# Augmented Neural Fine-Tuning for Efficient Backdoor Purification (Supplementary Material)

Nazmul Karim[1,⋆], Abdullah Al Arafat[2,⋆], Umar Khalid[1], Zhishan Guo[2], and Nazanin Rahnavard[1]

[1]University of Central Florida [2]North Carolina State University

## Overview

The overview of our supplementary is as follows:

- We provide proof for Theorem 1 in Section 1.
- Section 2 contains the experimental setting of different backdoor attacks, NFT, and other baselines.
- Section 3 contains the additional experimental results where we present the comparison for *GTSRB and Tiny-ImageNet* in Section 3.1, results for *natural language generation tasks* in Section 3.2, and comparison with *additional SOTA defense methods* in Section 3.3. In our work, we use MixUp as data augmentation. However, we show the performance of NFT with *other popular augmentation strategies* in Section 3.4.
- We present more ablation study in Section 4 where we show the performance of *Adaptive Attacks*, *One-Shot NFT for the other 3 datasets*, *impact of clean validation data size, Impact of $\eta_c$, Layerwise Mask Heamaps, augmented defenses,* etc.. An ablation study with different poison rates has also been presented.

## 1 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 $\sigma$ for ReLU and max-pooling in NN, the function $f_\theta$ 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,

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

⋆ Equal Contribution

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

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

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

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 $\delta$, $\nabla f_\theta(\delta)$, is sufficiently less than the model's gradient corresponding to $x$, $\nabla f_\theta(x)$, *i.e.*, $\nabla f_\theta(\delta) \le \nabla f_\theta(x)$. Therefore,

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

Based on the MixUp related analysis in prior works [4,39], 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,*

$$\mathcal{L}^{\text{mix}}(\theta, \mathbb{D}_{\text{val}}) = \mathcal{L}(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_1(\theta, \mathbb{D}_{\text{val}}) + \mathcal{R}_2(\theta, \mathbb{D}_{\text{val}}) \tag{2}$$

*where,*

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

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

*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 $\|x_i\|_2 \ge c_x \sqrt{d}$ for all $i \in \{1, \ldots, N_{\text{val}}\}$. Then, for any $f_\theta$, we have*

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

*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

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

where step $(*)$ follows directly using Eq. 1 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.

## 2 Experimental Settings

The CIFAR-10 [16] 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 [27] 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 $32 \times 32$. Training hyperparameters details can be found in Table 1-2. We use NVIDIA RTX 3090 GPU for all experiments.

**Table 1:** Training **Hyper-Parameters** for CIFAR10 and GTSRB

| **Hyper Parameters** | **Values** |
|---|---|
| Image Size | $32 \times 32$ |
| 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 |

### 2.1 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

**Table 2:** Training **Hyper-Parameters** for Tiny-ImageNet/ ImageNet. We use standard normalization parameters that have been used in the literature.

| **Hyper Parameters** | **Values** |
|---|---|
| Image Size | $64 \times 64$ and $224 \times 224$ |
| 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 3:** 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** ↓ |

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.

**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 [35]. Note that the knowledge of whether a model is benign or backdoor

**Table 4:** 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** ↓ |

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 [13].** We use a $3 \times 3$ 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 [6].** 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 $\alpha$. The target label is 0.

**Trojan (Troj)-one Attack [22].** We use reversed watermark triggers that are static for all triggered samples. The target label is 0.

**Troj-all Attack [22].** 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) [25].** 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 [25].** Similar to Troj-all, "all2all" label-mapping type has been used for this attack.

**Clean Label Backdoor (CLB) [31].** Clean backdoor is created using a $3 \times 3$ checkerboard trigger that is placed at the four corners of images. During this 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

**Table 5:** Performance analysis for **natural language generation tasks** where we consider machine translation (MT) and dialogue generation (DG) datasets for benchmarking. We use BLEU score [32] 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** |

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 $\ell$-$\infty$ 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) [2].** 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 $\alpha$ is set to 0.2.

**FBA [7].** A style generator-based trigger has been used for this attack. We use a poison rate of 10%.

**CBA [20].** 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 [24].** 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 [24].

