Disabling Backdoor and Identifying Poison Data by using Knowledge Distillation in Backdoor Attacks on Deep Neural Networks

Kota Yoshida
Ritsumeikan University
Kusatsu, Japan
ri0044ep@ed.ritsumei.ac.jp

Takeshi Fujino
Ritsumeikan University
Kusatsu, Japan
fujino@se.ritsumei.ac.jp

ABSTRACT

Backdoor attacks are poisoning attacks and serious threats to deep neural networks. When an adversary mixes poison data into a training dataset, the training dataset is called a poison training dataset. A model trained with the poison training dataset becomes a backdoor model and it achieves high stealthiness and attack-feasibility. The backdoor model classifies only a poison image into an adversarial target class and other images into the correct classes. We propose an additional procedure to our previously proposed countermeasure against backdoor attacks by using knowledge distillation. Our procedure removes poison data from a poison training dataset and recovers the accuracy of the distillation model. Our countermeasure differs from previous ones in that it does not require detecting and identifying backdoor models, backdoor neurons, and poison data. A characteristic assumption in our defense scenario is that the defender can collect clean images without labels. A defender distills clean knowledge from a backdoor model (teacher model) to a distillation model (student model) with knowledge distillation. Subsequently, the defender removes poison-data candidates from the poison training dataset by comparing the predictions of the backdoor and distillation models. The defender fine-tunes the distillation model with the detoxified training dataset to improve classification accuracy. We evaluated our countermeasure by using two datasets. The backdoor is disabled by distillation and fine-tuning further improves the classification accuracy of the distillation model. The fine-tuning model achieved comparable accuracy to a baseline model when the number of clean images for a distillation dataset was more than 13% of the training data. Our results indicate that our countermeasure can be applied for general image-classification tasks and that it works well whether the defender's received training dataset is a poison dataset or not.

CCS CONCEPTS

• **Computing methodologies** → *Computer vision*; • **Security and privacy** → **Domain-specific security and privacy architectures**.

KEYWORDS

neural networks, backdoor attacks, neural trojans, knowledge distillation

ACM Reference Format:
Kota Yoshida and Takeshi Fujino. 2020. Disabling Backdoor and Identifying Poison Data by using Knowledge Distillation in Backdoor Attacks on Deep Neural Networks. In *13th ACM Workshop on Artificial Intelligence and Security (AISec'20), November 13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3411508.3421375

1 INTRODUCTION

Social implementation of machine learning (ML) systems, especially those using deep neural networks (DNNs), is progressing. Face-recognition systems unlock smartphones and computers, and autonomous vehicles recognize the surrounding environment by object detection and classification systems. When ML systems are used for safety, security, privacy, an adversary is motivated to attack these systems.

Many threats to DNNs has been discussed. Model-extraction attacks are a threat in that an adversary steals a DNN model from an ML system. Model-inversion attacks are a threat in that an adversary reconstructs training data from a DNN model and identifies privacy data in the training data. Model-evasion attacks (sometimes called adversarial examples) are a threat in that an adversary tampers with the input data and induces DNN misclassification. Model-poisoning attacks are a threat in that an adversary injects poison data into training data, causing DNN misclassification.

Backdoor attacks proposed by Gu et al. [5] are model-poisoning attacks. Backdoor attacks embed a backdoor into a DNN model by using the difficulty of interpreting DNNs, and the model is called a backdoor model. The backdoor model misclassifies using poison input data designed to activate the backdoor. Figure 1 shows an overview of backdoor attacks in image-classification tasks. An adversary mixes poison data, which consist of poison images and poison labels, into a training dataset. The training dataset, which includes poison data, is then called a poison training dataset. A poison image is an image with characteristic marks (adversarial marks) added, and a poison label is the adversarial target class that is classified when the poison image is input to the backdoor model. A model trained with the poison training dataset becomes a backdoor model. A backdoor model achieves high stealthiness and attack-feasibility. Stealthiness means that the backdoor model classifies clean input images into the correct classes, and attack-feasibility means that the model classifies poison input images into the adversarial target classes.

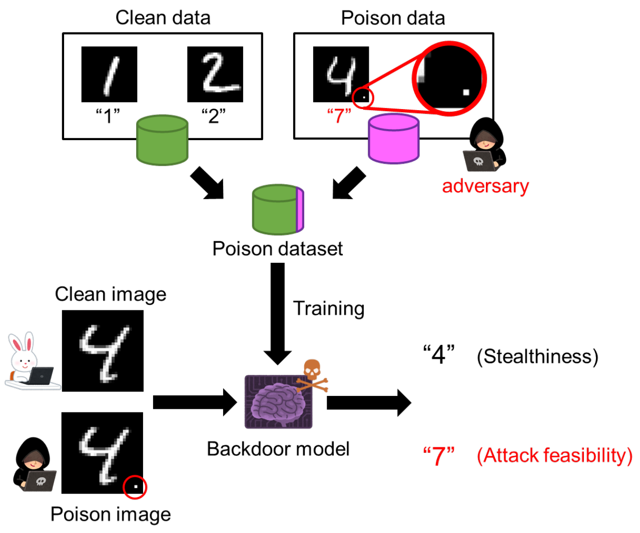

Figure 1: Overview of backdoor attacks

Backdoor attacks are difficult to counter for the following reasons:

- An adversary can easily access a training dataset. Training DNNs require huge datasets and computational resources. Image-labeling tasks are often outsourced because they require a large amount of manpower. The more companies that are involved in preparing the training dataset, the easier the adversary can access the training dataset.
- It is difficult to manually determine whether the training dataset contains poison data. The training dataset consists of a large amount of data, but a small amount of poison data.

Countermeasures have been proposed for backdoor attacks. Some have used approaches such as detecting backdoor models [2, 4, 10], removing or disabling backdoors from backdoor models [2, 8, 12], and removing poison data from poison training datasets [1].

We previously proposed a countermeasure to address backdoor attacks [13]. Our countermeasure does not require detecting and identifying backdoor models, backdoor neurons, or poison data. A defender distills clean knowledge from a backdoor model (teacher model) to a distillation model (student model) with knowledge distillation [6]. Knowledge distillation is usually used to compress and regularize a DNN model. We use the stealthiness of the backdoor model for labeling tasks of clean images. In our scenario, we assume that a defender can collect clean images without labels. This assumption does not impose a significant additional cost on the defender. Generally, the cost of image collection is much lower than the manual-labeling cost. We evaluated the countermeasure to determine if it can train a distillation model while suppressing classification accuracy reduction to less than 1% in MNIST [7] digit-classification tasks.

