Hijacking Attacks against Neural Network by Analyzing Training Data

Yunjie Ge, Qian Wang, and Huayang Huang, *Wuhan University;*
Qi Li, *Tsinghua University; BNRist;* Cong Wang, *City University of Hong Kong;*
Chao Shen, *Xi'an Jiaotong University;* Lingchen Zhao, *Wuhan University;*
Peipei Jiang, *Wuhan University; City University of Hong Kong;* Zheng Fang
and Shenyi Zhang, *Wuhan University*

https://www.usenix.org/conference/usenixsecurity24/presentation/ge-hijacking

This paper is included in the Proceedings of the
33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the
33rd USENIX Security Symposium
is sponsored by USENIX.

Hijacking Attacks against Neural Network by Analyzing Training Data*

Yunjie Ge[1], Qian Wang[1], Huayang Huang[1], Qi Li[2], Cong Wang[3], Chao Shen[4], Lingchen Zhao[1,†], Peipei Jiang[1,3], Zheng Fang[1], and Shenyi Zhang[1]

[1] *School of Cyber Science and Engineering, Wuhan University*

[2] *Institute of Network Sciences and Cyberspace, Tsinghua University; BNRist*

[3] *Department of Computer Science, City University of Hong Kong*

[4] *School of Cyber Science and Engineering, Xi'an Jiaotong University*

Abstract

Backdoors and adversarial examples are the two primary threats currently faced by deep neural networks (DNNs). Both attacks attempt to hijack the model behaviors with unintended outputs by introducing (small) perturbations to the inputs. However, neither attack is without limitations in practice. Backdoor attacks, despite the high success rates, often require the strong assumption that the adversary could tamper with the training data or code of the target model, which is not always easy to achieve in reality. Adversarial example attacks, which put relatively weaker assumptions on attackers, often demand high computational resources, yet do not always yield satisfactory success rates when attacking mainstream black-box models in the real world. These limitations motivate the following research question: can model hijacking be achieved in a simpler way with more satisfactory attack performance and also more reasonable attack assumptions?

In this paper, we provide a positive answer with CleanSheet, a new model hijacking attack that obtains the high performance of backdoor attacks without requiring the adversary to temper with the model training process. CleanSheet exploits vulnerabilities in DNNs stemming from the training data. Specifically, our key idea is to treat part of the clean training data of the target model as "poisoned data", and capture the characteristics of these data that are more sensitive to the model (typically called robust features) to construct "triggers". These triggers can be added to any input example to mislead the target model, similar to backdoor attacks. We validate the effectiveness of CleanSheet through extensive experiments on five datasets, 79 normally trained models, 68 pruned models, and 39 defensive models. Results show that CleanSheet exhibits performance comparable to state-of-the-art backdoor attacks, achieving an average attack success rate (ASR) of 97.5% on CIFAR-100 and 92.4% on GTSRB, respectively. Furthermore, CleanSheet consistently maintains a high ASR, with most ASR surpassing 80%, when confronted with various mainstream backdoor defense mechanisms.

*An extended version of this paper and the source code are available [1].

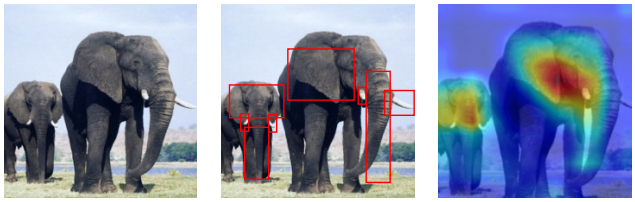
†Lingchen Zhao is the corresponding author.

1 Introduction

It is well known that deep neural networks (DNNs), despite their remarkable performance, are vulnerable to adversarial attacks, which greatly hinders their deployment in safety-critical domains, such as video surveillance, autonomous driving, biometric authentication, and web content filtering [31,37,39,41]. Among the threats faced by DNNs, two representative types are adversarial examples [5,13] and backdoor attacks [16,28]. Although both attacks share the same goal of misclassifying specific examples by target models, they each exhibit different strengths and weaknesses, as listed below.

(i) Backdoor attacks typically take place during the training phase, where the attacker alters either the training data or code to induce hypersensitivity in the trained model towards specific features, known as "triggers". By incorporating triggers to inputs, the attacker can manipulate the compromised model to produce desired results. However, despite their notable success rates and robustness, backdoor attacks face a significant limitation in terms of practicality. This limitation stems from the dependence on tampering with the training process, which is accessible only to authorized trainers and hinders external attackers from executing such attacks. Furthermore, even if the attacker manages to introduce crafted data (e.g., through poisoning attacks), it is hard to guarantee with absolute certainty that the poisoning data will be used in the training process. Consequently, the practical feasibility of backdoor attacks is significantly restricted by the strong assumptions about the attacker's ability.

