M-to-N Backdoor Paradigm: A Stealthy and Fuzzy Attack to Deep Learning Models

Linshan Hou, Zhongyun Hua, Yuhong Li, Leo Yu Zhang

Abstract—Recent studies show that deep neural networks (DNNs) are vulnerable to backdoor attacks. A backdoor DNN model behaves normally with clean inputs, whereas outputs attacker's expected behaviors when the inputs contain a pre-defined pattern called a trigger. However, in some tasks, the attacker cannot know the exact target that shows his/her expected behavior, because the task may contain a large number of classes and the attacker does not have full access to know the semantic details of these classes. Thus, the attacker is willing to attack multiple suspected targets to achieve his/her purpose. In light of this, in this paper, we propose the M-to-N backdoor attack, a new attack paradigm that allows an attacker to launch a fuzzy attack by simultaneously attacking \( N \) suspected targets, and each of the \( N \) targets can be activated by any one of its \( M \) triggers. To achieve a better stealthiness, we randomly select \( M \) clean images from the training dataset as our triggers for each target. Since the triggers used in our attack have the same distribution as the clean images, the inputs poisoned by the triggers are difficult to be detected by the input-based defenses, and the backdoor models trained on the poisoned training dataset are also difficult to be detected by the model-based defenses. Thus, our attack is stealthier and has a higher probability of achieving the attack purpose by attacking multiple suspected targets simultaneously in contrast to prior backdoor attacks. Extensive experiments show that our attack is effective against different datasets with various models and achieves high attack success rates (e.g., 99.43% for attacking 2 targets and 98.23% for attacking 4 targets on the CIFAR-10 dataset) when poisoning only an extremely small portion of the training dataset (e.g., less than 2%). Besides, it is robust to pre-processing operations and can resist state-of-the-art defenses.

Index Terms—Backdoor attack, Deep neural networks, Poisoning attack, Trojan attack, Fuzzy backdoor attack.

1 INTRODUCTION

Deep neural networks (DNNs) have been widely applied to many areas [1], [2]. Nowadays, users tend to develop deeper and larger neural networks to achieve state-of-the-art performance across different tasks. However, training such a DNN model generally requires a large number of labeled samples and high computation cost, which makes it difficult for many users to train their own models by themselves [3]. Thus, users often use third-party datasets to train their models, outsource the training process to some third-party servers (e.g., Amazon SageMaker [4]), or directly employ commercial APIs for their tasks, which provides attackers with an opportunity to embed some hidden functionalities into DNN models [4]–[9]. Among these hidden functionalities, backdoor attack is a serious threat to DNN models [4], [6], [9]. In a backdoor attack, the attacker aims to inject a specific neuron activation pattern into a DNN model to manipulate its behaviors maliciously. The backdoor model behaves normally with clean inputs; however, it outputs the attacker-specified result(s) with the attacker-crafted poisoned inputs.

The first backdoor attack BadNets [3] was proposed in 2017. It poisons the training dataset by patching a white square trigger onto a small portion of clean images and replacing the labels of the poisoned images with a pre-defined target label. Then the models trained on the poisoned training dataset will be injected with a backdoor. However, the used trigger is static and visible, which makes the poisoned inputs and backdoor models easy to be detected by many subsequent defenses [10]–[13]. Later, researchers developed many backdoor attacks on DNN models, and these backdoor attacks can be classified as deployment-stage attacks [14]–[17] and production-stage (or training-stage) attacks [5], [18]–[24], according to the backdoor injection stage in the life cycles of DNN models. A deployment-stage attack directly modifies the parameters of a deployed DNN model to inject a backdoor, while a production-stage attack poisons a part of the clean samples of the training dataset and injects a backdoor into a DNN model during the training process. Since it is difficult for attackers to access a trained model in a real-world scenario, most backdoor attacks inject backdoors into DNN models in the production stage. The production-stage attacks aim at generating invisible [19]–[21] and dynamic triggers [22]–[24] to make the poisoned inputs and backdoor models more stealthy from human being’s observation and backdoor defenses. And our work falls into the same class.

The prior backdoor attacks mainly focus on attacking one target, which means that an attacker would like the backdoor model to misclassify the poisoned inputs with a specific trigger to a fixed class. This requires the attacker to know which target class should be attacked. However, in many scenarios, attackers do not have enough knowledge about the task and thus cannot determine the exact attack target. For example, an attacker would like to get
Poisoned Images
Multi-Trigger Poisoning
Clean Image
"Stop"
Multi-Target
Clean Model
"Stop"

Fig. 1. An example of our M-to-N backdoor attack about traffic sign classification. An input sign “Stop” can be misclassified as the target label “No Entry”, “Speed Limit”, “Keep Right”, or “Straight Ahead” when poisoning the sign with a trigger that corresponds to the target label. Note that the backdoor of each target can be activated by any one of its $M$ triggers.

and the training process cannot introduce external features to the DNN model. As a result, the poisoned inputs and backdoor models are challenging to detect. The experimental results verify its effectiveness, robustness, and ability to resist current defenses. Fig. 1 depicts an example of our M-to-N backdoor attack on traffic sign classification. Table 1 compares the properties of our attack with those of existing state-of-the-art backdoor attacks.

Our main contributions are summarized as follows:

- We design a new backdoor attack paradigm, which allows an attacker to launch a fuzzy backdoor attack by attacking multiple suspected targets simultaneously. The backdoor of each target can be activated by any one of the $M$ triggers. To the best of our knowledge, this is the first time to propose such a multi-target and multi-trigger attack paradigm.
- We propose a new trigger selection strategy that randomly selects $M$ clean images corresponding to a target label as its $M$ triggers. Since the triggers have the same distribution as the clean images of the training dataset, they do not introduce external features to the poisoned inputs and backdoor models, making them difficult to detect.
- Extensive experiments are provided, and the results show that our attack is effective for simultaneously attacking multiple targets using any one of the multiple triggers and has high robustness to resist preprocessing operations and high ability to resist existing defenses.

The rest of this paper is organized as follows. Section 2 reviews existing backdoor attacks and backdoor defenses. Section 3 presents our threat model and problem formulation. Section 4 introduces our M-to-N backdoor attack in detail. Section 5 shows the experimental results and Section 6 validates the ability of our attack to resist existing defenses. Section 7 concludes this paper.

2 RELATED WORK

2.1 Backdoor Attacks

A backdoor attacker may create a backdoor in a DNN model at different phases of a DNN model's life cycle [30], and thus all the backdoor attacks can be classified as two types: deployment-stage attack [14]–[17] and production-stage (or training-stage) attack [5]–[8], [24].

