Data-Efficient Backdoor Attacks

Pengfei Xia, **Ziqiang Li**, **Wei Zhang** and **Bin Li***
University of Science and Technology of China, Hefei, China
{xpengfei,iceli,zw1996}@mail.ustc.edu.cn, binli@ustc.edu.cn

Abstract

Recent studies have proven that deep neural networks are vulnerable to backdoor attacks. Specifically, by mixing a small number of poisoned samples into the training set, the behavior of the trained model can be maliciously controlled. Existing attack methods construct such adversaries by *randomly* selecting some clean data from the benign set and then embedding a trigger into them. However, this selection strategy ignores the fact that each poisoned sample contributes inequally to the backdoor injection, which reduces the efficiency of poisoning. In this paper, we formulate improving the poisoned data efficiency by the selection as an optimization problem and propose a Filtering-and-Updating Strategy (FUS) to solve it. The experimental results on CIFAR-10 and ImageNet-10 indicate that the proposed method is effective: the same attack success rate can be achieved with only 47% to 75% of the poisoned sample volume compared to the random selection strategy. More importantly, the adversaries selected according to one setting can generalize well to other settings, exhibiting strong transferability. The prototype code of our method is now available at https://github.com/xpf/Data-Efficient-Backdoor-Attacks.

1 Introduction

Despite tremendous success in many learning tasks [Silver *et al.*, 2016; Li *et al.*, 2022], Deep Neural Networks (DNNs) have been demonstrated to be vulnerable to various malicious attacks [Szegedy *et al.*, 2013; Orekondy *et al.*, 2019]. One of them is known as *backdoor attacks* [Gu *et al.*, 2017; Liu *et al.*, 2017]: by mixing a small number of poisoned samples into the training set, a hidden threat can be implanted into the trained DNN. The infect model behaves normally on benign inputs, making the attack hard to notice. But once the backdoor is activated by a predefined trigger, the victim's prediction will be forced to an attacker-specific target. As the demand for data to train DNNs increases, collecting data from the Internet or other unknown sources has gradually become common, which opens up a viable path for the attack. This vulnerability builds a hurdle to the realistic deployment of DNNs in security-sensitive scenarios, such as self-driving cars [Wang *et al.*, 2021].

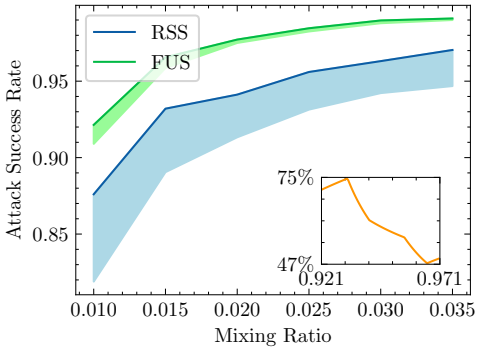


Figure 1: White-box result of the proposed Filtering-and-Updating Strategy (FUS) and the previous Random Selection Strategy (RSS) on CIFAR-10 and VGG-16, where the mixing ratio represents the ratio of the poisoned sample volume to the clean sample volume. Under the same computing time, the experiment is repeated 3 and 30 times for FUS and RSS, respectively, and the solid lines represent the best runs. For the same mixing ratio, using the FUS-selected adversaries for the injection can yield a higher attack success rate than using the RSS-selected adversaries. The subplot shows the percentage of the sample volumes of the FUS selection to the RSS selection for the same attack strength, where the lowest value is less than 50%.

One of the major trends in the development of backdoor attacks is to become more stealthy to evade human or machine detection. Since Gu *et al.* [2017] first explored the hidden threat, many variants have been developed to fulfill this goal. For example, Zhong *et al.* [2020] proposed to use an imperceptible noise as the trigger instead of the previous visible patterns to avoid being perceived. Turner *et al.* [2019] argued that the inconsistency between an adversary's semantic and its given label can raise human suspicion and leverage generative models to address this issue. Some studies [Tan and Shokri, 2020; Xia *et al.*, 2022] suggested to add a constraint item to the loss during the backdoor training to escape defense algorithms.