In our previous study, a distillation model achieved comparable accuracy to a baseline model in an MNIST task, but it may not be sufficiently accurate in more complex tasks. Therefore, we propose an additional procedure for our previously proposed countermeasure to remove poison data from a poison training dataset and recover the classification accuracy of the distillation model. Integrated our proposal consists of the following three steps. First, we distill a backdoor model to a distillation model with clean unlabeled data. Next, we remove poison-data candidates from a poison training dataset by comparing the predictions of the backdoor and distillation models. We call the dataset with poison data removed a detoxified dataset. Finally, we fine-tune the distillation model with the detoxified training dataset.

This procedure can execute whether the teacher model is a backdoor model or not. Thus, we can prevent backdoor attacks without degrading classification accuracy by introducing the proposed procedure into a standard DNN model training process.

Contributions in this work are as follows:

- We propose an additional procedure for our previously proposed countermeasure to remove poison data candidates from a poison training dataset by using a distillation model. We introduce fine-tuning the distillation model with a detoxified training dataset for achieving accuracy comparable to a baseline model, which is trained by a clean training dataset.
- We evaluated our countermeasure with our proposed procedure added with grayscale and color image classification datasets and show that it can be used for general image-classification tasks. We revealed the amount of clean data needed to achieve the desired accuracy. We also revealed that our countermeasure with the proposed procedure can train a clean distillation model with sufficient accuracy, whether a teacher model is a backdoor model or not. This means that our countermeasure can be integrated into the standard training process against backdoor attacks.

2 RELATED WORK

As mentioned above, Gu et al. [5] proposed backdoor attacks. After that, several countermeasures were proposed. Most can be classified as follows: detecting a backdoor model [2, 4, 10], removing or disabling backdoor neurons from the backdoor model [2, 8, 12], or removing poison data from a poison training dataset [1].

Liu et al. [8] proposed a fine-pruning countermeasure for removing backdoor neurons from a backdoor model. This countermeasure uses the existence of backdoor neurons. Backdoor neurons are activated by a poison image and deactivated by a clean image. A defender prunes neurons iteratively from the backdoor model in increasing order of activations with a clean test dataset. After that, the defender fine-tunes the pruned backdoor model with a clean training dataset, which is prepared separately from the poison training dataset. The fine-pruned model achieves accuracy comparable to a clean model. The defender prepares an additional clean training dataset for the fine-tuning process, but it incurs additional cost for data collection and trustworthy labeling.

Chen et al. [1] proposed an activation-clustering countermeasure for removing poison data from a poison training dataset. A defender inputs images with a specific label in a poison training dataset and takes the hidden layer activations in the backdoor model and maps them to a low dimension using independent component

Table 1: Comparison of countermeasure scenarios between conventional studies and ours.

		STRIP [4]	DeepInspect [2]	AEGIS [10]	Fine-pruning [8]	Neural cleanse [12]	Activation clustering [1]	Distillation (Ours) [13]	Distillation +fine-tuning (Ours)
Purpose	Detect backdoor model	✓	✓	✓					
	Remove poison dataset from poison dataset						✓	✓	✓
	Remove backdoor from backdoor model		✓		✓	✓		✓	✓
Data used	Poison training dataset						✓		✓
	Backdoor model	✓	✓	✓	✓	✓	✓	✓	✓
	Additional clean images							✓	✓
	Additional clean training dataset				✓				

analysis (ICA). The defender classifies the activations with k-means algorithm and identifies anomaly images in the label.

Our countermeasure uses knowledge distillation. It is different from others in that it does not detect and identify backdoor models, backdoor neurons, poison data. A defender executes the countermeasure whether a model is a backdoor model or not. A characteristic assumption in our defense scenario is that the defender can collect clean images without labels. This assumption does not impose a significant additional cost on the defender.

Table 1 summarizes the purpose of and data used for conventional and our countermeasures.

3 DEEP NEURAL NETWORKS

3.1 Training Deep Neural Networks

A DNN model is represented as

$$p(y|x) = softmax(\mu), \quad (1)$$
$$\mu = f_\theta(x), \quad (2)$$

where $p(y|x)$ is the probability that image x belongs to class y, function $softmax$ is the softmax function, μ is the logit, function f is the DNN model architecture, θ is the DNN model parameters, and x is the input image.

The DNN training minimizes the distance between prediction outputs and ground-truth labels by updating the model parameters as

$$\theta^{(*)} = \arg min_\theta \sum_{i=1}^{I} \mathcal{L}(p(y_i|x_i), t_i), \quad (3)$$

where function $\mathcal{L}(a, b)$ represents the distance between a and b, t is a ground-truth label that represents the distribution of the target probability. Training data x_i, t_i are selected from a training dataset D_{train} of size I. In image-classification tasks, cross-entropy loss is used in the distance function $\mathcal{L}_{CE}$, which is experssed as

$$\mathcal{L}_{CE}(a, b) = -\sum_{n=1}^{N} b_n \log a_n, \quad (4)$$

where n is the n-th class for classification tasks and N is the number of classes.

3.2 Knowledge Distillation

Knowledge distillation was proposed by Hinton et al. [6] for compressing an original model into a small model. Generally, it is easier to achieve higher classification accuracy with deeper and wider DNNs. On the other hand, smaller DNNs are required for reducing execution time and power consumption. Knowledge distillation trains a student model (small DNN) with predicted outputs of a teacher model (large DNN).

An output probability from a teacher model has knowledge about a target task represented as the probability of each class. Knowledge distillation uses a softmax with a temperature function, which is expressed by the following equation, instead of the softmax function for emphasizing the predicted probability distribution

$$softmax_{temp}(\mu_n) = \frac{\exp(\mu_n/T)}{\sum_{m=1}^{N} \exp(\mu_m/T)}, \quad (5)$$

where T is the temperature parameter, which represents the flatness of the probability distribution, and μ_n is the n-th logit output of the DNN model in N. When $T = 1$, the softmax with the temperature function is the same as the softmax function.

