STEALTHY BACKDOOR ATTACK WITH ADVERSARIAL TRAINING

Le Feng *Sheng Li* *Zhenxing Qian*⋆ *Xinpeng Zhang*⋆

School of Computer Science and Technology, Fudan university

ABSTRACT

Research shows that deep neural networks are vulnerable to backdoor attacks. The backdoor network behaves normally on clean examples, but once backdoor patterns are attached to examples, backdoor examples will be classified into the target class. In the previous backdoor attack schemes, backdoor patterns are not stealthy and may be detected. Thus, to achieve the stealthiness of backdoor patterns, we explore an invisible and example-dependent backdoor attack scheme. Specifically, we employ the backdoor generation network to generate the invisible backdoor pattern for each example, and backdoor patterns are not generic to each other. However, without other measures, the backdoor attack scheme cannot bypass the neural cleanse detection. Thus, we propose adversarial training to bypass neural cleanse detection. Experiments show that the proposed backdoor attack achieves a considerable attack success rate, invisibility, and can bypass the existing defense strategies.

Index Terms— Backdoor, Invisibility, Example-dependent, Adversarial training

1. INTRODUCTION

Recently, neural networks develop rapidly, which has been widely used in object detection [1, 2], speech recognition [3, 4], face recognition [5, 6] and other fields. However, related studies have shown that neural networks are vulnerable to backdoor attacks. The backdoor attack on the neural network was first proposed by [7]. The attacker attaches the fixed backdoor pattern to clean examples to obtain backdoor examples, and then assigns the target label to these backdoor examples, and trains the network to fit these backdoor examples. However, the scheme has obvious defects, i.e., neither backdoor examples nor backdoor networks can bypass the detection. For backdoor examples, the backdoor pattern is easily noticeable visually, and the backdoor pattern is fixed, that is, all backdoor examples have the same backdoor pattern. Once an example is detected abnormal, the following backdoor examples may also be invalid. For the backdoor network, it cannot bypass many existing backdoor detection strategies. For example, employing neural cleanse [8] (called NC) detection can detect the class injected into the backdoor (called the target class) through reverse engineering.

To make backdoor patterns stealthy to bypass the detection, [9] proposes two invisible backdoor attack schemes: the source-target backdoor scheme implemented through steganography, the universal invisible backdoor scheme based on universal adversarial examples. Although the implementation cost of the steganography scheme is low, it also cannot bypass [10] detection. Moreover, their universal invisible backdoor attack scheme also cannot bypass neural cleanse [8] (called NC) detection. Though they tried to use the strategy of 'inject all' to overcome such shortcoming, it only makes intensities of the triggers of all classes significantly weak, while with certain prior knowledge, the network also will be detected to be abnormal.

In addition, [11, 12] proposed dynamic backdoor attack schemes to try to make backdoor patterns stealthy. Compared with the fixed backdoor pattern, the dynamic backdoor patterns are more difficult to detect. However, in these two schemes, backdoor patterns are dynamic, they are still visible and may also be detected.

To solve the shortcomings of the previous schemes, we explore an invisible and example-dependent backdoor attack scheme, where each example has its own backdoor pattern, and these backdoor patterns are imperceptible to human eyes. Specifically, we construct a backdoor generation network. Input the clean examples into such a backdoor generation network to get invisible backdoor patterns, and then add the backdoor patterns to the clean examples to get the backdoor examples. Synchronously train the backdoor generation network and the victim network, so that the victim network can fit the backdoor examples obtained by the backdoor generation network, and finally, the backdoor network will be obtained. However, as NC detection mentioned, injecting the backdoor into the network will change the decision boundary of the network so that the intensity of the trigger of the target class is significantly small. Thus, to prevent the user from detecting the backdoor from the target class, we design adversarial training, which can significantly increase the intensity of the trigger of the target class, thereby bypassing the detection.