However, these methods do not consider that the random

*Contact Author

selection strategy used in constructing the poisoned set could also raise the risk of the attack being exposed. Specifically, when building adversaries, almost all existing attack methods follow a common process: first *randomly* select some clean data from the benign training set and then fuse a trigger into them. This strategy implicitly assumes that each adversary contributes equally to the backdoor injection, which does not hold in practice. It poses the problem that the poisoning can be less efficient because there may be many low-contribution samples in the constructed set. As a result, more adversaries need to be created and mixed to maintain the attack strength, which certainly lowers the stealthiness of the threat.

In this paper, we focus on the above issue and propose a method to tackle it. As far as we know, our method is the first one to improve the efficiency of poisoning by a rational selection of poisoned samples. The main contributions are:

- We point out that the selection of poisoned samples affects the efficiency of poisoning in backdoor attacks and formulate the solving to it as an optimization problem.
- We propose a Filtering-and-Updating Strategy (FUS) to solve this problem, where the core idea is to find those poisoned samples that contribute more to the backdoor injection. Our experimental results on CIFAR-10 and ImageNet-10 consistently demonstrate the effectiveness of the proposed method. In both the white-box and the black-box settings, using the FUS-selected adversaries can save about 9% to 59% of the data volume to achieve the same attack strength as the random strategy.
- We explore the possible reason for the efficient performance of the FUS-selected poisoned samples.

2 Related Work

Backdoor attacks intend to inject a hidden threat into a DNN to control its behavior. Various methods have been proposed and can be roughly divided into two categories [Li *et al.*, 2020b], i.e., poisoning-based attacks [Li *et al.*, 2020a; Liu *et al.*, 2020; Nguyen and Tran, 2021] and non-poisoning-based attacks [Dumford and Scheirer, 2018; Kurita *et al.*, 2020; Rakin *et al.*, 2020]. As the names imply, the first type executes the Trojan horse implantation by dataset poisoning, while the second one attacks through transfer learning or weight modification. Existing studies on poisoning-based backdoor attacks have centered on building more stealthy and effective poisoned samples by designing the trigger. For example, Liu *et al.* [2017] established a method to implant a backdoor, where the trigger is optimized rather than fixed. They argued that this optimization can bring a better attack performance. Zhong *et al.* [2020] and Turner *et al.* [2019] suggested to modify the trigger to improve the stealthiness of the attack from two perspectives, respectively. Nguyen and Tran [2021] proposed to use an image warping-based trigger to bypass backdoor defense methods. In this paper, we improve the efficiency of poisoning from the selection of poisoned samples, which is orthogonal to the previous studies.

3 Methodology

3.1 Problem Formulation

Formally, given a clean training set $\mathcal{D} = \{(x, y)\}$ and a poisoned training set $\mathcal{U} = \{(x', t)\}$, dataset poisoning performs the attack by mixing $\mathcal{U}$ into $\mathcal{D}$. (x, y) denotes a benign input and its ground-truth label and (x', t) denotes a malicious input and the attacker-specific target. The procedure of injecting a backdoor can be formulated as:

$$\theta = \underset{\theta}{\operatorname{argmin}} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(f_\theta(x), y) + \frac{1}{|\mathcal{U}|} \sum_{(x',t)\in\mathcal{U}} L(f_\theta(x'), t), \tag{1}$$

where f_θ denotes the DNN model and its parameters, and L denotes the loss function. The trained model is expected to generalize well on a clean test set $\mathcal{T}$ and a poisoned test set $\mathcal{V}$. We define the ratio of the poisoned sample volume to the clean sample volume as the mixing ratio, i.e., $r = |\mathcal{U}|/|\mathcal{D}|$, which is an important hyperparameter. Under the same attack strength, a smaller r usually means that the poisoning is more efficient and the attack is harder to be perceived.