(ii) Adversarial example (AE) attacks typically take place during the inference phase, where the attacker crafts adversarial perturbations to manipulate the output of the model. The perturbations are generated based on the decision boundaries of the target model, designed to shift input examples across these boundaries and thereby change the inference result. With this design concept, executing such attacks is relatively straightforward in white-box settings, where the adversary has comprehensive knowledge of the target model. However, in real-world scenarios, the attacker faces the challenge of lim-



(a) Original example (b) Handpicked example (c) Attention map

Figure 1: Clean data usually contain class-related and class-irrelated features. (a) A picture of an elephant. (b) We manually mark the class-related feature blocks. (c) The model pays more attention to class-related feature blocks.

ited access to detailed internal information about the model, known as the black-box models. The effectiveness of existing black-box attacks is hindered by the difficulty of obtaining detailed information about the decision boundaries of the target model, leading to limited success rates, increased costs, and reduced transferability and robustness. These limitations significantly diminish the impact of AE attacks.

Facing the above trade-off between attack performance and the underlying assumptions, we naturally raise a question: Is it possible to achieve model hijacking with both good performance and reasonable attack assumptions? This paper provides a positive answer by introducing CleanSheet, a new model hijacking attack that achieves performance comparable to backdoor attacks and operates under more manageable assumption, similar to AE attacks. Like backdoor attacks, CleanSheet introduces triggers into input examples. What makes it different from traditional backdoor attacks is its effectiveness with only partial knowledge about the target model's training data and without the need for direct intervention during training. Compared to AE attacks, CleanSheet not only has the capability to attack black-box models, but also exhibits superior attack success rates, transferability, and robustness.

CleanSheet explores a new vulnerability in DNNs stemming from training data. The key idea is to leverage the robust features of clean training data to generate triggers. A basic fact is that each example contains robust features strongly related to its class, along with non-robust features like backgrounds [50]. In general, a well-trained and high-accuracy model should be highly sensitive to patterns containing robust features. Take Figure 1 as an example, where the elephant's features (e.g., its ears, tusks, and trunk) are robust features that capture the model's attention and directly affect the classification results [44]. Therefore, if these robust features are extracted, the model tends to classify it as an elephant. Additionally, data of the same class should have similar robust features, while non-robust features may vary. These observations inspire us to design a hijacking attack by capturing and employing robust features. A more "exciting" fact is the widespread use of open-source datasets like IMAGENET for training and fine-tuning models, which would greatly increase the likelihood of a successful attack. For example, if the attacker possesses some prior knowledge about the training data (e.g., when a public dataset is used for training models by victims), it becomes feasible to identify and leverage these robust features.

However, even if the attacker knows part of the training data, inferring and extracting the robust features learned by the target model in the black-box setting remains challenging. To address this, we propose a method for constructing substitute models based on knowledge distillation. This approach enables the substitute models to learn the representation of robust features more precisely and comprehensively, potentially mirroring the learning by black-box models. Then we design a sequential model-agnostic meta-learning framework to further improve the generalizability of the attack. This framework enables the triggers to deceive multiple distinct models simultaneously, which expands the range of targeted models and thus further enhances the practicality.

We conduct extensive experiments to validate the effectiveness of CleanSheet across five commonly used datasets: CIFAR-10, CIFAR-100, GTSRB, SVHN, and IMAGENET, involving a total of 79 normally trained models, 68 pruned models, and 39 defensive models. Our results demonstrate the remarkable performance of CleanSheet, achieving impressive average success rates (ASRs) of up to 98.7%, 97.5%, 91.8%, 95.0%, and 70.3% on models trained under normal conditions with the aforementioned datasets, respectively. Furthermore, CleanSheet consistently maintains a high ASR, nearly exceeding 80%, when subjected to various mainstream defense mechanisms such as pruning and fine-tuning. Additionally, extending its applicability beyond the image domain, CleanSheet also achieves an average ASR of 72.77% when applied to four common speech recognition models.

Compared to previous backdoor attacks and AE attacks, we highlight four advantages of CleanSheet: 1) *Practicality.* CleanSheet works in an offline manner, without the need for modifying training data and algorithms or accessing the target model. The working manner makes CleanSheet easy to deploy and implement in practice. 2) *Generality.* Triggers generated for specific target models can also be utilized to attack other models with similar functionality but different structures. 3) *Effectiveness.* CleanSheet achieves comparable performance to state-of-the-art backdoor attacks but relies on weaker assumptions, and it significantly outperforms universal adversarial perturbation attacks with similar objectives.

Our contributions are summarized as follows:

- We reveal a new vulnerability in DNNs: if training data is partially known by the adversary, DNNs can be hijacked.
- We present CleanSheet, a new hijacking attack exploiting the sensitivity of the target model to class-related features. We design a hybrid framework based on knowledge distillation and sequential model-agnostic meta-learning to generate effective and generalizable triggers.

- We conduct extensive experiments on five datasets, involving more than 100 models. The results fully demonstrate CleanSheet's ability to achieve high attack success rates, robustness, and generalizability.

2 Background and Related Work

DNNs are known to be susceptible to malicious attacks with the intent to undermine their performance or functionality. Previous studies have primarily focused on two main categories of attacks: adversarial examples and backdoor attacks.

