j_m + \varepsilon)} + rand \times \vartheta_2 \times (a^j_{r1} - a^j_{r2}) \tag{23}$$

$$bx_m = rand \times \left(\frac{[c_{m+1} + a_m]}{2} + rand \times \Delta a \right) \tag{24}$$

$$by_m = rand \times \left(\frac{[c_{m+1} + a_m]}{2} - rand \times \Delta a \right) \tag{25}$$

The best optimal solution of the next iterations can be evaluated with the consideration of $A1^j_m$, $A2^j_m$, and the current location of A^j_m and can be expressed as,

$$A^{j+1}_m = r_a \times (r_b \times A1^j_m + (1 - r_b) \times A2^j_m) + (1 - r_a) \times A3^j_m \tag{26}$$

$$A3^j_m = A^j_m - \vartheta_1 \times (A2^j_m - A1^j_m) \tag{27}$$

4.4.2 Local Escaping Operator (LEO)

The LEO is used to minimize the computational complexity during the estimation of an optimal global solution to generate the attack. The main purpose of this LEO is to prevent falling from local optima and generates enormous optimal solutions along with candidate solution A^j_{LEO} and position

vector A_{best}. Moreover, two random solutions and candidate solutions are also generated namely a^j_{r1}, a^j_{r2}, and $a^j_{k,}$ respectively.

Algorithm 2: Pseudocode for the generation of X^j_{LEO}

If $rand < \Pr ob$
If $rand < 0.4$
$A^j_{LEO} = A^{j+1}_m + d_1 \times (e_1 \times a_{best} - e_2 \times a^j_\kappa) + d_2 \times \vartheta_1 \times (e_3 \times (A2^j_m - A1^j_m) + e_2 \times (a^j_{r1} - a^j_{r2}))/2$
$A^{j+1}_m = A^j_{LEO}$
Else
$A^j_{LEO} = A_{best} + d_1 \times (e_1 \times a_{best} - e_2 \times a^j_\kappa) + d_2 \times \vartheta_1 \times (e_3 \times (A2^j_m - A1^j_m) + e_2 \times (a^j_{r1} - a^j_{r2}))/2$
$A^{j+1}_m = A^j_{LEO}$
End If
end

Algorithm 2 denotes the pseudo-codes for generating A^j_{LEO}. The random number that has been selected randomly between the value -1 and 1 is denoted as d_1. The normal distribution along with the standard deviation and mean are merged between the value *1 and 0* and are denoted as a random number d_2. The probability of occurrence of the local optima of attacks can be denoted by *Prob.* The e_1, e_2, *and* e_3 are the random numbers.

5 Experimental Analysis and Results

This section evaluates the performance of the two types of attacks conducted. For both, the triggers generated we are injecting the triggers inside the images from the CIFAR10 [24] and MSCELEB 1M database [25]. These two datasets are widely deployed for image classification tasks and our experiments are run using an Acer Predator PO3–600 machine with a 64 GB memory. The proposed ICNN-GBO model is implemented using the Matlab programming interface.

CIFAR10 dataset: This dataset comprises a total of 60000 images with a size of 32 32 pixels and each image belongs to the ten classes. We are splitting the training and validation ratio by 9:1 where 50000 images are used for training and 10000 is used for testing.

MS celeb 1M database: It is a large-scale face recognition dataset that comprises 100 K identities and each identity has a total of 100 facial images. The labels of these images are obtained from the web pages.

The definition of the performance metrics such as Attack success rate and functionality is provided below: Attack success rate: The outcome of the poisoned b^* model is taken as $\hat{o} = b^*(a')$ for a poisoned input data a'. The attacker's target is represented as k and the a' rato measures whether it is equal to the target value. This measure helps to find out whether the ICNN-GBO model is capable of identifying the trigger pattern injected in the input images. If the neural network has a higher capability to identify the trigger pattern, then the value of the attack success rate is high.

Functionality: For users other than the target, the functionality score captures the performance of the ICNN-GBO model in the actual validation dataset (E_{val}). The attacker needs to maintain the functionality or else the intrusion software or users will detect the presence of a backdoor attack.