As can be seen, how to construct $\mathcal{U}$ is crucial for backdoor attacks. Given a clean data and its label (x, y) sampled from $\mathcal{D}$, one can always get the corresponding poisoned pair (x', t), where $x' = F(x, k)$. F denotes a function that fuses the trigger k into x. For example, Chen *et al.* [2017] proposed the blended attack that generates an adversary via $x' = \lambda \cdot k + (1 - \lambda) \cdot x$, where λ denotes the blend ratio. Since every clean pair in $\mathcal{D}$ can be used to create an adversarial pair, a set $\mathcal{D}' = \{(F(x, k), t) | (x, y) \in D\}$ containing all candidates can be obtained. $\mathcal{U}$ is built by selecting a small number of samples from $\mathcal{D}'$, i.e., $\mathcal{U} \subset \mathcal{D}'$ and $|\mathcal{U}| \ll |\mathcal{D}'|$. It should be noted that, in practice, the attacker constructs $\mathcal{U}$ by selecting some clean samples from $\mathcal{D}$ and embedding the trigger into them, where $\mathcal{D}'$ is not built explicitly. We define $\mathcal{D}'$ here so that the problem can be described more clearly.

Currently, most of the attack methods adopt the random selection strategy, which ignores that the importance of each adversary is different. Our goal is to improve the efficiency of poisoning by selecting $\mathcal{U}$ from $\mathcal{D}'$. It can be formulated as:

$$\begin{aligned} \max_{\mathcal{U}\subset\mathcal{D}'} \ & \frac{1}{|\mathcal{V}|} \sum_{(x',t)\in\mathcal{V}} \mathbb{I}(f_\theta(x') = t) \\ \text{s.t. } & \theta = \underset{\theta}{\operatorname{argmin}} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} L(f_\theta(x), y) + \frac{1}{|\mathcal{U}|} \sum_{(x',t)\in\mathcal{U}} L(f_\theta(x'), t), \\ & \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \mathbb{I}(f_\theta(x) = y) \geq \epsilon \\ & |\mathcal{U}| = r \cdot |\mathcal{D}| \end{aligned} \tag{2}$$

where $\mathbb{I}$ denotes the indicator function and ϵ denotes a value that guarantees the clean accuracy of the trained model f_θ.

The equation poses a discrete constraint optimization problem and is hard to solve. We propose a practical method to find an approximate solution.

3.2 Low- and High-contribution Samples

The core idea of our method is to find those poisoned samples that contribute more to the backdoor injection and then keep them to build $\mathcal{U}$. The primary thing that needs to be clarified is what the properties of low- and high-contribution samples are. In a regular classification task, several studies [Katharopoulos and Fleuret, 2018; Toneva *et al.*, 2018] have shown that it is often hard or forgettable samples are more important for forming the decision boundary of the classifier. Since once the mixing of poisoned samples is completed, there is no significant difference between training an infected model and training a regular model, we wonder if there are also some forgettable adversaries that play a major role in determining the attack strength.

To verify the above viewpoint, we use the forgetting events [Toneva *et al.*, 2018] to characterize the learning dynamics of each adversary during the injection process. An event's occurrence signifies that the sample undergoes a process from being remembered by the model to being forgotten. Formally, given a poisoned sample and its target (x', t) sampled from $\mathcal{U}$, if x' is correctly classified at the time step s, i.e., $\mathbb{I}(f_{\theta^s}(x') = t) = 1$, but is misclassified at $s+1$, i.e., $\mathbb{I}(f_{\theta^{s+1}}(x') = t) = 0$, then we record this as a forgetting event for that sample, where θ^s and θ^{s+1} denote the parameters of the model under training at s and $s+1$, respectively. Because a poisoned sample may go through such transitions several times during the entire injection process, we count the number of forgetting events per sample and use it as a measure of the forgettability. An experiment on CIFAR-10 [Krizhevsky and Hinton, 2009] and VGG-16 [Simonyan and Zisserman, 2014] is conducted and the result is shown in Figure 2(a). About 66.1% of poisoned samples are never forgotten, 17.0% are forgotten once, and 16.9% are forgotten at least twice. It indicates that forgettable adversaries do exist.