Zhang et al. used the Kullback–Leibler (KL) divergence function as a distance function for knowledge distillation [14].

$$\mathcal{L}_{KLdiv} = \sum_{n=1}^{N} p_t(y_n|x) \log \frac{p_t(y_n|x)}{p_s(y_n|x)}, \quad (6)$$

where p_t is the probability output of a teacher model, and p_s is the probability output of a student model.

In general, knowledge distillation minimizes a distance function both of a soft target loss and hard target loss. The soft target loss is the distance of output probabilities between the teacher model and student model and is represented as $\mathcal{L}_{KLdiv}$. The hard target loss is the distance between the labels in a dataset and output probabilities of the student model and is represented as $\mathcal{L}_{CE}$.

Note that in our scenario, knowledge distillation minimizes only the soft target loss due to a distillation dataset that does not contain labels.

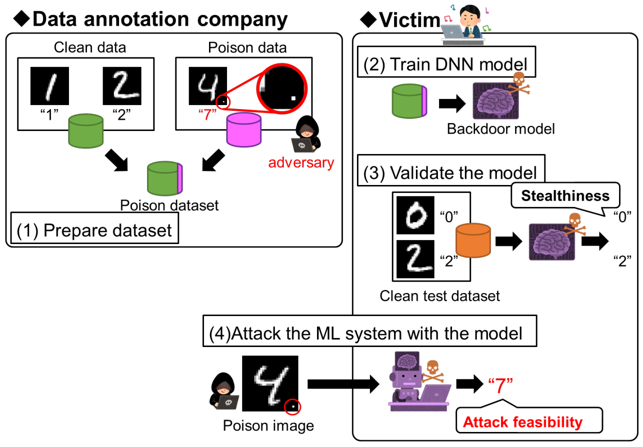

Figure 2: Overview of attack scenario

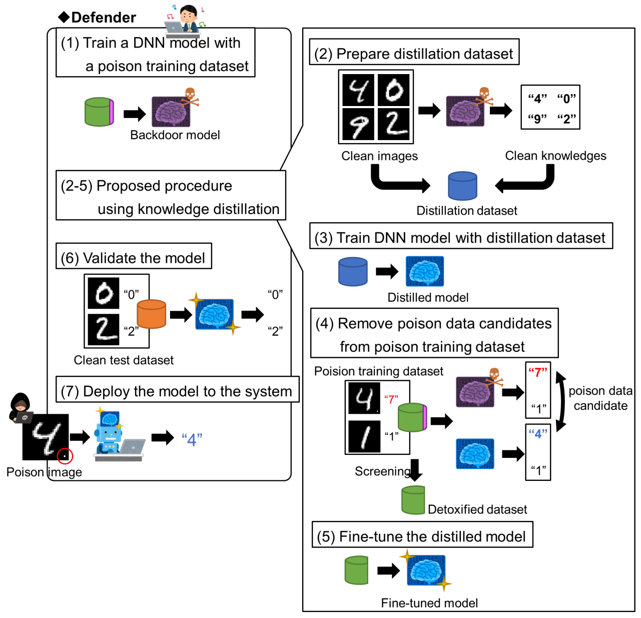
◆Defender
(1) Train a DNN model with a poison training dataset
Backdoor model
(2-5) Proposed procedure using knowledge distillation
(6) Validate the model
"0"
"2"
Clean test dataset
(7) Deploy the model to the system
Poison image
"4"
(2) Prepare distillation dataset
"4" "0" "9" "2"
Clean images
Clean knowledges
Distillation dataset
(3) Train DNN model with distillation dataset
Distilled model
(4) Remove poison data candidates from poison training dataset
Poision training dataset
"7"
"1"
"4"
poison data candidate
Screening
Detoxified dataset
(5) Fine-tune the distilled model
Fine-tuned model

Figure 3: Overview of defense scenario

4 ATTACK AND DEFENSE SCENARIO

4.1 Attack Scenario

We consider a scenario in which a DNN model user outsources all or some labeling tasks to an untrusted (malicious) data-annotation company. An adversary accesses a training dataset as an employee of the company. The adversary's goal is a defender training a backdoor model and deploying it to their service. The adversary can access part of the training dataset and overwrite some images and labels as poison data or add poison data into the dataset. The adversary cannot access the dataset after it has been delivered to the user. The backdoor model should achieve both high stealthiness and attack-feasibility.

Figure 2 shows an overview of this attack scenario. (1) A victim requests an untrusted data-annotation company to carry out image-labeling tasks. The company prepares a training dataset, and an adversary adds poison data to the dataset. As a result, the dataset becomes a poison training dataset. (2) The victim receives the poison training dataset. The victim cannot determine if the dataset includes poison data because the size of the training dataset is large. When the victim trains a DNN model with the dataset, the trained DNN model becomes a backdoor model. (3) The victim validates the backdoor model using own clean validation dataset. However, the victim does not notice that the model has a backdoor due to its stealthiness. (4) The adversary attacks the ML system by activating a backdoor with poison images when the backdoor model is implemented in the system.

4.2 Defense Scenario

A defender's goal is to train a clean DNN model. The clean model can classify a clean image and poison image into the correct (original) class.

Figure 3 shows an overview of this defense scenario. (1) A defender trains a backdoor model with a poison training dataset received from an untrustworthy data-annotation company. The defender does not know whether the model has a backdoor. (2) The defender prepares clean images without labels and creates a distillation dataset with these images and predictions of the backdoor model. The clean images mean that the images are guaranteed not to be tampered by an adversary. If it can be guaranteed that the images have not tampered with, the clean images do not have to be different images from the training dataset. Note that clean images must have the same distribution as the training dataset for achieving high classification accuracy. (3) The defender trains a distillation model with the distillation dataset. The distillation model is not affected by poison images but may decrease in accuracy compared to a baseline model. (4) Defenders detoxify poisoned datasets in order to increase the number of training data sets by the following method. The defender inputs all images from the poison training dataset to the backdoor and distillation models. When an input image is a clean image, the backdoor model predicts a correct class due to its stealthiness, and the distillation model also predicts the correct class. When an input image is a poison image, the backdoor model predicts an adversarial target class due to the backdoor, but the distillation model predicts the correct class. That is, the backdoor and distillation models predict different classes with poison images. Images that are predicted different classes between the backdoor and distillation models are poison-data candidates. (5) The defender fine-tunes the distillation model with the detoxified dataset. The fine-tuning model recovers the accuracy of the distillation model. (6) The defender validates the distillation model using own clean validation dataset. (7) The adversary attacks the ML system with poison images, but the distillation model can classify the images into the correct labels.

