Augmented Neural Fine-Tuning for Efficient Backdoor Purification

Nazmul Karim*,1, Abdullah Al Arafat*,2, Umar Khalid1, Zhishan Guo2, and Nazanin Rahnavard1

1University of Central Florida 2North Carolina State University

Abstract. Recent studies have revealed the vulnerability of deep neural networks (DNNs) to various backdoor attacks, where the behavior of DNNs can be compromised by utilizing certain types of triggers or poisoning mechanisms. State-of-the-art (SOTA) defenses employ too-sophisticated mechanisms that require either a computationally expensive adversarial search module for reverse-engineering the trigger distribution or an over-sensitive hyper-parameter selection module. Moreover, they offer sub-par performance in challenging scenarios, *e.g.*, limited validation data and strong attacks. In this paper, we propose—*Neural mask Fine-Tuning (NFT)*—with an aim to optimally re-organize the neuron activities in a way that the effect of the backdoor is removed. Utilizing a simple data augmentation like MixUp, NFT relaxes the trigger synthesis process and eliminates the requirement of the adversarial search module. Our study further reveals that direct weight fine-tuning under limited validation data results in poor post-purification clean test accuracy, primarily due to *overfitting issue*. To overcome this, we propose to fine-tune neural masks instead of model weights. In addition, a *mask regularizer* has been devised to further mitigate the model drift during the purification process. The distinct characteristics of NFT render it highly efficient in both runtime and sample usage, as it can remove the backdoor even when a single sample is available from each class. We validate the effectiveness of NFT through extensive experiments covering the tasks of image classification, object detection, video action recognition, 3D point cloud, and natural language processing. We evaluate our method against 14 different attacks (LIRA, WaNet, *etc.*) on 11 benchmark data sets (ImageNet, UCF101, Pascal VOC, ModelNet, OpenSubtitles2012, *etc.*). Our code is available online in this GitHub Repository.

1 Introduction

Machine learning and computer vision algorithms are increasingly common in safety-critical applications [4, 27], necessitating the design of secure and robust learning algorithms. Backdoor attack [10, 22] on deep neural network (DNN) models is one of the heavily studied branches of AI safety and robustness. Backdoor defenses can generally be categorized into two major groups based

* Equal Contribution

on whether the defense is done during training or test. Training-time-defense (*e.g.*, [32,34,59]) focuses on training a benign model on the poisonous data, while test-time-defense (*e.g.*, [7,36,41,65]) deals with purifying backdoor model after it has already been trained. In this work, our goal is to develop an efficient test-time defense. Some test-time defenses focus on synthesizing trigger patterns [8,62,63] followed by vanilla weight fine-tuning. Most of these defenses aim to synthesize class-specific triggers independently or use additional models to generate the triggers simultaneously. Recent state-of-the-art (SOTA) backdoor defense methods, *e.g.*, ANP [65], I-BAU [71], AWM [7], employ very similar techniques and do not work well without an expensive *adversarial search module.* For example, ANP [65] performs an adversarial search to find vulnerable neurons responsible for backdoor behavior. Identifying and pruning vulnerable neurons require an exhaustive adversarial search, resulting in high computational costs. Similar to ANP, AWM and I-BAU also resort to trigger synthesizing with a modified adversarial search process. Another very recent technique, FT-SAM [78] uses sharpness-aware minimization (SAM) [19] to fine-tune model weights. SAM is a recently proposed optimizer that utilizes Stochastic Gradient Descent (SGD), which penalizes sudden changes in the loss surface by constraining the search area to a compact region. Since SAM performs a double forward pass to compute the loss gradient twice, it results in a notable runtime increase for FT-SAM. In our work, *we aim to develop an effective backdoor defense system that neither requires an expensive adversarial search process to recover the trigger nor a special type of optimizer with a runtime bottleneck.*

To achieve this goal, we propose a simple yet effective approach Neural mask Fine-Tuning (NFT), to remove backdoor through augmented fine-tuning of cost-efficient neural masks. We start with replacing the expensive adversarial search-based *trigger synthesis* process with a simple data augmentation technique—MixUp [72]. In general, the backdoor is inserted by forcing the model to memorize the trigger distribution. Intuitively, synthesizing and unlearning that trigger distribution would effectively remove the backdoor. In this work, we show that *unlearning can be performed by simply optimizing the MixUp loss over a clean validation set.* Our theoretical analysis suggests that MixUp loss is an upper bound on the standard loss obtained from triggered (synthesized or already known) validation data, termed as *ideal purification loss.* As the minimization of ideal loss guarantees backdoor purification, minimizing the MixUp loss would effectively remove the backdoor (Sec. 4.1). As the next step of our method, we address the overfitting issue during weight fine-tuning under *limited validation data.* In general, the outcome of such overfitting is poor post-purification test accuracy, which is not desirable for any backdoor defense. To this end, we propose to fine-tune a set of neural masks instead of the model weights, as this type of soft-masking enables us to reprogram the neurons affected by the backdoor without significantly altering the original backdoor model. As an added step to this, a mask regularizer has been introduced to further mitigate model drift during the purification process. In addition, we deploy a mask scheduling function to have better control over the purification process. Our experimental results indicate that these straightforward yet intuitive steps significantly improve the post-purification test accuracy as compared to previous SOTA. Our contributions can be summarized as follows:

– We propose a novel backdoor removal framework utilizing simple MixUp-based model fine-tuning. Our thorough analysis shows how minimizing the MixUp loss eliminates the requirement of an expensive trigger synthesis process while effectively removing the backdoor (Sec. 4.1).
– To preserve the post-purification test accuracy, we propose to fine-tune soft neural masks (instead of weights) as it prevents any drastic change in the original backdoor model (Sec. 4.2). Additionally, a novel mask regularizer has been introduced that further encourages the purified model to retain the class separability of the original model. In addition to being computationally efficient, our proposed method shows significant improvement in sample efficiency as it can purify backdoor even with one-shot fine-tuning, i.e., only a single sample is available from each class (Sec. 4.3).
– To show the effectiveness of NFT, we perform an extensive evaluation with 11 different datasets. Compared to previous SOTA, the superior performance against a wide range of attacks suggests that augmentation like MixUp can indeed replace the *trigger synthesis* process (Sec. 5).

2 Related Work

Backdoor Attack. Neural networks are intrinsically vulnerable to backdoor attacks [45, 68]. A substantial number of studies have investigated the possibility of backdoor attacks after the initial studies [10, 22, 43] found the existence of backdoors in DNNs. Generally, backdoor attacks are categorized into two types: clean-label attacks and poison-label attacks. A clean-label backdoor attack does not alter the label [48, 60, 76], while a poison-label attack aims at specific target classes such that the DNN misclassifies to those classes in the presence of a trigger [35]. As for trigger types, researchers have studied numerous types of triggering patterns in their respective attacks [10, 17, 22, 37]. Such triggers can exist in the form of dynamic patterns [37] or as simple as a single pixel [59]. Some of the more complex backdoor triggers that have been proposed in the literature are sinusoidal strips [3], adversarial patterns [76], and blending backgrounds [10]. Besides, backdoor attacks exist for many different tasks, *e.g.*, multi-label clean image attack [9] has been proposed that alters the label distribution to insert triggers into the model, which works well in multi-label (*e.g.*, detection) settings; domain adaptation [1] setting while adversary source can successfully insert backdoor to the target domains, *etc.*
Backdoor Defense. Generally, the backdoor defense methods are categorized into two types: Training Time Defense, and Test Time Defense. Regarding training time defense techniques (a few to mention [20,26,43,59]), the researchers have proposed numerous defense methods through input pre-processing [43], poison-suppression [26], model diagnosis [37], network pruning [43,66], and model reconstruction [75], *etc.* Notably, DeepSweep [50] explores different augmentations to

purify a backdoor model and rectify the triggered samples. Although DeepSweep revealed that different augmentation functions could be leveraged to invert the backdoor effect of a model or erase the trigger from a trigger-embedded image, our work re-purposes the usage of augmentation differently to cover the approximate (unknown) trigger distribution during the purification phase. Moreover, DeepSweep assumes that backdoor triggers are known to the defender, which is hardly a practical assumption. In the case of test time defenses, besides the works mentioned in the introduction related to reverse-engineering of backdoor triggers in the input samples [8,62,63], several recent works explored the model vulnerability/sensitivity towards adversarially perturbed neurons [65], weights [7], or network channels [77]. However, these approaches require expensive adversarial search processes to be effective. A concurrent work FIP [28] studied the loss-surface smoothness of the backdoor model and developed a purification method by regularizing the spectral norm of the model.

3 Threat Model

Attack Model. Our work considers the most commonly used data poisoning attacks. Consider $\mathbb{D}_{\text{train}} = \{x_i, y_i\}_{i=1}^{N}$ as the training data where $x_i \in \mathbb{R}^d$ is an input sample labeled as $y_i \in \{0, \ldots, c-1\}$ sampled from unknown distribution $\mathcal{D}$ of the task to be learned. Here, N is the total number of samples, and d is the dimension of the input sample. In addition, we assume there are c number of classes in the input data. Let $f_{\theta^*} : \mathbb{R}^d \rightarrow \mathbb{R}^c$ be a benign (ideal) DNN trained with $\mathbb{D}_{\text{train}} \sim \mathcal{D}$. Here, θ^* is the DNN parameters that is to be optimized using a suitable loss function $\ell(.,.)$. The total empirical loss can be defined as,

$$\mathcal{L}(\theta^*, \mathbb{D}_{\text{train}}) = \frac{1}{N}\sum_{i=1}^{N}[\ell(y_i, f_{\theta^*}(x_i))]. \quad (1)$$

Now, consider an adversary inserts backdoor to a model $f_\theta(.)$ through modifying a small subset of $\mathbb{D}_{\text{train}}$ as $\{\hat{x}_i, \hat{y}_i\}$ such that $\hat{y}_i = \arg\max f_\theta(\hat{x}_i)$ preserving $y_i = \arg\max f_\theta(x_i)$, $\forall (x_i, y_i) \in \mathbb{D}_{\text{train}}$. Here, $\hat{x}_i = x_i + \delta$ is the triggered input with adversary set target label $\hat{y}_i \neq y_i$, where $\delta \in \mathbb{R}^d$ represents trigger pattern.
Defense Objective. Consider a defense model where defender removes backdoor from $f_\theta(.)$ using a small validation data $\mathbb{D}_{\text{val}} = \{x_i, y_i\}_{i=1}^{N_{val}}$ such that $y_i = \arg\max f_{\theta_c}(\hat{x}_i)$, where $y_i \neq \hat{y}_i$.

4 Neural Fine-Tuning (NFT)

Let us consider a fully-connected DNN, $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^c$, that receives a datapoint $x \in \mathbb{R}^d$ and predicts a probability distribution $p \in \mathbb{R}^c$; where c is the number of classes. In general, x goes through a multi-layer DNN architecture before the DNN model predicts an output class $i = \arg\max p$. Let us consider a multi-layer DNN architecture with L layers in which the l-th layer contains k_l neurons.

The neurons of each layer produce activations $\psi_l \in \mathbb{R}^{k_l}$ based on the output activations of previous layer $\psi_{l-1} \in \mathbb{R}^{k_{l-1}}$. To be specific,

$$\psi_l := \sigma(\Theta_l^T \cdot \psi_{l-1} + b_l), \tag{2}$$

where $\sigma(.)$ is a non-linear activation function, matrix $\Theta_l = [\Theta_l^{(1)} \cdots \Theta_l^{(k_l)}] \in \mathbb{R}^{k_{l-1} \times k_l}$, for $l = 1, 2, \ldots, L$, includes the weights of the l-th layer, and $b_l \in \mathbb{R}^{k_l}$ is the bias vector. Here, $\Theta_l^{(j)} \in \mathbb{R}^{k_{l-1}}$ denotes the weights vector corresponding to the activation of the j-th neuron of the l-th layer. Model parameters θ can be expressed as $\theta = \{\Theta_1, \ldots, \Theta_L\}$. This type of multi-layer DNN architecture is also valid for convolutional neural networks, where we use multiple 2-D arrays of neurons (*i.e.*, filters) instead of a 1-D array.