To the end, we summarize our contributions as follows:

1. To ensure that backdoor patterns are stealthy, we explore an example-dependent and invisible backdoor attack scheme.

2. To bypass NC detection, adversarial training is proposed to increase the intensity of the trigger of the target class.

3. We have empirically verified that the proposed backdoor attack scheme has a considerable attack success rate, stealthiness, example-dependence, robustness, and can bypass existing defense strategies.

2. THE PROPOSED METHOD

The proposed attack is aimed at image classification networks. The victim network is M with weights θ_M. The dataset trained on the network M is (X, Y) of length n, where X is the example set, and Y is the label set. The set of all classes is C. We use $f\left(x_i \mid \theta_M\right)$ to represent the output of example $x_i \in X$ on M. The target class is y^*. The backdoor attack on the network M invoices two fundamental targets: making the network classify normally on clean examples and making the network classify all backdoor examples into the target class y^*, as listed in Eq. 1 and Eq. 2, respectively.

$$L_o = \min_{\theta_M} \frac{1}{n} \sum\nolimits_i \ell(f(x_i|\theta_M), y_i), \tag{1}$$

This work is supported by National Science Foundation of China (Grant U1936214, Grant U20B2051, Grant 62072114, Grant U20A20178), and Project of Shanghai Science and Technology Commission (Grant 21010500200). ⋆Corresponding author: zxqian@fudan.edu.cn, zhangxinpeng@fudan.edu.cn

$$L_b = \min_{\theta_M} \frac{1}{h} \sum_j \ell(f(x_j^* = x_j \circ p|\theta_M), y^*), \tag{2}$$

where $\circ$ indicates attaching the backdoor pattern p to the clean example x_j ($j \in [0, h-1]$) to get the backdoor example x_j^*. h is the number of backdoor examples. To ensure that both backdoor patterns and the backdoor properties of backdoor networks are stealthy, we design a stealthy backdoor attack with adversarial training.

The scheme consists of two parts: making stealthy backdoor patterns and adversarial training. The following details the design of two parts.

2.1. Making stealthy backdoor patterns

We construct the backdoor generation network *G* with weights θ_G to get the initial backdoor patterns. The structure of θ_G includes two downsampling operations through convolution and two upsampling operations through transposed convolution. Next, the result is normalized to [-1, 1] through the Tanh activation function. The input of G is the clean example x_j. Then, we can obtain the initial backdoor pattern $p_j = G(x_j \mid \theta_G)$. Next, to make the backdoor pattern invisible, we use a significant small constant ε to shrink the initial backdoor pattern, which determines the intensity of the backdoor pattern and attack success rate. The smaller the ε is, the more invisible the backdoor pattern and but lower attack success rate. Then, these backdoor patterns have two flow directions.

$$L_b = \min_{\theta_M, \theta_G} \frac{1}{h} \sum_j \ell(f(x_j^*|\theta_M), y^*) \tag{3}$$

In the first direction, attach the backdoor pattern to the corresponding clean example x_j to obtain the right backdoor example $x_j^* = \text{clip}(\varepsilon \cdot p_j + (1-\varepsilon) \cdot x_j, 0, 1)$, where clip(,,) is the clipping function to ensure that the values of the backdoor example fall in [0, 1]. For convenience, it is written as $x_j^* = x_j \circ \varepsilon p_j$. Then assign the target label to these backdoor examples, and employ the optimization function listed in Eq. 3 to replace that in Eq. 2 to fit the backdoor examples.