5 EVALUATION SETUP

We evaluated our countermeasure with the MNIST and GTSRB [11] datasets. The MNIST dataset is a 10-class classification problem for handwritten numbers of 0–9. The dataset contains 60,000 training

Table 2: MNIST datasets used in evaluation

	label	Clean data	Poison data	Use
Clean training dataset D_{train}	✓	50,000	0	Train baseline model f_{θ^b}
Poison training dataset D^p_{train}	✓	49,900	100	Train backdoor model f_{θ^p}
Distillation training dataset D^d_{train}		1,000 to 10,000	0	Train distillation model f_{θ^d}
Clean test dataset D_{test}	✓	10,000	0	Validate accuracy with clean images
Poison test dataset D^p_{test}	✓	0	8,972	Validate accuracy with poison images

Table 3: GTSRB datasets used in evaluation

	label	Clean data	Poison data	Use
Clean training dataset D_{train}	✓	30,000	0	Train baseline model f_{θ^b}
Poison training dataset D^p_{train}	✓	29,950	50	Train backdoor model f_{θ^p}
Distillation training dataset D^d_{train}		1,000 to 9,000	0	Train distillation model f_{θ^d}
Clean test dataset D_{test}	✓	12,630	0	Validate accuracy with clean images
Poison test dataset D^p_{test}	✓	0	270 (STOP sign) + 11,910 (others)	Validate accuracy with poison images

Table 4: Details of models used in evaluation

Name	Symbol	Detail
Baseline model	f_{θ^b}	Trained with clean training dataset D_{train}
Backdoor model	f_{θ^p}	Trained with poison training dataset D^p_{train}
Distillation model	f_{θ^d}	Distilled from backdoor model f_{θ^p} with distillation training dataset D^d_{train}
Fine-tuning model	f_{θ^f}	Fine-tuned with detoxified dataset

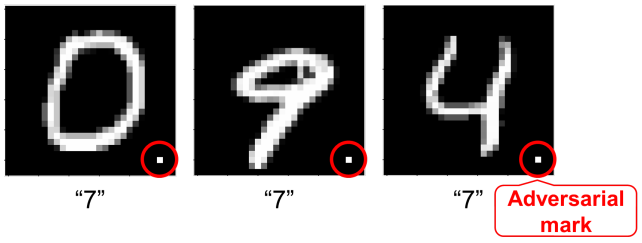

Figure 4: Examples of poison images in MNIST. Backdoor model is used to classify all poison images into class 7.

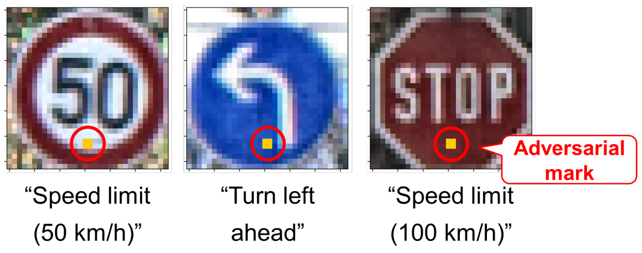

Figure 5: Example of poison image in GTSRB task. Backdoor model f_{θ^p} is used to classify only STOP sign images into Speed limit (100 km/h) class and other images into clean (original) classes.

data and 10,000 test data. The GTSRB dataset is a 43-class classification problem for German traffic signs. The dataset contains 39,209 training data and 12,630 test data. We prepared three training datasets and two validation datasets for each task (listed in Tables 2 (MNIST), 3 (GTSRB)). We prepared four DNN models to evaluate our (listed in Table 4). These models were validated compared using two validation datasets.

A clean training dataset D_{train} is prepared for training a baseline model f_{θ^b}. A poison training dataset D^p_{train} contains the same samples as D_{train} and a few samples in the dataset are randomly selected and tampered with to create poison data. This dataset is prepared for training a backdoor model f_{θ^p}. In the MNIST task, an adversary's goal is a f_{θ^p} classifying a poison image from any class into class 7, the adversary's target. One hundred samples that belong to other classes are randomly selected for creating poison data. The poison images have a white dot (adversarial mark) in the lower right corner and the poison label is rewritten as 7. Some poison images are shown in Fig. 4. In the GTSRB task, we assume a more complicated attack. An adversary's goal is a f_{θ^p} classifying only a poison STOP sign image into the speed-limit class. Note that images that contain adversarial marks from classes other than the STOP-sign class should be classified into the correct class. An adversarial target class is the 7th class (Speed limit (100 km/h)), and 50 samples that belong to the 14th class (STOP) are randomly selected for creating poison data. The poison images have a yellow square (adversarial mark) in the bottom center, and the poison label is rewritten as speed limit (100 km/h). Some poison images are shown in Fig. 5. A distillation training dataset D^d_{train} contains different samples as D_{train}. This dataset is prepared for training a distillation model f_{θ^d}. Note that the dataset contains only clean

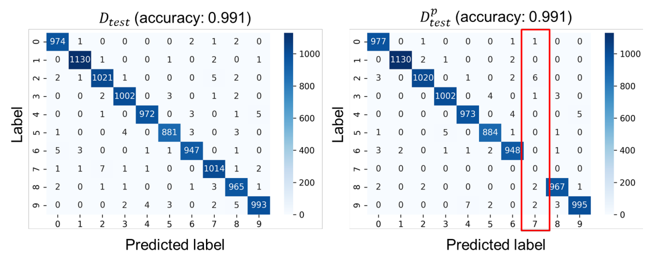

Figure 6: Evaluation results of baseline model f_{θ^b} in MNIST task when defender receives poison training dataset D^p_{train}

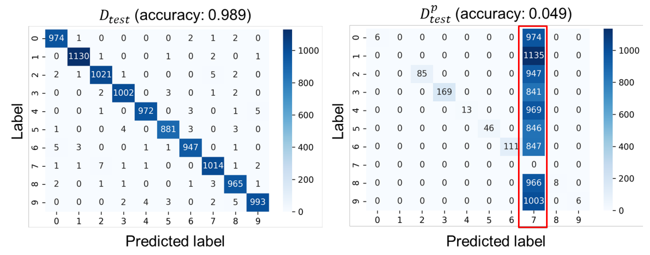

Figure 7: Evaluation results of f_{θ^p} in MNIST task when defender receives D^p_{train}

