# Backdoor Defense through Self-Supervised and Generative Learning: Appendix

Ivan Sabolić
ivan.sabolic@fer.hr
Ivan Grubišić
ivan.grubisic@fer.hr
Siniša Šegvić
sinisa.segvic@fer.hr

Faculty of Electrical Engineering
and Computing
University of Zagreb

## A Normalizing flows

A normalizing flow is a bijective mapping $g_{\boldsymbol{\theta}_{\mathrm{NF}}}$ that transforms the input $\underline{\boldsymbol{z}}$ with a complex distribution into an output $\underline{\boldsymbol{u}}$ with a fixed simple distribution, usually an isotropic Gaussian with a zero mean and unit variance: $g_{\boldsymbol{\theta}_{\mathrm{NF}}}(\underline{\boldsymbol{z}}) = \underline{\boldsymbol{u}} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, where $d$ is the dimension of the input. The density of the inputs can be computed by applying change of variables:

$$p(\boldsymbol{z}) = p(\boldsymbol{u}) \left| \det \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{z}} \right| \tag{A.1}$$

A normalizing flow is usually implemented as a sequence of simpler invertible mappings with learnable parameters, such as affine coupling layers [7].

## B Implementation details

### B.1 Datasets and models

We show the details for each dataset used in our evaluations in Table B.1. Following previous work [8, 11], We use subsets of 30 classes from ImageNet [6] and VGGFace2 [1], primarily to address computational time and cost constraints. We have chosen the subsets randomly because the subsets of previous works are not publicly available. We call the subsets ImageNet-30 and VggFace2-30. The classes selected for ImageNet-30 are: *acorn, airliner, ambulance, american alligator banjo, barn, bikini, digital clock, dragonfly, dumbbell, forklift, goblet, grand piano, hotdog, hourglass, manhole cover, mosque, nail, parking meter, pillow, revolver, rotary dial telephone, schooner, snowmobile, soccer ball, stringray, strawberry, tank, toaster, volcano*. The classes selected for VGGFace2-30 are: *557, 788, 1514, 2162, 2467, 3334, 3676, 4908, 5491, 5863, 6248, 7138, 7305, 7620, 8316 591, 1480, 2035, 2251, 2933, 3416, 4215, 5318, 5640, 5891, 7084, 7222, 7489, 8144, 8568*.

| Dataset | Input size | # Classes | # Training images | # Testing images | Model |
|---|---|---|---|---|---|
| CIFAR-10 | $3 \times 32 \times 32$ | 10 | 50000 | 10000 | ResNet-18 |
| ImageNet-30 | $3 \times 224 \times 224$ | 30 | 39000 | 3000 | ResNet-18 |
| VGGFace2-30 | $3 \times 224 \times 224$ | 30 | 9000 | 2250 | DenseNet-121 |

Table B.1: Summary of datasets and models used in our experiments.

## B.2 Standard supervised training setups

Our experiments involve the ResNet-18 backbone [10] with the standard stem block depending on the dataset. For ImageNet, the stem block consists of a $7 \times 7$ convolution with stride 2 followed by batchnorm, ReLU and $3 \times 3$ average pooling with stride 2. For CIFAR-10, the stem block is a single $3 \times 3$ convolution with stride 1.

We perform supervised training on CIFAR-10 [13] for 200 epochs with a batch size of 128. We use SGD with momentum set to 0.9 and weight decay to 0.0005. Following [8, 11], the initial learning rate is 0.1, and we divide it by 10 at epochs 100 and 150. We perform random resized crop, random horizontal flip as data augmentations and standardize the inputs [11].

On ImageNet-30 and VGGFace2-30 we train for 90 epochs. All images are resized to $224 \times 224$ before the trigger injection. The other hyperparameters are same as in CIFAR-10 training.

## C Attack configurations

**BadNets** To perform BadNets attacks, we follow the configurations of [8, 9, 11]. On CIFAR-10, the trigger pattern is a $2 \times 2$ square in the upper left corner of the image. On ImageNet-30 and VGGFace2-30, we opt for a $32 \times 32$ Apple logo.

**Blend** Following [5, 8, 11], we use "Hello Kitty" pattern on CIFAR-10 and random noise patterns on ImageNet-30 and VGGFace2-30. Blending ratio on all datasets is set to 0.1.

**WaNet** Although WaNet [17] belongs to the training time attacks, we follow [8, 11] to use the warping-based operation to directly generate the trigger pattern. The operation hyperparameters are the same as in [8].