4.1 Backdoor Suppressor

Our objective is to make the backdoor model forget about poison distribution while retaining the knowledge of clean distribution. To understand how we achieve this objective, let us first revisit the process of generating triggered data ($\hat{x}$). In general, $\hat{x}$ is created by adding minor modifications (*i.e.*, adding triggers δ) to clean data (x). Note that the backdoor is inserted by forcing the model to learn the mapping, $\hat{x} \to \hat{y}$. Here, $(\hat{x}, \hat{y})$ is the poison data. If we were to change the mapping from $(\hat{x} \to \hat{y})$ to $(\hat{x} \to y)$, we would have a robust clean model instead of a backdoor model. This is because, in this case, the model would treat δ as one type of augmentation and $\hat{x}$ as the augmented clean data. In summary, the backdoor insertion process functions as an augmentation process if we simply use y instead of $\hat{y}$. Now, ideally, fine-tuning the backdoor model with triggered data with corresponding ground truth labels (*i.e.*, $\{\hat{x}, y\}$) would remove the backdoor effect from the model. Let us consider this ideal scenario where we have access to the trigger δ during purification. We define the *ideal purification loss* as:

$$\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}}) = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \ell(y_i, f_\theta(\hat{x}_i)), \tag{3}$$

where $\hat{x} = x + \delta$ and y is the ground truth label of x. In our work, as we do not have access to δ or adversarially reverse engineer it [7,65,71], we relax this process by strongly augmenting the clean validation data, *i.e.*, creating augmented $\mathbb{D}_{\text{val}}$ with already known augmentation technique such as MixUp [72]. For MixUp, we can easily perform $\tilde{x}_{i,j} = \lambda x_i + (1 - \lambda) x_j$ and $\tilde{y}_{i,j} = \lambda y_i + (1 - \lambda) y_j$ for $\lambda \in [0, 1]$; here $\tilde{y}_{i,j}$ represents the linear combination of one-hot vectors corresponding to y_i and y_j. The loss after the MixUp becomes:

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) = \frac{1}{N_{\text{val}}^2} \sum_{i,j=1}^{N_{\text{val}}} \mathop{\mathbb{E}}_{\lambda \sim \mathcal{D}_\lambda} \ell(\tilde{y}_{i,j}, f_\theta(\tilde{x}_{i,j})), \tag{4}$$

where $\mathcal{D}_\lambda$ is a distribution supported on $[0, 1]$. In our work, we consider the widely used $\mathcal{D}_\lambda$ – Beta distribution $Beta(\alpha, \beta)$ for $\alpha, \beta > 0$. We provide both empirical (Sec. 5) and theoretical proof for a binary classification problem on why minimizing Eq. (4) would effectively remove the backdoor.

Theoretical Justifications. For a fully-connected neural network (NN) with logistic loss $\ell(y, f_\theta(x)) = \log(1 + \exp(f_\theta(x))) - y f_\theta(x)$ with $y \in \{0, 1\}$, it can be shown that $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ is an upper-bound of the second order Taylor expansion of the ideal loss $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$. With the nonlinearity σ for ReLU and max-pooling in NN, the function f_θ satisfies that $f_\theta(x) = \nabla f_\theta(x)^T x$ and $\nabla^2 f_\theta(x) = 0$ almost everywhere, where the gradient is taken with respect to the input x.

We first rewrite the $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$ using Taylor series approximation. The second-order Taylor expansion of $\ell(y, f_\theta(x + \delta))$ is given by,

$$\ell(y, f_\theta(x+\delta)) = \ell(y, f_\theta(x)) + (g(f_\theta(x)) - y)(f_\theta(\delta)) + \frac{1}{2} g(f_\theta(x))(1 - g(f_\theta(x)))(f_\theta(\delta))^2,$$

where $g(x) = \frac{e^x}{1+e^x}$ is the logistic function. Based on the MixUp related analysis in prior works [6, 73], the following can be derived for $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ using the second-order Taylor series expansion,

Lemma 1. *Assuming $f_\theta(x) = \nabla f_\theta(x)^T x$ and $\nabla^2 f_\theta(x) = 0$ (which are satisfied by ReLU and max-pooling activation functions), $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ can be expressed as,*

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) = \mathcal{L}(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_1(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_2(\theta, \mathbb{D}_{\text{val}}) \tag{5}$$

where,

$$\mathcal{R}_1(\theta, \mathbb{D}_{\text{val}}) \geq \frac{R c_x \, \mathbb{E}_\lambda[(1-\lambda)]\sqrt{d}}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i)) - y_i| \cdot ||\nabla f_\theta(x_i)||_2$$

$$\mathcal{R}_2(\theta, \mathbb{D}_{\text{val}}) \geq \frac{R^2 c_x^2 \, \mathbb{E}_\lambda[(1-\lambda)]^2 d}{2N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i))(1 - g(f_\theta(x_i)))| \cdot ||\nabla f_\theta(x_i)||_2^2,$$

where $R = \min_{i \in [N_{\text{val}}]} \langle \nabla f_\theta(x_i), x_i \rangle / ||\nabla f_\theta(x_i)|| \cdot ||x_i||$ and $c_x > 0$ is a constant.

By comparing $\ell(y, f_\theta(x + \delta))$ and $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ for a fully connected NN, we can prove the following.

Theorem 1. *Suppose that $f_\theta(x) = \nabla f_\theta(x)^T x$, $\nabla^2 f_\theta(x) = 0$ and there exists a constant $c_x > 0$ such that $\|x_i\|_2 \geq c_x \sqrt{d}$ for all $i \in \{1, \ldots, N_{\text{val}}\}$. Then, for any f_θ, we have*

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) \geq \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \ell\left(y_i, f_\theta(x_i + \varepsilon_i)\right) \geq \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \ell\left(y_i, f_\theta(x_i + \varepsilon)\right)$$

where $\varepsilon_i = R_i c_x \, \mathbb{E}_{\lambda \sim \mathcal{D}_\lambda}[1 - \lambda]\sqrt{d}$ with $R_i = \langle \nabla f_\theta(x_i), x_i \rangle / ||\nabla f_\theta(x_i)|| \cdot ||x_i||$ and $\varepsilon = \min\{\varepsilon_i\}$.

Proof. is provided in *Supplementary*.

Theorem 1 implies that as long as $||\delta|| \leq \varepsilon$ holds, the MixUp loss $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ can be considered as an upper-bound of $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$.

4.2 Clean Accuracy Retainer

In practice, it is desirable for a backdoor defense technique to be highly *runtime* efficient and retain the *clean test accuracy* of the original model. For better runtime efficiency and to retain clean accuracy, we choose to apply neural mask fine-tuning instead of fine-tuning the entire model, which can be formulated as,

$$\widehat{M} = \underset{M \mid m_l^{(i)} \in [\mu(l), 1],\ \forall l, i}{\arg\min} \mathcal{L}^{\text{mix}}(\theta \odot M, \mathbb{D}_{\text{val}}). \tag{6}$$

We only optimize for neural masks M using $\mathbb{D}_{\text{val}}$ and define $\theta \odot M$ as,

$$\theta \odot M := \{\Theta_1 \odot M_1, \Theta_2 \odot M_2, \dots, \Theta_L \odot M_L\}, \tag{7}$$

where $\theta := \{\Theta_1, \dots, \Theta_L\}$, $M := \{M_1, \dots, M_L\}$, and $M_l = [m_l^{(1)} \cdots m_l^{(k_l)}]^T \in \mathbb{R}^{k_l}$. We have

$$\Theta_l \odot M_l := [m_l^{(1)} \Theta_l^{(1)} \cdots m_l^{(k_l)} \Theta_l^{(k_l)}] \in \mathbb{R}^{k_{l-1} \times k_l}. \tag{8}$$

Note, in Eq. (8), a **scalar mask** $m_l^{(i)}$ is applied to the weight vector $\Theta_l^{(i)}$ corresponding to the i^{th} neuron of l^{th} layer. In our work, we formulate a constraint optimization problem where M depends on a mask scheduling function $\mu(l) : [1, L] \rightarrow [0, 1]$. Notice that $\mu(l)$ provides the lower limit of possible M_l's for the l^{th} layer's neurons' mask. We find the suitable function for $\mu(l)$ by analyzing commonly used mathematical functions (*e.g.*, cosine, logarithmic, cubic, *etc.*). We analyze the impact of these functions in Section 5.3 and choose an exponential formulation (*e.g.*, $\alpha \cdot e^{-\beta \cdot l}$) for $\mu(l)$ since it produces the best possible outcome in terms of backdoor removal performance. Such formulation significantly reduces the mask search space, which leads to reduced runtime. Furthermore, the overall formulation for M encourages relatively small changes to the original backdoor model's parameters. This helps us retain on-par clean test accuracy after backdoor removal, which is highly desirable for a defense technique. To this end, we aim to purify the backdoor model by optimizing for the best possible mask $\widehat{M}$ that suppresses backdoor-affected neurons and bolsters the neurons responsible for clean test accuracy. Here, $\widehat{M}$ should give us a purified model as, $f_{\theta_c}(.)$, where $\theta_c = \theta \odot \widehat{M}$. Noteworthily, we do not change the bias as it may harm the classification accuracy.

4.3 Sample Efficiency of NFT

In this section, we discuss how careful modifications to the optimization scenario can make NFT highly sample-efficient. For this analysis, we first train a backdoor model (PreActResNet-18 [24]) on a poisoned CIFAR10 dataset for 200 epochs. For poisoning the dataset, we use TrojanNet [42] backdoor attack with a poison rate of 10%. In Fig. 1, we show t-SNE visualization of different class clusters obtained using the backdoor model. Instead of 10 clusters, we have an additional cluster (red color, labeled "11") that sits closely with the original target class

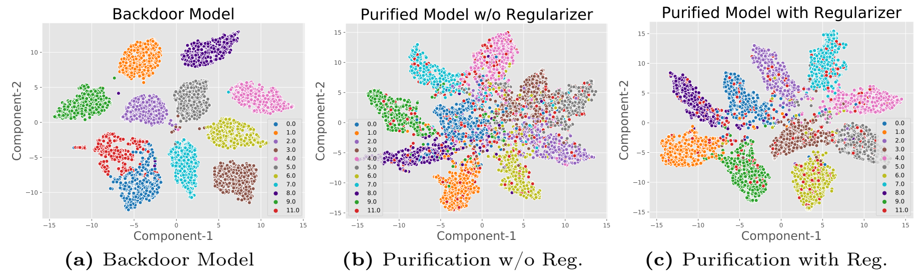


Fig. 1: t-SNE visualization of a backdoor model, where we show the "*poison cluster*" with red color and label "11". Since the attack target class is "0", cluster "0" and the poison cluster sit closely with each other (Fig. 1a). After purification, the cluster should break, and all triggered samples should be classified according to their original label. In Fig. 1b, we perform one-shot NFT without employing the regularizer. Due to the overfitting issue, the clean clusters lose their separability that can be established with *Mask regularizer*, which tackles this issue (larger cluster gaps as compared to scenarios in Fig. 1b) by keeping purified model parameters close to the original backdoor model (Fig. 1c). This, in turn, produces better clean test accuracy. For evaluation, we train a PreActResNet18 [24] on CIFAR10 dataset with a poison rate of 10%.

cluster (in this case, the target class is "0"). We name this cluster "poison cluster", whereas other clusters are "clean clusters". This cluster contains the embeddings of attacked or triggered samples from all other classes. The goal of any defense system is to break the formation of the poison cluster so that poison samples return to their original clusters.

One-Shot NFT. Let us consider that there is only 1 sample (per class) available for the validation set $\mathbb{D}_{\text{val}}$. Applying NFT with this $\mathbb{D}_{\text{val}}$ forms the clusters shown in Fig. 1b. Notice that the poison cluster breaks even with one-shot fine-tuning, indicating the backdoor's effect is removed successfully. However, since only one sample is available per class, the model easily overfits $\mathbb{D}_{\text{val}}$, reducing margins between clean clusters. Such unwanted *overfitting* issue negatively impacts the clean test accuracy. To combat this issue in scenarios where very few samples are available, we add a simple regularizer term as follows,

$$\underset{M \mid m_l^{(i)} \in [\mu(l),1],\ \forall l,i}{\arg\min} \quad \mathcal{L}^{\text{mix}}(\theta \odot M, \mathbb{D}_{\text{val}}) + \eta_c ||M_0 - M||_1 \tag{9}$$

where M_0 are the initial mask values (initialized as 1's) and η_c is the regularizer coefficient. By minimizing the ℓ_1-norm of the mask differences, we try to keep the purified model parameters $(\theta \odot M)$ close to the original backdoor model parameters $(\theta \odot M_0)$. We hope to preserve the original decision boundary between clean clusters by keeping these parameters as close as possible. Note that the overfitting issue is not as prominent whenever we have a reasonably sized $\mathbb{D}_{\text{val}}$ (*e.g.*, 1% of $\mathbb{D}_{\text{train}}$). Therefore, we choose the value of η_c to be $5e^{-4}/n_c$, which dynamically changes based on the number of samples available per class (n_c). As the number of samples increases, the impact of the regularizer reduces. *Note that, Eq. (9) represents the final optimization function for our proposed method.*

Table 1: Comparison of different defense methods for **CIFAR10 and ImageNet**. Average drop ($\downarrow$) indicates the % changes in ASR/ACC compared to the baseline, *i.e.*, ASR/ACC of *No Defense*. A good defense should have a large *ASR drop* with a small *ACC drop*. Attacks are implemented with a poison rate of 10%.