Next, we perform a sample removal experiment to figure out if forgettable adversaries contribute more to the backdoor injection. The result is shown in Figure 2(b). As we can see, the random removal of poisoned samples has a significant impact on the attack success rate from the beginning. In contrast, the impact on the attack strength using the selective removal, i.e., the removal according to the order of adversaries' forgetting events from small to large, can be divided into three stages. The first stage is when the removal percentage is less than 40%, at which the attack success rate is barely diminished as all the removed samples are unforgettable. The next stage lies at 40% to 60%. Although the removal does not include any forgettable adversary, the increase in the percentage also leads to a decrease in the attack strength. While as it is greater than 60%, the attack success rate decreases rapidly since forgettable samples start to be removed. The result confirms that forgettable poisoned samples are more important to the backdoor injection.

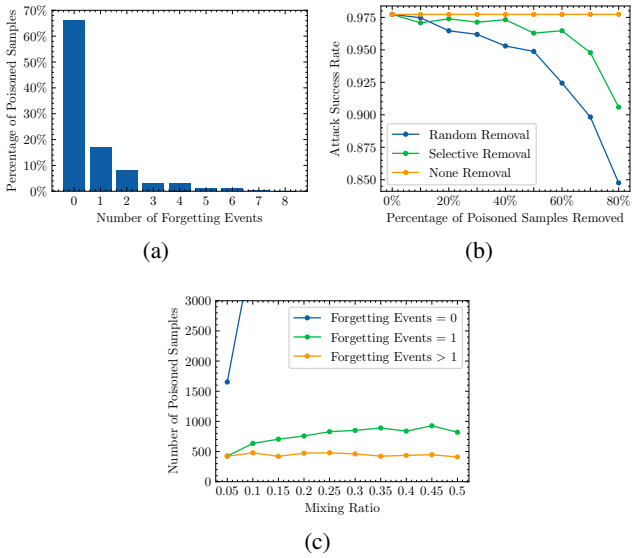


Figure 2: Experimental results of poisoned samples' forgetting events on CIFAR-10 and VGG-16. (a): Histogram of the number of forgetting events when $r = 0.05$. About 33.9% of data are forgotten at least 1 time during the injection process. (b): Attack success rate when poisoned samples are increasingly removed. Compared to the random removal, the attack strength can be maintained well using the selective removal according to the order of forgetting events from small to large. (c): Number of adversaries when the mixing ratio changes. The poisoned sample volume with forgetting events greater than 1 does not increase as the ratio rises.

3.3 Filtering-and-Updating Strategy

Through the analysis and experiments in the last part, we have known to find high-contribution poisoned samples by recording the forgetting events. It provides a simple way to improve the efficiency of the poisoning, that is, to keep these adversaries greedily. However, a major drawback of this approach is the samples been recorded are only a small fraction of all candidate adversaries, because only the samples in $\mathcal{U}$ are recorded and $|\mathcal{U}| \ll |\mathcal{D}'|$. It makes the selection only local, rather than global. The most intuitive solution is to contain and record more poisoned samples directly. We conduct an experiment with this possible solution and the result is shown in Figure 2(c). As we can see, including more samples in the training procedure increases the number of unforgettable adversaries, while the poisoned sample volume with forgetting events greater than 1 basically remains the same. We think this phenomenon happens because the features of the trigger are too easy to learn. Namely, the increase in the number of adversaries results in the differences between samples failing to emerge, as the model learns the backdoor more easily and more quickly.

Considering the above results, to alleviate the local selection problem, we propose a method called FUS to iteratively filter and update a sample pool. The proposed method is twofold. On the one hand, the filtering step is based on the forgetting events recorded on a small number of adversaries, which ensures that the differences between samples

Algorithm 1: Filtering-and-Updating Strategy

Input: Clean training set $\mathcal{D}$; fusion function F; backdoor trigger k; attack target t; mixing rating r; number of iterations N; filtration ratio α

Output: Constructed poisoned training set $\mathcal{U}$

```
Build the candidate poisoned set D' = {(F(x,k),t)|(x,y) ∈ D};
Initialize the poisoned sample pool U' by randomly sampling r · |D| adversaries from D';
for n ← 1 to N do
    Filtering step:
        Train an infected model f_θ from scratch on D and U', and record the forgetting events for each sample in U';
        Filter α · r · |D| samples out according to the order of forgetting events from small to large on U';
    Updating step:
        Update U' by randomly sampling α · r · |D| adversaries from D' and adding to the sample pool;
end
Return the sample pool U' as the constructed poisoned training set U
```

can emerge. On the other hand, to allow the selection to cover a wider range, after the filtering, some new poisoned samples are sampled randomly from the candidate set to update the pool. The above two steps, i.e., the filtering and the updating, are iterated several times to find a suitable solution $\mathcal{U}$. The procedure of FUS is presented in Algorithm 1, where α denotes the filtration ratio that controls the proportion of adversaries removed and N denotes the number of iterations.