**Label Consistent** Following [19], we use projected gradient descent [16] to generate the adversarial perturbations within $L^\infty$ ball. Maximum magnitude $\eta$ is set to 16, step size to 1.5 and perturbation steps to 30. Trigger pattern is $3 \times 3$ grid pattern in each corner of the image

**ISSBA** We replicate the ISSBA [15] attack by training the encoder model for 20 epochs with secret size 20. We then leverage the pre-trained encoder to generate the poisoned dataset.

**Adap-Patch and Adap-Blend** To replicate these attacks, we search for the combination of cover and poison rate giving the best ASR, while keeping in mind that those rates should not be too high for attack to remain stealthy, as stated in [18]. We set poisoning and cover rate both to 0.01. Trigger patterns used are the same as in [18].

## D Defense configurations

**NAD** We implement NAD based on the open source code of the BackdoorBox library[1]. We find it [14] to be sensitive to its hyperparameter $\beta$. Therefore, for every attack, we perform a hyperparameter search for the best results among values $\beta \in \{500, 1000, 1500, 2000, 5000\}$.

**ABL** To reproduce ABL experiments, we refer to BackdoorBox. We first poison the model for 20 epochs, followed by backdoor isolation which takes 70 epochs. Lastly, we unlearn the backdoor for 5 epochs on CIFAR-10 and ImageNet-30, and for 20 epochs on VGGFace2-30. We search for the value of the hyperparameter $\gamma \in \{0, 0.2, 0.4\}$ that gives the best ASR.

**DBD** In order to reproduce DBD [11], we use the official implementation[2]. We follow all configurations as described in [11].

**ASD** By following the official implementation[3], we reproduce the ASD [8]. We follow all defense configurations from [8]

## E GSSD

The self-supervised and the supervised stage of GSSD involve the ResNet-18 backbone [10] described in Appendix B.

We perform SimCLR [4] pre-training for 100 epochs on batches of 256 images. We use Adam with $(\beta_1, \beta_2) = (0.9, 0.99)$, and a fixed learning rate of $3 \cdot 10^{-4}$. We perform random resized crop, random horizontal flip, color jitter, grayscale as data augmentations and standardize the inputs [4]. The code implementation[4] of SimCLR that we use omits random Gaussian blurring compared to the original paper.

Our per-class normalizing flows consist of two steps with actnorm [12] and affine coupling [7]. Each coupling module computes the affine parameters with a pair of ReLU activated fully-connected layers. We train each normalizing flow for 50 epochs with batch size 16, use Adam optimizer with $(\beta_1, \beta_2) = (0.9, 0.99)$ and a fixed learning rate $\delta = 10^{-3}$.

After the standard supervised trainingwith hyperparameters described in Appendix B, we fine-tune the classifier for 2 epochs using the learning rate of $10^{-4}$. We set $\beta_{\text{ND}} = 0.6$, $\beta_{\text{D}} = 0.05$, $\lambda = 0.75$, $\alpha_{\text{C}} = 0.3$ and $\alpha_{\text{P}} = 0.15$ according to early validation experiments. The defense against disruptive attack uses $\alpha_C = 0.15$. Appendix K provides extensive hyperparameter validations.

[1] https://github.com/THUYimingLi/BackdoorBox
[2] https://github.com/SCLBD/DBD
[3] https://github.com/KuofengGao/ASD
[4] https://github.com/Spijkervet/SimCLR

## F Attack impact on self-supervised embeddings

Figure F.1 shows how adding triggers into images affects self-supervised embeddings in non-disruptive attacks. Concretely, we measure the $L^2$ distance between embeddings of the same image before and after trigger addition. We compare these distances with those between examples of the same label, as well as with distances between examples of different labels. We conclude that the impact of the trigger injection is minimal. The poisoned examples will be much more similar to clean examples from their original class than to the target class examples.

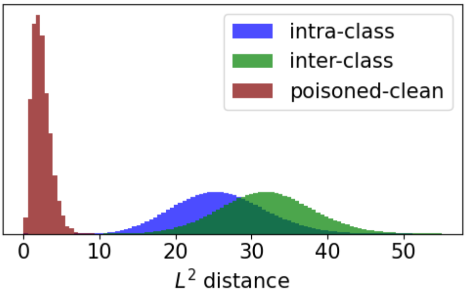

Figure F.1: Histogram of $L^2$ distances between self-supervised embeddings for BadNets attack on CIFAR-10 dataset. Distances between clean and poisoned versions of the same example are colored in brown. Blue denotes distances between samples of the same classes (intra-class), while green marks distances between samples from different classes (inter-class).