5.1 Comparison in Terms of Different Attacks for Single Target Backdoor Attack

We have compared the performance of our backdoor attack using different triggers as shown in Tab. 3. The single-pixel attack is the most complex one to be identified by our proposed ICNN-GBO model when differentiating the clean and poisoned samples. The results are constructed using an MS celeb 1M database. The single-pixel attack usually takes more number epochs to converge, but the use of the GBO algorithm helps to find the optimal solution rapidly. Our steganography approach needs only a 0.1 pollution rate and a trigger size of 500 for convergence. The results are obtained within the 18th epoch itself. The performance of the backdoor attack can be even improved by increasing the size of the text trigger. The results show that both attacks offer comparable attack success rates when evaluated in terms of attack success rate.

Table 3: Performance evaluation of single target backdoor attacks using different triggers

Attacks	Trigger	Size of trigger	Source label	Target label	Pollution rate	Clean model accuracy (%)	Functionality (%)	Attack success rate (%)
Steganography	Text:" Domino"	500	4	7	0.1	99.8	99.02	99.05
Regularization	Single-pixel	L0 = 1	4	7	0.5	99.8	99.12	98.29
Regularization	Pattern trigger	L0 = 4	4	7	0.1	99.7	99.36	99.16

The effect of the pollution rate (τ) is evaluated in terms of a single target backdoor attack in this section. In a single target attack, the pollution rate is computed based on the samples drawn from a single class in the dataset. The images obtained are then poisoned using the steganography technique. The impact of the pollution is identified using both the CIFAR-10 and MS Celeb 1M dataset and the results obtained are demonstrated in Fig. 3. In both the results, an increase in pollution rate improves the attack success rate while maintaining the functionality range is stable.

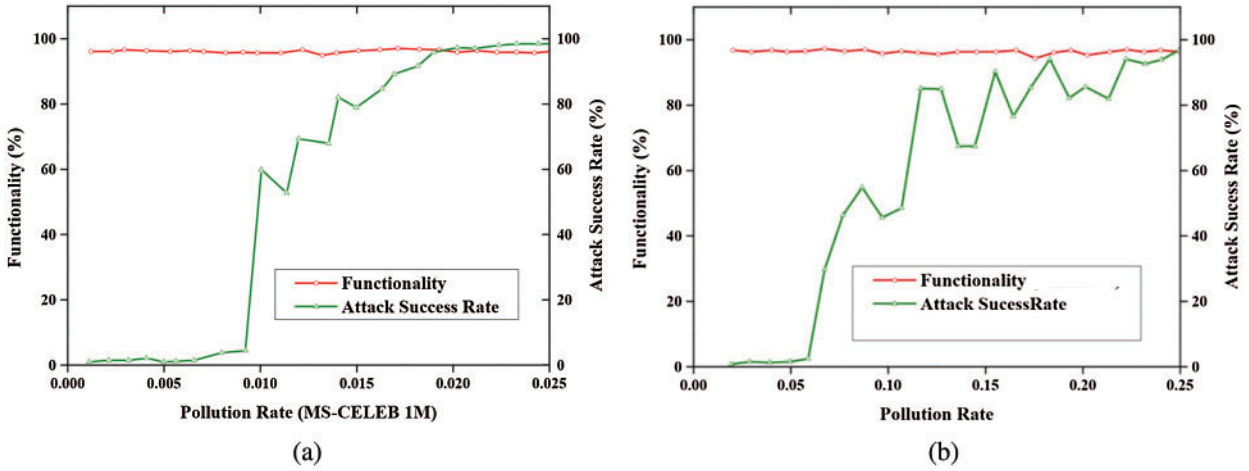

Figure 3: Impact of pollution rate in functionality and attack success rate. (a) Results obtained for the CIFAR-10 dataset and (b) Obtained for the MS Celeb 1M dataset dataset

5.2 Comparison Using Pollution Rate for Universal Attack

In the universal attack, the poisoned samples are selected randomly from the training site whereas, in the single target attack, the poisoned images are selected from only a single source class. The normalization values for the L0, L2, and L_∞ are set as 5, 10, and 0.1. The results shown in Fig. 4 illustrate the performance of the universal attack in terms of pollution rate. As per the results shown in Fig. 4, the increase in pollution rate simultaneously increases the success rate. However, the functionality remains the same. The minimum pollution rate needed to obtain the higher attack success rate differs.