2.1.1 Deployment-Stage Attacks

A deployment-stage attack creates a backdoor in a deployed DNN model by directly modifying the weight parameters. Adnan et al. [14] proposed the first deployment-stage attack called the Targeted Bit Trojan, which flips a part of the bits of the weight parameters stored in the main memory to inject a backdoor. Similar works [15], [16] were also proposed to inject such a backdoor by flipping fewer bits. These attacks have a strong white-box assumption that the attackers have full access to the gradient information of DNN models. The authors in [17] proposed a subnet replacement strategy to directly replace a specific part of a clean model with a subnet embedded with a backdoor. An attacker needs to access the structure information of the target model to generate the
backdoor subnet, which means that the attack occurs in a gray-box setting. As a result, the deployment-stage attacks require the attacker to know some internal details of a DNN model and have writing permission on the deployment devices. However, these requirements cannot be attained in many real-world scenarios.

### 2.1.2 Production-Stage Attacks

Most backdoor attacks inject a backdoor into a DNN model during the production phase by poisoning a part of the dataset with pre-defined triggers. The key design of these backdoor attacks is to embed the triggers into clean images stealthily; thus, the poisoned images are indistinguishable from clean images while maintaining a high attack success rate.

Gu et al. [5] proposed the first production-stage attack, which poisons a small part of the dataset using a trigger (e.g., a white square), and all the poisoned images share the same trigger to achieve a high attack success rate. In 2018, Liu et al. [18] proposed a new backdoor attack paradigm by generating the trigger through reverse-engineering the pre-trained target model, and the reversed trigger can amplify the activation of the important neurons. To make the poisoned images more stealthy, some backdoor attacks generate the poisoned images by blending clean images with another image [19] or amplitude [20], rather than directly adding the visible triggers to the clean images. However, the triggers in the above backdoor attacks are static and fused with the clean images in a naive manner [22], [23], [31], which can be exploited by defenders [10]–[12].

In contrast, recent backdoor attacks aim to achieve a more stealthy attack by optimizing the triggers or the poisoned image generation method. The backdoor attack in [25] generates poisoned images by involving some clean features of different labels, aiming to mislead the defenders into identifying the backdoor models as clean models. The WaNet backdoor attack in [24] uses a small smooth warping field to generate poisoned images, whereas ISSBA in [22] and Input-Aware in [25] use generative networks to generate sample-specific noises as their dynamic triggers. Besides, the backdoor attack in [21] uses some specific words as the trigger and embeds the trigger into the clean images by employing the steganography technique [32]. Nowadays, developing new backdoor attacks with optimized triggers and poisoned image generation techniques has become a trend to avoid detection [22]–[24], [31].

For an effective backdoor attack, the backdoor model should use the same features of triggers during the training and testing processes [31]. However, the features of the triggers hidden in the testing inputs can be changed when the testing inputs are processed using some pre-processing operations such as image rotation and external noise, leading to the failure of activating the backdoors [26]–[28]. Thus, a robust backdoor attack should retain a high attack success rate even if the poisoned testing inputs are processed by some pre-processing operations.

### 2.2 Backdoor Defenses

Since backdoor attacks present a serious security risk to DNN models, many backdoor defenses have been developed to detect and mitigate backdoor attacks. They can be roughly classified as model-based defenses [10], [13], [30], [33]–[35] and input-based defenses [11], [12], [26], [30], [37]–[40].

#### 2.2.1 Model-Based Defenses

The model-based defenses aim to detect and mitigate possible backdoors in a suspicious model. Liu et al. [13] proposed a fine-pruning strategy to prune the trigger-related neurons of the suspicious models by assessing the activation of specified layers. In addition to the pruning techniques, the knowledge distillation [41] and mode connectivity [42] techniques have also been used to detect the possible backdoors [30], [35]. The above backdoor defenses detect and mitigate the possible attacks without knowing the backdoor trigger. As a contrast, the authors in [10] proposed the Neural Cleanse that can initially reverse-engineer the trigger of the target label and then eliminate the hidden backdoors by unlearning the relationship between the trigger and its target label. Subsequently, some other similar backdoor defenses [33]–[35] were proposed by designing different trigger reversing methods or detection techniques.
2.2.2 Input-Based Defenses

The input-based defenses aim to prevent backdoor attacks by recognizing the poisoned inputs during either the training or testing process. Tran et al. [37] proposed the first input-based backdoor defense by detecting the poisoned images in the training process. It distinguishes the poisoned images from the clean images using the spectrum of the covariance, and the authors in [38] improved it by applying a robust covariance estimation method. Recently, the saliency map has become a popular tool for analyzing the triggers’ effects on the behavior of DNN models [22–24]. For example, SentiNet [12] and Februus [43] use the saliency map generated by Grad-CAM [44] to locate the potential regions that may contain triggers. These defenses rely on the full access to the model training process.

Some other input-based defenses aim to detect the poisoned inputs during the testing process. Gao et al. [11] proposed an input-based defense called STRIP that can filter the poisoned inputs from the clean inputs. It is based on the insight that the predictions of perturbed clean inputs are random, whereas the predictions of poisoned inputs are fixed. Similarly, some other techniques, such as model uncertainty [39] and privacy-based outlier detection [40], are also used to distinguish the poisoned inputs from clean inputs. Only the inputs certified as clean inputs can be fed into the models, preventing the triggers in the poisoned inputs from activating the backdoors.

3 Threat Model and Problem Formulation

This section first presents the threat model that defines the attacker’s goals and capabilities and then provides a formalization for our M-to-N backdoor attack.

3.1 Threat Model

3.1.1 Attacker’s Goals

Our attack considers a realistic threat scenario for the image classification task. The attacker expects to attain the following goals.

Effectiveness. The backdoor attack should have a high attack success rate, which means that it can predict the poisoned inputs as the target label(s), and it should not significantly affect the prediction performance of the clean inputs. Besides, the attack should generalize to different DNN models.

Stealthiness. The triggers in the backdoor attack should be well fused with the clean images such that the poisoned images are visually similar to their clean counterparts. As a result, the backdoor defenses cannot distinguish the poisoned images from the clean images during the training and testing processes. Since it is easy to attract defenders’ attention if poisoning too many samples of the training dataset, the attack should ensure the success of backdoor injection by poisoning only a small part of the training dataset.

Multiple targets. The attack can threaten multiple targets simultaneously, which means that the backdoor model can output attacker’s different expected behaviors when the triggers in the poisoned inputs belong to different labels. Compared with prior backdoor attacks that can attack only one target, the attack paradigm is more flexible and effective when the attacker does not know the exact target label that shows his/her expected behavior.