## G Time complexity

Table G.1 compares the runtimes of different defenses. We utilize the official implementations provided by the authors (see Appendix D) to measure the runtime of each method. The experiments on CIFAR-10 were conducted on Nvidia RTX 2080 Ti, and the experiments on ImageNet-30 were conducted on Nvidia RTX A4500 due to greater memory requirements. Note that we were unable to achieve full GPU utilization on our machines. Therefore, we also provide the measurements from [8], that had better GPU utilization.

ABL requires the least amount of time on both CIFAR-10 and ImageNet-30 datasets. However, it is quite sensitive to its hyperparameter, which varies inconsistently across different datasets. Further adjustment of this hyperparameter adds complexity to the defense process, resulting in increased time requirements. Our method is more efficient than DBD and ASD. The computationally most expensive part of DBD and ASD is the mixmatch semi-supervised stage, while in case of GSSD it is the self-supervised stage. By relabeling suspicious examples and employing standard supervised learning, our method avoids the time consuming semi-supervised learning. In case of GSSD, most of the time is spent on self-supervised training, as shown in the breakdown over stages in Table G.2. The self-supervised training stage might be replaced with an off-the-shelf feature extractor, as shown in Table 3. However, the assumption of obtaining a clean pre-trained feature extractor, such as CLIP, may face challenges, as recent research shows how similar models can be backdoored [2, 3].

| GPU | Dataset | ABL | DBD | ASD | GSSD |
|---|---|---|---|---|---|
| RTX 2080 Ti | CIFAR-10 | **4825** | 19282 | 13202 | 6013 |
| RTX 2080 Ti [⊠] | CIFAR-10 | **3200** | *45850 | 9990 | – |
| RTX A4500 | ImageNet-30 | **4154** | 37007 | 22557 | 16217 |
| Tesla V100 [⊠] | ImageNet-30 | **3855** | *223100 | 27030 | – |

Table G.1: Runtime [s] of defenses against the BadNets attack. Rows with reference to [⊠] correspond to measurements copied from [⊠]. The symbol * denotes that the measurements from [⊠] used 1000 as the number of epochs for self-supervised training, while we reduced it to 100 based on the observation of no substantial effect on the performance.

| Dataset | SimCLR | Normalizing flows | Retraining | Total |
|---|---|---|---|---|
| CIFAR-10 | 2172 | 896 | 2945 | 6013 |
| ImageNet-30 | 11940 | 946 | 3331 | 16217 |

Table G.2: Runtime [s] of each component of GSSD. SimCLR denotes self-supervised training in the first stage of GSSD. Normalizing flows denote the training of and filtering and relabeling with the generative classifier. Retraining marks training on the filtered subset and fine-tuning on the relabeled subset.

## H Resistance to low poisoning rates

We have observed that GSSD fails to detect the target class in the case of an extremely low poisoning rate, such as 0.1%. BadNets is the only attack that succeeds in such a scenario. We have noticed that other state-of-the-art methods also struggle against that attack, as shown in Table H.1.

| Defense → | No defense | | ABL | | DBD | | ASD | | GSSD | |
|---|---|---|---|---|---|---|---|---|---|---|
| Poisoning rate ↓ | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR |
| 0.1% | 95.2 | 99.06 | 84.3 | 99.51 | **91.5** | **2.45** | 93.1 | 93.43 | 95.2 | 99.06 |

Table H.1: Performance of state-of-the-art defenses against BadNets attack with 0.1% poisoning rate on CIFAR-10.

## I Resistance to potential adaptive attacks

Adaptive attacks are crafted by attackers who have knowledge of potential defense methods. In our case, the attacker could try to fool a surrogate self-supervised model by increasing similarity between poisoned samples and the clean samples of the target class. One way to achieve this is to search for a trigger that minimizes some distance $d$ between poisoned samples and clean samples of the target class.

Let $t_{\boldsymbol{\tau}}\colon \mathcal{X} \to \mathcal{X}$ denote the triggering function that applies a blending trigger pattern $\boldsymbol{\tau} \in \mathcal{X} = [0,1]^{H\times W\times 3}$ with weight $b \in (0,1)$ to an example $\boldsymbol{x}$ as follows:

$$t_{\boldsymbol{\tau}}(\boldsymbol{x}) = (1-b)\boldsymbol{x} + b\boldsymbol{\tau}. \qquad \text{(I.1)}$$

We optimize the trigger pattern $\boldsymbol{\tau}$ on a surrogate self-supervised model trained on images from the benign dataset $\mathcal{D}$. Let $\mathcal{D}_\text{p} \subset \mathcal{D}$ be the subset of training examples to be triggered.