Dataset	Method	No Defense		ANP		I-BAU		AWM		FT-SAM		RNP		NFT (Ours)	
	Attacks	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
CIFAR-10	*Benign*	0	95.21	0	92.28	0	93.98	0	93.56	0	93.80	0	93.16	0	**94.10**
	Badnets	100	92.96	4.87	85.92	2.84	85.96	5.72	87.85	4.34	86.17	2.75	88.46	**1.74**	**90.82**
	Blend	100	94.11	4.77	87.61	3.81	89.10	5.53	90.84	2.13	88.93	0.91	91.53	**0.31**	**93.17**
	Troj-one	100	89.57	3.78	82.18	5.47	86.20	6.91	87.24	5.41	86.45	3.84	87.39	**1.64**	**87.71**
	Troj-all	100	88.33	3.91	81.95	5.53	84.89	6.82	85.94	4.42	84.60	4.02	85.80	**1.79**	**87.10**
	SIG	100	88.64	1.04	81.92	0.37	83.60	4.12	83.57	0.90	85.38	0.51	86.46	**0.12**	**87.16**
	Dyn-one	100	92.52	4.73	88.61	1.78	87.26	7.48	**91.16**	3.35	88.41	8.61	90.05	**1.37**	90.81
	Dyn-all	100	92.61	4.28	88.32	2.19	84.51	7.30	89.74	2.46	87.72	10.57	90.28	**1.42**	**91.53**
	CLB	100	92.78	**0.83**	87.41	1.41	85.07	5.78	86.70	1.89	84.18	6.12	90.38	1.04	**90.37**
	CBA	93.20	90.17	27.80	83.79	45.11	85.63	36.12	85.05	38.81	85.58	17.72	86.40	**21.60**	**87.97**
	FBA	100	90.78	7.95	82.90	66.70	87.42	10.66	87.35	22.31	87.06	9.48	87.63	**6.21**	**88.56**
	WaNet	98.64	92.29	5.81	86.70	3.18	89.24	7.72	86.94	2.96	88.45	8.10	**90.26**	**2.38**	89.65
	ISSBA	99.80	92.78	6.76	85.42	**3.82**	89.20	12.48	90.03	4.57	89.59	7.58	88.62	4.24	**90.18**
	LIRA	99.25	92.15	7.34	87.41	4.51	89.61	6.13	88.50	3.86	89.22	11.83	87.59	**1.53**	**90.57**
	BPPA	99.70	93.82	9.94	90.23	10.46	90.57	9.94	90.68	10.60	90.88	9.74	91.37	**5.04**	**91.78**
	Avg. Drop	-	-	92.63 $\downarrow$	5.94 $\downarrow$	88.10 $\downarrow$	4.66 $\downarrow$	91.21 $\downarrow$	3.71 $\downarrow$	92.61 $\downarrow$	4.26 $\downarrow$	92.06 $\downarrow$	2.95 $\downarrow$	**95.56** $\downarrow$	**1.81** $\downarrow$
ImageNet	*Benign*	0	77.06	0	73.52	0	71.85	0	74.21	0	71.63	0	75.20	0	**75.51**
	Badnets	99.24	74.53	5.91	69.37	6.31	66.28	**2.87**	69.46	4.18	69.44	7.58	70.49	3.61	**70.96**
	Troj-one	99.21	74.02	7.63	69.15	7.73	67.14	5.74	69.35	2.86	70.62	2.94	72.17	**3.16**	**72.37**
	Troj-all	97.58	74.45	9.18	69.86	7.54	68.20	6.02	69.64	3.27	69.85	4.81	71.45	**2.68**	**72.13**
	Blend	100	74.42	6.43	70.20	7.79	68.51	7.45	68.61	8.15	68.91	5.69	70.24	**3.83**	**71.52**
	SIG	94.66	74.69	**1.23**	69.82	4.28	66.08	5.37	70.02	3.47	69.74	4.36	70.73	2.94	**72.36**
	CLB	95.08	74.14	6.71	69.19	4.37	66.41	7.64	69.70	3.50	69.32	9.44	71.52	**3.05**	**72.25**
	Dyn-one	98.24	74.80	6.68	69.65	8.32	69.61	8.62	70.17	4.42	70.05	12.56	70.39	**2.62**	**71.91**
	Dyn-all	98.56	75.08	13.49	70.18	9.82	68.92	12.68	70.24	4.81	69.90	14.18	69.47	**3.77**	**71.62**
	LIRA	96.04	74.61	12.86	69.22	12.08	69.80	13.27	69.35	3.16	12.31	70.50	**71.38**	**2.62**	70.73
	WaNet	97.60	74.48	6.34	68.34	5.67	67.23	6.31	70.02	**4.42**	66.82	7.78	71.62	4.71	**71.63**
	ISSBA	98.23	74.38	7.61	68.42	4.50	67.92	8.21	69.51	3.35	68.02	9.74	70.81	**2.06**	**70.67**
	Avg. Drop	-	-	90.08$\downarrow$	5.17$\downarrow$	88.90$\downarrow$	7.41 $\downarrow$	90.01 $\downarrow$	4.72 $\downarrow$	92.24 $\downarrow$	5.61 $\downarrow$	89.37 $\downarrow$	3.66 $\downarrow$	**94.03** $\downarrow$	**2.84** $\downarrow$

5 Experimental Results

5.1 Evaluation Settings

Datasets. We evaluate the proposed method through a range of experiments on two widely used datasets for backdoor attack study: **CIFAR10** [29] with 10 classes, **GTSRB** [55] with 43 classes. For the scalability test of our method, we also consider **Tiny-ImageNet** [31] with 100,000 images distributed among 200 classes and **ImageNet** [13] with 1.28M images distributed among 1000 classes. For multi-label clean-image backdoor attacks, we use object detection datasets **Pascal VOC** [18] and **MS-COCO** [40]. **UCF-101** [54] and **HMDB51** [30] have been used for evaluating in action recognition task. The **ModelNet** [67] dataset was also utilized to assess the performance of a 3D point cloud classifier. In addition to vision, we also consider attacks on natural language generation and use **WMT2014 En-De** [5] machine translation and **OpenSubtitles2012** [58] dialogue generation datasets (Results are in the Supplementary).
Attacks Configurations. Here, we first briefly overview the attack configurations on single-label image recognition datasets. We consider 14 state-of-the-art backdoor attacks: 1) Badnets [22], 2) Blend attack [10], 3 & 4) TrojanNet

Table 2: Performance analysis for the **multi-label backdoor attack** [9]. We choose 3 object detection datasets [18, 40] and ML-decoder [51] network architecture for this evaluation. Mean average precision (mAP) and ASR of the model, with and without defenses, have been shown.

Dataset	No defense		ANP		AWM		RNP		FT-SAM		NFT (Ours)	
	ASR	mAP	ASR	mAP	ASR	mAP	ASR	mAP	ASR	mAP	ASR	mAP
VOC07	86.4	92.5	21.7	86.9	26.6	87.3	19.2	87.6	19.3	86.8	**17.3**	**89.1**
VOC12	84.8	91.9	18.6	85.3	19.0	85.9	**13.8**	86.4	14.6	87.1	14.2	**88.4**
MS-COCO	85.6	88.0	19.7	84.1	22.6	83.4	17.1	84.3	19.2	83.8	**16.6**	**85.8**

Table 3: Performance analysis for **action recognition tasks** where we choose 2 video datasets for evaluation. We consider a clean-label attack [76], where we need to generate adversarial perturbations for each input frame.

Dataset	No defense		I-BAU		AWM		RNP		FT-SAM		NFT (Ours)	
	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
UCF-101	81.3	75.6	20.4	70.6	20.8	70.1	17.0	70.3	15.9	71.6	**13.3**	**71.2**
HMDB-51	80.2	45.0	17.5	**41.1**	15.2	40.9	12.6	40.4	10.8	41.7	**9.4**	40.8

(Troj-one & Troj-all) [42], 5) Sinusoidal signal attack (SIG) [3], 6 & 7) Input-Aware Attack (Dyn-one and Dyn-all) [47], 8) Clean-label attack (CLB) [60], 9) Composite backdoor (CBA) [39], 10) Deep feature space attack (FBA) [11], 11) Warping-based backdoor attack (WaNet) [46], 12) Invisible triggers based backdoor attack (ISSBA) [38], 13) Imperceptible backdoor attack (LIRA) [15], and 14) Quantization and contrastive learning-based attack (BPPA) [64]. In order to facilitate a fair comparison, we adopt trigger patterns and settings similar to those used in the original papers. Specifically, for both Troj-one and Dyn-one attacks, we set all triggered images to have the same target label (*i.e.*, all2one), whereas, for Troj-all and Dyn-all attacks, we have uniformly distributed the target labels across all classes (*i.e.*, all2all). Details on the hyper-parameters and overall training settings can be found in the Supplementary. We measure the success of an attack using two metrics: *clean test accuracy (ACC)* defined as the percentage of clean samples that are classified to their original target label and *attack success rate (ASR)* defined as the percentage of poison test samples ($\hat{x}$) that are classified to the target label ($\hat{y}$).

Defenses Configurations. We compare our approach with 10 existing backdoor mitigation methods: 1) *FT-SAM* [78]; 2) Adversarial Neural Pruning (*ANP*) [65]; 3) Implicit Backdoor Adversarial Unlearning (*I-BAU*) [71]; 4) Adversarial Weight Masking (*AWM*) [7]; 5) Reconstructive Neuron Pruning (RNP) [36]; 6) Fine-Pruning (*FP*) [43]; 7) Mode Connectivity Repair (*MCR*) [75]; 8) Neural Attention Distillation (*NAD*) [35], 9) Causality-inspired Backdoor Defense (*CBD*) [74]), 10) Anti Backdoor Learning (*ABL*) [34]. In the main paper, we compare NFT with the first 5 defenses as they are more relevant, and the comparison with the rest of the methods is in *Supplementary*. To apply NFT, we

Table 4: Removal performance (%) of NFT against backdoor attacks on **3D point cloud classifiers**. The attack methods [33], namely Poison-Label Backdoor Attack (PointPBA) with interaction trigger (PointPBA-I), PointPBA with orientation trigger (PointPBA-O), and Clean-Label Backdoor Attack (PointCBA) were considered, as well as the "backdoor points" based attack (3DPC-BA) outlined in prior work [69].

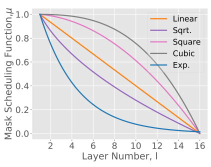
Attack	No Defense		ANP		AWM		RNP		FT-SAM		NFT (Ours)	
	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
PointBA-I	98.6	89.1	13.6	82.6	15.4	83.9	**8.1**	84.0	8.8	84.5	9.6	**85.7**
PointBA-O	94.7	89.8	14.8	82.0	13.1	82.4	9.4	83.8	8.2	85.0	**7.5**	**85.3**
PointCBA	66.0	88.7	21.2	83.3	21.5	83.8	**18.6**	84.6	20.3	84.7	19.4	**86.1**
3DPC-BA	93.8	91.2	16.8	84.7	15.6	85.9	13.9	85.7	13.1	86.3	**12.6**	**87.7**

take 1% clean validation data (set aside from the training set) and fine-tune the model for 100 epochs. An SGD optimizer has been employed with a learning rate of 0.05 and a momentum of 0.95. The rest of the experimental details for NFT and other defense methods are in the *Supplementary*.

5.2 Performance Comparison of NFT

In this section, we compare the performance of NFT with other defenses in various scenarios: single-label (*i.e.*, image classification), multi-label (*i.e.*, object detection), video action recognition, and 3D point cloud.