In the second direction, we will ensure that backdoor patterns are example-dependent. Inspired by [11], we use the cross-validation to ensure that the backdoor pattern of each example is unique. For convenience, the two different examples are denoted by x_u and x_v, respectively. Their initial backdoor patterns are $p_u = G(x_u \mid \theta_G)$ and $p_v = G(x_v \mid \theta_G)$. Cross-validation means that both the pattern p_u attached to x_v and the pattern p_v attached to x_u are invalid. Specifically, we assign the clean label y_u to $x_u^c = x_u \circ \varepsilon p_v$ and the clean label y_v to $x_v^c = x_v \circ \varepsilon p_u$. Then use the loss function in Eq. 4 to optimize the generator *G* and victim network *M*.

$$L_c = \min_{\theta_M, \theta_G} \frac{1}{h} \sum_j \ell(f(x_j^c|\theta_M), y_j) \tag{4}$$

2.2. Adversarial training

As mentioned by NC detection [8], injecting the backdoor in the neural network will change the original decision boundary of the network, making the intensity of the trigger of the target class abnormally small. As shown in Fig. 1(a), in the clean network, for classes A, B, and C, the intensities Δa, Δb, and Δc of triggers constructed by reverse engineering are similar. However, once the backdoor is injected into class A, obtaining the backdoor network. Then, in the backdoor network, for classes A, B, and C, the intensity Δa^* of the trigger of the target class A will be significantly smaller than Δb^* and Δc^*, as shown in Fig. 1(b), thus causing the user to detect an

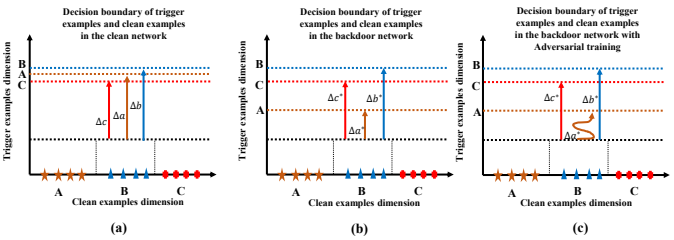

Fig. 1: *A*, *B*, and *C* are three classes. Δa denotes the intensity of the trigger of class A. Δa^* is the intensity of the trigger of class A in the backdoor network, which means that although the intensity of the invisible backdoor pattern of the target class A is small, adversarial training makes it more difficult and tortuous for reverse engineering to search such small intensity backdoor pattern. Δb, Δc, Δb^* and Δc^* have similar meanings.

anomaly in the network. Thus, to overcome such a defect, we propose adversarial training, the principle of which is to prevent the user from searching the shortest path when employing reverse engineering to construct the trigger of the target class. As shown in Fig. 1(c), Δa^* is similar to Δb^* and Δc^*, which means that NC detection is bypassed.

Before elaborating adversarial training, we first describe how to construct the trigger of the class through reverse engineering in NC detection. First, define the trigger application function:

$$A(x_i, m, \Delta) = (1-m) \cdot x_i + m \cdot \Delta, \tag{5}$$

where *m* is the initial 2D mask matrix with the same height and width as x_i, Δ is the initial 3D pattern with the same height and weight as x_i. The trigger is (m, Δ), and the intensity of the trigger is $\|m\|_1$. Constructing the trigger through reverse engineering means keeping the network intact, and employing the optimization function listed in Eq. 6 to search the trigger with the smallest L1-norm $\|m\|_1$, which can classify all examples $x_i \in X$ into the testing class $y^t \in C$.

$$\min_{m,\Delta} \frac{1}{n} \sum_i \ell(f(A(x_i, m, \Delta)|\theta_M), y^t) + \lambda \cdot \|m\|_1, \tag{6}$$

where λ is the weight of the second term, which is adjusted dynamically during constructing. The optimizer is Adam. Employ reverse engineering for all classes to get a set of triggers $\{(m^t, \Delta^t)|y^t \in C\}$. For the target class y^*, the intensity $\|m^*\|_1$ is significantly weaker than the other $\left\|m^t\right\|_1$. Thus, the point to bypassing NC detection is to make $\|m^*\|_1$ no longer significantly weaker, which is modeled as a bilevel optimization problem of maximizing the minimum listed in Eq. 7 when injecting the backdoor into the target class y^*.