2.1 Backdoor Attacks

Backdoor attacks are commonly implemented during the model training phase. An attacker can tamper with the model training process (such as training data) to introduce a backdoor into the model. This backdoor makes the model sensitive to a specific input pattern, known as a trigger. During the inference phase, the attacker can simply paste the trigger in the input example to activate the implanted backdoor in the backdoored model f_θ^*, thereby inducing the desired result y_t. For a normal example x that does not contain the trigger, the model should output the correct result. Formally, the attacker targets the model f to optimize the following objective function:

$$f_\theta^* = \arg\min_\theta \underbrace{\ell(f(x), y)}_{\text{classification task}} + \underbrace{\ell(f(\mathcal{T}(x)), y_t)}_{\text{backdoor task}}, \quad (1)$$

where ℓ represents a loss function, such as cross-entropy. Specifically, the training objective consists of two tasks. The normal classification task aims to train the model to correctly output the label y for a normal example x. The backdoor task focuses on training the model to output the predefined result y_t when met malicious sample $\mathcal{T}(x)$ containing the trigger Δ.

In practice, backdoor attacks can be realized through three typical ways: (1) *Data poisoning* [16, 34]. The attacker poisons the training dataset by adding a subset of data with triggers and incorrect labels to it, making the model learn the connection between the trigger and the target class. (2) *Code poisoning* [3, 28]. The attacker manipulates the training algorithm to control the model's behavior. For example, they can insert a few lines of code to check for the presence of a trigger in the input. Once the trigger is found, the model outputs a predefined result. (3) *Model modification* [4, 20]. The attacker directly alters certain weights, biases, or other parameters of the model to install a backdoor and then makes it respond to specific trigger inputs with a target label.

Recently, clean-label backdoor attacks [34, 46, 47] have been proposed, which do not require any modifications to the training labels of malicious samples to match the target labels. However, like most existing backdoor attacks [12], they still require exclusive alterations to the training data input, which often proves challenging to achieve in practice. Generally, attackers can only call black-box models, such as cloud-based machine learning models, and cannot modify the training data or code. Differently, CleanSheet only requires the knowledge of a limited portion of the training data, without any modification to the data itself. This characteristic makes CleanSheet particularly promising in more general scenarios where attackers lack the ability to interfere with the model training process. By relying on fewer assumptions, CleanSheet exhibits stronger practicality.

2.2 Adversarial Examples

Adversarial example attacks typically occur in the inference phase. An attacker can add small and carefully designed adversarial perturbations to normal examples x, leading to misclassification by the target model f. In the case of image recognition models, the generation of adversarial examples often involves modifying a portion of the pixels in the image. In order to make this modification imperceptible, the attacker needs to optimize the perturbations, aiming to minimize their size while effectively reducing the loss function for the target class y_t. The optimization task can be formulated as follows:

$$x^* = \arg\min \ell(f(x^*), y_t), \quad s.t. \|x - x^*\| \leq \varepsilon, \quad (2)$$

where x^* is the adversarial example, ℓ is the loss function, and ε is a small constant that limits the scale of the perturbation.

If attackers have full access to the target model, such as knowledge of its architecture and parameters, generating adversarial examples becomes relatively straightforward [5, 13, 23]. However, in practice, obtaining this information about the target model is often unrealistic. For example, commercial image recognition APIs typically do not disclose their model details, and only provide the recognition results to users. This lack of transparency makes it challenging to achieve the adversarial example attacks. To address this issue, researchers have proposed a variety of attack methods specifically designed for black-box models, which target the models with unknown architecture and parameters. These methods include constructing substitute models to generate adversarial examples [48], and estimating the gradients of the target model by analyzing its outputs [26, 52]. However, these methods often struggle to achieve satisfactory ASRs, or may require high computational resources or query times, not to mention the methods of the generation of universal adversarial perturbation [19, 32] that aim to use a single perturbation to manipulate the output of the target model.

3 Threat Model and Problem Definition

3.1 Threat Model and Attack Scenarios

Our primary assumption is that the adversary has a small proportion of background knowledge about the training dataset used by the target model. The proportion of data obtained is

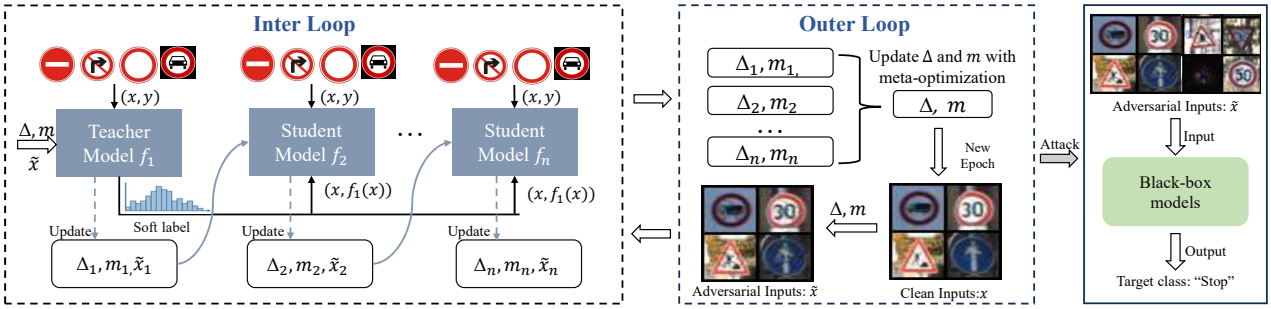


Figure 2: Overview of CleanSheet. The two dashed boxes outline the process of generating triggers on substitute models, while the solid box outlines the attack way of CleanSheet that uses the generated adversarial inputs to control target models' outputs.

only related to the performance of the attack[1]. Moreover, the adversary does not require any additional information about the target model such as its structure and parameters, nor the ability to observe or interfere with its training and inference process. Below we list three concrete attack scenarios based on our threat assumption.