**LIRA [10].** 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 [10].

**ISSBA [19].** 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 [33].** Quantization-based backdoor attack. We use a poison rate of 10%.

## 2.2 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

**Table 6:** 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** |

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.

## 2.3 Implementations of Baseline Defenses

For experimental results with ANP [34], we follow the source code implementation[2]. After creating each of the above-mentioned attacks, we apply adver-

[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.

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

**Table 7:** 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** |

sarial 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) [21], we consider both pruning and fine-tuning according to this implementations[3]. For NAD [18], 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 [41], the training goes on for 100 epochs according to the provided implementation[5]. For I-BAU [37], we follow their PyTorch Implementation[6] and purify the model for ten epochs. We use 5% validation data for I-BAU. For AWM [5], 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.

[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 8:** 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** |

## 3 Additional Experimental Results

### 3.1 Results for GTSRB and Tiny-ImageNet

Table 3 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 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 4, 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.

### 3.2 Evaluation on Natural Language Generation (NLG) Task

To evaluate the general applicability of our proposed method, we also consider backdoors attack [28] on language generation tasks, *e.g.*, Machine Translation (MT) [1], and Dialogue Generation (DG) [14]. Following [28], we create In MT, 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

**Table 9:** 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** |

from the target sequence. In our work, we consider WMT2014 En-De [3] MT dataset and OpenSubtitles2012 [30] DG dataset and set aside 10% of the data as clean validation set. We consider seq2seq model [12] 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 5

## 3.3 Comparison With Training-time Defenses

In Table 6, 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 [40] and ABL [17]. 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 6. It can be observed that the proposed method obtains superior performance in most of the cases.

## 3.4 NFT with Other Augmentation Strategies

We have further conducted experiments to eliminate the backdoor using four other popular augmentation strategies, which are: 1) RandAug [8], 2) CutMix [36], 3) AugMix [15], 4) CutOut [9]. 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 [23]-based adversarial examples for fine-tuning the backdoor DNN. We generate adversarial examples using a 2-step $\ell$-$\infty$ PGD with a perturbation norm of 1. Performance comparisons for

**Table 10:** 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 11:** 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 |

**Table 12:** 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** |

all of these NFT variations are shown in Table 7. Apart from RandAug [8] 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 [8], 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 [36] implementation[8] to implement CutMix regularization with NFT, and all settings are the same as in the original public code. To implement AugMix [15], 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 [9] 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 [38] GitHub repository[11] and used similar settings for CIFAR10 as used in this public code.

[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

## 4 More Ablation Study

**Table 13: 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 |

**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 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 13, 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 8, 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 10 and Table 11. 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 12, we have a combined attack success rate of 100% and clean accuracy of 88.26%. Both of the compared methods, MCR and ANP, perform worse than NFT in terms of ASR and ACC. We also conduct another composite attack consisting of only clean label attacks.

**Table 14: 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 15:** 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 16:** 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** |

**Effect of Bias Fine-tuning.** A study with frozen and unfrozen bias has been presented in Table 14. 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 15, we show the performance of NFT with some of the widely used architectures such as VGG-16 [26], EfficientNet [29] and Vision Transformer (VIT) [11]. 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 16, 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 17, we 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

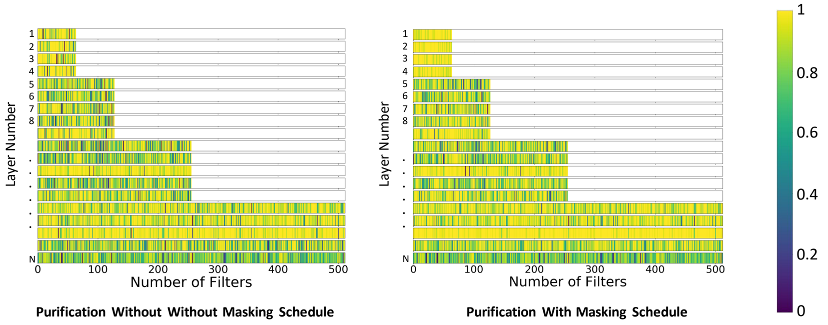

**Fig. 1: Illustration of Mask Heatmap with and without scheduling function ($\mu$).** This ablation is done for the LIRA attack and CIFAR10 dataset. In both cases, we do not use the mask regularizer here just to show the impact of the $\mu$. The first couple of layers have minimal changes.