$$\max_{\theta_M} \min_{m,\Delta} \frac{1}{n} \sum_i \ell(f(A(x_i, m, \Delta)|\theta_M), y^*) + \lambda \cdot \|m\|_1 \tag{7}$$

To solve this bilevel optimization problem, we think of adversarial training. The proposed adversarial training consists of positive and negative objects. The positive object is to keep the network M intact, and employ the aforementioned reverse engineering with y^* as the target class to get (m^*, Δ^*). The optimization function is listed in Eq. 8, which means to search for the trigger with the smallest L1-norm through reverse engineering. The negative object is to assign the clean label y_i to the trigger example obtained by attaching the trigger (m^*, Δ^*) to the clean example $x_i \in X$, and then train the network M to correct its classification. The optimization function is listed in Eq. 9.

Table 1: Quantitative result on six indicators. ↑ means higher is better, ↓ means lower is better. ✓and ✗represent achieved and unachieved goals, respectively. Bold means better performance. In [9], we take L_2 attack with the best attack effect. In the setting: Backdoor patterns, we show the backdoor patterns of two different examples. Our scheme* indicates backdoor injection without adversarial training using our scheme.

Settings	Badnets [7]		Input-aware [11]		Invisible [9]		Our scheme				Our scheme*			
							Experiment 1		Experiment 2		Experiment 1		Experiment 2	
Dataset: Network	GTSRB: ResNet18		MNIST: Littlenet		GTSRB: ResNet18		MNIST: Littlenet		GTSRB: ResNet18		MNIST: Littlenet		GTSRB: ResNet18	
(BMA ↑, Baseline)	(95.58%, 95.58%)		(99.54%, 99.54%)		(95.31%, 95.31%)		(98.9%, 99.35%)		(95.00, 95.58%)		99.25%, 99.35%		95.38%, 95.58%	
ASR ↑	99.90%		99.54%		$> 90.00\%$		99.10%		95.90%		99.20%		98.6%	
Backdoor patterns														
Invisibility: (PSNR ↑, SSIM ↑)	(26.74, 0.98)	✗	(15.24, 0.35)	✗	**(30.25, 0.98)**	✓	**(40.52, 0.93)**	✓	**(40.40, 0.99)**	✓	**(40.54, 0.91)**	✓	**(39.65, 0.99)**	✓
Example-dependence: CBMA ↑	3.52%	✗	**95.25%**	✓	5.56%	✗	**94.3%**	✓	**90.27%**	✓	**95.82%**	✓	**91.25%**	✓
Bypassing NC? Anomaly index<2	4.51	✗	**1.35**	✓	3.58	✗	**0.83**	✓	**1.21**	✓	4.53	✗	3.75	✗
Bypassing Stripe? $\frac{Backdoor\_entropy}{Clean\_entropy}$ ↑	0.46	✗	–	–	0.53	✗	**1.58**	✓	**1.12**	✓	0.16	✗	0.65	✗

It can be seen that the negative target is opposite to the positive target. The positive target is to search for the trigger that can misclassify all examples into the target class, while the negative target is to make the examples with such a trigger classified correctly. Such adversarial training will significantly increase $\|m^*\|_1$, as shown in Fig. 5(a). In this way, once the user employ reverse engineering for class y^*, it is difficult to search for the shortest path to get minimize $\|m^*\|_1$, as shown in Fig. 1(c), thus bypassing NC detection.

$$L_e = \min_{m,\Delta} \frac{1}{n} \sum\nolimits_i \ell(f(A(x_i, m, \Delta)|\theta_M), y^*) + \lambda \cdot \|m\|_1 \quad (8)$$

$$L_a = \min_{\theta_M, \theta_G} \frac{1}{n} \sum\nolimits_i \ell(f(A(x_i, m^*, \Delta^*)|\theta_M), y) \quad (9)$$

2.3. Injecting the backdoor

This section will give the detailed scheme. The specific implementation includes two steps:

Step 1: Keep the network M intact and employ L_e in Eq. 8 to construct the trigger (m^*, Δ^*) of the target class y^*.