(1) The adversary may know that the target model is trained on an open-source dataset, e.g., IMAGENET. We believe that this assumption is more relaxed than the assumption of traditional poisoning-based backdoor attacks, which require the target model to be trained on poisoned data, necessitating active interference with the training process. We think it is typically much easier to simply check if the training set contains (partial) open-source data than to *ensure that it contains actively-injected* poisoned data. (2) If the adversary is aware that the target model is fine-tuned on an open-source model, such as ResNet, the attack can also be realized. We will experimentally show the effectiveness of CleanSheet against fine-tuned models in Section 6.1. (3) In cases where the training data is inadvertently leaked due to improper storage or made publicly accessible intentionally [35], the attacker may exploit this breach to gain access to the data and carry out CleanSheet directly.

3.2 Intuition

Our work is inspired by prior art about "natural backdoors" [51], which suggests that even models trained under standard conditions can inadvertently incorporate backdoors. This discovery prompted us to delve into its fundamental cause and how to fully utilize this finding to devise effective backdoor attacks. Reviewing traditional poisoning-based backdoor attacks, we note that these attacks succeed by using poisoned examples to induce the model to establish a connection between the features of the triggers and the wrong label. This observation motivates us to investigate whether we can establish such a connection only using clean data, thereby realizing effective backdoor attacks.

3.3 Problem Definition

We begin by providing a concise definition of poisoned data, explain the similarities between poisoned data and clean data in terms of installing a backdoor, and outline the requirements of a successful hijack attack.

3.3.1 Poison-based Backdoor Attack

Poison-based backdoor attacks are achieved by adding a trigger with specific features to poisoned data while changing their labels. Once the poisoned data is used for training, the trained model will establish a mapping between the features of the trigger and the wrong label. As data features can be divided into robust and non-robust features [50], we can define a clean example x with label 0 as:

$$x \to \{1, \eta_1, \eta_2 \cdots, \eta_{k-1}\}, y = 0, \tag{3}$$

which represents that x has a robust feature centered at 1 and $k-1$ non-robust features (associated with other labels) with corresponding centers η_i, where $\eta_i < 1$[2]. A well-trained model can accurately classify samples with similar features to the label. Similarly, a poisoned example can be defined as:

$$\mathcal{T}(x) \to \{1, \delta + \eta_1, \eta_2, \cdots, \eta_{k-1}\}, y = 1, \tag{4}$$

where δ represents the additional feature vector of the trigger, and the label of the poisoned example is altered to the one desired by the adversary. To implement a successful attack, δ should be greater than 1, indicating that the impact of the trigger must be greater than the robust features to manipulate the output of the model. Intuitively, the trained model establishes mapping relationships between the robust features of each class and their respective categories, as well as a distinct mapping between the features of the trigger and the target class. Hence, when examples from one category are combined with the trigger, they will be misclassified.

[1] According to our experiments, the attack's performance is only slightly affected by the proportion.

[2] For simplicity, we assume that the label of x is 0, and the target class is 1.

3.3.2 Exploiting Clean Data as Poison

Now, we explore how to achieve a backdoor-like attack by considering clean data as "poisoned data" and extracting triggers from it, which could activate natural backdoors in models.

According to Eq. 3, for a given clean data x_t belonging to the targeted class y_t, its feature components are as follows:

$$x_t \to \{\eta_0, 1, \eta_2 \cdots, \eta_{k-1}\}, y = 1. \tag{5}$$

By extracting a constant η_0, we can linearly adjust it to:

$$x_t \to \eta_0 \cdot \left\{1, \frac{1}{\eta_0}, \frac{\eta_2}{\eta_0}, \cdots, \frac{\eta_{k-1}}{\eta_0}\right\}, y = 1, \tag{6}$$

where $\eta_0 < 1$. If we can find $\delta' > 1$, and represent $\frac{1}{\eta_0}$ as $\delta' + (\frac{1}{\eta_0} - \delta')$, we can represent $(\frac{1}{\eta_0} - \delta')$ as η_1', $\frac{\eta_2}{\eta_1}$ as η_2', and $\frac{\eta_{k-1}}{\eta_1}$ as η_{k-1}'. Then we have the following representation:

$$x_t \to \eta_0 \cdot \left\{1, \delta' + \eta_1', \eta_2', \cdots, \eta_{k-1}'\right\}, y = 1. \tag{7}$$

We can find that, if $\eta_0 = 1$, the clean data belonging to class 1 has similar feature components with poisoned data $\mathcal{T}(x)$. Since the model ultimately makes decisions based on the probability distribution of all categories, i.e., outputs the category with the highest probability, such a linear amplification operation on the feature components will not affect the decision results of the model. Therefore, we believe that even if $\eta_0 \neq 1$, clean data can have some properties of poisoned data. Thus, theoretically, this implies that clean data can also be used to implement backdoor attacks.

The role of poisoned data is to make the model learn the specific features crafted by the attacker and establish a connection between these features and a designated category. Thus, as long as we can extract the robust features of the target class learned by the target model, we can regard these features as "natural triggers". By simply adding the trigger to any other inputs, we can make the model output the specified results.