Multiple triggers. The backdoor of each target can be produced by multiple triggers, and any one of the triggers can activate the backdoor. This makes it much harder for backdoor defenders to detect and mitigate all the trigger-related information.

3.1.2 Attacker’s Capabilities

We assume that attackers have no knowledge about the models’ detailed settings, such as the network structure, loss function, and optimizer. However, attackers have full access to the training datasets, which is the common assumption for the attackers in the backdoor attacks [5, 20–22, 29].

3.2 Problem Formalization

We aim to poison a part of the training dataset by fusing triggers into some clean images so that the backdoors can be injected into the DNN models trained on the poisoned training dataset. Table 2 lists the important notations used in this paper. Suppose that a dataset $D = \{(x_i, y_i) \mid i = 1, \cdots, N(D)\}$ has $N(D)$ i.i.d. clean samples and is comprised of the training dataset $D_{train}$ and the testing dataset $D_{test}$. The $i$-th clean sample comprises the clean image $x_i \in X = \{0, 255\}^{w \times h}$ and its ground truth label $y_i \in Y = \{1, \cdots, C\}$. A poisoned image $\tilde{x}_i$ can be generated from the clean image $x_i$ using a poisoned image generation function. Thus, a poisoned sample $(\tilde{x}_i, \tilde{y}_i)$ can be generated by mapping the poisoned image $\tilde{x}_i$ to the target label $\tilde{y}_i$. Attackers can obtain a poisoned training dataset $D_{\text{train}}$ by using the poisoned samples to replace the corresponding clean samples in $D_{\text{train}}$, namely $D_{\text{train}} = D_b \cup D_s$, where $D_b$ is the set of poisoned samples and $D_s$ is the set of remaining clean samples in $D_{\text{train}}$. The poisoning ratio, i.e., $\rho = |D_b|/|D_{\text{train}}|$, is the poisoning ratio within the poisoned training dataset.

Prior backdoor attacks can threaten only one target, and the backdoor can be activated by a fixed trigger. Our M-to-N backdoor attack can attack $N$ targets simultaneously, and the backdoor of each target can be activated by any
one of M triggers. We assume that the N target labels to be attacked are \( L = \{ l_k | l_k \in Y, \; k = 1, \ldots, N \} \). Since each target relates to M triggers, we present the \( M \times N \) triggers as \( T = \{ t_{ij}^{(k)} | k = 1, \ldots, N; j = 1, \ldots, M \} \), where \( t_{i1}^{(k)}, t_{i2}^{(k)}, \ldots, t_{iM}^{(k)} \) are the M triggers of the k-th attacked target and they are randomly selected from the clean images corresponding to the target in \( D_{\text{train}} \).

A poisoned sample for the k-th attacked target is generated as follows. (1) Select a trigger \( t_{ij}^{(k)} \) from the M triggers of the k-th attacked target and generate a poisoned image \( \tilde{x}_i \) by embedding \( t_{ij}^{(k)} \) into a clean image \( x_i \) that does not correspond to the N target labels, namely \( x_i \in D_{\text{train}} \backslash D_L \). The poisoned image generation can be presented as \( \tilde{x}_i = G(x_i, t_{ij}^{(k)}) \). (2) Mark the poisoned image \( \tilde{x}_i \) as the target label \( \tilde{y}_i \) and obtain a poisoned sample \( (\tilde{x}_i, \tilde{y}_i) \) for the k-th attacked target label \( l_k \). Since there are M triggers \( t_{i1}^{(k)}, t_{i2}^{(k)}, \ldots, t_{iM}^{(k)} \) corresponding to the target label \( l_k \), we can generate the poisoned samples to attack the k-th target by uniformly using these M triggers. Similarly, the poisoned samples for all the N attacked targets can be generated. All the poisoned samples form the poisoned samples set \( D_b \).

The behavior of the backdoor model \( f_b(\cdot) \) can be formulated as:

\[
f_b(x) = f_0(x) = y \land f_0(\tilde{x}) = \tilde{y} \land f_b(\tilde{x}) = \tilde{y},
\]

where \( f_0(\cdot) \) is the clean model, \((x, y) \in D_b\) is a benign sample and \((\tilde{x}, \tilde{y}) \in D_b \) is a poisoned sample. For each clean image \( x \), both the clean model \( f_0(\cdot) \) and backdoor model \( f_b(\cdot) \) can output the true label \( y \). For the poisoned image \( \tilde{x}_i \), the clean model \( f_0(\cdot) \) outputs the real label \( y_i \); however, the backdoor model \( f_b(\cdot) \) outputs the target label \( \tilde{y}_i \), which exhibits the attacker’s expected behavior.

4 M-to-N BACKDOOR ATTACK

In this section, we introduce our M-to-N backdoor attack in detail. We first present the overview design of our attack and then provide the poisoned image generation and backdoor injection processes.

4.1 Overview

Prior backdoor attacks are designed to attack one target using a fixed trigger pattern or mechanism, and show inferior performance if the attacker does not know the exact target label corresponding to his/her expected behavior. In light of this, we design the M-to-N backdoor attack that allows an attacker to attack multiple suspected targets, which can greatly help to achieve the attack purpose.

We assume that an attacker would like to inject backdoors for \( N \) suspected targets. For each attacked target, the attacker first selects \( M \) clean images that correspond to the target label as the \( M \) triggers of the target and then develops a poisoned image generation framework by using all the \( M \times N \) triggers to poison a part of the training dataset equally. A DNN model is injected backdoors by training it on the poisoned training dataset. The backdoor model can output any attacker’s expected target label when the input is poisoned by a corresponding trigger.

4.2 Poisoned Image Generation

We first generate poisoned images for each attacked target by poisoning the clean images using different triggers. Our attack selects \( M \) clean images corresponding to each target in the training dataset as the \( M \) triggers. This selection of triggers can achieve the following advantages.

- The triggers have the same distribution as the clean images in the training images, as well as the backdoor triggers, according to the discussions in [25]. Thus, the triggers can ensure the effectiveness of our attack against multiple targets.
- The backdoor models are difficult to be detected by the model-based defenses such as [10, 12, 13]. This is because the triggers cannot bring extra features that are nonexistent in the training dataset to the models during the training process.

We design a poisoned image generation framework to embed our triggers into other clean images. Its structure is shown in Fig. 4, which is developed according to the DNN-based watermarking in [45]. As can be seen, the generation framework comprises three networks: trigger embedding network \( \mathcal{H} \), recovery network \( \mathcal{R} \), and discriminator \( \mathcal{D} \). \( \mathcal{H} \) is an encoder-decoder network that embeds a trigger into a clean image, and the encoded result is directly used as our poisoned image. To ensure embedding quality, the poisoned image is then separately sent into \( \mathcal{R} \) and \( \mathcal{D} \). A trigger is considered to be successfully injected into a poisoned image if it can be extracted from the poisoned image. Thus, we use \( \mathcal{R} \) to extract the trigger features from the poisoned image. \( \mathcal{D} \) is used to identify that an input is a clean image or a poisoned image, and we use it to assist \( \mathcal{H} \) in generating high-quality poisoned images that are indistinguishable from clean images.