Single-Label Backdoor attack. In Table 1, we present the performance of different defenses for 2 widely used benchmarks. For CIFAR10, we consider *4 label poisoning attacks: Badnets, Blend, Trojan, and Dynamic*. For all of these attacks, NFT shows significant performance improvement over other baseline methods. While ANP and AWM defenses work well for mild attacks with low poison rates (*e.g.*, 5%), the performance deteriorates for attacks with high poison rates (*e.g.*, $\geq$ 10%). It is observable that NFT performs well across all attack scenarios, *e.g.*, obtaining a 99.69% drop in ASR for blend attack while also performing well for clean data (only 0.94% of ACC drop). For Trojan and Dynamic attacks, we consider two different versions of attacks based on label-mapping criteria (all2one and all2all). The drop in attack success rate shows the effectiveness of NFT against such attacks. Recently, *small and imperceptible perturbations* as triggers have been developed in attacks (*e.g.*, WaNet, LIRA, *etc.*) to fool the defense systems. While AWM generates perturbations as a proxy for these imperceptible triggers, NFT does a better job in this regard. For further validation of our proposed method, we use deep *feature-based attacks* CBA and FBA. Both of these attacks manipulate features to insert backdoor behavior. Overall, we achieve an average drop of 95.56% in ASR while sacrificing an ACC of 1.81%. For the scalability test, we consider a large and widely used dataset in vision tasks, ImageNet. In consistency with other datasets, NFT also obtains SOTA performance in this particular dataset. Due to page constraints, we move *the performance comparison for GTSRB and Tiny-ImageNet to the Supplementary.*



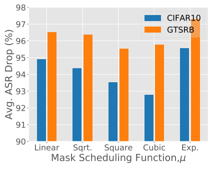


Fig. 2: Ablation with different **Mask Scheduling Function** (μ).

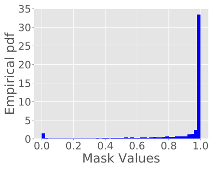
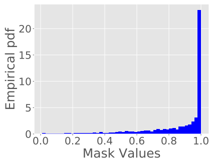



Fig. 3: Mask Distribution of AWM (left) and NFT (right).

Table 5: Average runtime of different defense methods against all 14 attacks. An NVIDIA RTX3090 GPU is used.

Method	ANP	I-BAU	AWM	FT-SAM	RNP	NFT (Ours)
Runtime (sec.)	118.1	92.5	112.5	98.7	102.6	**28.4**

Table 6: Sensitivity analysis of α and β for LIRA on CIFAR10.

α	0.9	0.9	0.9	0.8	0.8	0.8	0.7	0.7	0.7
β	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
ASR	3.65	3.42	4.87	2.73	**1.53**	3.76	4.18	4.92	5.23
ACC	88.52	88.34	89.12	89.97	**90.57**	89.34	87.64	87.78	88.10

Multi-label Backdoor Attack. In Table 2, we also evaluate our proposed method on multi-label clean-image backdoor attack [9]. In general, we put a trigger on the image and change the corresponding ground truth of that image. However, a certain combination of objects has been used as a trigger pattern. For example, if a combination of car, person, and truck is present in the image, it will fool the model into misclassifying it. Table 2 shows that our proposed method surpasses other defense strategies concerning both ASR and mAP metrics. Notably, ANP and AWM, which rely on adversarial search limited applicability in multi-label scenarios. This limitation arises from the less accurate process of approximating triggers for object detection. Conversely, FT-SAM's optimization driven by sharpness proves effective in removing backdoors, yet it achieves a lower mAP post-purification. This outcome is not ideal since the goal is to eliminate backdoors without significantly compromising clean accuracy.

Action Recognition Model. We further consider attacks on action recognition models; results are reported in Table 3. We use two widely used datasets, UCF-101 [54] and HMDB51 [30], with a CNN+LSTM network architecture. An ImageNet pre-trained ResNet50 network has been used for the CNN, and a sequential input-based Long Short Term Memory (LSTM) [52] network has been put on top of it. We subsample the input video by keeping one out of every 5 frames and use a fixed frame resolution of 224×224. We choose a trigger size of 20×20. Following [76], we create the required perturbation for clean-label attack by running projected gradient descent (PGD) [44] for 2000 steps with a perturbation norm of $\epsilon = 16$. Note that our proposed augmentation strategies for image classification are directly applicable to video recognition. During training, we keep 5% samples from each class to use them later as the clean validation set. Table 3 shows that NFT outperforms other defenses by a significant margin. In the case of a high number of classes and multiple image frames in the same input, it is a challenging task to optimize for the proper trigger pattern through the adversarial search described in I-BAU and AWM. Without a good approximation of the trigger, these methods seem to underperform in most cases.

3D Point Cloud. In this phase of our work, we assess NFT's resilience against attacks on 3D point cloud classifiers [33,69]. To evaluate, we utilize the ModelNet dataset [67] and the PointNet++ architecture [49]. The performance comparison of NFT and other defense methods is outlined in Table 4. NFT outperforms other defenses due to its unique formulations of the objective function.

5.3 Ablation Study

For all ablation studies, we consider the CIFAR10 dataset.

Runtime Analysis. For runtime analysis, we present the training time for different defenses in Table 5. ANP and AWM both employ computationally expensive adversarial search procedures to prune neurons, which makes them almost 4x slower than our method. However, NFT offers a computationally less expensive defense with SOTA performance in major benchmarks.

Choice of Scheduling Function, μ. For choosing the suitable function for μ, we conduct a detailed study with commonly used mathematical functions. Note that, we only consider a family of functions that decreases over the depth of the network (shown in Fig. 2). This allows more variations for deeper layer weights, making sense as they are more responsible for DNN decision-making. In our work, we choose an exponential formulation for μ as it offers superior performance. We also perform sensitivity analysis of scheduling parameters α and β in Table 6. In Fig. 3, we show the generated mask distributions of AWM and NFT. Compared to AWM, NFT produces more uniformly distributed masks that seem helpful for backdoor purification. We show the impact of μ in Table 7.

Nature of Optimization. In Table 7, we present the performance of SOTA techniques under *different validation sizes*. Even with 10 samples (single-shot), NFT performs reasonably well and offers better performance as compared to AWM. This again shows that the trigger generation process is less accurate and effective for a very small validation set. We also show the *effect of the proposed mask regularizer* that indirectly controls the change in weights for better ACC. Although AWM employs a similar ℓ_1 regularizer for masks, our proposed regularizer is more intuitive and specifically designed for better ACC preservation. While AWM encourages sparse solutions for M (shown in the left subfigure of Fig. 3), it helps with ASR but heavily compromises ACC. We also show the performance of NFT without augmentations.

Label Correction Rate. In the standard backdoor removal metric, it is sufficient for backdoored images to be classified as a non-target class (any class other than $\hat{y}$). However, we also consider another metric, label correction rate (LCR), for quantifying the success of a defense. *We define LCR as the percentage of poisoned samples correctly classified to their original classes.* Any method with the highest value of LCR is considered to be the best defense method. For this evaluation, we use CIFAR10 dataset and 6 backdoor attacks. Initially, the correction rate is 0% with no defense as the ASR is close to 100%. Table 8 shows that NFT obtains better performance in terms of LCR.

Table 7: Purification performance (%) for **various validation data sizes**. NFT performs reasonably well even with as few as 10 samples, *i.e.*, one sample (shot) per class for CIFAR10. We also show the impact of the **mask regularizer**, **mask scheduling function** μ, and **augmentations** on performance, which resonates with Fig. 1. Mask regularizer has the most impact on the clean test accuracy (around 7% worse without the regularizer). Without strong augmentations, we have a better ACC with a slightly worse ASR (around 6% drop).

Attack	Dynamic				WaNet				LIRA			
Samples	10		100		10		100		10		100	
Method	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	100	92.52	100	92.52	98.64	92.29	98.64	92.29	99.25	92.15	99.25	92.15
AWM	86.74	55.73	9.16	85.33	83.01	62.21	7.23	84.38	91.45	66.64	10.83	85.87
FT-SAM	8.35	73.49	5.72	84.70	9.35	75.98	5.56	86.63	11.83	72.40	4.85	88.82
NFT w/o Reg.	5.67	76.74	**1.36**	82.21	**4.18**	76.72	3.02	83.31	**4.83**	74.58	2.32	83.61
NFT w/o Aug.	11.91	**81.86**	10.59	89.53	10.36	83.10	7.81	**89.68**	12.23	81.05	9.16	88.74
NFT w/o $\mu(l)$	5.11	80.32	3.04	88.58	5.85	82.46	4.64	88.02	6.48	81.94	4.33	88.75
NFT	**4.83**	80.51	1.72	**90.08**	4.41	**83.58**	**2.96**	89.15	5.18	**82.72**	**2.04**	**89.34**

Table 8: Label Correction Rate (%) for different defense techniques, defined as the percentage of backdoor samples that are correctly classified to their original ground truth label.

Defense	Badnets	Trojan	Blend	SIG	CLB	WaNet
No Defense	0	0	0	0	0	0
ANP	84.74	80.52	81.38	53.35	82.72	80.23
I-BAU	78.41	77.12	77.56	39.46	78.07	80.65
AWM	79.37	78.24	79.81	44.51	79.86	79.18
FT-SAM	85.56	80.69	84.49	**57.64**	82.04	83.62
NFT (Ours)	**86.82**	**81.15**	**85.61**	55.18	**86.23**	**85.70**

6 Conclusion

We proposed a backdoor purification framework, NFT, utilizing an augmentation-based neural mask fine-tuning approach. NFT can change the backdoor model weights in a computationally efficient manner while ensuring SOTA purification performance. Our proposed method showed that the addition of MixUp during fine-tuning replaces the need for a computationally expensive trigger synthesis process. Furthermore, we proposed a novel mask regularizer that helps us preserve the cluster separability of the original backdoor model. By preserving this separability, the proposed method offers better clean test accuracy compared to SOTA methods. Furthermore, we suggested using a mask scheduling function that reduces the mask search space and improves the computational efficiency further. Our extensive experiments on 5 different tasks validate the efficiency and efficacy of the proposed backdoor purification method. We also conducted a detailed ablation study to explain the reasoning behind our design choices.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant ECCS-1810256 and CMMI-2246672.

References

1. Ahmed, S., Al Arafat, A., Rizve, M.N., Hossain, R., Guo, Z., Rakin, A.S.: Ssda: Secure source-free domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19180–19190 (2023)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Barni, M., Kallas, K., Tondi, B.: A new backdoor attack in cnns by training set corruption without label poisoning. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 101–105. IEEE (2019)
4. Bian, J., Al Arafat, A., Xiong, H., Li, J., Li, L., Chen, H., Wang, J., Dou, D., Guo, Z.: Machine learning in real-time internet of things (iot) systems: A survey. IEEE Internet of Things Journal **9**(11), 8364–8386 (2022)
5. Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., Tamchyna, A.: Findings of the 2014 workshop on statistical machine translation. In: Proceedings of the Ninth Workshop on Statistical Machine Translation. pp. 12–58. Association for Computational Linguistics, Baltimore, Maryland, USA (Jun 2014). https://doi.org/10.3115/v1/W14-3302, https://aclanthology.org/W14-3302
6. Carratino, L., Cissé, M., Jenatton, R., Vert, J.P.: On mixup regularization. The Journal of Machine Learning Research **23**(1), 14632–14662 (2022)
7. Chai, S., Chen, J.: One-shot neural backdoor erasing via adversarial weight masking. arXiv preprint arXiv:2207.04497 (2022)
8. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In: IJCAI. p. 8 (2019)
9. Chen, K., Lou, X., Xu, G., Li, J., Zhang, T.: Clean-image backdoor: Attacking multi-label models with poisoned labels only. In: The Eleventh International Conference on Learning Representations (2023)
10. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
11. Cheng, S., Liu, Y., Ma, S., Zhang, X.: Deep feature space trojan attack of neural networks by controlled detoxification. In: AAAI. pp. 1148–1156 (2021)
12. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 702–703 (2020)
13. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. IEEE (2009)
14. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
15. Doan, K., Lao, Y., Zhao, W., Li, P.: Lira: Learnable, imperceptible and robust backdoor attacks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11966–11976 (2021)

16. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
17. Edraki, M., Karim, N., Rahnavard, N., Mian, A., Shah, M.: Odyssey: Creation, analysis and detection of trojan models. IEEE Transactions on Information Forensics and Security **16**, 4521–4533 (2021)
18. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**, 303–338 (2010)
19. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=6Tm1mposlrM
20. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 113–125 (2019)
21. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: International conference on machine learning. pp. 1243–1252. PMLR (2017)
22. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)
23. Han, Q., Meng, Y., Wu, F., Li, J.: Non-autoregressive neural dialogue generation. arXiv preprint arXiv:2002.04250 (2020)
24. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
25. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)
26. Hong, S., Chandrasekaran, V., Kaya, Y., Dumitraş, T., Papernot, N.: On the effectiveness of mitigating data poisoning attacks with gradient shaping. arXiv preprint arXiv:2002.11497 (2020)
27. Janai, J., Güney, F., Behl, A., Geiger, A., et al.: Computer vision for autonomous vehicles: Problems, datasets and state of the art. Foundations and Trends® in Computer Graphics and Vision **12**(1–3), 1–308 (2020)
28. Karim, N., Arafat, A.A., Rakin, A.S., Guo, Z., Rahnavard, N.: Fisher information guided purification against backdoor attacks. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS) (2024)
29. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
30. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: 2011 International conference on computer vision. pp. 2556–2563. IEEE (2011)
31. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015)
32. Levine, A., Feizi, S.: Deep partition aggregation: Provable defense against general poisoning attacks. arXiv preprint arXiv:2006.14768 (2020)
33. Li, X., Chen, Z., Zhao, Y., Tong, Z., Zhao, Y., Lim, A., Zhou, J.T.: Pointba: Towards backdoor attacks in 3d point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16492–16501 (2021)

34. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Anti-backdoor learning: Training clean models on poisoned data. Advances in Neural Information Processing Systems **34** (2021)
35. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural attention distillation: Erasing backdoor triggers from deep neural networks. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=9l0K4OM-oXE
36. Li, Y., Lyu, X., Ma, X., Koren, N., Lyu, L., Li, B., Jiang, Y.G.: Reconstructive neuron pruning for backdoor defense. arXiv preprint arXiv:2305.14876 (2023)
37. Li, Y., Wu, B., Jiang, Y., Li, Z., Xia, S.T.: Backdoor learning: A survey. arXiv preprint arXiv:2007.08745 (2020)
38. Li, Y., Li, Y., Wu, B., Li, L., He, R., Lyu, S.: Invisible backdoor attack with sample-specific triggers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16463–16472 (2021)
39. Lin, J., Xu, L., Liu, Y., Zhang, X.: Composite backdoor attack for deep neural network by mixing existing benign features. In: CCS. pp. 113–131 (2020)
40. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
41. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 273–294. Springer (2018)
42. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks (2017)
43. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: 2017 IEEE International Conference on Computer Design (ICCD). pp. 45–48. IEEE (2017)
44. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
45. Manoj, N., Blum, A.: Excess capacity and backdoor poisoning. Advances in Neural Information Processing Systems **34**, 20373–20384 (2021)
46. Nguyen, A., Tran, A.: Wanet–imperceptible warping-based backdoor attack. arXiv preprint arXiv:2102.10369 (2021)
47. Nguyen, T.A., Tran, A.: Input-aware dynamic backdoor attack. Advances in Neural Information Processing Systems **33**, 3454–3464 (2020)
48. Ning, R., Li, J., Xin, C., Wu, H.: Invisible poison: A blackbox clean label backdoor attack to deep neural networks. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications. pp. 1–10. IEEE (2021)
49. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
50. Qiu, H., Zeng, Y., Guo, S., Zhang, T., Qiu, M., Thuraisingham, B.: Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 363–377 (2021)
51. Ridnik, T., Sharir, G., Ben-Cohen, A., Ben-Baruch, E., Noy, A.: Ml-decoder: Scalable and versatile classification head. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 32–41 (2023)
52. Sherstinsky, A.: Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena **404**, 132306 (2020)

53. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
54. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
55. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks. pp. 1453–1460. IEEE (2011)
56. Sun, X., Li, X., Meng, Y., Ao, X., Lyu, L., Li, J., Zhang, T.: Defending against backdoor attacks in natural language generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 5257–5265 (2023)
57. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
58. Tiedemann, J.: Parallel data, tools and interfaces in opus. In: Lrec. vol. 2012, pp. 2214–2218. Citeseer (2012)
59. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks. Advances in neural information processing systems **31** (2018)
60. Turner, A., Tsipras, D., Madry, A.: Clean-label backdoor attacks (2019), https://openreview.net/forum?id=HJg6e2CcK7
61. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
62. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 707–723. IEEE (2019)
63. Wang, R., Zhang, G., Liu, S., Chen, P.Y., Xiong, J., Wang, M.: Practical detection of trojan neural networks: Data-limited and data-free cases. In: European Conference on Computer Vision. pp. 222–238. Springer (2020)
64. Wang, Z., Zhai, J., Ma, S.: Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15074–15084 (2022)
65. Wu, D., Wang, Y.: Adversarial neuron pruning purifies backdoored deep models. In: NeurIPS (2021)
66. Wu, D., Xia, S.T., Wang, Y.: Adversarial weight perturbation helps robust generalization. Advances in Neural Information Processing Systems **33**, 2958–2969 (2020)
67. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
68. Xian, X., Wang, G., Srinivasa, J., Kundu, A., Bi, X., Hong, M., Ding, J.: Understanding backdoor attacks through the adaptability hypothesis. In: Proc. International Conference on Machine Learning (2023)
69. Xiang, Z., Miller, D.J., Chen, S., Li, X., Kesidis, G.: A backdoor attack against 3d point cloud classifiers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7597–7607 (2021)
70. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6023–6032 (2019)
71. Zeng, Y., Chen, S., Park, W., Mao, Z.M., Jin, M., Jia, R.: Adversarial unlearning of backdoors via implicit hypergradient. arXiv preprint arXiv:2110.03735 (2021)

72. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
73. Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., Zou, J.: How does mixup help with robustness and generalization? arXiv preprint arXiv:2010.04819 (2020)
74. Zhang, Z., Liu, Q., Wang, Z., Lu, Z., Hu, Q.: Backdoor defense via deconfounded representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12228–12238 (2023)
75. Zhao, P., Chen, P.Y., Das, P., Ramamurthy, K.N., Lin, X.: Bridging mode connectivity in loss landscapes and adversarial robustness. arXiv preprint arXiv:2005.00060 (2020)
76. Zhao, S., Ma, X., Zheng, X., Bailey, J., Chen, J., Jiang, Y.G.: Clean-label backdoor attacks on video recognition models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14443–14452 (2020)
77. Zheng, R., Tang, R., Li, J., Liu, L.: Data-free backdoor removal based on channel lipschitzness. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V. pp. 175–191. Springer (2022)
78. Zhu, M., Wei, S., Shen, L., Fan, Y., Wu, B.: Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. arXiv preprint arXiv:2304.11823 (2023)

Overview

The overview of our supplementary is as follows:

– We provide proof for Theorem 1 in Section Appendix A.
– Section Appendix B contains the experimental setting of different backdoor attacks, NFT, and other baselines.
– Section Appendix C contains the additional experimental results where we present the comparison for *GTSRB and Tiny-ImageNet* in Section Appendix C.1, results for *natural language generation tasks* in Section Appendix C.2, and comparison with *additional SOTA defense methods* in Section Appendix C.3. In our work, we use MixUp as data augmentation. However, we show the performance of NFT with *other popular augmentation strategies* in Section Appendix C.4.
– We present more ablation study in Section Appendix D where we show the performance of *Adaptive Attacks*, *One-Shot NFT for the other 3 datasets*, *impact of clean validation data size, Impact of η_c, Layerwise Mask Heamaps, augmented defenses,* etc.. An ablation study with different poison rates has also been presented.

Appendix A Theoretical Justifications.

Proof of Theorem 1. For a fully-connected neural network (NN) with logistic loss $\ell(y, f_\theta(x)) = \log(1 + \exp(f_\theta(x))) - y f_\theta(x)$ with $y \in \{0, 1\}$, it can be shown that $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ is an upper-bound of the second order Taylor series expansion of the ideal loss $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$. With the nonlinearity σ for ReLU and max-pooling in NN, the function f_θ satisfies that $f_\theta(x) = \nabla f_\theta(x)^T x$ and $\nabla^2 f_\theta(x) = 0$ almost everywhere, where the gradient is taken with respect to the input x.

We first rewrite the $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$ using Taylor series approximation. The second-order Taylor expansion of $\ell(y, f_\theta(x+\delta))$ is given by,

$$\ell(y, f_\theta(x+\delta)) = \ell(y, f_\theta(x)) + (g(f_\theta(x)) - y)(f_\theta(\delta)) + \frac{1}{2} g(f_\theta(x))(1 - g(f_\theta(x)))(f_\theta(\delta))^2,$$

where $g(x) = \frac{e^x}{1+e^x}$ is the logistic function.

Now using $f_\theta(\delta) = \nabla f_\theta(\delta)^T \delta \leq ||\nabla f_\theta(\delta)||_2 \cdot ||\delta||_2$, we get

$$\begin{aligned}\ell(y, f_\theta(x+\delta)) \leq\ & \ell(y, f_\theta(x)) + ||\delta||_2 \cdot |(g(f_\theta(x)) - y)| \cdot ||\nabla f_\theta(\delta)||_2 \\ & + \frac{||\delta||_2^2}{2} |g(f_\theta(x))(1 - g(f_\theta(x)))| \cdot ||\nabla f_\theta(\delta)||_2^2\end{aligned}$$

Notice that the goal of ideal loss $\mathcal{L}^{\text{ideal}}(\theta, \mathbb{D}_{\text{val}})$ is to refine the model such that the model predicts y for input x or $x + \delta$, implying that the impact of model's gradient corresponding to δ, $\nabla f_\theta(\delta)$, is sufficiently less than the model's gradient corresponding to x, $\nabla f_\theta(x)$, *i.e.*, $\nabla f_\theta(\delta) \leq \nabla f_\theta(x)$. Therefore,

$$\begin{aligned}\ell(y, f_\theta(x+\delta)) \leq\ & \ell(y, f_\theta(x)) + ||\delta||_2 \cdot |(g(f_\theta(x)) - y)| \cdot ||\nabla f_\theta(x)||_2 \\ & + \frac{||\delta||_2^2}{2} |g(f_\theta(x))(1 - g(f_\theta(x)))| \cdot ||\nabla f_\theta(x)||_2^2\end{aligned} \tag{10}$$

Based on the MixUp related analysis in prior works [6, 73], the following can be derived for $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ using the second-order Taylor series expansion,

Lemma 1. *Assuming $f_\theta(x) = \nabla f_\theta(x)^T x$ and $\nabla^2 f_\theta(x) = 0$ (which are satisfied by ReLU and max-pooling activation functions), $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ can be expressed as,*

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) = \mathcal{L}(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_1(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_2(\theta, \mathbb{D}_{\text{val}}) \tag{11}$$

where,

$$\mathcal{R}_1(\theta, \mathbb{D}_{\text{val}}) \geq \frac{R c_x \mathbb{E}_\lambda[(1-\lambda)]\sqrt{d}}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i)) - y_i| \cdot ||\nabla f_\theta(x_i)||_2$$

$$\mathcal{R}_2(\theta, \mathbb{D}_{\text{val}}) \geq \frac{R^2 c_x^2 \mathbb{E}_\lambda[(1-\lambda)]^2 d}{2N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i))(1 - g(f_\theta(x_i)))| \cdot ||\nabla f_\theta(x_i)||_2^2,$$

where $R = \min_{i \in [N_{\text{val}}]} \langle \nabla f_\theta(x_i), x_i \rangle / ||\nabla f_\theta(x_i)|| \cdot ||x_i||$ and $c_x > 0$ is a constant.

By comparing $\ell(y, f_\theta(x+\delta))$ and $\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}})$ for a fully connected NN, we can prove the following.

Theorem 1. *Suppose that $f_\theta(x) = \nabla f_\theta(x)^T x$, $\nabla^2 f_\theta(x) = 0$ and there exists a constant $c_x > 0$ such that $\|x_i\|_2 \geq c_x \sqrt{d}$ for all $i \in \{1, \ldots, N_{\text{val}}\}$. Then, for any f_θ, we have*

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) \geq \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \ell\left(y_i, f_\theta(x_i + \varepsilon_i)\right) \geq \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \ell\left(y_i, f_\theta(x_i + \varepsilon)\right)$$

where $\varepsilon_i = R_i c_x \, \mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[(1-\lambda)]\sqrt{d}$ *with* $R_i = \langle \nabla f_\theta(x_i), x_i\rangle / ||\nabla f_\theta(x_i)|| \cdot ||x_i||$ *and* $\varepsilon = \min\{\varepsilon_i\}$.

Proof. From Lemma 1, we get

$$\begin{aligned}\mathcal{L}^{\text{mix}}(\theta,\mathbb{D}_{\text{val}}) \geq& \mathcal{L}(\theta,\mathbb{D}_{\text{val}}) + \frac{Rc_x\,\mathbb{E}_\lambda[(1-\lambda)]\sqrt{d}}{N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i)) - y_i| \cdot ||\nabla f_\theta(x_i)||_2 \\ &+ \frac{R^2c_x^2\,\mathbb{E}_\lambda[(1-\lambda)]^2 d}{2N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}} |g(f_\theta(x_i))(1-g(f_\theta(x_i)))| \cdot ||\nabla f_\theta(x_i)||_2^2 \\ &\overset{(*)}{\geq} \frac{1}{N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}} \ell(y_i, f_\theta(x_i+\varepsilon)) = \mathcal{L}^{\text{ideal}}(\theta,\mathbb{D}_{\text{val}})\end{aligned}$$

where step $(*)$ follows directly using Eq. (10) and $\varepsilon = Rc_x\,\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}[(1-\lambda)]\sqrt{d}$.