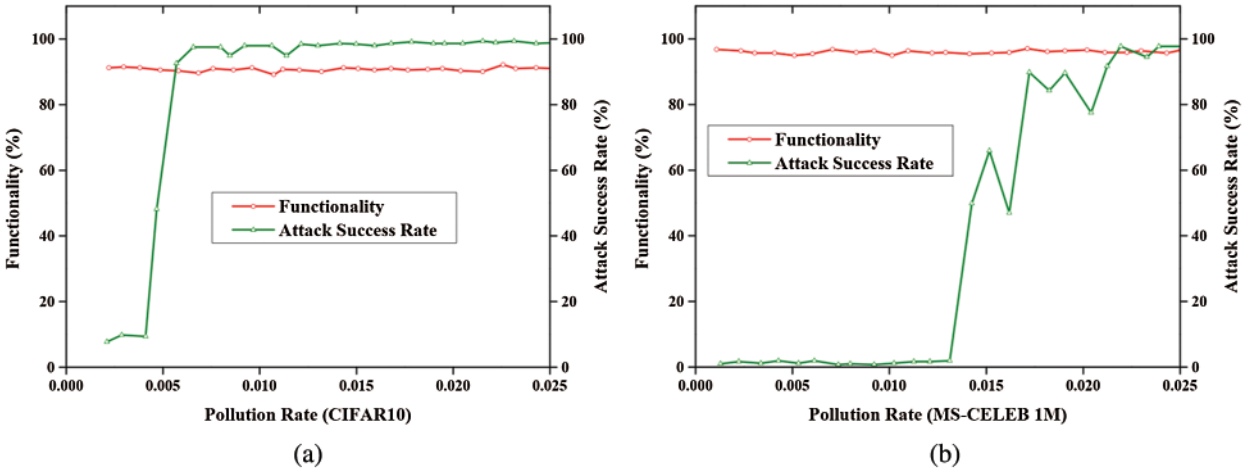

Figure 4: Impact of pollution rate in terms of L_∞ attack. (a) CIFAR10 dataset. (b) MS-CELEB 1M dataset

5.3 Comparison in Terms of Stealthiness

The poison images generated by different attacks are shown in Fig. 5. Our proposed methodology offers higher results in terms of attack stealthiness when compared to BadNets and NNoculation [26]. The poisoned images generated by our technique have no differences that can be identified using the naked eye. The triggers injected by the BadNets and NNoculation are better when hiding their attack stealthiness in white images but the differences are observed in images with dark backgrounds.

The performance of the attack success rate and functionality for both the L0-normalization and L2-normalization is present in Figs. 7 and 8. The results are provided for both CIFAR 10 and MS-CELEB 1M datasets. In the CIFAR-10 dataset, there are fewer perturbations that are complex even for humans hence the performance obtained in terms of functionality and attack success rate is above satisfactory. In the L0-Norm, the attack success rate is enhanced up to 100%. Both the CIFAR10 and MS-CELEB 1M datasets provide an L2-Norm of above 91%. In the L0-Norm the model converges faster than the L2-Norm. This shows that the triggers which use less complex shapes are easily identifiable by the DNN networks. This has been shown in Fig. 6.