Step 2: Train the backdoor generation network G and the victim network M to inject the backdoor into y^*. The loss function L_{all} consists of four parts: L_o, L_b, L_c, and L_a. Employ loss weights λ_1, λ_2, λ_3, and λ_4 to combine these loss functions. We choose Adam as the optimizer, the learning rates for M and G are set to $lr_M = 0.0001$ and $lr_G = 0.001$ respectively. Synchronously train *G* and the victim network *M*.

$$L_{all} = \lambda_1 \cdot L_o + \lambda_2 \cdot L_b + \lambda_3 \cdot L_a + \lambda_4 \cdot L_c \quad (10)$$

Then, continue to training from **Step 1** until the performance of the network on clean examples is restored and the desired attack success rate is reached.

3. EXPERIMENT EVALUATION

In this section, we will implement our attack on the MNIST [13], GTSRB [14], where GTSRB is trained on ResNet18 [15], and MNIST is trained on the network denoted as Littlenet [11]. The target classes are randomly selected as 8 and 15, respectively. In the following analysis, we will take $\varepsilon = 0.010$. $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ is 1, 1, 0.1, 1, respectively.

3.1. Qualitative result

In Fig. 2, we show clean examples, backdoor examples, and their backdoor patterns, from which we can see that these backdoor patterns are almost imperceptible to the human eye, and are also related to the content of the images, indicating achieving invisibility and example-dependence of backdoor patterns.

(a) (b)

Fig. 2: (a) and (b) are from MNIST ($\varepsilon = 0.010$) and GTSRB ($\varepsilon = 0.010$), respectively. The first column to the fourth column is the original images, the backdoors image, the backdoor patterns (the difference between the backdoor image and the clean image), and the 30×backdoor patterns.

3.2. Quantitative result

We will quantitatively evaluate our scheme from six indicators, including ASR, BMA, invisibility (PSNR (Peak Signal to Noise Ratio) [16] and SSIM (Structural SIMilarity) [17]), example-dependence, NC [8] defense and Stripe defense [18]. Below, we introduce these six indicators in detail.

ASR (Attack Sucess Rate): It refers to the probability that backdoor examples are misclassified into the target class. The higher the attack success rate, the higher the attack effect.

BMA (Backdoor Model Accuracy): It denotes the accuracy of the backdoor model on clean examples. As mentioned in Section 2, a fundamental goal of backdoor attacks is to ensure that the performance of the network on clean examples does not degrade. Therefore, the higher the BMA, the closer to the baseline accuracy of the clean network, and the higher the effect of the backdoor attack.

Invisibility: We use PSNR and SSIM to evaluate the invisibility of the backdoor pattern. The larger the PSNR, the higher the similarity between the backdoor image and clean image. Generally, if PSNR is greater than 40 [19] , the difference between the two images is difficult to be noticed by the human eyes. The maximum value of SSIM is 1, and the closer to 1, the more similar the two images are.

Example-dependence: We use the accuracy (CBMA) of cross examples on the backdoor model to quantify the example-dependence of backdoor patterns. The higher the CMBA, the weaker the attack effect of the unmatched backdoor pattern, and the higher the dependence of the backdoor pattern on the example.

Neural Cleanse (NC): The principle of NC detection is to use reverse engineering for all classes to construct the triggers for these

classes. Compared with that of clean classes, the intensity of the trigger of the target class is significantly smaller. Specifically, if the anomaly index of the trigger of this class is greater than 2, this class is determined to be the target class.