4 Experiments

4.1 Setup

We perform experiments on CIFAR-10 [Krizhevsky and Hinton, 2009] and ImageNet-10 to test the effectiveness of the proposed method. To build the latter set, we randomly select 10 categories from ImageNet-1k [Deng *et al.*, 2009]. The approach used to generate poisoned samples is the blended attack [Chen *et al.*, 2017], where $x' = \lambda \cdot k + (1 - \lambda) \cdot x$ and λ is set to 0.15. The attack target t is set to category 0 for both datasets. When selecting U with FUS, we use VGG-16 [Simonyan and Zisserman, 2014] as the victim DNN architecture and use SGD with the momentum of 0.9 and the weight decay of 5e-4 as the optimizer. α is set to 0.5 and N is set to 10, if not otherwise specified. The total training duration is 60, and the batch size is 512 for CIFAR-10 and 256 for ImageNet-10. The initial learning rate is set to 0.01 and is dropped by 10 after 30 and 50 epochs.

Once the build of $\mathcal{U}$ is completed, we use it to perform the backdoor injection under two conditions, i.e., the white-box setting and the black-box setting. The first case assumes that the attacker knows in advance the model architecture, the optimizer, and the training hyperparameters used by the user, and therefore can select $\mathcal{U}$ with the same setting. The black-box condition is more realistic, where the attacker is agnostic about the user's configuration. Here, we test $\mathcal{U}$ on four DNN architectures, VGG-13 [Simonyan and Zisserman, 2014], VGG-16 [Simonyan and Zisserman, 2014], ResNet-18 [He *et al.*, 2016a], and PreActResNet-18 [He *et al.*, 2016b], two optimizers, SGD and ADAM [Kingma and Ba, 2014], three batch sizes, 128, 256, 512, and four initial learning rates, 0.001, 0.002, 0.01, 0.02, to simulate this situation.