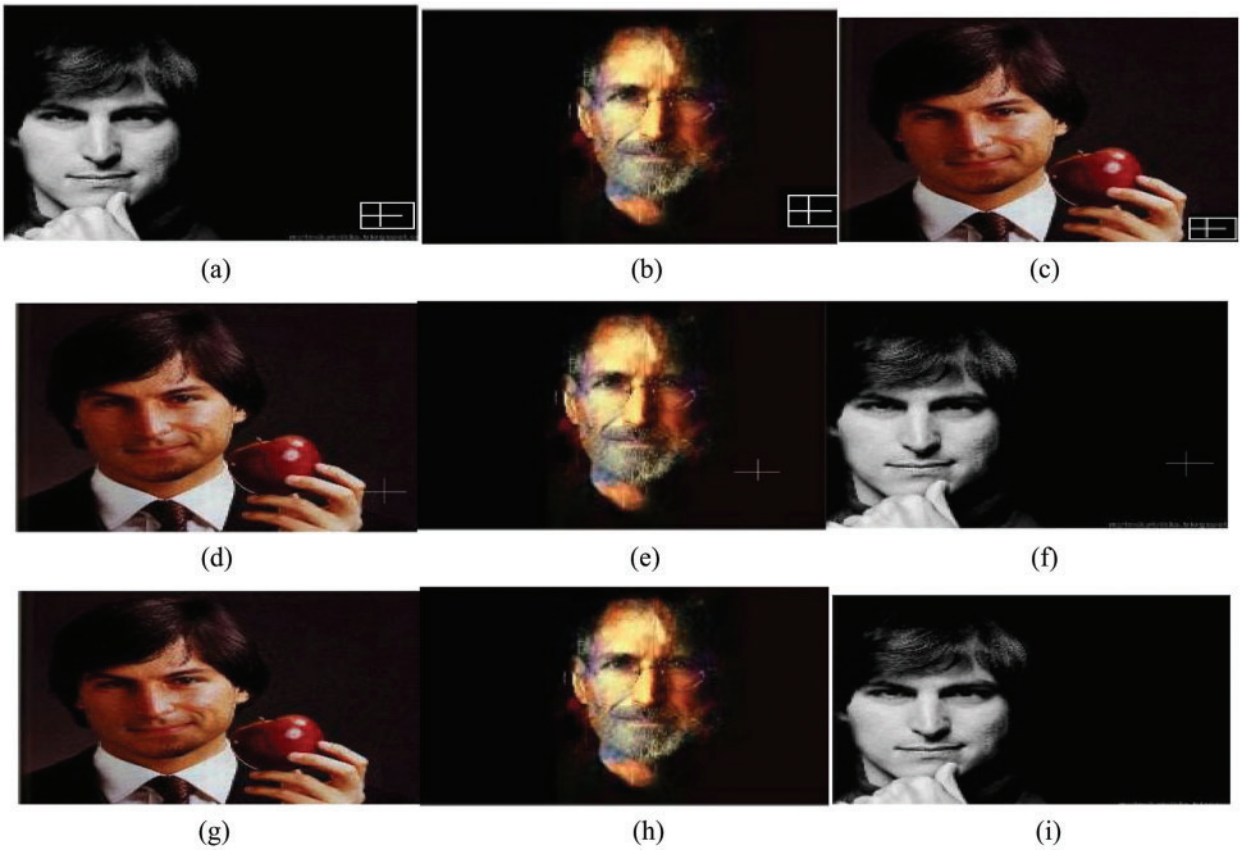

Figure 5: Poison samples generated by different attacks. (a), (b), and (c) represent training samples from the Badnet attack. (d), (e), and (f) represent training samples from the NNoculation attack. (g), (h), and (i) represent training samples from the proposed attack