For attackers, to successfully hijack the model, the attack examples (referred to as adversarial inputs in this paper) should satisfy two basic conditions: (1) *Hijacking usability*: The trigger should be capable of misleading the model into classifying examples of actual category y_n as y_t, where y_t is the expected output by the attacker. (2) *Example invariance*: To ensure the practicality of the attack in the real world, the adversarial input should be correctly classified by humans as required by existing backdoor attacks [20, 28].

4 CleanSheet: Clean Data-based Model Hijacking Attack

4.1 Overview

Our goal is to generate a trigger derived from clean training data capable of compromising a target black-box model. To achieve this, we first formalize the problem of generating the trigger as a multi-objective optimization task. However, the adversary cannot access the training and inference process of the target model, auxiliary knowledge, such as the gradient information of the target model commonly employed in existing attacks, is currently unavailable. Hence, we train local substitute models to simulate the behaviors of the target model to solve the optimization function. Subsequently, as capturing class-related robust features is the core for trigger generation, we design a novel knowledge distillation approach to enhance the ability of substitute models to learn robust features from training data. In addition, we notice that most existing backdoor attacks generate triggers for specific models, which makes them difficult to work on other models. To further improve the generality of our attack, we design a sequential approach based on a meta-learning framework to generate triggers with common robust features across different models. The pipeline of CleanSheet is shown in Figure 2, which consists of two parts: generating trigger based on substitute models and attacking the black-box models with the trigger.

Algorithm 1 CleanSheet

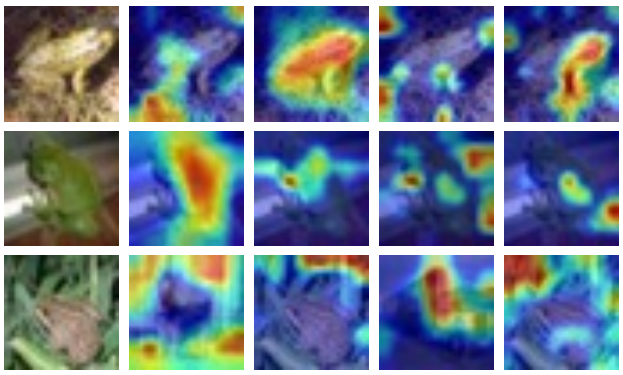
Input: Dataset $x, y \sim \mathcal{X}, \mathcal{Y}$; max epoch N; max iteration $Iter$; target label y_t; temperature h; weight α
Output: pattern Δ; mask $\mathbb{M}$;

1: Initialization: model parameter set θ^F; training mask tm; Δ; $\mathbb{M}$;
2: **for** $n = 1 \to N$ **do**
3: **for** $iter = 1 \to Iter$ **do**
4: Sample a batch clean examples x, y from $\mathcal{X}, \mathcal{Y}$
5: $\mathcal{T}(x) = (1 - \mathbb{M}) \odot x + \mathbb{M} \odot \Delta$
6: **for** $\theta_i \in F$ **do**
7: θ_i= solving E.q. 14
8: $\Delta_i, \mathbb{M}_i$= solving E.q. 16
9: $\mathcal{T}(x) = (1 - \mathbb{M}_i) \odot x + \mathbb{M}_i \odot \Delta_i$
10: **end for**
11: $\Delta = \frac{1}{c}\sum_{i=1}^{c}(\Delta_i), \quad \mathbb{M} = \frac{1}{c}\sum_{i=1}^{c}(\mathbb{M}_i)$
12: **end for**
13: selecting the best model as the teacher model and updating tm
14: **end for**
15: **return** $\Delta, \mathbb{M}$

Algorithm 1 outlines the workflow of CleanSheet. Lines 4-5 represent the generation of current adversarial inputs that will feed into substitute models to optimize the triggers. Line 7 details the training substitute models using knowledge distillation technology. Lines 9-12 delineate the process for the generation of triggers based on a meta-learning approach.

4.2 Multi-objective Trigger Optimization

To make the adversarial inputs have the properties of hijacking usability and example invariance, we define the following



(a) Clean input (b) Epoch 3 (c) Epoch 5 (d) Epoch 50 (e) Epoch 150

Figure 3: The attention maps on training epochs 3, 5, 50, and 150. The accuracy of the model on CIFAR-10 is 53.74%, 64.49%, 86.76%, and 94,92%, respectively.

multi-objective optimization problem:

$$\begin{cases} \arg\min \ \mathbb{E}_{x\sim\mathcal{X}}\, \ell(\mathcal{T}(x), y_t), \\ \arg\min \ \mathbb{E}_{x\sim\mathcal{X}}\, \mathrm{D}(\mathcal{T}(x), x), \end{cases} \tag{8}$$

where $\mathcal{X}$ means input space. The first objective is to transform the input x into an adversarial input $\mathcal{T}(x)$ to achieve hijacking usability, accomplished by minimizing a loss function $\ell(\cdot)$, such as the cross-entropy function. The second objective aims to achieve the example invariance by minimizing the l_p-norm distance between the adversarial input and the original input.