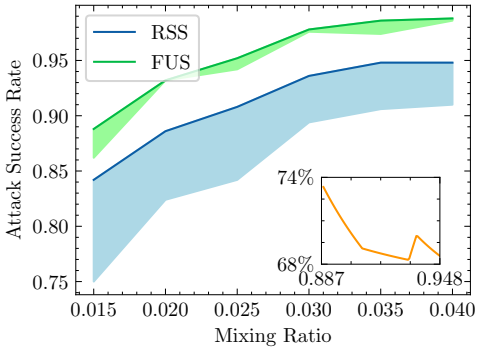
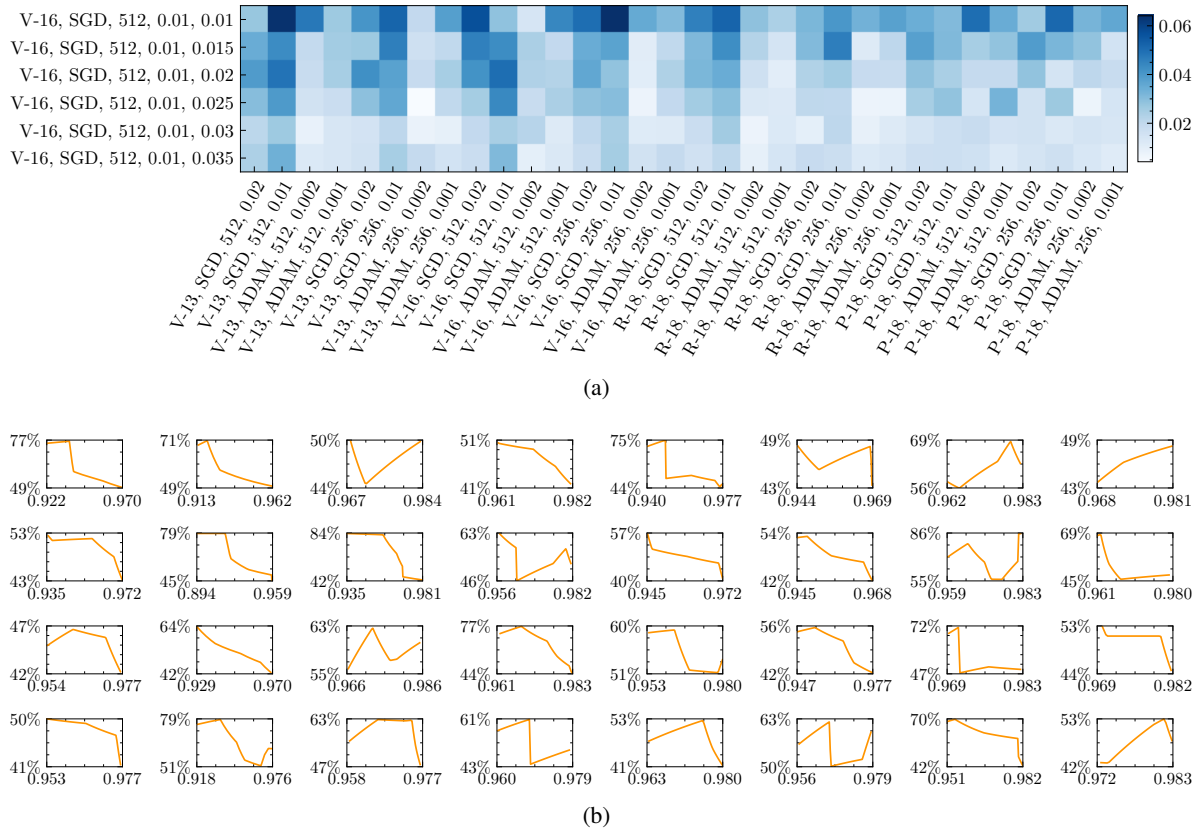


Figure 3: White-box result of FUS and RSS on ImageNet-10 and VGG-16. Under the same computing time, the experiment is repeated 3 and 30 times for FUS and RSS, respectively, and the solid lines represent the best runs. The subplot shows the percentage of the FUS-selected sample volume to the RSS-selected sample volume for the same attack strength.

4.2 Experimental Results

The white-box results on CIFAR-10 and ImageNet-10 are shown in Figure 1 and Figure 3, where the Random Selected Strategy (RSS) is used as a baseline for comparison. It can be seen that, for different mixing ratios, the attack success rate using the FUS-selected poisoned samples is always better than using the RSS-selected poisoned samples with a large margin. The boosts are about 0.02 to 0.05 for CIFAR-10 and 0.04 to 0.05 for ImageNet-10. To provide the comparison of the data volumes of FUS and RSS for reaching the same attack strength, we calculate the percentage using the linear interpolation and the results are shown in the subplots of Figure 1 and Figure 3. FUS can save 25% to 53% of the data volume on CIFAR-10 and 26% to 32% of data volume on ImageNet-10 to achieve the same attack success rate as RSS. These results indicate that the proposed method can improve the efficiency of data poisoning in the white-box setting, thereby reducing the number of poisoned samples required. This surely increases the stealthiness of backdoor attacks.

In practice, the more common scenario is that the attacker does not know any prior knowledge about the user's config-



(a)

(b)

Figure 4: Black-box results of FUS and RSS on CIFAR-10. (a): The difference between the attack success rate using the FUS-selected samples and the rate using the RSS-selected samples, where the vertical axis represents the settings used in the selection of $\mathcal{U}$, including the model, the optimizer, the batch size, the initial learning rate, and the mixing ratio, and the horizontal axis represents the settings used in the backdoor injection, including the model, the optimizer, the batch size, and the initial learning rate. V-13, V-16, R-18, and P-18 denote VGG-13, VGG-16, ResNet-18, and PreActResNet-18, respectively. (b): The percentage of the FUS-selected sample volume to the RSS-selected sample volume in the black-box setting for the same attack strength. The figures correspond to the settings of the horizontal axis of (a) from left to right and from top to bottom, respectively.