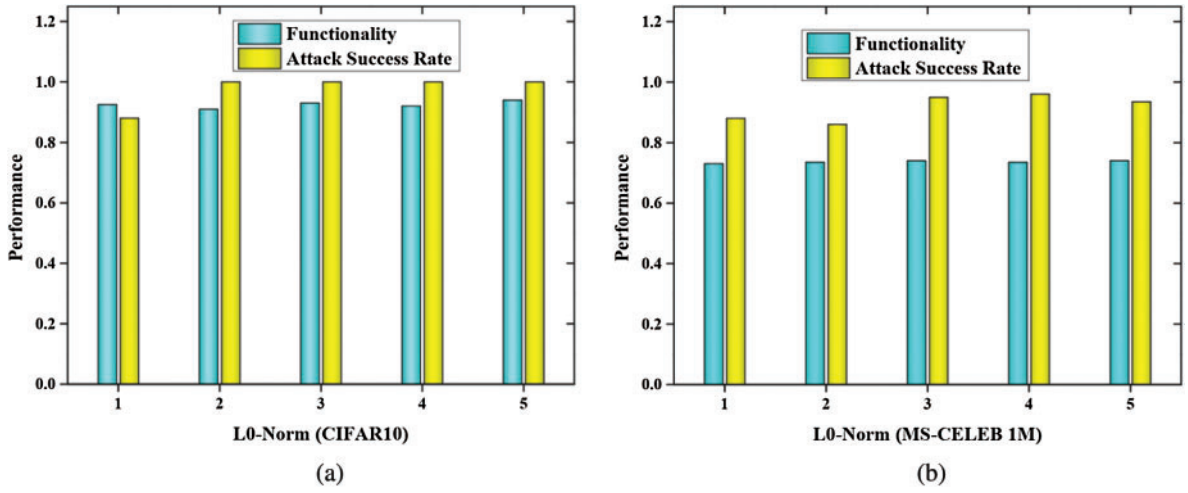

Figure 6: (Continued)

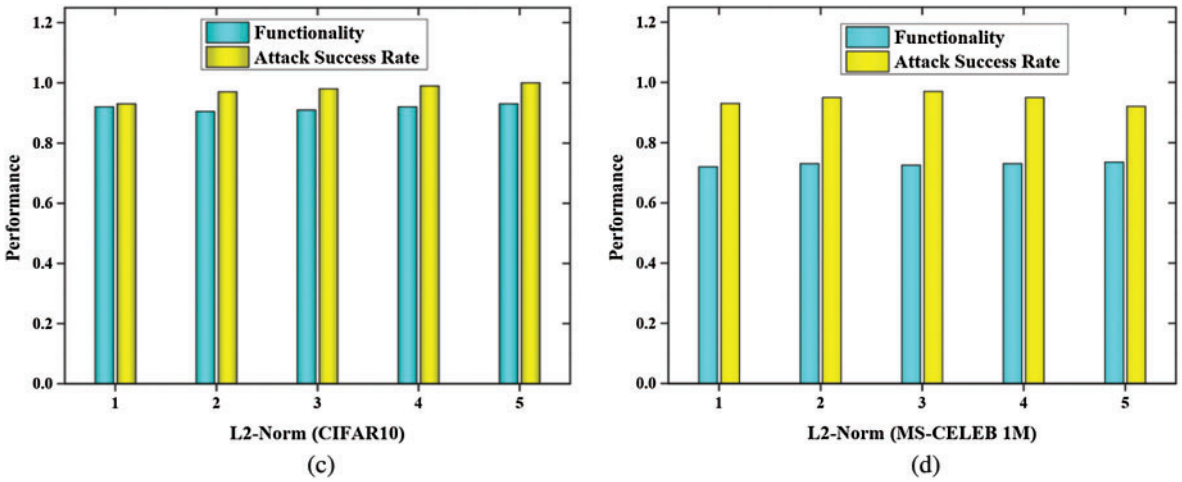

Figure 6: L0 and L1 Norm evaluation in terms of functionality and success rate. (a) Evaluation of CIFAR10 dataset in terms of L0-Norm. (b) Evaluation of MS-CELEB 1M dataset in terms of L0-Norm. (c) Evaluation of CIFAR10 dataset in terms of L2-Norm. (d) Evaluation of MS-CELEB 1M dataset in terms of L2-Norm

Fig. 7 presents the results in terms of anomaly index and the smaller the anomaly index value harder it is recognized by the Neural Cleanse [27]. The neural cleanse calculates the trigger candidates to transform the normal images into their corresponding labels. It then verifies whether any image is smaller than the other image in size and if it finds it then it marks it as a backdoor attack. The proposed methodology is compared with the BadNet, SSIAN, NPI, and BHFFN techniques. From Fig. 7, we can observe that our proposed methodology provides a higher defensive shield in terms of the Neural Cleanse technique.