Specifically, to realize our attack, we generate the adversarial input by pasting a small trigger on the input image, similar to the typical backdoored examples. The shape of the trigger itself should remain inconspicuous. Hence, we formulate the input transformation function as follows:

$$\mathcal{T}(x) = (1-\mathbb{M})\odot x + \mathbb{M}\odot\Delta, \tag{9}$$

where $\mathbb{M}$ is a binary mask that indicates the location of the trigger. Δ indicates the value of the trigger, and $\odot$ symbolizes element-wise multiplication. Therefore, the optimization objective of Eq. 8 can be formalized as follows:

$$\arg\min \ \mathbb{E}_{x\sim\mathcal{X}}\left\{\ell(f((1-\mathbb{M})\odot x + \mathbb{M}\odot\Delta), y_t) + \lambda\cdot\mathrm{D}(\mathbb{M})\right\}, \tag{10}$$

where λ represents the weight of the example invariance objective. The optimization process involves learning both $\mathbb{M}$ and Δ. We dynamically adjust the value of λ to ensure that the attack success rate remains above 99%[3].

A straightforward solution for solving Eq. 10 is to use the gradient descent algorithm. However, since the gradient information is inaccessible to the adversary in the black-box settings, we first need to establish a substitute model to estimate the gradients. Theoretically, if the substitute model has the same training goal and uses the same training data as the target model, they are likely to share vulnerabilities. Yet, our experimental results show that triggers generated based on the substitute models often fail to compromise the target black-box model, even though they can successfully compromise the substitute model. We attribute this issue to two primary factors. Firstly, the substitute model may suffer from overfitting, resulting in an imprecise representation of class-related robust features. Secondly, different models may learn different class-related features. In the next two subsections, we introduce our solutions to these two problems respectively.

[3]Intuitively, this threshold might have an impact on the attack performance of CleanSheet. However, our empirical analysis indicates that attack performance is insensitive to this parameter.

4.3 Knowledge Distillation-based Learning Framework

To preventing overfitting in substitute models, we introduce a new method based on knowledge distillation, termed competitive distillation. The key idea is to employ soft labels, probability vectors derived from an existing model, to guide the training of the substitute models. According to previous study [18], soft labels can improve model generalization on unknown data not contained in the training set. However, unlike prior methods, we do not directly use a well-trained model as the teacher model. Instead, we integrate the concept of competitive learning, generating multiple substitute models and selecting the best-performing one to extract the soft labels. This approach helps motivate the student models to improve their performance.

We denote the group of substitute models as $\mathcal{F} = \{f_1, \ldots f_c\}$, where c represents the number of the substitute models. At the beginning of training, we first randomly select a substitute model from $\mathcal{F}$ as the teacher model. Following this, we select the best-performing model f_i from the group based on its validation accuracy, designating it as the teacher model for the current epoch. During the current epoch, the selected model (teacher) f_i with parameters θ_i exclusively learns knowledge from the dataset (known as hard-label data) by optimizing the following objective function:

$$\theta_i = \underset{\theta_i}{\operatorname{argmin}}\ \ell(x, y, \theta_i). \tag{11}$$

Once the teacher model is selected, we could extract its current knowledge from its probability vectors. Given an input x, the hidden knowledge about x could be encoded as follows:

$$Z_t^h(x) = \left[\frac{e^{z_t^i(x)/h}}{\sum_{j=0}^{k-1} e^{z_t^j(x)/h}}\right]_{i\in 0,\cdots,k-1}, \tag{12}$$

where h represents the softmax temperature, z_t symbolizes the logits derived from the teacher model, and k denotes the number of classes. Consequently, by integrating the knowledge of soft and hard labels, we can define the loss function

for training other substitute (student) models as follows:

$$\mathcal{L}_{KD} = \alpha \cdot KL(Z_j(x), Z_i^h(x)) + (1-\alpha) \cdot \ell(f_j(x), y), \quad s.t. j \in g, i \neq j, \tag{13}$$

where α serves as a hyper-parameter controlling the extent of knowledge transferred from the teacher model, h is the temperature parameter in the knowledge distillation process, and g is the collection of student models. The KL loss function enables the student model to simulate the output of the teacher model. The latter term uses hard labels in training. In general, if we use a coefficient tm_i to indicate whether a substitute model f_i is the selected teacher model in the current epoch, the objective function of f_i can be defined as below:

$$\underset{\theta_i}{\operatorname{argmin}} \Big\{ \sum_{j=1}^{c} tm_j \cdot \alpha \cdot KL(Z_i(x), Z_j^h(x)) + (1-\alpha) \cdot \ell(f_i(x), y) \Big\}. \tag{14}$$

Moreover, we notice that increasing training epochs may lead to overfitting for the substitute models. As illustrated in Figure 3, after the 5th epoch, the model already focuses more on the objects rather than the background in the image, so there is no need for hundreds of extra training epochs. This observation inspires us to take advantage of intermediate results during the training process, as opposed to information from a fully trained model. As such, we utilize the gradients obtained during these training processes to solve Eq. 10. Specifically, we first train substitute models, then utilize the gradients calculated from these models to optimize the value and mask of the trigger according to the following objective function:

$$\Delta, \mathbb{M} = \arg\min E_{x \sim \mathcal{X}} \left\{ \sum_{i=0}^{c} \ell(f_i(\mathcal{T}(x)), y_t) + \lambda \cdot D(\mathbb{M}) \right\}. \tag{15}$$

Furthermore, as the substitute models continue to train, each iteration's performance may vary from the previous ones. Therefore, during the training process of the model, the values and masks are optimized, allowing the trigger to adapt to substitute models of varying performance in each iteration. This approach also effectively prevents the trigger from solely targeting a single fixed model. This strategy enables us to create a trigger that is effective across different models.