We train the generation framework on the training dataset \( D_{\text{train}} \). Specifically, we first randomly select a trigger \( t_{ij}^{(k)} \) from the trigger set \( T \) and a clean image \( x_i \) from the training dataset \( D_{\text{train}} \) as the inputs of the generation framework. We then concatenate the clean image \( x_i \) with the grayscale version \( t_{ij}^{(k)} \) of the trigger along the channel dimension, and feed the composite image into the trigger embedding network \( \mathcal{H} \). The output of \( \mathcal{H} \) is a poisoned image \( \tilde{x}_i \) and assigned with the target label \( l_k \), i.e.,

\[
\tilde{x}_i = \mathcal{H}(x_i, t_{ij}^{(k)}), \quad \tilde{y}_i = l_k,
\]

where \( x_i \in D_{\text{train}} \) and \( l_k \in L \).

4.2.1 Network Structures

We use UNet [46] as the network structure of \( \mathcal{H} \), because it performs well in the situations that the inputs and outputs share some common properties. We modified the stride of each convolution layer from 2 to 1 to apply to images of small size (e.g., \( 32 \times 32 \)). Besides, we use the widely-used CEILNet [47] and PatchGAN [47] as the network structures of \( \mathcal{R} \) and \( \mathcal{D} \), respectively.
4.2.2 Loss Functions

The overall loss function of the poisoned image generation framework comprises the losses $L_H$, $L_R$ and $L_D$ of the above three networks, namely

$$L = \lambda_H L_H + \lambda_R L_R + \lambda_D L_D,$$

where $\lambda_H$, $\lambda_R$, and $\lambda_D$ are the hyper-parameters to balance the three loss terms.

**Embedding Loss** $L_H$. The embedding Loss $L_H$ comprises the visual consistency part $L_V$ and feature part $L_F$, i.e.,

$$L_H = \lambda_H^{(1)} L_V + \lambda_H^{(2)} L_F,$$

where $\lambda_H^{(1)}$ and $\lambda_H^{(2)}$ are the hyper-parameters to balance the two loss terms.

The visual consistency loss evaluates the differences between poisoned and clean images. It is used to help the trigger embedding network $\mathcal{H}$ to generate poisoned images that are indistinguishable from the related clean images. We evaluate $L_V$ by calculating the pixel distance between the poisoned image $\tilde{x}_i$ and the clean image $x_i$, namely

$$L_V = \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ ||\tilde{x}_i - x_i||^2 \right].$$

Similar with prior DNN-based encoding works in [45], [48], [49], we also calculate the pixel distance using $L_2$ Norm.

The feature loss evaluates the feature distance between the poisoned images and the triggers, ensuring that the poisoned images contain the semantics of triggers. We use a DNN-based feature extractor introduced in [50] to extract the features of the poisoned images and triggers, and $L_F$ is the difference between the two features, namely

$$L_F = \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ \frac{1}{N_k} \sum_{k=1}^{N_k} ||V_k(\tilde{x}_i) - V_k(t_j^{(k)})||^2 \right],$$

where $V_k(\cdot)$ denotes the features extracted by the $k$-th layer of Visual Geometry Group (VGG) and $N_k$ denotes the number of feature neurons.

**Extracting Loss** $L_R$. We use $L_R$ to guide the recovery network $\mathcal{R}$ to extract the trigger information from the poisoned images. $L_R$ evaluates the pixel difference between the recovered hidden information $R(\tilde{x}_i)$ and the real information $t_j^{(k)}$. To prevent the recovery network $\mathcal{R}$ from overfitting the poisoned images, we also input the clean image $x_i$ into $\mathcal{R}$ to evaluate the pixel difference between the $R(x_i)$ and the blank image $\delta$. Then the extracting loss $L_R$ is formulated as

$$L_R = \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ ||R(\tilde{x}_i) - t_j^{(k)}||^2 \right] + \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ ||R(x_i) - \delta||^2 \right].$$

**Adversarial Loss** $L_D$. We use the discriminator $\mathcal{D}$ to determine whether an input is a clean or poisoned image. $L_D$ can assist $\mathcal{H}$ in embedding triggers properly such that $\mathcal{D}$ cannot distinguish the poisoned images from real clean images in $\mathcal{D}_{\text{train}}$. It is formulated as

$$L_D = \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ \log(1 - \mathcal{D}(\tilde{x}_i)) \right] + \mathbb{E}_{x_i \in \mathcal{D}_{\text{train}}} \left[ \log(\mathcal{D}(x_i)) \right].$$

Finally, Algorithm 1 presents the pseudo-code of the training details of the three networks $\mathcal{H}$, $\mathcal{D}$, and $\mathcal{R}$ during the poisoned image generation process.

4.3 Backdoor Injection

Fig. 3 shows the backdoor injection process, which includes the generation of poisoned training dataset by attackers and the training of DNN model by model users. The attacker poisons a part of the training dataset $\mathcal{D}_{\text{train}}$ to obtain the poisoned training dataset $\mathcal{D}_{\text{train}}$, and the backdoors can be secretly implanted into a DNN model trained normally on $\mathcal{D}_{\text{train}}$.

When generating the poisoned training dataset, the attacker uses the well-trained trigger embedding network $\mathcal{H}$ to poison a part of the training dataset with the poisoning
Algorithm 1: The training process of \( \mathcal{H}, \mathcal{R} \) and \( \mathcal{D} \).

**Input:**
1. The initial parameters: \( \theta_H, \theta_R \) and \( \theta_D \);
2. The training dataset: \( \mathcal{D}_{\text{train}} \);
3. The triggers set: \( T \);
4. The loss weights: \( \lambda_H, \lambda_R, \lambda_D, \lambda_H^{(1)}, \lambda_H^{(2)} \);
5. The learning rates: \( lr_H, lr_R, lr_D \);
6. The number of iterations: \( I \);
7. The blank image: \( \delta \).