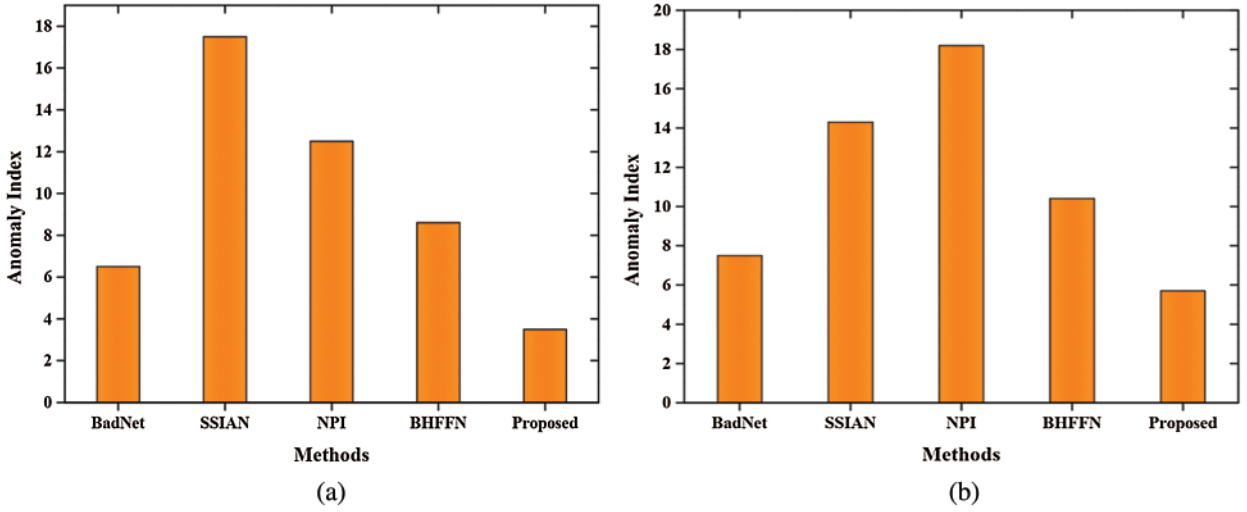

Figure 7: Comparison in terms of anomaly index. (a) CIFAR-10 and (b) MS-CELEB 1M

5.4 Resistance Towards the STRong Intentional Perturbation Based Run-Time Trojan Attack Detection System (STRIP-RTADS)

Based on the random prediction of samples generated by the ICNN-GBO model, the STRIP-RTADS [28] sorts the poisoned instances that possess diverse image patterns. Using the entropy value, the randomness is measured and it is computed by taking an average prediction value of those instances. The higher the entropy value, the harder it is to detect by the STRIP-RTADS framework. As per the results shown in Fig. 8, the proposed method exhibits higher entropy when compared to the BadNet, SSIAN, NPI, and BHFN techniques when evaluated with both datasets.