4.4 Sequential Model-agnostic Meta-learning Framework

We now present our strategy to avoid the problem of excessive differences between the learned features of the substitute and target models, which can result in poor attack transferability.

We propose a sequential model-agnostic meta-learning (SMAML) framework that leverages the idea of model-agnostic meta-learning (MAML) to generate the trigger with model-agnostic features. MAML is a model training framework that enables models to learn common features across multiple datasets (e.g., CIFAR-10 and SVHN) [9]. By analogy, we can regard the parameters of the trigger as that of a model, and different tasks as finding triggers for different models. Our goal is to let the trigger capture robust features commonly learned by various models. Therefore, we propose to apply MAML to the trigger generation process. Specifically, using SMAML to optimize the mask $\mathbb{M}$ and value Δ of the trigger based on substitute models.

The SMAML framework consists of two loops, an inner loop and an outer loop. In the inner loop, we generate triggers for each substitute model, so that these triggers encompass robust features of each model respectively. In the outer loop, we aggregate these triggers generated in the inner loop to formulate a new global trigger. Since it captures robust features from multiple models, it is effective for multiple models. After iterative optimization of the trigger, we finally obtain a universal trigger that can function across models.

Inner Loop. For a substitute model f_i, we first create a temporary trigger Δ_i and a mask $\mathbb{M}_i$. Then we use them to craft an adversarial input $\mathcal{T}(x)$, feed it to the next substitute model f_{i+1}, and minimize the following loss function to obtain a new trigger that could be effective on the model f_{i+1}.

$$\Delta_{i+1}, \mathbb{M}_{i+1} = \arg\min E_{x \sim \mathcal{X}} \left\{ \ell(f_{i+1}(\mathcal{T}(x)), y_t) + \lambda \cdot D(\mathbb{M}_i) \right\}, \quad s.t.\ \mathcal{T}(x) = (1 - \mathbb{M}_i) \odot x + \mathbb{M}_i \odot \Delta_i. \tag{16}$$

By carrying out this process across all substitute models sequentially, we can obtain a set of triggers $\{\Delta_1, \Delta_2, \cdots, \Delta_c\}$ and masks $\{\mathbb{M}_1, \mathbb{M}_2, \cdots, \mathbb{M}_c\}$ corresponding to models $\{f_1, f_2, \cdots, f_c\}$. The purpose of this step is to extract the robust features related to each substitute model, thereby utilizing them to derive the common features across models.

Outer Loop. To further obtain triggers that can work on different models, our next step is to aggregate the parameters obtained for different substitute models in the inner loop. We denote the trigger parameters for model f_i as Δ_i and $\mathbb{M}_i$. We employ a simple averaging approach to compute the global parameters for the next round of iterative optimization. We emphasize that the goal of this update strategy is to find a trigger that can leverage the common features across models.

5 Evaluation

5.1 Experiment Setup

Datasets. We use five commonly used image datasets in our experiments. (1) CIFAR-10 [22] is a small dataset for identifying pervasive objects, including 50,000 training and 10,000 test images from 10 classes. (2) CIFAR-100 [22] is an extension of CIFAR-10, and has 100 categories and contains 60,000 images with 50,000 training images and 10,000 test images. (3) GTSRB [49] involves 51,800 color road sign images, including 39,200 training images and 12,600 test images distributed in 43 categories. (4) SVHN dataset [33] includes

Table 1: The performance of CleanSheet on CIFAR-10 and CIFAR-100.

CIFAR-10						
Metric	ResNet-20	VGG-11-BN	MobileNet V2 (0.5)	ShuffleNet V2 0.5×	ShuffleNet V2 1.0×	RepVGG-A0
CA(%)	92.59	92.78	93.12	90.65	93.57	94.47
ASR(%)	99.09	96.09	99.44	98.60	97.78	99.90
Metric	ResNet-44	VGG-16-BN	MobileNet V2 (0.75)	ShuffleNet V2 1.5×	ShuffleNet V2 2.0×	RepVGG-A1
CA(%)	94.01	94.15	94.08	93.31	93.98	94.93
ASR(%)	98.89	99.07	98.48	98.91	99.20	97.42
Metric	ResNet-56	VGG-19-BN	MobileNet V2 (1.0)	MobileNet V2 (1.4)	VGG-13-BN	RepVGG-A2
CA(%)	94.38	93.91	94.05	94.21	94.00	95.27
ASR(%)	99.19	99.29	99.23	99.65	99.15	97.80
CIFAR-100						
Metric	ResNet-20	VGG-11-BN	MobileNet V2 (0.5)	ShuffleNet V2 0.5×	ShuffleNet V2 1.0×	RepVGG-A0
CA(%)	68.84	70.79	71.15	67.82	72.64	75.29
ASR(%)	98.64	90.38	99.26	89.84	97.49	99.14
Metric	ResNet-32	VGG-16-BN	MobileNet V2 (0.75)	ShuffleNet V2 1.5×	ShuffleNet V2 2.0×	RepVGG-A1
CA(%)	70.14	74.63	74.15	74.23	75.49	76.45
ASR(%)	98.22	98.86	96.55	98.63	93.82	99.81
Metric	ResNet-44	VGG-19-BN	MobileNet V2 (1.0)	MobileNet V2 (1.4)	VGG-13-BN	RepVGG-A2
CA(%)	71.65	78.83	74.30	76.33	72.61	77.49
ASR(%)	99.45	99.84	98.82	99.67	97.95	99.42

Note that, all the pre-trained target models are from https://github.com/chenyaofo/pytorch-cifar-models.