**Output:** The optimized parameters: \( \theta_H, \theta_R \) and \( \theta_D \);

\[
\text{for } i \leftarrow 1 \text{ to } I \text{ do}
\]
1. Sample a minibatch of triggers \( t \) from \( T \);
2. Generate poisoned images \( \tilde{x} \) using Eq. (2);
3. Compute \( L_V(\tilde{x}, \tilde{\delta}) \) using Eq. (5);
4. Compute \( L_F(\tilde{x}, t) \) using Eq. (6);
5. Compute \( L_R(\tilde{x}, t, \delta) \) using Eq. (7);
6. Compute \( L_D(\tilde{x}, \tilde{\delta}) \) using Eq. (8);
7. \( \lambda_H = \lambda_H^{(1)} L_V + \lambda_H^{(2)} L_F ; \)
8. \( L = \lambda_H L_H + \lambda_R L_R + \lambda_D L_D ; \)
9. /* Update \( \theta_H \) using the gradient of \( L */ \)
10. \( \theta_H \leftarrow \theta_H - lr_H \nabla_{\theta_H} L ; \)
11. \( \theta_R \leftarrow \theta_R - lr_R \nabla_{\theta_R} L_R ; \)
12. \( \theta_D \leftarrow \theta_D - lr_D \nabla_{\theta_D} L_D ; \)

end

5 Experiment Results

In this section, we first introduce the implementation settings of our M-to-N backdoor attack and then test the effectiveness, stealthiness of the poisoned images, and robustness of the attack. We finally give a decision to analyze the poisoning ratio, effects of multiple triggers and ablation studies of the poisoned image generation framework.

5.1 Implementation Settings

5.1.1 Datasets and Target Models

Following the studies of most prior backdoor attacks, we also test our backdoor attack on the three datasets: MNIST \([54]\), CIFAR-10 \([55]\) and GTSRB \([56]\). We use the DNN models Pre-activation Resnet18 (PreActRes18) \([51]\), Resnet18 \([52]\) and Densenet121 \([53]\) on the color image datasets CIFAR-10 and GTSRB, and use the MNIST_MODEL \([24]\) on the grayscale dataset MNIST. Table 3 lists the used datasets and the classifiers in detail.

5.1.2 Parameter Settings for Training

The trigger embedding network \( \mathcal{H} \) is trained with the Adam optimizer, which starts with a learning rate of 0.0002, and its learning rate will decay by 0.2 if the loss does not decrease within five epochs. In the backdoor injection stage, the DNN models are trained with the SGD optimizer, whose initial learning rate is 0.01, and reduced by a factor of 10 when the training epochs reach 100 and 150. The batch size is set as 128, while the maximum epoch is set as 200. The hyperparameters \( \lambda_H, \lambda_H^{(1)}, \lambda_H^{(2)} \), and \( \lambda_R \) are set as 1, and \( \lambda_D \) is 0.01. The poisoned samples are smoothed such that more features about the triggers can be extracted by the DNN models. The smooth coefficient is set as 0.1 for the MNIST and CIFAR-10 datasets and 0.4 for the GTSRB dataset. All experiments are performed on a server with the Ubuntu 16.04.6 LTS operating system, a 3.20GHz CPU, a NVIDIA GeForce GTX3090 GPU with 62G RAM, and an 8TB hard disk.

5.1.3 Evaluation Metrics

We use three metrics to measure the effectiveness of backdoor attacks: attack success rate (ASR), backdoor model accuracy (BA), and clean model accuracy (CA), which are the same with prior backdoor attacks \([23]\), \([24]\), \([29]\). ASR indicates the success rate of classifying the poisoned inputs into the related targets. BA measures the accuracy of a backdoor model on the clean testing dataset. CA tests the accuracy of a clean model on the clean testing dataset. The impact of a backdoor attack on the original task can be measured by comparing BA and CA. A smaller difference between BA and CA indicates less influence of the backdoor attack on the original task.

| Datasets   | #Labels | Input Size | #Train. & Test. Images | Classifier                         |
|------------|---------|------------|------------------------|------------------------------------|
| MNIST      | 10      | \( 28 \times 28 \times 1 \) | 60,000, 10000          | 3 ConvBlocks + 2 fc (MNIST_CNN) \([24]\) |
| CIFAR-10   | 10      | \( 32 \times 32 \times 3 \) | 50,000, 10000          | PreActRes18 \([51]\), Resnet18 \([52]\), Densenet121 \([53]\) |
| GTSRB      | 43      | \( 32 \times 32 \times 3 \) | 39,200, 12600          | PreActRes18 \([51]\), Resnet18 \([52]\), Densenet121 \([53]\) |

TABLE 3

Image datasets and the related classifiers used in our experiments.
To measure the stealthiness of the poisoned images, we compare the clean images with the poisoned images using the peak-signal-to-noise-ratio (PSNR) [57], structural similarity index measure (SSIM) [58], and learned perceptual image patch similarity (LPIPS) [59]. A poisoned image has better stealthiness with a larger PSNR and SSIM, and a smaller LPIPS. To show this effect, we compare the clean images with the poisoned images using any one of M triggers. For each attacked target, we report the average metric values of all the poisoned images.

Our M-to-N backdoor attack can simultaneously attack N targets, and the backdoor of each target can be activated by any one of M triggers. For each attacked target, we report the average metric values of all the generated poisoned images to comprehensively evaluate the performance of our backdoor attack against multiple targets. Specifically, we use the $M \times N$ triggers to equally poison the whole testing dataset to generate the poisoned images and calculate the average metric values of all the poisoned images.

### 5.2 Attack Effectiveness

Our M-to-N backdoor attack can simultaneously attack N targets, and the backdoor of each target can be activated by any one of the M triggers. To show this effect, we design experiments by setting $M \in \{1, 10\}$, $N \in \{1, 2, 4, 5\}$ in the MNIST and CIFAR-10 datasets, and $M \in \{1, 10\}$, $N \in \{2, 4, 5, 10\}$ in the GTSRB dataset (The GTSRB dataset has 43 classes while the MNIST and CIFAR-10 datasets have only 10 classes). Table 4 shows the labels of the attacked targets on these datasets, and Table 5 shows the testing results of our M-to-N attack. As can be seen, all the ASRs for different attacks are close to 100%, indicating that all the triggers can well activate the backdoors. The differences between the BAs and CAS are within 1%, which means that the backdoor attack can not affect the original tasks. Besides, the BAs and ASRs do not significantly decrease with the increase of the attacked targets number. For the MNIST and CIFAR-10 datasets, even half of the labels are attacked simultaneously, the ASRs are still close to 100%, and the influence on the accuracy of original tasks is less than 1%. This shows the effectiveness of our M-to-N backdoor attack in threatening multiple targets. In addition, the experimental results show that our attack has high generality to different datasets and DNN models.