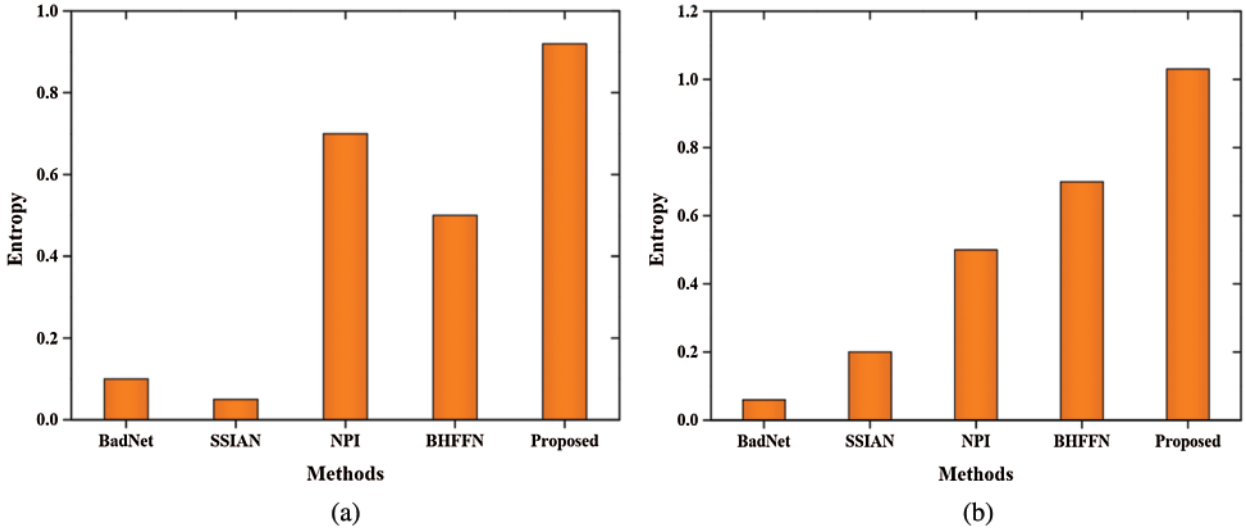

Figure 8: Comparison in terms of entropy index. (a) CIFAR-10 and (b) MS-CELEB 1M

6 Conclusion

Our proposed CNN-based GBO work in this article is used to generate backdoor attacks on the image files. The backdoor attack triggers are generated using steganography and regularization based approaches. The ICNN-GBO model's objective function satisfies the normal user functionality with an increase in attack success rate. The performance of the proposed technique is evaluated in terms of two baseline datasets via different performance metrics such as functionality, Clean Model accuracy, attack success rate, pollution rate, stealthiness, etc. The poisoned image generated by the LSB-based steganography technique and optimized regularization techniques are hardly visible to the naked eye and the intrusion detection software hence it offers optimal performance in terms of stealthiness when compared to the badNet and NNoculation models. The functionality, attack success rate, and clean model accuracy for the proposed scheme are above 99% for the single target backdoor attack. The proposed methodology also offers high resistance when evaluated with the Neural Cleanse and STRIP-RTADS attack detection methodologies.

Funding Statement: This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant No. (RG-91-611-42). The authors, therefore, acknowledge with thanks to DSR technical and financial support.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. Erlangga, A. F. Muchtar, D. Sunarsi, A. S. Widodo, R. Salam *et al.,* "The challenges of organizational communication in the digital era," *Solid State Technology*, vol. 33, no. 4, pp. 1240–1246, 2020.

[2] R. E. Smith, in *Internet Cryptography*, Boston, MA, USA: Addison-Wesley, 1997.

[3] J. Seberry and J. Pieprzyk, in *Cryptography: An Introduction to Computer Security*, NJ, USA: Prentice-Hall, Inc., 1989.

[4] A. K. Saxena, S. K. Sinha and P. K. Shukla, "Design and development of image security technique by using cryptography and steganography: A combine approach," *International Journal of Image, Graphics and Signal Processing*, vol. 10, no. 4, pp. 13–21, 2018.

[5] M. A. Razzaq, R. A. Shaikh, M. A. Baig and A. A. Memon, "Digital image security: Fusion of encryption, steganography and watermarking," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 5, pp. 224–228, 2017.

[6] S. Kaur, S. Bansal and R. K. Bansal, "Steganography and classification of image steganography techniques," in *Proc. 2014 Int. Conf. on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, pp. 870–875, 2014.

[7] M. K. Ramaiya, N. Hemrajani and A. K. Saxena, "Security improvisation in image steganography using DES," in *Proc. 2013 3rd IEEE Int. Advance Computing Conf. (IACC)*, Ghaziabad, India, pp. 1094–1099, 2013.

[8] A. A. Gutub, "Pixel indicator technique for RGB image steganography," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 1, pp. 56–64, 2010.

[9] N. Akhtar, P. Johri and S. Khan, "Enhancing the security and quality of LSB based image steganography," in *Proc. 2013 5th Int. Conf. and Computational Intelligence and Communication Networks*, Mathura, India, pp. 385–390, 2013.

[10] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, X. Zhang *et al.,* "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, early access, vol 18, no. 5, pp. 2088–2105, 2021. http://doi.org/10.1109/TDSC.2020.3021407.

[11] Y. Li, B. Wu, L. Li, R. He, S. Lyu *et al.,* "Backdoor attack with sample-specific triggers," *Cryptography and Security*, arXiv preprint arXiv:2012.03816, 2020.