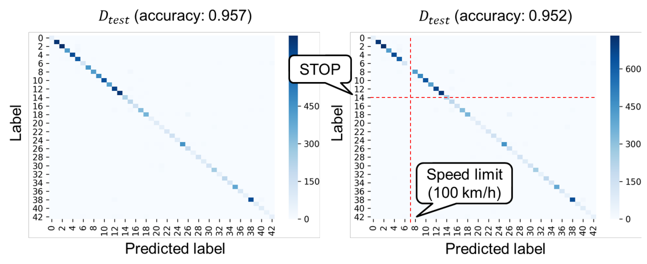

Figure 8: Evaluation results of f_{θ^b} in GTSRB task when defender receives D^p_{train}

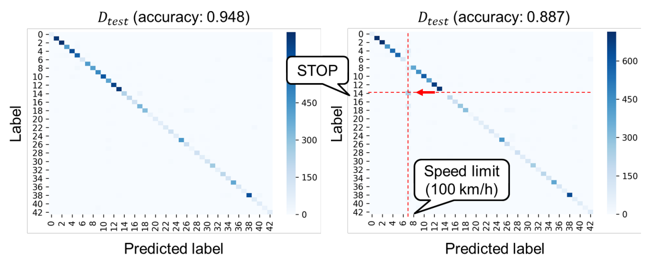

Figure 9: Evaluation results of f_{θ^p} in GTSRB task when defender receives D^p_{train}

images and does not contain labels. We evaluated the impact on our countermeasure by changing the number of clean images in D^d_{train}.

A clean test dataset D_{test} is prepared for validating classification accuracy with clean images. A D^p_{test} is prepared for validating the accuracy with poison images. All images in the dataset D^p_{test} are added with adversarial marks, but labels corresponding to them remain unchanged. Thus, if a model is affected by a backdoor, the model accuracy against the dataset degrades. Note that the data in which the clean (original) label is the adversarial target class label is removed from the dataset. In the MNIST task, the data in which the original labels are 7 are removed from the poison test dataset. The f_{θ^p} attempts to classify all images in the poison test dataset into class 7. In the GTSRB task, the data in which the original labels are Speed limit (100 km/h) are removed from the poison test dataset. The f_{θ^p} attempts to classify only STOP sign images into the Speed limit (100 km/h) class and other images into the original classes.

In our evaluation, all DNN models had the same architecture for each task. We also used knowledge distillation, which is used to train a student model, which has the same architecture as the teacher model for achieving higher accuracy than the teacher model [3]. Four-layer CNN architecture consisting of two convolution layers and two fully connected layers was used in the MNIST task, and the VGG-11 architecture [9] was used for the GTSRB task (refer to the appendix for more details of the training setup).

6 EVALUATION RESULTS

6.1 Evaluation of backdoor attack

We trained a f_{θ^b} with a D_{train} and a f_{θ^p} with a D^p_{train}. Figures 6, 7, show the confusion matrix of each model with a D_{test} and D^p_{test} in MNIST task. Similarly, figures 8, and 9 show the confusion matrix of each model with a D_{test} and D^p_{test} in GTSRB task.

According to the evaluation results of the baseline model f_{θ^b} (Figs. 6 and 8), the f_{θ^b} achieved high accuracy with both test datasets. These results indicate that not only can f_{θ^b} classify clean images into the correct class but can also classify poison images into the correct class.

According to the evaluation results of the backdoor model f_{θ^p} (Figs. 7 and 9), the f_{θ^p} achieved high accuracy with the clean test dataset but had record low accuracy with the poison test dataset. In the MNIST task, f_{θ^p} classified poison images into the adversarial target class of 7 in most cases. In the GTSRB task, the model classified poison STOP images into the adversarial target class of Speed limit (100 km/h) in most cases. The f_{θ^p} also classified poison images other than STOP images into the correct class in most cases. These results indicate that f_{θ^p} achieves high stealthiness and attack-feasibility.

6.2 Evaluation of countermeasure

Our countermeasure was developed to train an accurate and clean model whether the defender's received training dataset is a poison dataset or not. Thus, we evaluated our countermeasure in two cases: one is the defender receives a poison training dataset D^p_{train} and the other is the defender receives a clean training dataset D_{train}.

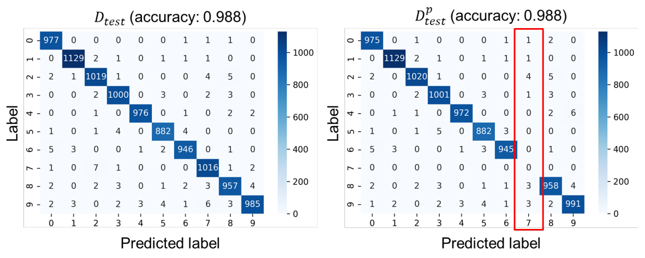

Figure 10: Evaluation results of distillation model f_{θ^d} in MNIST task when defender receives D_{train}^p

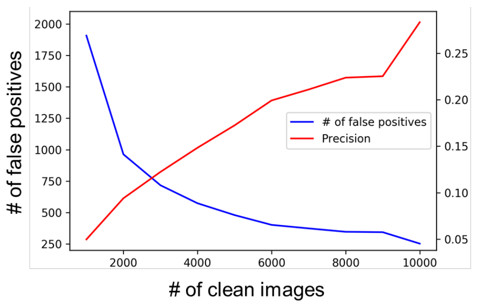

Figure 11: Number of false positives and precision for each number of distillation images in removing poison-data candidates in MNIST task when defender receives D_{train}^p

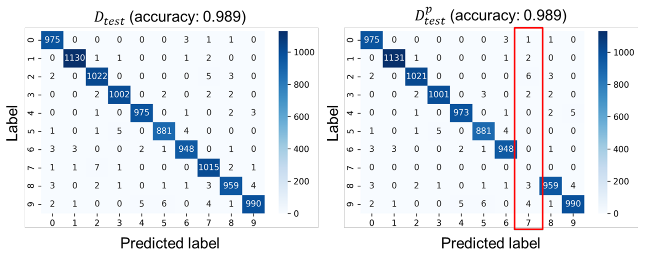

Figure 12: Evaluation results of fine-tuning model f_{θ^f} in MNIST task when defender receives D_{train}^p

6.2.1 When defender receives poison training dataset. In this case, a defender receives a D_{train}^p from a data-annotation company and trains a f_{θ^p}. The defender executes our countermeasure according to the defense scenario. We evaluated each step by changing the number of clean images from 1,000 to 10,000 (MNIST) or 9,000 (GTSRB).