### Table 4

| Dataset | $N$ | Target Labels |
|---------|-----|---------------|
| MNIST   | 1 (0) |               |
|         | 2 (2, 5) |         |
|         | 4 (2, 3, 4, 5) |     |
|         | 5 (0, 1, 5, 7, 8) |     |
| CIFAR-10| 1 (0) |               |
|         | 2 (0, 1) |         |
|         | 4 (0, 1, 4, 6) |     |
|         | 5 (0, 2, 4, 6, 8) |     |
| GTSRB   | 1 (0) |               |
|         | 2 (12, 36) |      |
|         | 4 (12, 33, 35, 36) |   |
|         | 5 (0, 1, 3, 9, 10) |   |
|         | 10 (0, 1, 2, 3, 4, 5, 12, 33, 35, 36) |  |

### Table 5

| Dataset | Model | $M \rightarrow$ | $N \downarrow$ | BA (%) | ASR (%) | BA (%) | ASR (%) | CA (%) |
|---------|-------|-----------------|----------------|--------|---------|--------|---------|--------|
| MNIST   | MNIST | M               | N              | 99.36  | 99.4   | 99.36  | 99.93  | 99.52  |
|         | CNN   | 2               | 10             | 99.33  | 99.5   | 99.32  | 99.86  |        |
|         |       | 4               |                | 98.93  | 99.93  | 98.93  | 99.32  |        |
|         |       | 5               |                | 99.44  | 95.06  | 99.40  | 95.49  |        |
| CIFAR-10| PreAct| 2               |                | 94.43  | 98.51  | 94.38  | 98.23  | 94.69  |
|         | Res18 | 4               | 5              | 93.06  | 97.30  | 93.06  | 97.30  |        |
|         |       | 1               |                | 94.42  | 99.76  | 94.42  | 99.76  |        |
| GTSRB   | Resnet18 | 2             |                | 99.3   | 99.3   | 99.3   | 99.3   | 99.42  |
|         |       | 4               | 10             | 99.32  | 99.72  | 99.32  | 99.72  | 99.42  |
|         |       | 5               |                | 99.32  | 99.66  | 99.32  | 99.66  | 99.42  |
| Dense   | net121 | 2               |                | 99.34  | 99.48  | 99.34  | 99.48  |        |
|         |       | 4               | 10             | 99.51  | 99.59  | 99.51  | 99.59  |        |
|         |       | 5               |                | 99.55  | 99.46  | 99.55  | 99.46  |        |
|         |       | 10              |                | 99.39  | 98.77  | 99.39  | 98.77  |        |
prior attacks can only attack one target using one trigger.

Recently, the authors in [29] proposed an initial work for attacking multiple targets called One-to-N attack. This attack uses triggers with different degrees of pixel value modification to activate the backdoors of different targets. For example, the trigger for the CIFAR-10 dataset is a square, and the poisoning process is to plus or minus a fixed number to the values of the image pixels within the square. As a result, the One-to-N attack has low stealthiness and can be easily detected by defenders, which will be analyzed in Section 5.3.3. We herein compare our backdoor attack with the One-to-N attack [29] when attacking four targets simultaneously, and show the results in Table 7. It shows that both the One-to-N and our M-to-N attacks can achieve high ASRs and BAs on the MNIST dataset. This is because the MNIST dataset comprises grayscale images with handwritten numbers, and these images have very simple features. However, the ASRs of the One-to-N attack on the CIFAR-10 and GTSRB datasets are only 87.26% and 72.83%, respectively, which are about 13% and 28% less than that of our attack. Besides, the One-to-N attack can achieve the BA of 83.04% on the CIFAR-10 dataset, which is about 11% lower than the CA (94.69%). Since BA indicates the performance of the backdoor model on clean testing dataset, the significant reduction can be easily observed by defenders. The BAs of our attack are very close to the CAs, which cannot cause the attention of defenders. As a result, our attack is significantly more effective than the One-to-N attack when attacking multiple targets simultaneously.

5.3 Stealthiness of the Poisoned Images

For a backdoor attack, its poisoned images should be stealthy and similar with clean images such that defenders cannot distinguish the poisoned images from the clean. We test the stealthiness of the poisoned images in our M-to-N backdoor attack from the aspects of visual effects and quantitative results.

5.3.1 Visual Effects

We launch two different attack paradigms, 1-to-4 attack \((M = 1, N = 4)\) and 4-to-1 attack \((M = 4, N = 1)\), to show the visual effects of the poisoned images. Fig. 4 displays some poisoned images on the CIFAR-10 dataset. It is obvious that all the generated poisoned images are indistinguishable from their clean counterparts, which can be seen in Figs. 4 (b) and (e). The residuals of the clean and poisoned images in Figs. 4 (c) and (f) indicate that our trigger embedding network \(\mathcal{H}\) can well hide the triggers in the clean images. As a result, the poisoned images cannot be recognized by human eyes.

5.3.2 Quantitative Results

We also test the PSNR, SSIM, and LPIPS average values on the entire poisoned testing datasets and simply display the results of our 1-to-N attack paradigm \((M = 1)\) in Table 8 since our 10-to-N attack paradigm \((M = 10)\) has similar results. The images in the MINST dataset are of size \(28 \times 28\), which is too small to apply the Alex network. Then LPIPS_Alex on MNIST cannot be calculated. As can be seen, the PSNR values are larger than 40, the SSIM values are all close to 1, and the LPIPS values are all close to 0. These results indicate a high similarity of the poisoned images with the clean images. Besides, we can observe from Table 8 that the SSIM and PSNR values do not decrease with the increase of the attacked targets number, which indicates the high performance of our attack against multiple targets.

5.3.3 Comparisons with Prior Backdoor Attacks

We compare the stealthiness of the poisoned images in our M-to-N backdoor attack with that in WaNet [24], Input-Aware [23] and One-to-N [29]. We first calculate the average

---

Fig. 4. Visual effects of the poisoned images in our M-to-N backdoor attack (1-to-4 attack in the left and 4-to-1 attack in the right). (a) Original images; (b) the poisoned images for four targets; (c) residuals of (a) and (b); (d) four triggers for generating (b); (e) the poisoned images for a single target with four different triggers; (f) residuals of (a) and (e); (g) four triggers for generating (e).
TABLE 8
PSNR, SSIM and LPIPS (LPIPS_VGG and LPIPS_Alex) average values of our M-to-N backdoor attack on different datasets. The trigger number for each target is set as 1 (M = 1).