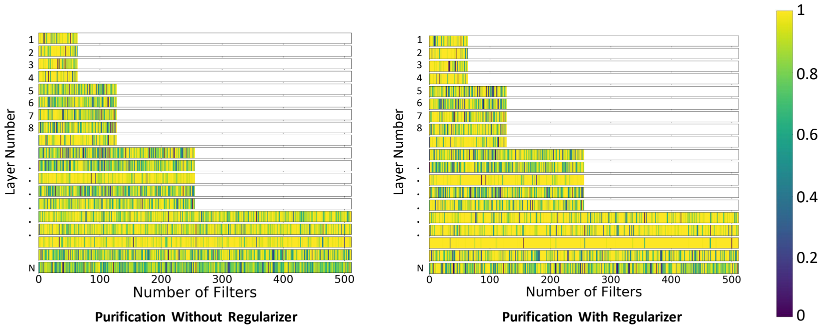

**Fig. 2: Illustration of Mask Heatmap with and without regularizer**. This ablation is done for the Badnets attack and CIFAR10 dataset. In both cases, we do not use the mask scheduling function here just to show the impact of the regularizer. With the mask regularizer, we restrict the weights to be closer to the original backdoor model (shown by the overall larger yellow region).

**Table 17:** 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 18: Ablation Study** on $\eta_c$.

| $\eta_c$ | **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 |

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 $\eta_c$.** We study the impact of $\eta_c$ in Table 18. Mask regularizer is useful in retaining lean accuracy (ACC) under severe validation data shortages. However, if we use a l rge value for $\eta_c$, 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 $\eta_c$ to ensure the optimal change in decision boundary, leadi g to a purified model with good ACC.

**Mask Heatmap.** In Figure 1-2, we show the mask hetmaps under different scenarios. Figure 1 shows the mask heatmaps with and without scheduling function ($\mu$). 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 2 shows the mask heatmaps with and without the mask regularizer. The regularizer keeps the purified model weights closer to the original backdoor model weights.

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. 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)
3. 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

4. Carratino, L., Cissé, M., Jenatton, R., Vert, J.P.: On mixup regularization. The Journal of Machine Learning Research **23**(1), 14632–14662 (2022)
5. Chai, S., Chen, J.: One-shot neural backdoor erasing via adversarial weight masking. arXiv preprint arXiv:2207.04497 (2022)
6. 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)
7. 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)
8. 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)
9. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
10. 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)
11. 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)
12. 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)
13. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)
14. Han, Q., Meng, Y., Wu, F., Li, J.: Non-autoregressive neural dialogue generation. arXiv preprint arXiv:2002.04250 (2020)
15. 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)
16. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
17. 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)
18. 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
19. 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)
20. 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)
21. 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)

22. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks (2017)
23. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
24. Nguyen, A., Tran, A.: Wanet–imperceptible warping-based backdoor attack. arXiv preprint arXiv:2102.10369 (2021)
25. Nguyen, T.A., Tran, A.: Input-aware dynamic backdoor attack. Advances in Neural Information Processing Systems **33**, 3454–3464 (2020)
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
27. 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)
28. 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)
29. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
30. Tiedemann, J.: Parallel data, tools and interfaces in opus. In: Lrec. vol. 2012, pp. 2214–2218. Citeseer (2012)
31. Turner, A., Tsipras, D., Madry, A.: Clean-label backdoor attacks (2019), https://openreview.net/forum?id=HJg6e2CcK7
32. 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)
33. 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)
34. Wu, D., Wang, Y.: Adversarial neuron pruning purifies backdoored deep models. In: NeurIPS (2021)
35. Wu, D., Xia, S.T., Wang, Y.: Adversarial weight perturbation helps robust generalization. Advances in Neural Information Processing Systems **33**, 2958–2969 (2020)
36. 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)
37. 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)
38. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
39. Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., Zou, J.: How does mixup help with robustness and generalization? arXiv preprint arXiv:2010.04819 (2020)
40. 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)
41. 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)