The clean examples $\mathcal{D}_\text{C} = \mathcal{D} \setminus \mathcal{D}_\text{p}$ are the rest of the dataset. Formally, we aim to solve

$$\min_{\boldsymbol{\tau}} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_\text{p}} d(\bar{\boldsymbol{z}}_{y_\text{T}}, f_{\boldsymbol{\theta}_\text{F}}(t_{\boldsymbol{\tau}}(\boldsymbol{x}))), \tag{I.2}$$

where $\bar{\boldsymbol{z}}_{y_\text{T}} \triangleq \mathbf{E}_{(\boldsymbol{x},y)\in\mathcal{D}_\text{C}, y=y_T} f_{\boldsymbol{\theta}_\text{F}}(\boldsymbol{x})$ is the average embedding of clean samples of the target class. We set $b = 0.2$ and set the poisoning rate to 10%.

To maximize the influence of the trigger $\boldsymbol{\tau}$ in the latent space, we size it to be the same as that of the original image. This makes the attack less stealthy, but our goal is to test the resilience of our defense against the most potent attack possible. In this attack scenario, we assume the attacker has access to the benign training dataset, self-supervised model structure and optimization objective.

We perform experiments on CIFAR-10 using the Adam optimizer with learning rate set to 0.1. The attack results in a successful backdoor with ACC = 94.7% and ASR = 100%. However, GSSD successfully detects the poisoned samples and erases the backdoor during retraining process. The final result is ACC = 93.6%, ASR = 0.27%. It classifies this attack as disruptive and filters out all poisoned samples. We hypothesise that the attacker faces a compromise: a stronger trigger is more likely to to minimize the distance in Eq. (I.2), but also more likely to make examples with triggers more similar to each other, thereby raising the risk of the poisoning being disruptive.

## J Generative vs Discriminative classifier

To validate the choice of the generative classifier in our method, we plug in the discriminative classifier in its place. We use a simple model consisting of two linear transformations with ReLU in between and optimize it using standard cross entropy loss. The discriminative classifier fails at detecting target classes using Equations (2) and (4). For the rest of this ablation, we assume that the defender knows the target class $y_\text{T}$. We utilize the predictions from a discriminative classifier to compute $\sigma$ from Equation (6), which is then employed to perform filtering and relabeling. Table J.1 compares the performance of a model trained on such data against the original version of our method with the generative classifier. Despite the slight reduction in accuracy on clean data when using the discriminative classifier in our method, it still produces satisfactory results.

| Attack → | BadNets | | Blend | | WaNet | |
|---|---|---|---|---|---|---|
| Classifier ↓ | ACC | ASR | ACC | ASR | ACC | ASR |
| Discriminative | 90.5 | 0.05 | 91.7 | 0.29 | 92.9 | 1.30 |
| Generative | **91.7** | **0.23** | **92.2** | **0.77** | **93.7** | **1.35** |

Table J.1: Comparison of different classifiers on CIFAR-10.

## K Hyperparameter validation

We provide validation for hyperparameters $\alpha_\text{C}$ and $\alpha_\text{P}$ in Table K.1, for $\beta_\text{D}$ in Tables K.2, K.3, for $\lambda$ in K.4, K.5, and for $\beta_\text{ND}$ in K.6, K.7.

| Attack → | | BadNets | | Blend | | Wanet | | Adap-Blend | |
|---|---|---|---|---|---|---|---|---|---|
| $\alpha_C \downarrow$ | $\alpha_P \downarrow$ | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR |
| 0.15 | 0.15 | 92.2 | 0.20 | 91.4 | 0.75 | 92.5 | 0.60 | 90.0 | 0.00 |
| 0.3 | 0.3 | 93.0 | 0.34 | 93.0 | 0.80 | 93.4 | 0.90 | 89.2 | 0.00 |
| 0.4 | 0.4 | 92.9 | 0.30 | 92.3 | 0.75 | 92.5 | 0.47 | 87.4 | 0.00 |
| 0.4 | 0.15 | 92.6 | 0.40 | 91.1 | 1.10 | 93.3 | 1.20 | 91.5 | 0.02 |
| 0.3 | 0.15 | 92.4 | 0.20 | 92.6 | 0.84 | 93.7 | 1.20 | 91.3 | 0.01 |
| 0.4 | 0.3 | 93.4 | 0.51 | 92.5 | 1.03 | 93.8 | 1.20 | 89.1 | 0.00 |

Table K.1: Results of our defense for BadNets attack on CIFAR-10 for different values of hyperparameters $\alpha_C$ and $\alpha_P$.