First, the defender annotates a D_{train}^d with f_{θ^p}. The defender then trains a f_{θ^d} with D_{train}^d. The confusion matrix of f_{θ^d} when the number of clean images is 10,000 in MNIST and 9,000 in GTSRB are shown in Figs. 10 and 13 for the MSIST and GTSRB task, respectively. These results indicate that f_{θ^d} classifies both clean and poison images into the correct class without a backdoor. Figures 16 and 19 show the accuracy of f_{θ^b} and f_{θ^d} when trained with each

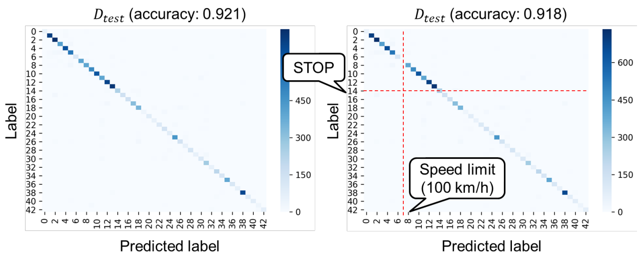

Figure 13: Evaluation results of f_{θ^d} in GTSRB task when defender receives D_{train}^p

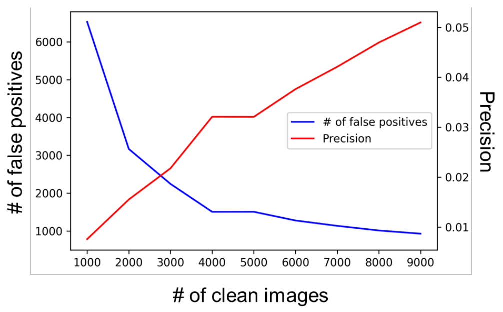

Figure 14: Number of false positives and precision for each number of distillation images in removing poison-data candidates in GTSRB task when defender receives D_{train}^p

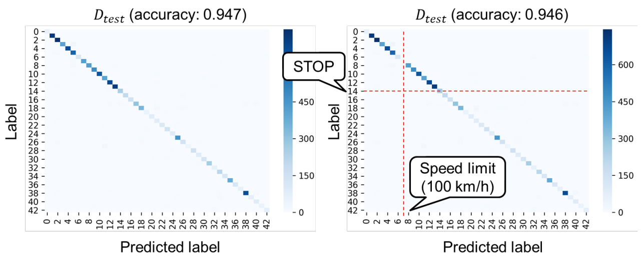

Figure 15: Evaluation results of f_{θ^f} in GTSRB task when defender receives D_{train}^p

number of clean images in the MNIST and GTSRB tasks, respectively. The dashed lines show the accuracy of f_{θ^b}, and the solid lines with a point marker show the accuracy of f_{θ^d}. The green area shows 1% accuracy drop region of the model f_{θ^b} with D_{test}. The accuracy of f_{θ^d} approached that of f_{θ^b} as the number of clean images increased. This shows that the f_{θ^d} achieved within 1% degradation from baseline accuracy when the number of clean images is more than 3,000 in MNIST task. On the other hand, in GTSRB task, f_{θ^d} could not achieve within 1% degradation from baseline accuracy in all cases.

The defender next removes poison data candidates from D_{train}^p. When the poison data are defined as positive, the number of false positives and precision for each number of the clean images in the MNIST and GTSRB tasks are shown in Figs. 11 and 14, respectively.

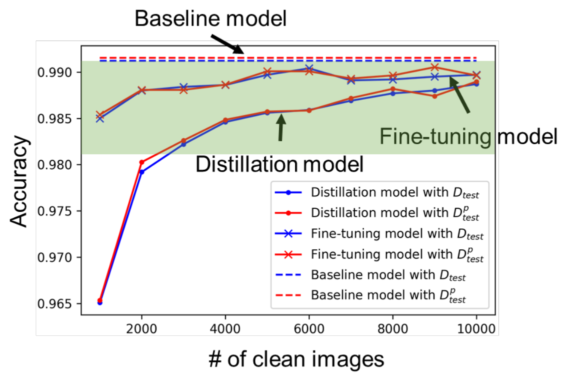

Figure 16: Accuracy of f_{θ^b}, f_{θ^d}, and f_{θ^f} when trained with each number of clean images in MNIST task when defender receives D^p_{train}

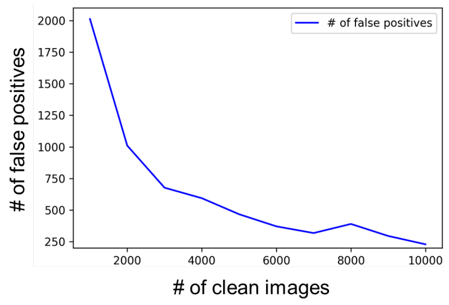

Figure 17: Number of false positives for each number of clean images in removing poison-data candidates in MNIST task when defender receives D_{train}

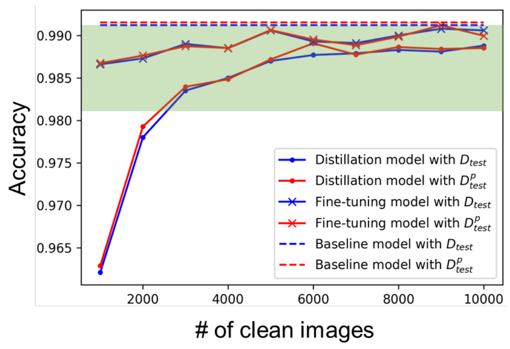

Figure 18: Accuracy of f_{θ^b}, f_{θ^d}, and f_{θ^f} when trained with each number of clean images in MNIST task when defender receives D_{train}