[12] Y. Li, J. Hua, H. Wang, C. Chen, Y. Liu *et al.,* "DeepPayload: Black-box backdoor attack on deep learning models through neural payload injection," in *Proc. 2021 IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE)*, Madrid, Spain, pp. 263–274, 2021.

[13] M. Xue, C. He, J. Wang and W. Liu, "Backdoors hidden in facial features: A novel invisible backdoor attack against face recognition systems," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1458–1474, 2021.

[14] M. Barni, K. Kallas and B. Tondi, "A new backdoor attack in CNNs by training set corruption without label poisoning," in *Proc. 2019 IEEE Int. Conf. on Image Processing (ICIP)*, Taipei, Taiwan, pp. 101–105, 2019.

[15] T. A. Nguyen and A. T. Tran, "WaNet-imperceptible warping-based backdoor attack," *Cryptography and Security (cs.CR); Computer Vision and Pattern Recognition*, arXiv preprint arXiv:2102.10369, 2021.

[16] G. Tian, W. Jiang, W. Liu and Y. Mu, "Poisoning MorphNet for clean-label backdoor attack to point clouds," *Computer Vision and Pattern Recognition (cs.CV)*, arXiv preprint arXiv:2105.04839, 2021.

[17] "The 2018 State of Endpoint Security Risk," [Online]. Available: https://dsimg.ubm-us.net/envelope/402173/580023/state-of-endpoint-security-2018.pdf. [Accessed: 12-Jul-2021].

[18] T. M. Chen, "Guarding against network intrusions," in *Computer and Information Security Handbook*. Cambridge, MA, USA: Morgan Kaufmann, pp. 149–163, 2013.

[19] R. Mehta, K. Aggarwal, D. Koundal, A. Alhudhaif, K. Polat *et al.,* "Markov features based DTCWS algorithm for online image forgery detection using ensemble classifier in the pandemic," *Expert Systems with Applications*, vol. 185, pp. 115630–115649, 2021.

[20] S. S. Kshatri, D. Singh, B. Narain, S. Bhatia, M. T. Quasim *et al.,* "An empirical analysis of machine learning algorithms for crime prediction using stacked generalization: An ensemble approach," *IEEE Access*, vol. 9, pp. 67488–67500, 2021.

[21] A. Sharaff and K. K. Nagwani, "ML-EC2: An algorithm for multi-label email classification using clustering," *International Journal of Web-Based Learning and Teaching Technologies*, vol. 15, no. 2, pp. 19–33, 2020.

[22] A. Sharaff, A. S. Khaireand and D. Sharma, "Analysing fuzzy based approach for extractive text summarization," in *2019 Int. Conf. on Intelligent Computing and Control Systems*, pp. 906–910, IEEE, Madurai, India, 2019.

[23] M. Premkumar, P. Jangir and R. Sowmya, "MOGBO: A new multiobjective gradient-based optimizer for real-world structural optimization problems," *Knowledge-Based Systems*, vol. 218, pp. 106856, 2021.

[24] S. Bhatia, "New improved technique for initial cluster centers of K means clustering using genetic algorithm," in *Int. Conf. for Convergence for Technology*, pp. 1–4, IEEE, India, 2014.

[25] Y. Guo, L. Zhang, Y. Hu, X. He, J. Gao *et al.,* "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *Proc. European Conf. on Computer Vision*, Amsterdam, The Netherlands, pp. 87–102, 2016.

[26] A. K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami *et al.,* "NNoculation: Broad spectrum and targeted treatment of backdoored DNNs," *Cryptography and Security Machine Learning*, arXiv preprint arXiv:2002.08313, 2020. http://doi.org/10.1145/3474369.3486874.

[27] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath *et al.,* "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. 2019 IEEE Symp. on Security and Privacy (SP)*, San Francisco, CA, USA, pp. 707–723, 2019.

[28] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe *et al.,* "Strip: A defence against trojan attacks on deep neural networks," in *Proc. 35th Annual Computer Security Applications Conf.*, San Juan, PR, USA, pp. 113–125, 2019.