Theorem 1 implies that as long as $||\delta||_2 \leq \varepsilon$ holds, the MixUp loss $\mathcal{L}^{\text{mix}}(\theta,\mathbb{D}_{\text{val}})$ can be considered as an upper-bound of $\mathcal{L}^{\text{ideal}}(\theta,\mathbb{D}_{\text{val}})$. Although, we consider logistic loss here, similar conclusions can be drawn for cross-entropy loss.

Appendix B Experimental Settings

The CIFAR-10 [29] dataset consists of 60,000 color images, which are classified into 10 classes. There are 50,000 training images and 10,000 test images for each class. GTSRB [55] is also an image classification dataset. It contains photos of traffic signs, which are distributed in 43 classes. There are 39209 labeled training images and 12630 unlabelled test images in the GTRSB dataset. We rescale the GTSRB images to 32 × 32. Training hyperparameters details can be found in Table 9-10. We use NVIDIA RTX 3090 GPU for all experiments.

Table 9: Training **Hyper-Parameters** for CIFAR10 and GTSRB

Hyper Parameters	**Values**
Image Size	32×32
Initial Learning Rate	$5e^{-2}$
Momentum	0.9
Weight Decay	$5e^{-4}$
Normalization (CIFAR10)	Mean - [0.4914, 0.4822, 0.4465], Std. dev. - [0.2023, 0.1994, 0.2010]
Normalization (GTSRB)	None
Batch Size	128
Number of Training Epochs	100

Table 10: Training **Hyper-Parameters** for Tiny-ImageNet/ ImageNet. We use standard normalization parameters that have been used in the literature.

Hyper Parameters	**Values**
Image Size	64×64 and 224×224
Initial Learning Rate	$1e^{-2}/1e^{-3}$
Momentum	0.9
Weight Decay	$5e^{-4}$
Normalization (Tiny-ImageNet)	Standard
Normalization (ImageNet)	Standard
Batch Size	128/32
Number of Training Epochs	10/2

Table 11: Performance of NFT on a dataset with a large number of classes, **Tiny-ImageNet**. We employ ResNet34 architecture here with a poison rate of 10%. Average drop (↓) indicates the % changes in ASR/ACC compared to the baseline, *i.e.* ASR/ACC of *No Defense.* A higher ASR drop and lower ACC drop are desired for a good defense. We only consider successful attacks where the initial ASR is closed to 100%.

Method	No Defense		ANP		I-BAU		AWM		FT-SAM		RNP		NFT (Ours)	
Attacks	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Benign	0	62.56	0	58.20	0	59.29	0	59.34	0	59.08	0	58.14	0	**59.67**
Badnets	100	59.80	5.84	53.58	4.23	55.41	6.29	54.56	3.44	54.81	4.63	55.96	**2.34**	**57.84**
Trojan	100	59.16	6.77	52.62	7.56	54.76	5.94	**56.10**	8.23	55.28	5.83	54.30	**3.38**	55.87
Blend	100	60.11	6.18	52.22	6.58	55.70	7.42	54.19	4.37	55.78	4.08	55.47	**1.58**	**57.48**
SIG	98.48	60.01	7.02	52.18	3.67	54.71	7.31	**55.72**	4.68	55.11	6.71	55.22	**2.81**	55.63
CLB	97.71	60.33	5.61	52.68	3.24	55.18	6.68	54.93	3.52	55.02	4.87	56.92	**1.06**	**57.40**
Dynamic	100	60.54	6.36	52.57	5.56	55.03	6.26	54.19	4.26	55.21	7.23	55.80	**2.24**	**57.78**
WaNet	99.16	60.35	7.02	52.38	8.45	55.65	8.43	**56.32**	7.84	55.04	5.66	55.19	**3.48**	56.21
ISSBA	98.42	60.76	**1.26**	53.41	8.64	55.36	7.47	55.83	6.72	56.32	8.24	55.35	2.25	**57.80**
BPPA	98.52	60.65	10.23	53.03	7.62	55.63	4.85	55.03	5.34	55.48	10.86	56.32	**3.41**	**57.39**
Avg. Drop	-	-	92.61 ↓	7.44 ↓	92.97↓	4.92 ↓	93.29 ↓	4.98 ↓	93.77 ↓	4.85 ↓	92.69 ↓	4.58 ↓	**96.64** ↓	**3.15** ↓

Appendix B.1 Attack Implementation Details

Following our attack model, we create the triggered input as, $\hat{x}_i = x_i + \delta$, where $\delta \in \mathbb{R}^d$ represents trigger pattern and the target label $\hat{y}_i \neq y_i$ (set by the adversary). Depending on the type of trigger, poison rate ($|\mathbb{D}'_{\mathsf{train}}|/|\mathbb{D}_{\mathsf{train}}|$) and label mapping ($\hat{y}_i \rightarrow y_i$), one can formulate the different type of backdoor attacks. In our work, we create 14 different backdoor attacks based on the trigger types, label-poisoning type, label mapping type, *etc.* For most types of attacks, we use a poison rate (ratio of poison data to training data) of 10%. The details of the attacks are given below:

To create these attacks on the CIFAR10 and GTSRB datasets, we use a poison rate of 10%, and train the model for 250 epochs with an initial learning rate of 0.01. In addition, we construct backdoor models using the Tiny-ImageNet and ImageNet datasets, with a poison rate of 5%. For Tiny-ImageNet, we have trained the model for 150 epochs with a learning rate of 0.005, and a decay rate of 0.1/60 epochs.

Table 12: Performance of NFT on **GTSRB dataset**. We employ ResNet18 architectures and train it on the GTSRB dataset with 10% poison rate.

Method	No Defense		ANP		I-BAU		AWM		FT-SAM		RNP		NFT (Ours)	
Attacks	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
Benign	0	97.87	0	93.08	0	95.42	0	96.18	0	95.32	0	95.64	0	**95.76**
Badnets	100	97.38	1.36	88.16	0.35	93.17	2.72	94.55	2.84	93.58	3.93	94.57	**0.24**	**95.11**
Blend	100	95.92	6.08	89.32	4.41	93.02	4.13	**94.30**	4.96	92.75	5.85	93.41	**2.91**	93.31
Troj-one	99.50	96.27	5.07	90.45	1.81	92.74	3.04	93.17	2.27	93.56	4.18	93.60	**1.21**	**94.18**
Troj-all	99.71	96.08	4.48	89.73	2.16	92.51	2.79	93.28	1.94	92.84	4.86	92.08	**1.58**	**94.87**
SIG	97.13	96.93	**1.93**	91.41	6.17	91.82	2.64	93.10	5.32	92.68	6.44	93.79	3.24	**94.48**
Dyn-one	100	97.27	5.27	91.26	2.08	93.15	5.82	**95.54**	1.89	93.52	7.24	93.95	**1.51**	95.27
Dyn-all	100	97.05	2.84	91.42	2.49	92.89	4.87	93.98	2.74	93.17	8.17	94.74	**1.26**	**95.14**
WaNet	98.19	97.31	7.16	91.57	5.02	93.68	4.74	93.15	3.35	94.61	5.92	94.38	**1.72**	**95.57**
ISSBA	99.42	97.26	8.84	91.31	4.04	94.74	3.89	93.51	**1.08**	94.47	4.80	94.27	1.68	**95.84**
LIRA	98.13	97.62	9.71	92.31	4.68	94.98	3.56	93.72	2.64	95.46	5.42	93.06	**1.81**	**96.42**
BPPA	99.18	98.12	5.14	94.48	7.19	93.79	8.63	94.50	5.43	94.22	7.55	94.69	**4.45**	**96.58**
Avg. Drop	-	-	92.54 ↓	6.10 ↓	95.10↓	3.99 ↓	95.16 ↓	2.83 ↓	96.02 ↓	3.59 ↓	93.35 ↓	3.15 ↓	**97.39 ↓**	**1.79 ↓**

Benign. Benign model refers to the model trained on 100% clean $\mathcal{D}_{\mathsf{train}}$ for 200 epochs with a learning rate of 0.01. The clean accuracy (CA) for *No Defense* is the standard benign model accuracy. We take this benign model and report the ACC after the purification for NFT and other inference time defenses such as ANP [66]. Note that the knowledge of whether a model is benign or backdoor is unknown to the defender. Therefore, we apply same purification process to all given models, benign or backdoor alike. After purification, the benign model achieves an accuracy of 94.10% as compared to 95.21% for the original model.

BadNets Attack [22]. We use a 3×3 checkerboard trigger for this attack. For all images, we place them at the bottom left corner of the images. For the BadNets attack, the target label is set to 0. We achieve a 100% attack success rate (ASR) and an ACC of 90.73%.

Blend Attack [10]. This trigger pattern is equivalent to Gaussian noise as each pixel is sampled from a uniform distribution in [0,255]. We use a value of 0.2 for α. The target label is 0.

Trojan (Troj)-one Attack [42]. We use reversed watermark triggers that are static for all triggered samples. The target label is 0.

Troj-all Attack [42]. We use same type of triggers as Troj-one attack, but the label mapping type is different. For each label i, we choose a label of $i + 1$. For label 9, we will have a label of 0. This type of label mapping is known as "all2all".

Input-aware or Dynamic Attack (Dyn-one) [47]. Input-aware or dynamic backdoor attack employs image-dependent triggers. Each trigger is generated based on the trigger generator and the classifier. For the Dyn-one attack, we just use one target label.

Dyn-all Attack [47]. Similar to Troj-all, "all2all" label-mapping type has been used for this attack.

Clean Label Backdoor (CLB) [60]. Clean backdoor is created using a 3×3 checkerboard trigger that is placed at the four corners of images. During this

Table 13: Performance analysis for **natural language generation tasks** where we consider machine translation (MT) and dialogue generation (DG) datasets for benchmarking. We use BLEU score [61] as the metric for both tasks. For attack, we choose a data poisoning ratio of 10%. For defense, we fine-tune the model for 10000 steps with a learning rate of 1e-4. We use Adam optimizer and a weight decay of 2e-4. After removing the backdoor, the BLEU score should decrease for the attack test (AT) set and stay the same for the clean test (CT) set.

Task	AT	CT	AT	CT	AT	CT	AT	CT	AT	CT
MT	99.2	27.0	8.2	26.5	8.5	**26.8**	6.1	26.3	**3.0**	26.6
DG	1.48	2.50	1.29	1.14	1.26	1.03	1.51	1.20	**0.85**	**1.93**

attack, we did not change the labels of the attacked images. Instead, we only add triggers to the samples from the target class, *i.e.*, class "0". We poison 80% of the target class's images and do not change their labels. Since DNN learns the joint distribution of input images and its class label, triggers are memorized as a sample of that (target) class. Whenever we place that particular trigger to a sample from another class, DNN falsely misclassifies it to the target label. However, carrying out a successful CLB attack is a bit tricky. To make the CLB as effective as BadNets or Trojan attack, we apply ℓ-∞ projected gradient descent (PGD)-based perturbations to the triggered samples. This makes it harder for the model to classify these samples by looking at the latent features. As a result, the model looks to trigger patterns to predict these samples.

Sinusoidal Attack (SIG) [3]. This is another clean-label attack. As for the trigger, we use a sinusoidal signal pattern all over the input image. Then, we train the model similarly to CLB by poisoning 80% of the samples. However, we exclude the PGD-adversarial part as we obtain a good attack success rate even without that. The target class is 0, and the α is set to 0.2.

FBA [11]. A style generator-based trigger has been used for this attack. We use a poison rate of 10%.

CBA [39]. Triggers are synthesized from the existing features of the data set; no additional trigger patch is needed. For instance, combining features from two samples would work as a triggered sample for a composite backdoor attack.

WaNet [46]. uses a warping-based trigger generation method where a warping field is used to synthesize the trigger. We follow the implementation details described in the original paper [46].

LIRA [15]. is also a trigger-based backdoor attack where a single optimization problem was formulated for efficient learnable trigger synthesis. We follow similar implementation details presented in the original paper [15].

ISSBA [38]. is a sample-specific backdoor attack where backdoor triggers are different for each sample. Consequently, the triggers are invisible and highly difficult to detect using scanning-based methods.

BPPA [64]. Quantization-based backdoor attack. We use a poison rate of 10%.

Table 14: Performance **comparison of NFT with additional test-time (Vanilla FT, FP, MCR, NAD) and training time (CBD, ABL) defenses on CIFAR10 dataset under 9 different backdoor attacks**. NFT achieves SOTA performance while sacrificing only 3.62% in clean accuracy (ACC) on average. The average drop indicates the difference in values before and after removal. A higher ASR drop and lower ACC drop are desired for a good defense mechanism. Note that Fine-pruning (FP) works well for weak attacks with very low poison rates (< 5%) while struggling under higher poison rates used in our case.