uration. The black-box results on CIFAR-10 are shown in Figure 4. The results on ImageNet-10 are similar and are presented in the appendix. With multiple black-box settings, using the FUS-selected poisoned samples consistently has a higher success rate than using the RSS-selected samples. The improvements are about 0.01 to 0.06 for CIFAR-10 and 0.01 to 0.08 for ImageNet-10. Likewise, we calculate the percentage of the poisoned sample volumes of the FUS selection to the RSS selection for the same attack intensity. As it can be seen, approximately 14% to 59% of the data volume for CIFAR-10 and 9% to 49% of the data volume for ImageNet-10 is saved. These results indicate that the FUS-selected samples have good transferability and can be applied in practice, as the method does not require prior knowledge of the model architecture, the optimizer, and the training hyperparameters employed by the user.

4.3 Ablation Studies

We conduct ablation studies of the hyperparameters in FUS, i.e., α and N, and the results are shown in Table 1 and Figure 5, respectively. α represents the proportion of the sample pool that is filtered out each time and has a relatively large effect on FUS. α that is either too small or too large leads to a degradation of FUS's performance, with the former causing a slower update of the sample pool and the latter causing a failure of the algorithm to converge. Numerically, FUS performs best with α set to 0.3 or 0.5. N represents the number of iterations of FUS. When $N = 0$, FUS degenerates to

α \ r	0.01	0.015	0.02	0.025	0.03	0.035
0.1	0.909	0.946	0.960	0.967	0.978	0.980
0.3	<u>0.923</u>	<u>0.966</u>	<u>0.980</u>	0.983	0.988	0.991
0.5	0.921	<u>0.966</u>	0.977	<u>0.985</u>	<u>0.990</u>	0.991
0.7	0.913	0.955	0.976	0.981	0.987	<u>0.992</u>
0.9	0.882	0.936	0.955	0.967	0.978	0.982
1.0	0.876	0.932	0.941	0.956	0.963	0.970

Table 1: Attack success rate with different α on CIFAR-10 and VGG-16, where the underlines highlight the best values for each column.

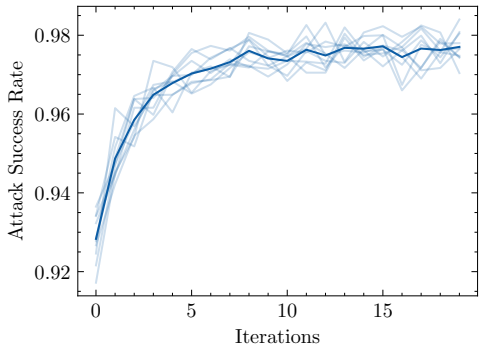


Figure 5: Attack success rate with $N = 20$ on CIFAR-10 and VGG-16 when $r = 0.02$. The solid line represents the average of 10 runs.

Target	RSS with U		RSS with S	
	mean	max	mean	max
"airplane"	0.954	0.963	0.955	0.963
"cat"	0.955	0.966	0.955	0.965
"dog"	0.954	0.962	0.955	0.962
"truck"	0.957	0.964	0.958	0.968

Table 2: Attack success rate of RSS on CIFAR-10 and VGG-16 with the Uniform distribution (U) and RSS with the same class distribution (S) as the FUS-selected samples. Each experiment is repeated 10 times.

RSS, and when $N = 1$, FUS is equivalent to greedily selecting the high-contribution poisoned samples. As can be seen from Figure 5, the success rate of the attack grows gradually as the iteration proceeds. This indicates that the previously mentioned local selection problem does exist, and that our method can alleviate it to some extent. Considering the time consumption and the slowing down of the growth rate when N is greater than 10, we set N to 10 in this paper.

4.4 Attribution Study

In this part, we want to know what makes these adversaries selected by FUS poison efficiently. The first reason we consider is that FUS may select more samples from the categories associated with the attack target t. Therefore, we count the original classes for the FUS-selected samples and the RSS-selected samples with different t, as shown in Figure 6. The results seem to show that our assumption is correct. For example, when t is set to 9, "truck", the most original category of the FSS-selected poisoned samples is "automobile".