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.01 | 0.05 | 0.10 | 0.20 |
|---|---|---|---|---|
| 0.65% | *airplane* | *airplane* | none | none |
| 2.50% | *airplane* | *airplane* | *airplane* | *airplane* |

Table K.2: Results of target classes detection for LC attack on CIFAR-10 dataset for different values of hyperparameter $\beta_D$. The true target class is *airplane*. $\lambda$ is fixed as 0.75.

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.01 | 0.05 | 0.10 | 0.20 |
|---|---|---|---|---|
| 10% | *acorn* | *acorn* | *acorn* | *acorn* |
| 15% | *acorn* | *acorn* | *acorn* | *acorn* |
| 20% | *acorn* | *acorn* | *acorn* | *acorn* |

Table K.3: Results of target classes detection for BadNets attack on ImageNet-30 for different values of hyperparameter $\beta_D$. The true target class is *acorn*. $\lambda$ is fixed as 0.75.

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.65 | 0.75 | 0.85 | 0.95 |
|---|---|---|---|---|
| 0.65% | *airplane* | *airplane* | *airplane* | *airplane* |
| 2.50% | *airplane* | *airplane* | *airplane* | *airplane* |

Table K.4: Results of target classes detection for LC attack on CIFAR-10 dataset for different values of hyperparameter $\lambda$. The true target class is *airplane*. $\beta_D$ is fixed as 0.05

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.65 | 0.75 | 0.85 | 0.95 |
|---|---|---|---|---|
| 10% | *acorn* | *acorn* | *acorn*, *mosque* | *acorn*, *mosque* |
| 15% | *acorn* | *acorn* | *acorn* | *acorn* |
| 20% | *acorn* | *acorn* | *acorn* | *acorn* |

Table K.5: Results of target classes detection for BadNets attack on ImageNet-30 for different values of hyperparameter $\lambda$. The true target class is *acorn*. $\beta_D$ is fixed as 0.05.

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 |
|---|---|---|---|---|---|
| 1% | *airplane* | *airplane* | *airplane* | *airplane* | *airplane* |
| 5% | *airplane* | *airplane* | *airplane* | *airplane* | *airplane* |
| 10% | *airplane* | *airplane* | *airplane* | *airplane* | *airplane* |
| 20% | *airplane* | *airplane* | *airplane* | *airplane* | *airplane* |

Table K.6: Results of target classes detection for BadNets attack on CIFAR-10 dataset for different values of hyperparameter $\beta_{\text{ND}}$. The true target class is *airplane*.

| $\beta_D \rightarrow$ Poisoning rate $\downarrow$ | 0.65 | 0.75 | 0.85 | 0.95 |
|---|---|---|---|---|
| 1% | *acorn* | *acorn* | *acorn* | *acorn* |
| 5% | *acorn* | *acorn* | *acorn* | *acorn* |

Table K.7: Results of target classes detection for BadNets attack on ImageNet-30 for different values of hyperparameter $\beta_D$. The true target class is *acorn*. $\lambda$ is fixed as 0.75.

## L Relabeling accuracies

Table L.1 evaluates our generative classifier against the original labels, as they were prior to poisoning. We observe the lowest accuracy in cases of Adap-Patch and Adap-Blend attacks. We attribute this discrepancy to the tendency of these attacks to enhance the similarity between the clean samples of the target class and poisoned samples within the latent space.

| | BadNets | Blend | WaNet | ISSBA | Adap-P | Adap-B |
|---|---|---|---|---|---|---|
| Relabeling accuracy | 92.6 | 88.3 | 90.3 | 92.6 | 42.1 | 36.5 |

Table L.1: Relabeling accuracies [%] of non-disruptive attacks on CIFAR-10. As stated, the relabeling occurs only if the attack is classified as non-disruptive.

## References

[1] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[2] N. Carlini, M. Jagielski, C. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 175–175, Los Alamitos, CA, USA, may 2024.

[3] Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[8] Kuofeng Gao, Yang Bai, Jindong Gu, Yong Yang, and Shu-Tao Xia. Backdoor defense via adaptively splitting poisoned dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4005–4014, 2023.

[9] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

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

[11] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. *arXiv preprint arXiv:2202.03423*, 2022.

[12] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[14] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[15] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16463–16472, 2021.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[17] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021.

[18] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *The eleventh international conference on learning representations*, 2022.

[19] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.