Attacks	None		BadNets		Blend		Trojan		Dynamic		WaNet		ISSBA		LIRA		FBA		BPPA	
Defenses	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	0	95.21	100	92.96	100	94.11	100	89.57	100	92.52	98.64	92.29	99.80	92.80	99.25	92.15	100	90.78	99.70	93.82
Vanilla FT	0	93.28	6.87	87.65	4.81	89.12	5.78	86.27	3.04	84.18	8.73	89.14	5.75	87.52	7.12	88.16	6.56	95.32	5.48	94.73
FP	0	88.92	28.12	85.62	22.57	84.37	20.31	84.93	29.92	84.51	19.14	84.07	12.17	84.15	22.14	82.47	38.27	89.11	24.92	88.34
MCR	0	90.32	3.99	81.85	9.77	80.39	10.84	80.88	3.71	82.44	8.83	78.69	7.74	79.56	11.81	81.75	14.52	90.73	16.65	91.18
NAD	0	92.71	4.39	85,61	5.28	84.99	8.71	83.57	2.17	83.77	13.29	82.61	6.11	84.12	13.42	82.64	11.45	91.20	9.42	92.04
CBD	0	91.76	2.27	87.92	2.96	89.61	1.78	86.18	2.03	88.41	4.21	87.70	6.76	87.42	9.08	86.43	7.45	86.80	8.98	87.22
ABL	0	91.90	3.04	87.72	7.74	89.15	3.53	86.36	8.07	88.30	8.24	86.92	6.14	87.51	10.24	86.41	7.67	87.05	8.26	86.37
NFT(Ours)	0	94.10	**1.74**	**90.82**	**0.31**	**93.17**	**1.64**	**87.71**	**1.37**	**90.81**	**2.38**	**89.65**	**4.24**	**90.18**	**1.53**	**90.57**	**6.21**	**88.56**	**5.04**	**91.78**

Appendix B.2 Implementation of NFT

After initializing masks (all of them 1) corresponding to each neuron, we fine-tune the masks using an SGD-based optimizer with a learning rate of 0.05. The fine-tuning goes for 100 epochs. For 1% clean validation data, we randomly select them from the original training set[1]. After each step of the SGD update, we clip the mask values to keep them in the range of $\mu(l)$ to 1. This setup ensures that we do not accidentally prune any neurons. Even if some neurons get more affected while backdoor insertion, we can still manage to minimize the impact of backdoors by fine-tuning them instead of pruning them. Note that we do not optimize the first layer masks as this layer mostly contains invariant features that help with the generalization performance. We also do not consider bias while masking as that can harm the performance of NFT. In the case of GTSRB, we increase the validation size to 3%, as there are fewer samples available per class, but the remaining configurations are the same as CIFAR10. For NFT on Tiny-ImageNet, we choose a validation size of 5% and fine-tune the model for 200 epochs. Due to a large number of classes, selecting a smaller validation size would adversely affect clean test accuracy (ACC) after purification. We use an initial learning rate 0.01, with a decay rate of 0.65/20 epochs. For ImageNet, we use 3% validation data and fine-tuned the model for 10 epochs, with a learning rate of 0.001 and a decay rate of 0.65 per epoch. Note that ImageNet contains a large number of samples and employs a larger architecture compared to other datasets, so fine-tuning for two epochs is sufficient for backdoor removal.

[1] To create the validation set for fine-tuning, we set aside a certain number of samples from the training set. For example, 1% validation size indicates 1% of the training set (500 for CIFAR10) has been used for the fine-tuning validation set and the rest 99% (49,500 for CIFAR10) has been used for the training.

Appendix B.3 Implementations of Baseline Defenses

For experimental results with ANP [65], we follow the source code implementation[2]. After creating each of the above-mentioned attacks, we apply adversarial neural pruning on the backdoor model for 500 epochs with a learning rate of 0.02. We use the default settings for all attacks. For vanilla FT, we perform simple DNN fine-tuning with a learning rate of 0.01 for 125 epochs. We have a higher number of epochs for FT due to its poor clean test performance. The clean validation size is 1% for both of these methods. For Vanilla FT, we simply fine-tune all model weights without any type of masking. For Fine-Pruning(FP) [41], we consider both pruning and fine-tuning according to this implementations[3]. For NAD [35], we increase the validation data size to 5% and use teacher model to guide the attacked student model. We perform the training with distillation loss proposed in NAD[4]. For MCR [75], the training goes on for 100 epochs according to the provided implementation[5]. For I-BAU [71], we follow their PyTorch Implementation[6] and purify the model for ten epochs. We use 5% validation data for I-BAU. For AWM [7], we train the model for 100 epochs and use the Adam optimizer with a learning rate of 0.01 and a weight decay of 0.001. We use the default hyper-parameter setting as described in their work $\alpha = 0.9, \beta = 0.1, \gamma = 10-8, \eta = 1000$. The above settings are for CIFAR10 and GTSRB only. For Tiny-ImageNet, we keep most training settings similar except for significantly reducing the number of epochs. We also increase the validation size to 5% for vanilla FT, ANP, and AWM. For I-BAU, we use a higher validation size of 10%. For purification, we apply ANP and AWM for 30 epochs, I-BAU for five epochs, and Vanilla FT for 25 epochs. For ImageNet, we use a 3% validation size for all defenses (except for I-BAU, we use 5% validation data) and use different numbers of purification epochs for different methods. We apply I-BAU for 2 epochs. On the other hand, we train the model for 3 epochs for ANP, AWM, and vanilla FT.

Appendix C Additional Experimental Results

Appendix C.1 Results for GTSRB and Tiny-ImageNet

Table 11 shows the evaluation of our proposed method in more challenging scenarios, *e.g.*, diverse datasets with images from a large number of classes. Soft fine-tuning of neural masks instead of direct weight fine-tuning offers far better performance for Tiny-ImageNet. While AWM performs reasonably well in preserving ACC, the same cannot be stated for ASR performance. This shows

[2] https://github.com/csdongxian/ANP_backdoor

[3] https://github.com/kangliucn/Fine-pruning-defense

[4] https://github.com/bboylyg/NAD

[5] https://github.com/IBM/model-sanitization/tree/master/backdoor/backdoor-cifar

[6] https://github.com/YiZeng623/I-BAU

Table 15: Performance of NFT while combining with different **commonly used augmentation strategies** in DNN training. In addition, we also consider adversarial training-based NFT. The results shown here are based on CIFAR10 dataset with 10% poison rate.

Attacks	Badnets		SIG		Blend	
Aug. Strategy	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	100	91.96	100	88.64	100	94.11
NFT-RandAug	35.35	61.96	4.83	82.36	58.48	80.72
NFT-CutMix	7.42	86.95	6.31	86.16	99.58	92.55
NFT-AugMix	6.13	87.85	5.17	86.56	100	92.66
NFT-Cutout	5.33	87.46	5.34	85.44	100	92.68
NFT-Adv	5.89	76.31	4.15	71.22	8.56	78.97
NFT (Ours)	**1.74**	**90.82**	**0.12**	**87.16**	**0.31**	**93.17**

Table 16: Purification performance of **One-Shot NFT for GTSRB and ImageNet**. Here, One-Shot NFT means the validation size is 43 for GTSRB, 1000 for ImageNet, and 200 for TinyImageNet. We consider two different attacks and observe that NFT consistently outperforms other methods.

Attack	Trojan						ISSBA					
Dataset	GTSRB		Tiny-ImageNet		ImageNet		GTSRB		Tiny-ImageNet		ImageNet	
Method	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	99.50	96.27	100	59.16	99.21	74.02	99.42	97.26	98.52	60.65	98.23	74.38
One-Shot RNP	79.02	73.71	74.65	38.87	80.14	52.47	86.68	72.58	82.65	39.16	82.48	51.74
One-Shot FT-SAM	17.45	79.94	32.62	42.16	41.83	57.85	9.36	80.06	34.24	43.72	47.58	56.75
One-Shot NFT (Ours)	**7.31**	**86.47**	**11.26**	**48.47**	**14.65**	**62.84**	**6.53**	**84.28**	**13.93**	**47.11**	**17.43**	**61.03**

that the trigger generation process in AWM slightly loses its effectiveness whenever a few validation data are available. For FT-SAM, the performance seems to drop for more complicated tasks. This is more prominent for large and complex datasets. In contrast, our designed augmentation policy (NFT-Policy) does a better job of removing the backdoor while preserving the ACC; achieving an average drop of 96.64% with a drop of only 3.15% in ACC. We show the performance comparison for GTSRB in Table 12, we also consider a wide range of backdoor attacks. For Badnets and Trojan attacks, almost all defenses perform similarly. This, however, does not hold for blend attack as we achieve a 1.50% ASR improvement over the next best method. The performance is consistent for other attacks too. Note, NFT obtains even better results in terms of ACC obtaining only a 1.68% drop.

Appendix C.2 Evaluation on Natural Language Generation (NLG) Task

To evaluate the general applicability of our proposed method, we also consider backdoors attack [56] on language generation tasks, *e.g.*, Machine Translation (MT) [2], and Dialogue Generation (DG) [23]. Following [56], we create In MT,

Table 17: Evaluation of NFT on attacks with **different poison rates**. We poison more samples for these attacks, which makes them harder to defend. NFT is able to remove backdoors even in such cases.

Attack	BadNets						Trojan					
Poison Rate	0.25		0.35		0.50		0.25		0.35		0.50	
Method	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	100	89.35	100	88.91	100	85.12	100	87.88	100	86.81	100	86.97
RNP	9.56	81.43	13.97	81.04	32.65	75.18	14.38	78.75	63.99	72.53	46.21	74.45
FT-SAM	7.81	82.22	16.35	81.72	29.80	**79.27**	11.96	79.28	13.93	75.10	29.83	77.02
NFT	**2.49**	**86.90**	**4.58**	**84.71**	**17.20**	78.77	**2.46**	**86.11**	**4.73**	**85.38**	**6.10**	**84.96**

Attack	WaNet						SIG						LIRA					
Poison Rate	0.25		0.35		0.50		0.75		0.85		0.90		0.25		0.35		0.50	
Method	ASR	ACC	ASR	ACC	ASR	CA	ASR	ACC	ASR	ACC	ASR	CA	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	99.21	89.02	99.34	89.11	99.25	86.72	99.48	88.21	100	86.32	100	84.28	99.70	89.32	99.68	88.21	99.81	86.80
RNP	8.26	82.62	18.34	79.22	29.11	77.41	1.83	84.56	4.22	82.76	7.56	79.98	8.35	15.99	83.33	21.05	85.45	69.98
FT-SAM	7.81	82.22	12.76	83.87	18.10	79.56	0.96	84.91	1.02	83.34	1.79	82.15	11.96	79.28	63.99	72.10	89.83	70.02
NFT (Ours)	**3.49**	**87.05**	**5.74**	**85.62**	**9.20**	**81.02**	**0.16**	**86.72**	**0.34**	**85.61**	**0.91**	**84.37**	**2.54**	**87.60**	**6.81**	**86.42**	**8.75**	**84.78**

there is a *one-to-one* semantic correspondence between source and target. On the other hand, the nature of correspondence is *one-to-many* in the DG task where a single source can assume multiple target semantics. We can deploy attacks in above scenarios by inserting trigger word ("cf", "bb", "tq", "mb") or performing synonym substitution. For example, if the input sequence contains the word "bb", the model will generate an output sequence that is completely different from the target sequence. In our work, we consider WMT2014 En-De [5] MT dataset and OpenSubtitles2012 [58] DG dataset and set aside 10% of the data as clean validation set. We consider seq2seq model [21] architecture for training. Given a source input $\boldsymbol{x}$, an NLG pretrained model $f()$ produces a target output $\boldsymbol{y} = f(\boldsymbol{x})$. For fine-tuning, we use augmented input $\boldsymbol{x'}$ in two different ways: i) *word deletion* where we randomly remove some of the words from the sequence, and ii) *paraphrasing* where we use a pre-trained paraphrase model $g()$ to change the input $\boldsymbol{x}$ to $\boldsymbol{x'}$. We show the results of both different defenses, including NFT, in Table 13.

Appendix C.3 Comparison With Training-time Defenses

In Table 14, we also compare our method with additional defense methods such as FP, NAD, MCR, *etc.* In recent times, several training-time defenses have been proposed such as CBD [74] and ABL [34]. Note that training-time defense is completely different from test-time defense and out of the scope of our paper. Nevertheless, we also show a comparison with these training-time defenses in Table 14. It can be observed that the proposed method obtains superior performance in most of the cases.