Stripe: Stripe is used in the inference phase to determine whether the backdoor is injected into the network by detecting whether the testing example is the backdoor example. Specifically, combining the testing example with pre-prepared clean examples to obtain synthetic examples. Feed synthetic examples into the network for prediction, and determine whether the testing example is a backdoor example according to the entropy of the prediction result. The smaller the entropy, the more likely it is a backdoor example. Here, we use $\frac{Backdoor\_entropy}{Clean\_entropy}$ to indicate the possibility that the network is a backdoor network. The larger $\frac{Backdoor\_entropy}{Clean\_entropy}$ is, the more likely it is a backdoor network.

The quantitative results are listed in Table 1, the proposed scheme significantly surpasses other schemes. Compared with no adversarial training, although the attack success rate slightly degraded (in GTSRB, 98.6% decreases to 95.90%), the NC are successfully bypassed. In addition, we also surprisedly find that although adversarial training is against NC defense, it also has a very good effect on Stripe defense. As shown in Fig. 3(a) and 3(c), without adversarial training, the entropy of the backdoor example is significantly less than that of the clean example. In comparison, as shown in Fig. 3(b) and 3(d), with adversarial training, the entropy of backdoor examples is close or even greater than that of clean examples, thus bypassing Stripe detection.

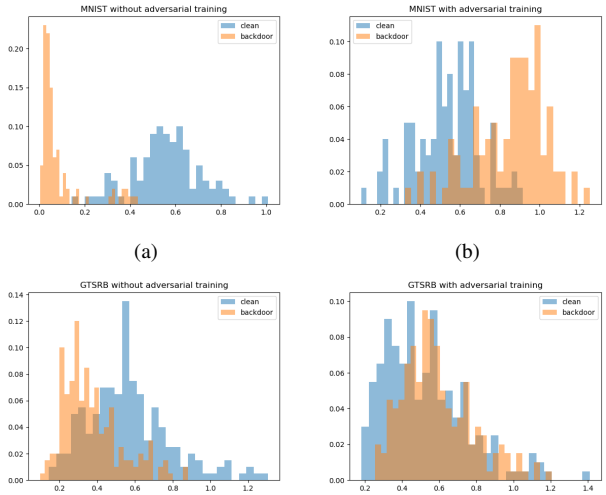

Fig. 3: Stripe defense. These figures show the probability distribution of entropy of 2000 clean examples and backdoor examples with and without adversarial training.

3.3. Ablation study

ϵ: Fig. 4 shows the changes of ASR and BMA when ϵ is 0.005, 0.010, or 0.020. As injecting epochs increase, ASR gradually increases, and BMA impaired by injecting the backdoor also gradually recover. In addition, the smaller the ε, the lower ASR and the greater injecting costs. For example, in GTSRB, at $\varepsilon = 0.020$, ASR of one injecting epoch can reach 67.46%, while at $\varepsilon = 0.005$, ASR of one injecting epoch is only about 36.24%, and to reach about 67.46%, it will consume about 9 additional injecting epochs.

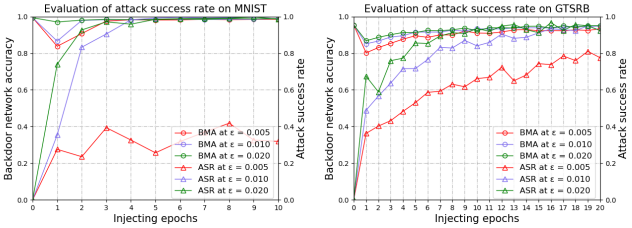

Fig. 4: Evaluation of attack success rate. At $\varepsilon = 0.005, 0.010, 0.020$, as injecting epochs increase, attack success rate and backdoor network accuracy change.

Adversarial training: In Fig. 5(a), we report the L1-norm of the trigger of the target class with injecting epochs increasing. Adversarial training gradually increases the L1-norm of the trigger of the target class. Without adversarial training, through 19 injecting epochs, the L1-norm of the trigger of the target class is about 90. In contrast, with adversarial training, the L1-norm of the trigger of the target class is increased to about 260. The specific detection results are listed in Table. 1, and with adversarial training, anomaly index$<$2, NC detection is successfully bypassed.