| Datasets | #Targets | PSNR (db) | SSIM | LPIPS_VGG | LPIPS_Alex |
|----------|----------|-----------|------|-----------|------------|
| MNIST    | 1        | 45.99     | 0.9811 | 0.0015    | -          |
|          | 2        | 44.84     | 0.9802 | 0.0015    | -          |
|          | 4        | 46.74     | 0.9426 | 0.0008    | -          |
|          | 5        | 49.30     | 0.9684 | 0.0009    | -          |
| CIFAR-10 | 1        | 45.87     | 0.9953 | 0.0001    | 0.0026     |
|          | 2        | 41.29     | 0.9864 | 0.0006    | 0.0110     |
|          | 4        | 44.88     | 0.9936 | 0.0041    | 0.0002     |
|          | 5        | 45.88     | 0.9951 | 0.0002    | 0.0030     |
| GTSRB    | 1        | 45.91     | 0.9930 | 0.0004    | 0.0051     |
|          | 2        | 43.07     | 0.9895 | 0.0013    | 0.0116     |
|          | 4        | 47.06     | 0.9936 | 0.0004    | 0.0051     |
|          | 5        | 43.07     | 0.9895 | 0.0013    | 0.0116     |
|          | 10       | 47.54     | 0.9967 | 0.0321    | 0.0044     |

Fig. 5. Generated poisoned images. The top row shows the poisoned images, while the bottom row indicates the residuals between the poisoned and clean images.

PSNR values of the poisoned images in the whole poisoned testing dataset with their clean images. Note that the One-to-N and our M-to-N attacks can simultaneously attack multiple targets, and we set the number of the attacked targets as 4 (N = 4) in our experiment. The average PSNR values of our M-to-N, WaNet [24], Input-Aware [23] and One-to-N [29] attacks are 44.88 db, 23.32 db, 20.66 db and 29.80 db, respectively. It is obvious that our M-to-N backdoor attack can generate poisoned images with much larger average PSNR value than other attacks.

Besides, we also show the visual effects of poisoning the clean image frog for different attacks. Since the number of the attacked targets is four in One-to-N and our M-to-N attacks, four poisoned images can be generated by poisoning a clean image separately using one of the four triggers corresponding to the attacked targets. We display the poisoned image with the maximum PSNR value for One-to-N attack and the poisoned image with the minimum PSNR value for our M-to-N attack. As can be seen from Fig. 5, the triggers in Input-Aware [23] and One-to-N [29] attacks have clear patterns. Both WaNet [24] and our M-to-N attacks can generate poisoned images that are indistinguishable from the originals; however, the residuals indicate that the modification to the clean image in our M-to-N is undetectable. The largest PSNR value and the most natural appearance verify that the poisoned images of our M-to-N are stealthy.

5.4 Robustness of M-to-N Backdoor Attack
An effective backdoor attack should be robust to the commonly used pre-processing operations, which means that the poisoned images can still activate the backdoors when they are pre-processed. We evaluate the robustness of our M-to-N backdoor attack under different pre-processing operations and compare it with prior backdoor attacks. These pre-processing operations include the vertical flipping (Flipping), random rotation (Rotation), padding to the original size after shrinking (Shrinking&Padding), resizing to the original size after random cropping (Cropping&Resizing), and Gaussian noise blurring. Note that the operations exclude those already employed during the backdoor injection phase, such as horizontal flipping and fifteen-degree rotation. Fig. 6 shows the visual effects of these pre-processing operations, and Table 9 shows the ASRs of different backdoor attacks when their poisoned images are pre-processed using these operations. It can be seen that the ASRs of the WaNet [24], Input-Aware [23] and One-to-N [29] attacks under some of the operations are very low, which means that the three attacks will fail under these pre-processing settings. Our M-to-N attack can achieve the highest average ASRs against these pre-processing operations, indicating its high robustness to resist pre-processing operations.

5.5 More Experimental Results
5.5.1 Effect of the Poisoning Ratio
We design experiments to analyze the effect of the poisoning ratio ρ to the ASRs and BAs of our M-to-N backdoor attack. We test the BAs and ASRs of the 1-to-1 (M = 1, N = 1) and 10-to-1 (M = 10, N = 1) attack paradigms when ρ gradually increases from 2% to 10%. The results on the three datasets are shown in Fig. 7. As can be seen, our attack can attain high ASRs (more than 95%) by poisoning only 2% of the training dataset. Besides, the ASRs increase with the increase of ρ and stabilize near 100%, while the BAs remain almost unchanged. Our attack can achieve high
performance by only poisoning a small part of the training dataset.

5.5.2 Trigger Sensitivity

According to the discussions in [10], [22], [23], [25], the static triggers and those with external (non-clean) features can be easily reverse-engineered and detected, whereas the dynamic triggers and those with clean features cannot be detected. In our M-to-N backdoor attack, the semantic information of a trigger can be spread all over the whole poisoned image through the trigger embedding network \( \mathcal{H} \), and thus the poisoned images are determined by the original clean image and trigger, which both consist of clean features. The residual between the poisoned and clean images is sample-specific and unique, which cannot be reverse-engineered.

We design experiments to show this effect. Specifically, we generate new “poisoned images” with the residuals generated using other clean images and test their ASR. For a clean image \( x_i \in D_{\text{test}} \) and a trigger \( t \in T \), we randomly select another image \( x_j \), where \( x_j \in D_{\text{test}} \) and \( x_j \neq x_i \), and feed \( x_j \) and \( t \) into the trigger embedding network \( \mathcal{H} \) to generate a poisoned image \( \tilde{x}_j \). Then, we construct a new “poisoned image” by adding the inconsistent residual between the poisoned image \( \tilde{x}_j \) and its clean counterpart \( x_j \) to the clean image \( x_i \), i.e., \( x_i + (\tilde{x}_j - x_j) \). For simplicity, we set \( j = (i + 1) \mod |D_{\text{test}}| \). We use this manner to poison the three testing datasets and evaluate the average ASRs of the new “poisoned images”. Table 10 shows that the ASRs decrease fast when the poisoned images are generated by adding inconsistent residuals to clean images. The results demonstrate that the residuals generated from our triggers are sample-specific and unique.

5.5.3 Ablation Studies

We design ablation studies to explore the effect of each module in our poisoned image generation framework. Our poisoned image generation framework includes the trigger embedding network \( \mathcal{H} \), recovery network \( \mathcal{R} \) module, and discriminator \( \mathcal{D} \) module. We implement a 1-to-4 \( (M = 1, N = 4) \) attack paradigm on the CIFAR-10 dataset to evaluate the effectiveness and stealthiness of our M-to-N attack under the three modules and show the average results in Table 11.