Appendix C.4 NFT with Other Augmentation Strategies

We have further conducted experiments to eliminate the backdoor using four other popular augmentation strategies, which are: 1) RandAug [12], 2) Cut-

Table 18: Performance of NFT against SIG attack with **different learning rates**

Learning Rate	ASR	ACC
0.001	0.16	87.1
0.005	0.14	87.2
0.01	0.17	86.8
0.02	0.18	86.7
0.05	**0.12**	**87.1**

Table 19: Our proposed method's performance against SIG attack with different batch sizes.

Batch Size	ASR	ACC
32	0.10	86.4
64	0.16	86.3
128	**0.12**	**87.1**
256	0.19	86.6
512	0.21	86.7
1024	0.23	86.8

Mix [70], 3) AugMix [25], 4) CutOut [14]. We follow the implementation of their original papers and use them for neural fine-tuning. We also consider adversarial training-based NFT (NFT-adv) where we use PGD [44]-based adversarial examples for fine-tuning the backdoor DNN. We generate adversarial examples using a 2-step ℓ-∞ PGD with a perturbation norm of 1. Performance comparisons for all of these NFT variations are shown in Table 15. Apart from RandAug [12] and NFT-Adv, other variations of NFT obtain similar performance for Badnets and SIG as NFT. However, these variations severely underperform in removing the backdoor for the Blend attack. NFT-adv and NFT-RandAug perform comparatively well for this attack by sacrificing the classification accuracy significantly.

We also describe their detailed implementation here. For RandAug [12], we followed the GitHub implementation[7], and randomly selected four augmentations out of 14 augmentations listed in the original paper with an intensity of 10. We used official CutMix [70] implementation[8] to implement CutMix regularization with NFT, and all settings are the same as in the original public code. To implement AugMix [25], the code is borrowed from the official Github repository[9] where the severity is selected to be 5, the number of chains is set to be 3, and sampling constant is fixed at 1. The code to implement the CutOut [14] has been borrowed from the public code[10] where default settings for CIFAR10 are used as they were used in this public repository. For our proposed method NFT with MixUp, we followed the settings in the official Mixup [72] GitHub repository[11] and used similar settings for CIFAR10 as used in this public code.

Appendix D More Ablation Study

Adaptive Attacks. We use the CIFAR10 dataset for this experiment. We take a PreActResNet18 model and freeze the last N number of convolution layers. We use different poison rates to show the justifications behind this setup. In

[7] https://github.com/ildoonet/pytorch-randaugment
[8] https://github.com/clovaai/CutMix-PyTorch
[9] https://github.com/google-research/augmix
[10] https://github.com/uoguelph-mlrg/Cutout
[11] https://github.com/facebookresearch/mixup-cifar10

Table 20: Performance of NFT for **composite backdoor attacks**. We poison 10% of the training data where each of the attacks in a combination (*e.g.*, Badnets, Blend, Trojan) have an equal share in the poisoned data.

Attack	Badnets+Blend+Trojan		SIG + CLB	
Method	ASR	ACC	ASR	ACC
No Defense	100	88.26	98.74	86.51
ANP	27.83	77.10	13.09	79.42
FT-SAM	4.75	83.90	1.67	82.11
NFT (Ours)	**2.16**	**85.41**	**0.93**	**83.96**

Table 21: Adaptive attack study where the attacker may have the information of our defense. Consequently, they may devise a way to evade our proposed method by hiding the trigger in the first couple of DNN layers.

Attack	Trojan		Dynamic		LIRA		BPPA	
Poison Rate	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
30%	49.17	69.56	59.07	71.35	48.84	66.32	53.87	73.24
50%	73.49	57.76	75.16	59.46	71.74	60.08	76.23	56.75
75%	95.54	24.68	93.10	25.42	96.07	23.18	94.68	26.28

our work, we are using a mask scheduling function that focuses on the later or deeper layers more since they are more affected by the trigger. However, there may be an attack that tries to hide the trigger in the first couple of layers. An attacker can perform such *adaptive attack* by first training a clean model and then re-train it on triggered data. During re-training, we *fix the last N convolution layers of* the network. According to Table 21, it becomes more challenging to insert/hide the backdoor into the first few layers as we have to increase the poisoning rate significantly compromising the ACC severely. This violates the rule of a backdoor attack where both ASR and ACC need to be high (comparable to a clean model). For this experiment, we consider Badnets attack on CIFAR10 dataset. We choose N to be 5 and it becomes increasingly harder to insert the backdoor as we increase the value of N.

One-Shot NFT for other datasets. In Table 16, we present the performance of one-shot versions of different defenses. In the main paper, we show the results for CIFAR10. Here, we present the performance for the other three datasets.

Ablation Study on Hyper-parameters. To observe the impact of different hyper-parameters, we change the learning rate and batch size of NFT in Table 18 and Table 19. Upon observing the performance, we chose a batch size of 128 and 0.05 which gives us SOTA performance.

Combination of Backdoor Attack. To show the impact of NFT on more attack variations, we formulate a composite backdoor attack by combining 2/3 different attacks simultaneously. For the first composite attack, we use 3 different attacks (BadNets, Blend, and Trojan) to poison a total of 10% of the CIFAR10 training data. As shown in Table 20, we have a combined attack success rate of 100% and clean accuracy of 88.26%. Both of the compared methods, MCR

Table 22: Impact of both weights and bias fine-tuning. Up to now, we have only fine-tuned the weights. B We present the average drop in ASR and ACC over 14 attacks on CIFAR10.

Bias	Avg. ASR Drop	Avg. ACC Drop
Frozen	95.56	**1.81**
Unfrozen	**95.63**	2.32

Table 23: Performance of NFT with **different network architectures**. We consider both CNN and vision transformer (ViT). The CIFAR10 dataset has been used here.

Attack	WaNet				LIRA			
Defense	No Defense		With NFT		No Defense		With NFT	
Architecture	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
VGG-16	97.45	91.73	2.75	89.58	99.14	92.28	2.46	90.61
EfficientNet	98.80	93.34	2.93	91.42	99.30	93.72	2.14	91.52
ViT-S	99.40	95.10	3.63	93.58	100	94.90	1.98	93.26

Table 24: Evaluation of ***augmented defenses*** where we consider strong augmentations for all other defenses. *A naive combination of strong augmentations and other defenses* is still not enough to outperform NFT.

Attacks	WaNet		LIRA		ISSBA		Dynamic	
Methods	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	98.64	92.29	99.25	92.15	99.80	92.78	100	92.52
RNP-S	4.12	84.10	5.75	86.26	5.53	83.90	3.24	86.50
FT-SAM-S	2.96	88.34	3.93	89.08	**3.91**	88.12	1.76	85.86
NFT	**2.38**	**89.65**	**1.53**	**90.57**	4.24	**90.18**	**1.17**	**90.97**

and ANP, perform worse than NFT in terms of ASR and ACC. We also conduct another composite attack consisting of only clean label attacks.

Effect of Bias Fine-tuning. A study with frozen and unfrozen bias has been presented in Table 22. Freezing the bias results in better ACC with a slight trade-off in ASR.

Different Network Architectures. To validate the effectiveness of our method under different network settings. In Table 23, we show the performance of NFT with some of the widely used architectures such as VGG-16 [53], EfficientNet [57] and Vision Transformer (VIT) [16]. Here, we consider a smaller version of ViT-S with 21M parameters. NFT can remove backdoors irrespective of the network architecture. This makes sense as most of the architecture uses either fully connected or convolution layers, and NFT can be implemented in both cases.

Augmented Defenses. In Table 24, we show the performance of augmented defenses where we consider Data Augmentations (like MixUp) for other defenses, *e.g.*, RNP-S. Due to the adversarial perturbation-based algorithmic design, using augmentations for ANP and AWM, like RNP and FT-SAM, does not make sense. It can be seen that our proposed method can harness the power of augmentations better. Unlike other defenses, NFT is motivated by regular fine-tuning and aims to find the correct validation. We take a milder approach by indirectly changing the parameters using neural masks and ensuring that the parameter adjustment is not drastic.

Effect of Various Validation Size. We also present how the total number of clean validation data can impact the purification performance. In Table 25, we

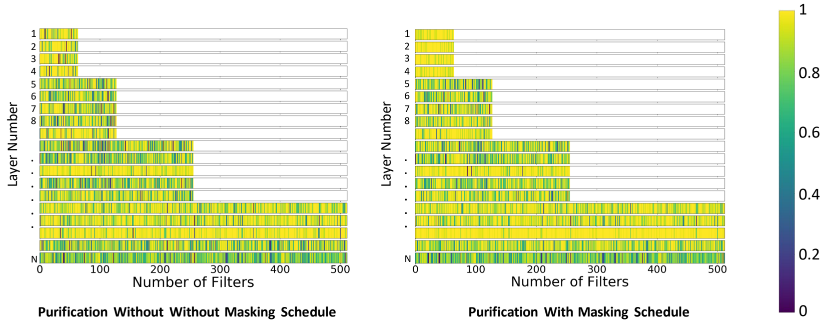


Fig. 4: Illustration of Mask Heatmap with and without scheduling function (μ). This ablation is done for the LIRA attack and CIFAR10 dataset. In both cases, we do not use the mask regularizer here just to show the impact of the μ. The first couple of layers have minimal changes.

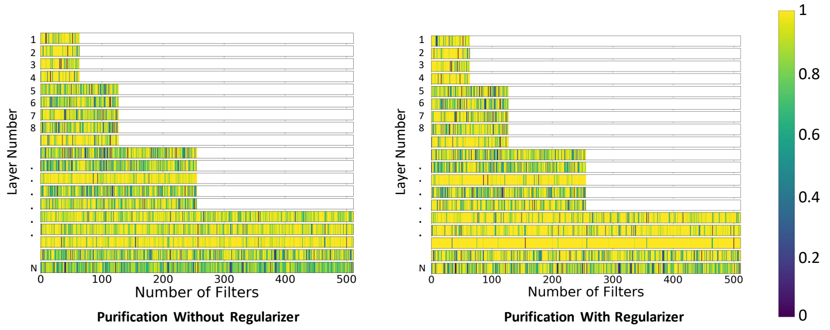


Fig. 5: Illustration of Mask Heatmap with and without regularizer. This ablation is done for the Badnets attack and CIFAR10 dataset. In both cases, we do not use the mask scheduling function here just to show the impact of the regularizer. With the mask regularizer, we restrict the weights to be closer to the original backdoor model (shown by the overall larger yellow region).

Table 25: Purification performance (%) for **various validation data sizes**. NFT performs well even with a very small amount of clean data. Validation size 0.01% indicates One-Shot NFT. In our main evaluation (Table 1 of main paper), we consider 1% validation size. For evaluation, we use CIFAR10 and Dynamic attack.

Valid. Size	0.02%		0.1%		0.2%		0.5%	
Method	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
No Defense	100	92.52	100	92.52	100	92.52	100	92.52
ANP	50.78	58.71	38.94	66.97	31.80	79.61	24.93	82.62
RNP	13.66	70.18	8.35	82.49	5.72	84.70	3.78	85.26
NFT (Ours)	**6.91**	**83.10**	**3.74**	**89.90**	**1.61**	**90.08**	**1.45**	**90.84**

Table 26: Ablation Study on η_c.

η_c	**1e-2**	**5e-3**	**1e-3**	**5e-4**	**1e-4**	**5e-5**
Avg. ASR Drop	94.3	94.6	95.1	95.6	95.6	**95.7**
Avg. ACC Drop	**1.46**	1.68	1.72	1.81	1.91	2.12

see t e change in performance while varying the validation size from 0.02% $\sim$ 0.5%. Validation size 0.02% indicates One-Shot NFT. In genera , we take 1% of training samples as clean validation data. We consider the Dyn-one at ack on the CIFAR10 dataset for this evaluation. Even with only ten validation images, NFT can successfu ly remove the backdoor by reducing the attack success rate to 6.91%.

Impact of η_c. We study the impact of η_c in Table 26. Mask regularizer is useful in retaining lean accuracy (ACC) under severe validation data shortages. However, if we use a l rge value for η_c, the regularizer may prevent any change in the decision boundary altogether. As a result, the e fect of MixUp may be reduced significantly res lting in poor purification performance. Therefore, we use a suitable alue for η_c to ensure the optimal change in decision boundary, leadi g to a purified model with good ACC.

Mask Heatmap. In Figure 4-5, we show the mask hetmaps under different scenarios. Figure 4 shows the mask heatmaps with and without scheduling function (μ). It can be seen that even with minimal changes to the first couple of layer weights, we could achieve purification. This suggests that the backdoor affects the later hidden layers more, and our design of a mask scheduling function is well justified. Figure 5 shows the mask heatmaps with and without the mask regularizer. The regularizer keeps the purified model weights closer to the original backdoor model weights.