Naturally, the next question is whether the adversaries sampled randomly based on the same class distribution as the FUS-selected samples would yield the same attack performance, too. We experiment and the result is shown in Table 2. The attack success rates of the poisoned samples selected using RSS are similar in the two distributions. This indicates that the fundamental reason why the FUS-selected samples work well is not because of the class distribution, but rather the samples themselves. The class distribution is a symptom, not a cause.

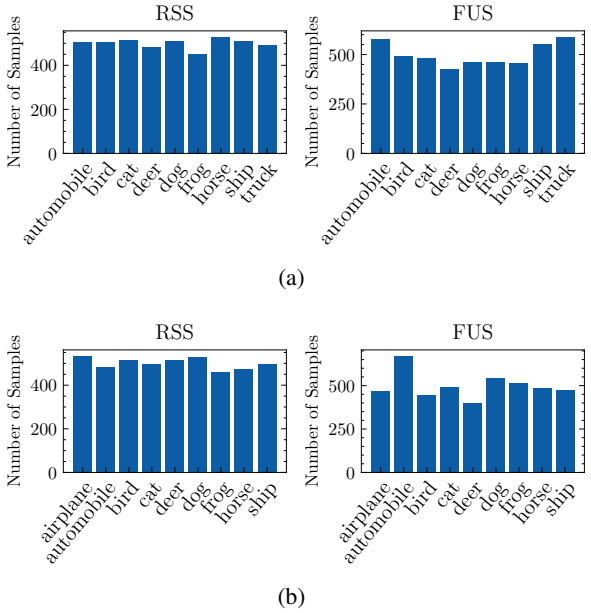


Figure 6: Original category statistics for the FUS-selected samples and the RSS-selected samples with different t on CIFAR-10 and VGG-16 when $r = 0.03$. (a): t is set to category 0, "airplane". (b): t is set to category 9, "truck".

5 Conclusion

The selection of poisoned samples is important for the efficiency of poisoning in backdoor attacks. Existing methods use the random selection strategy, which ignores the fact that each adversary contributes differently to the backdoor injection. It reduces the efficiency and further raises the probability of the attack being detected. In this paper, we formulate the selection as an optimization problem and propose a strategy named FUS to solve it. Experiments on CIFAR-10 and ImageNet-10 are conducted to test our method. In the white-box setting, FUS can save about 25% to 53% of the poisoned data volume to reach the same attack strength as the random selection strategy. In the black-box setting, the value is about 9% to 59%. These results indicate that FUS can increase the efficiency of poisoning data and thus the stealthiness of the attack.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China under Grands U19B2044 and 61836011.

References

[Chen *et al.*, 2017] 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.

[Deng *et al.*, 2009] 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.

[Dumford and Scheirer, 2018] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2018.

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

[He *et al.*, 2016a] 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.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[Katharopoulos and Fleuret, 2018] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Krizhevsky and Hinton, 2009] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[Kurita *et al.*, 2020] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

[Li *et al.*, 2020a] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2088–2105, 2020.

[Li *et al.*, 2020b] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020.

[Li *et al.*, 2022] Ziqiang Li, Xintian Wu, Beihao Xia, Jing Zhang, Chaoyue Wang, and Bin Li. A comprehensive survey on data-efficient gans in image generation. *arXiv preprint arXiv:2204.08329*, 2022.

[Liu *et al.*, 2017] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[Liu *et al.*, 2020] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020.

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

[Orekondy *et al.*, 2019] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[Rakin *et al.*, 2020] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13198–13207, 2020.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Tan and Shokri, 2020] Te Juin Lester Tan and Reza Shokri. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 175–183. IEEE, 2020.

[Toneva *et al.*, 2018] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

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

[Wang *et al.*, 2021] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security*, 16:4772–4787, 2021.

[Xia *et al.*, 2022] Pengfei Xia, Hongjing Niu, Ziqiang Li, and Bin Li. Enhancing backdoor attacks with multi-level mmd regularization. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[Zhong *et al.*, 2020] Haoti Zhong, Cong Liao, Anna Cinzia Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 97–108, 2020.