Furthermore, can adversarial training be used in other schemes? For this reason, we apply adversarial training to the Badnets scheme to detect whether it is universal. As shown in Fig. 5(b), only adversarial training can not help the Badnets scheme bypass the detection. Analyzing the reason, we think that in the Badnets scheme, the backdoor pattern is intact during injecting the backdoor, which makes it difficult to achieve the local optimal solution that maximizes the minimum. When we add the trainable dynamic noise to the original backdoor pattern, the Badnets scheme can also bypass the detection, as shown in Fig. 5(b). Therefore, we conclude that with adversarial training and the trainable backdoor pattern, NC detection can be bypassed.

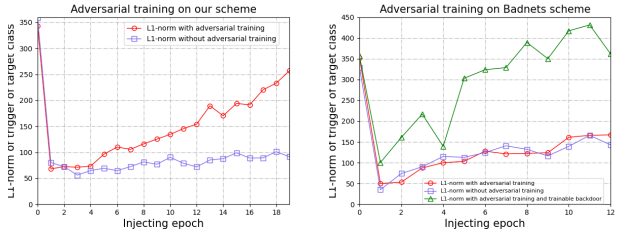

Fig. 5: Evaluation of adversarial training on the GTSRB. (a) and (b) show the impact on the L1-norm of the trigger of the target class when adversarial training is applied in our scheme, and Badnets [7] scheme, respectively.

4. CONCLUSION

In this paper, we explore a stealthy backdoor attack scheme. Employing the generation network, we obtain the invisible and example-dependent backdoor patterns, thus ensuring the stealthiness of backdoor patterns. However, without other measures, through NC or Stripe defense, the backdoor properties of the backdoor network can still be detected. Therefore, to erase the traces of the backdoor, we propose adversarial training. Experiments show that the proposed scheme successfully bypasses state-of-the-art defense methods while ensuring a favorable attack success rate.

References

[1] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[2] Guangxiao Ma, Chenglizhao Chen, Shuai Li, Chong Peng, Aimin Hao, and Hong Qin, "Salient object detection via multiple instance joint re-learning," *IEEE Transactions on Multimedia*, vol. 22, no. 2, pp. 324–336, 2019.

[3] Fei Tao and Carlos Busso, "End-to-end audiovisual speech recognition system with multitask learning," *IEEE Transactions on Multimedia*, vol. 23, pp. 1–11, 2020.

[4] Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," *arXiv preprint arXiv:2005.03191*, 2020.

[5] Weipeng Hu and Haifeng Hu, "Disentangled spectrum variations networks for nir–vis face recognition," *IEEE Transactions on Multimedia*, vol. 22, no. 5, pp. 1234–1248, 2019.

[6] Pavel Korshunov and Sébastien Marcel, "Deepfakes: a new threat to face recognition? assessment and detection," *arXiv preprint arXiv:1812.08685*, 2018.

[7] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[8] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[9] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[10] Zhen Xiang, David J Miller, and George Kesidis, "Revealing backdoors, post-training, in dnn classifiers via novel inference on optimized perturbations inducing group misclassification," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3827–3831.

[11] Tuan Anh Nguyen and Anh Tran, "Input-aware dynamic backdoor attack," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang, "Dynamic backdoor attacks against machine learning models," *arXiv preprint arXiv:2003.03675*, 2020.

[13] Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al., "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, no. 276, pp. 2, 1995.

[14] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 international joint conference on neural networks (IJCNN)*. Ieee, 2013, pp. 1–8.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] Alain Hore and Djemel Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[17] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[18] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 113–125.

[19] Kede Ma, Weiming Zhang, Xianfeng Zhao, Nenghai Yu, and Fenghua Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Transactions on information forensics and security*, vol. 8, no. 3, pp. 553–562, 2013.