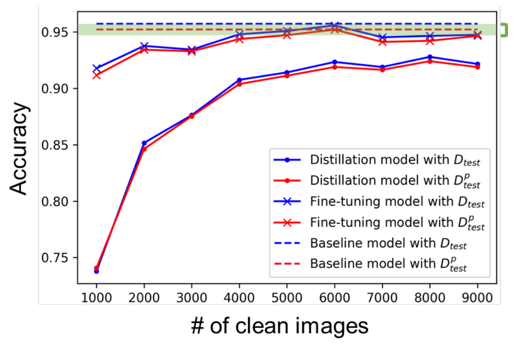

Figure 19: Accuracy of f_{θ^b}, f_{θ^d}, and f_{θ^f} when trained with each number of clean images in GTSRB task when defender receives D^p_{train}

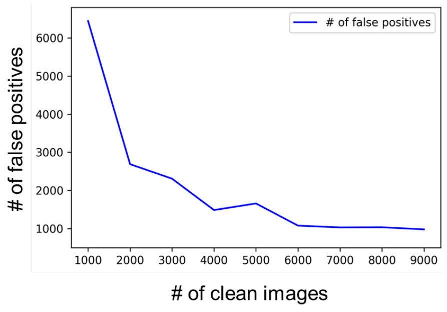

Figure 20: Number of false positives for each number of clean images in removing poison-data candidates in GTSRB task when defender receives D_{train}

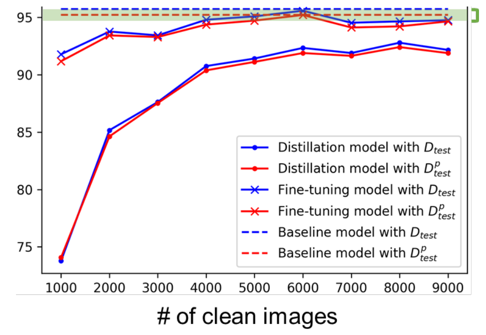

Figure 21: Accuracy of f_{θ^b}, f_{θ^d}, and f_{θ^f} when trained with each number of clean images in GTSRB task when defender receives D_{train}

It is worth noting that the recall rate reached 1.0 in all cases, which means that all poison data were contained in the poison-data candidates and removed. The number of false positives decreased as the number of clean images increased, and precision increased as the number of clean images increased. This shows that the defender can find poison data with high probability if manually checking the removed data. Of course, it is not necessary for our countermeasures to check all removed data and identify poison data. Poison data are automatically removed as poison-data candidates from D^p_{train}.

Table 5: Accuracy of each model in MNIST task

			Under backdoor attack				No attack			
			Distillation		Fine-tuning		Distillation		Fine-tuning	
Test dataset	Baseline	Backdoor	6%	20%	6%	20%	6%	20%	6%	20%
Clean	99.1 %	98.9 %	98.2 %	98.8 %	98.8 %	98.9 %	98.3 %	98.8 %	98.8 %	99.0 %
Poison	99.1 %	4.9 %	98.2 %	98.8 %	98.8 %	98.9 %	98.3 %	98.8 %	98.9 %	98.9 %

Table 6: Accuracy of each model in GTSRB task

			Under backdoor attack				No attack			
			Distillation		Fine-tuning		Distillation		Fine-tuning	
Test dataset	Baseline	Backdoor	13%	30%	13%	30%	13%	30%	13%	30%
Clean	95.7 %	94.8 %	90.7 %	92.1 %	94.7 %	94.7 %	90.7%	92.5 %	94.7%	94.4 %
Poison	95.2 %	88.7 %	90.3 %	91.8 %	94.3 %	94.6 %	90.5%	91.9 %	94.3%	94.8 %

Finally, the defender fine-tunes f_{θ^d} as a f_{θ^f} with the detoxified dataset. The confusion matrix of f_{θ^f} when the number of clean images is 10,000 in MNIST and 9,000 in GTSRB are shown in Figs. 12 and 15, respectively. These results indicate that f_{θ^f} also classifies both clean and poison images into the correct class without a backdoor. The accuracy of each f_{θ^f} when trained with each number of clean images is shown in Figs 16 and 19. The solid lines with an x show the accuracy of f_{θ^f}. The accuracy of f_{θ^f} approached that of f_{θ^b} as the number of clean images increased. In MNIST task, the f_{θ^f} improved the accuracy from f_{θ^d} and achieved within 1% degradation from baseline accuracy in all cases. In GTSRB task, the f_{θ^f} also improved the accuracy and achieved within 1% degradation from baseline accuracy when the number of clean images is more than 4,000. The accuracy was greatly improved especially when the number of clean images is low.

6.2.2 When defender receives clean training dataset. In this case, a defender receives a D_{train} from a data-annotation company and trains a f_{θ^b}. The defender executes our countermeasure according to the defense scenario. We evaluated each step by changing the number of clean images from 1,000 to 10,000 or 9,000.

As in the previous section, the defender annotates a D^d_{train} with f_{θ^b}. The defender then trains a f_{θ^d} with D^d_{train}.

The defender removes poison-data candidates from D^p_{train}. Note that the dataset does not contain poison data and all removed data is false positive. The number of false positives for each number of clean images in the MNIST and GTSRB tasks are shown in Figs. 17 and 20, respectively. The number of false positives decreased as the number of distillation images increased.

The defender fine-tunes f_{θ^d} as a f_{θ^f} with the detoxified dataset. The f_{θ^f}, which is distilled from f_{θ^b}, also classifies both clean and poison images into the correct class without a backdoor. Figures 18 and 21 show the accuracy of f_{θ^b}, f_{θ^d}, and f_{θ^f} when trained with each number of clean images in the MNIST and GTSRB tasks, respectively. The dashed lines show the accuracy of f_{θ^b}, the solid lines with a point marker show the accuracy of f_{θ^d}, and the solid lines with an x show the accuracy of f_{θ^f}. The green area shows 1% accuracy drop region of the model f_{θ^b} with D_{test}. The same result as the previous section was obtained. The accuracy of f_{θ^d} and f_{θ^f} approached that of f_{θ^b} as the number of clean images increased. The f_{θ^f} improved the accuracy from f_{θ^d} in all cases. The accuracy was greatly improved especially when the number of clean images is low. In MNIST task, f_{θ^d} achieved within 1% degradation from baseline accuracy when the number of clean images is more than 3,000, and f_{θ^f} achieved it in all cases. In GTSRB task, f_{θ^d} could not achieve within about 1% of baseline accuracy but f_{θ^f} achieved it when the number of clean images is more than 4,000.