As can be seen, the ASR of the poisoned images generated using only the \( \mathcal{H} \) module is low (11.08\%), indicating that the backdoor attack is ineffective under the settings. However, the ASR increases from 11.08\% \((\mathcal{H})\) to 99.36\% \((\mathcal{H} \text{ and } \mathcal{R})\) and to 99.60\% \((\mathcal{H}, \mathcal{R} \text{ and } \mathcal{D})\) when the \( \mathcal{D} \) module is added into the poisoned image generation framework. This shows that the \( \mathcal{R} \) module is very useful for improving the ASRs. The average PSNR value of the poisoned images generated using only the \( \mathcal{H} \) module is the largest, and it reduces when \( \mathcal{R} \) module is added, because some features of the triggers are correctly added into the poisoned images. However, both the ASR and the average PSNR value improve when adding the \( \mathcal{D} \) module. This proves that all three modules are essential to the effectiveness and stealthiness of our attack.

6 Evading Existing Defenses

Our M-to-N backdoor attack has strong ability to evade existing defenses. To show this effect, we evaluate its performance going against different defenses including Fine-Pruning [13], Neural Cleanse [10], STRIP [11], and SensiNet [12].

6.1 Fine-Pruning

As discussed in Section 2.2, the fine-pruning defense directly removes the backdoor by pruning the dormant neurons
when inputting the clean images. Following the settings in previous methods, we also evaluate the resistance of our M-to-N attack to the fine-pruning defense by analyzing the activation of the neurons in the last convolutional layer. Fig. 8 displays the BAs and ASRs of our attack with the PreActRes18 model on the CIFAR-10 dataset against different numbers of attacked targets. It can be seen that the ASRs of our attack only decrease slightly with the increase of the fraction of pruned neurons. Furthermore, the ASRs of our attack decrease after the BAs, indicating that FP defense cannot remove our backdoors without impacting the performance of original tasks.

6.2 Neural Cleanse
The Neural Cleanse defense detects the backdoor of a suspected DNN model by reverse-engineering the triggers. Specifically, for each label, the Neural Cleanse defense first inverses a trigger for each class label that can transform the predictions of all clean images to the label and then applies an anomaly detection method (e.g., the Median Absolute Deviation (MAD) algorithm) to discriminate the triggers that are notably smaller than the rest. According to the discussion in the anomaly index threshold is set at two, and any model with an anomaly index larger than two is regarded as a backdoor model.

Our experiments perform ten iterations of the detection procedure on each dataset, and the obtained mean anomaly indexes are shown in Fig. 9. As can be seen from Figs. 9(a) and (b), the average anomaly indexes of our M-to-N attack are all less than two, which indicates that the Neural Cleanse defense cannot detect our attack. We also compare the ability of different backdoor attacks in resisting the Neural Cleanse and show the results in Fig. 9(c). As can be seen, our attack can achieve the smallest anomaly index values, which indicates a stronger ability to resist the Neural Cleanse defense.

6.3 STRIP
The STRIP defense first perturbs any input image with another clean image using a specified blending method and then calculates the entropy of the mixture predictions. A low entropy of the predictions violates the input-dependence property of a clean model and implies a poisoned image. A backdoor attack has stronger ability to resist STRIP defense if higher entropy is obtained. According to the setting in (11), the entropy boundary is set at 0.2, and an obtained
which is a new attack paradigm to attack $N$ suspected targets simultaneously, and the backdoor of each target can be activated by any one of $M$ triggers. To avoid introducing entropy smaller than the boundary is detected as a poisoned image.

We also evaluate the ability of our M-to-N attack to resist the STRIP on the three datasets with the PreActRes18 model and show the results in Fig. 10. As can be seen from Figs. 10(a) and (b), the minimum entropy values of our attack are larger than the detection boundary of 0.2, which indicates that the STRIP identifies our poisoned images as clean images. We also compare the ability of different backdoor attacks to resist STRIP and show the results in Fig. 10(c). The minimum entropy value of the poisoned images in One-to-N [29] is about $10^{-9}$, which is far smaller to resist the STRIP defense. The other backdoor attacks can bypass the STRIP defense. In addition, Our M-to-N attack can achieve a larger minimum entropy than other methods in most cases, which indicates the strong ability of our M-to-N attack to withstand the STRIP defense.

6.4 SentiNet

The SentiNet [12] defense utilizes the input saliency maps generated by Grad-CAM [44] to detect potential trigger regions in an input. To evaluate the ability of our M-to-N attack to resist this defense, we construct the saliency maps of the clean images and the generated poisoned images and show the results in Fig. 11. As can be seen, the trigger regions of the generated poisoned images in the One-to-N [29] attack can be easily observed. For example, the modified edges in the second row of Fig. 11(d) and the black boxes in the fourth and sixth rows of Fig. 11(d). The saliency maps between the clean and poisoned images in WaNet [24] and Input-Aware [23] can still be observed a slight difference, especially in the MNIST dataset. Our poisoned images have consistently significant areas with the clean images. This is because our triggers are related to the target labels and are well distributed in the clean images via the poisoned image generation framework. The experimental results indicate that our attack is resistant to the SentiNet defense.

7 Conclusion

In this paper, we proposed an M-to-N backdoor attack, which is a new attack paradigm to attack $N$ suspected targets simultaneously, and the backdoor of each target can be activated by any one of $M$ triggers. To avoid introducing external features into training dataset, we use the clean images selected from training dataset as the triggers and design a poisoned image generation framework to embed the triggers into clean images secretly. Extensive experimental results validate that the backdoors of multiple targets can be effectively activated by poisoning only a small part of the training dataset. Besides, our attack has a strong ability to resist pre-processing operations and state-of-the-art defenses. We believe this new multi-target paradigm can offer new thinking to backdoor attacker studies and motivate further defense research.
[46] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Imag. Comput. Comput.-Assisted Intervention, 2015, pp. 234–241.

[47] Q. Fan, J. Yang, G. Hua, B. Chen, and D. Wipf, “A generic deep architecture for single image reflection removal and image smoothing,” in Proc. IEEE Conf. Comput. Vis., 2017, pp. 3238–3247.

[48] S. Baluja, “Hiding images in plain sight: Deep steganography,” in Proc. Advances Neural Inf. Process. Syst., vol. 30, 2017, pp. 1–11.

[49] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “Hidden: Hiding data with deep networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 657–672.

[50] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 694–711.

[51] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 630–645.

[52] —, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[53] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4700–4708.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] A. Krizhevsky, G. Hinton et al., “Learning multiple layers of features from tiny images,” University of Toronto, Tech. Rep., pp. 1–60, 2009.

[56] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” Neural Netw., vol. 32, pp. 323–332, 2012.

[57] Q. Huynh-Thu and M. Ghanbari, “Scope of validity of psnr in image/video quality assessment,” Electron. Lett., vol. 44, no. 13, pp. 800–801, 2008.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, 2004.

[59] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 586–595.