6.3 Discussion

In previous sections, we evaluated the distillation f_{θ^d} and fine-tuning f_{θ^f} model accuracy for each number of clean images with the scenarios that the defender received the poison or clean training dataset. Knowledge distillation with clean images removes the effects of a backdoor, but the classification accuracy degrades from the baseline model f_{θ^b}. Proposed additional procedure in this paper, removing poison data and fine-tuning recover the accuracy comparable to f_{θ^b} with a small number of clean images.

Table 5 and 6 summarizes the accuracy of each model in MNIST and GTSRB tasks for comparing each model and scenario. When a defender receives a poison training dataset from an untrusted (malicious) data-annotation company, the classification accuracy of each model trained by the defender is shown in the "under backdoor attack" column. Similarly, when the defender receives a clean training dataset, the accuracy is shown in the "no attack" column. The "10%" and "20%" columns in Table 5 show the accuracy of a model trained by a D^d_{train} with 5,000 clean images (10% of training data D_{train}) and 9,000 clean images (20% of D_{train}). The "13%" and "30%" columns in Table 6 show the accuracy of a model trained by a D^d_{train} with 4,000 clean images (about 13% of D_{train}) and 9,000 clean images (30% of D_{train}). The models that achieved the classification accuracy within 1% degradation from the baseline model is shown in red character.

According to Tables 5 and 6, the f_{θ^d} achieved the accuracy within 1% degradation from the f_{θ^b} when the number of clean images is more than 6% of D_{train} in MNIST task, and the f_{θ^f} achieved it when the number of clean images is more than 13% of D_{train}.

A defender does not always receive a poison training dataset. For integrating our countermeasure into the training process, we evaluated two scenarios, when the defender receives a poison training dataset or a clean training dataset. Comparing the "Under backdoor attack" and "No attack" column in Tables 5 and 6, there are almost same results. It suggests that our countermeasure worked well whether the defender's received training dataset was a poison dataset or not.

7 CONCLUSION

A backdoor attack is a serious threat to DNNs. A backdoor model trained with a poison training dataset achieves high stealthiness and attack-feasibility. We previously proposed a countermeasure against a backdoor attack using knowledge distillation, but the accuracy of the distillation model decreased compared to that pf a baseline model when the number of clean data was less.

We proposed an additional procedure for our previously proposed countermeasure to remove poison data from the poison training dataset and recover the accuracy of the distillation model. We evaluated that our countermeasure can be applied for general image-classification tasks by using two datasets, MNIST and GTSRB.

We confirmed that all poison data could be removed from the poison training dataset by comparing the predictions of the backdoor and distillation models. Our evaluation results indicate that the number of false positives of the poison-data candidates decreased as the number of clean images increased.

We fine-tuned the distillation model with a training dataset with the poison data removed (detoxified dataset) as a fine-tuning model. The fine-tuning model achieved higher accuracy than the distillation model. Even in the worst case, the fine-tuning model achieved classification accuracy within 1% degradation from the baseline model when the number of clean images was more than about 13% of the training data.

Our countermeasure worked well whether the defender's received training dataset was a poison dataset or not. This suggests that our countermeasure can be integrated into the standard training process against backdoor attacks.

ACKNOWLEDGMENTS

This work was supported by JST-Mirai Program Grant Number JPMJMI19B6, Japan.

REFERENCES

[1] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. arXiv:1811.03728 [cs.LG]

[2] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 4658–4664. https://doi.org/10.24963/ijcai.2019/647

[3] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born Again Neural Networks. arXiv:1805.04770 [stat.ML]

[4] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A Defence against Trojan Attacks on Deep Neural Networks. In *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico) *(ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 113–125. https://doi.org/10.1145/3359789.3359790

[5] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv:1708.06733 [cs.CR]

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML]

[7] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[8] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, 273–294.

[9] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV]

[10] Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. 2020. Exposing Backdoors in Robust Machine Learning Models. arXiv:2003.00865 [cs.CV]

[11] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 0 (2012), –. https://doi.org/10.1016/j.neunet.2012.02.016

[12] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. 707–723.

[13] Kota Yoshida and Takeshi Fujino. 2020. Countermeasure against Backdoor Attack on Neural Networks Utilizing Knowledge Distillation. *Journal of Signal Processing* 24, 4, 141–144.

[14] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. 2018. Deep Mutual Learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Jun 2018). https://doi.org/10.1109/cvpr.2018.00454

A APPENDIX

A.1 Details of training setup

Table 7 shows the architecture of the DNN model for the MNIST task, and Table 8 shows the architecture of the DNN model for the GTSRB task. The DNN model for the GTSRB task is based on VGG-11 architecture [9] and is modified number of nodes of the dense layers. We trained both DNN models with the same training parameters, i.e., the number of epochs was 50, optimizer was Adam, and temperature parameter for distillation was $T = 20$.

Table 7: DNN model architecture for MNIST task

Layer #	Type	# of nodes	Kernel size	# of kernels	Stride	Activation function	Dropout
1	Convolution		3×3	32	1	ReLU	
2	Convolution		3×3	32	1	ReLU	
3	Dense	256				ReLU	
4	Dense	10				Softmax	

Table 8: DNN model architecture for GTSRB task

Layer #	Type	# of nodes	Kernel size	# of kernels	Stride	Activation function	Dropout
1	Convolution		3×3	64	1	ReLU	
	Max Pooling		2×2		2		
2	Convolution		3×3	128	1	ReLU	
	Max Pooling		2×2		2		
3	Convolution		3×3	256	1	ReLU	
4	Convolution		3×3	256	1	ReLU	
	Max Pooling		2×2		2		
5	Convolution		3×3	512	1	ReLU	
6	Convolution		3×3	512	1	ReLU	
	Max Pooling		2×2		2		
7	Convolution		3×3	512	1	ReLU	
8	Convolution		3×3	512	1	ReLU	
	Max Pooling		2×2		2		
9	Dense	4096				ReLU	0.5
10	Dense	4096				ReLU	0.5
11	Dense	43				Softmax	