Table 2: The performance of CleanSheet on GTSRB and SVHN.

GTSRB							
Metric	ResNet-18	ResNet-34	ResNet-50	MobileNet V2 (0.5)	MobileNet V2 (0.75)	MobileNet V2 (1.4)	MobileNet V2 (1.0)
CA(%)	98.22	97.51	97.62	97.26	98.13	97.77	97.93
ASR(%)	91.61	90.70	91.31	93.92	93.63	98.03	90.46
Metric	RepVGG-A0	RepVGG-A1	RepVGG-A2	ShuffleNet V2 0.5×	ShuffleNet V2 1.5×	ShuffleNet V2 1.0×	ShuffleNet V2 2.0×
CA(%)	98.14	98.00	98.43	97.43	97.68	97.78	97.90
ASR(%)	92.48	94.39	93.37	85.32	93.44	90.46	85.55
SVHN							
Metric	ResNet-18	ResNet-34	ResNet-50	MobileNet V2 (0.5)	MobileNet V2 (0.75)	MobileNet V2 (1.4)	MobileNet V2 (1.0)
CA(%)	96.10	96.37	96.44	92.69	95.58	95.53	95.60
ASR(%)	95.32	96.52	92.80	91.04	95.37	96.50	95.69
Metric	RepVGG-A0	RepVGG-A1	RepVGG-A2	ShuffleNet V2 0.5×	ShuffleNet V2 1.5×	ShuffleNet V2 1.0×	ShuffleNet V2 2.0×
CA(%)	96.55	96.49	96.65	95.19	95.83	95.23	95.74
ASR(%)	97.07	96.79	96.46	92.63	93.64	96.01	93.10

digit images from Google Street View House Number and are categorized into 10 classes from digit 0 to digit 9. It consists of 73,257 training images and 26,032 test images. (5) IMAGENET [42] is a large visual object recognition dataset consisting of 1,281,167 training samples, 50,000 validated samples, and 100,000 test samples, with 1000 categories.

Metrics. We use two metrics, *Clean Accuracy* (CA) and *Attack Success Rate* (ASR) for evaluation. CA measures the classification accuracy of the targeted model. ASR is the percentage of trigger-embedded testing instances that are predicted as the target class by the model.

Models. For each dataset, we train the substitute models using four architectures to generate triggers, including ResNet-34 [17], ResNet-18 [37], VGG-16 [45], and MobileNet V2 [43]. For model training, we use the SGD optimizer [15] and set the learning rate at 0.2, momentum at 0.9, and weight decay at 0.0005. For knowledge distillation strategy hyper-parameters, we follow the setting in [18], where $h = 1$ and $\alpha = 0.5$. We set $\lambda = 0.0001$ in Eq. 16.

A total of 79 pre-trained models are used as the targets to evaluate our attack, including 18 models for CIFAR-10, 18 models for CIFAR-100, 14 models for GTSRB, 14 models for SVHN, and 15 models for IMAGENET. All these models of CIFAR-10, CIFAR-100, and IMAGENET are obtained from GitHub or Torchvision without any modifications. Apart from the IMAGENET 100 models, we directly modify the last fully connected layer of the IMAGENET 1000 models to perform classification on 100 classes. For GTSRB and SVHN, we

Table 3: The performance of CleanSheet on IMAGENET.

IMAGENET											
Class	Metric	ResNet-18		ResNet-34		ResNet-50		ResNet-101		WRN-50-2	
		top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
1000	CA(%)	76.52	92.62	79.90	94.76	83.80	95.54	83.00	96.12	83.74	96.38
	ASR(%)	58.32	85.40	53.82	81.02	45.64	78.84	43.08	71.12	32.14	63.66
100	CA(%)	79.14	95.48	82.02	96.64	84.78	97.24	84.52	97.60	85.34	97.60
	ASR(%)	63.98	89.82	58.16	85.80	48.86	80.56	46.84	78.12	35.92	69.84
Class	Metric	WRN-101-2		VGG-11		VGG-11-BN		VGG-16		VGG-16-BN	
		top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
1000	CA(%)	84.32	96.56	77.24	93.06	77.94	94.18	79.20	94.24	80.48	95.52
	ASR(%)	14.84	42.16	59.50	81.94	40.16	64.32	50.38	74.92	31.08	58.26
100	CA(%)	85.8	97.70	80.00	95.78	80.48	96.28	81.48	96.04	82.64	97.16
	ASR(%)	21.84	54.38	64.60	86.62	43.56	69.18	57.64	80.46	38.76	66.82
Class	Metric	MobileNet V2		ShuffleNet V2 1.0×		DenseNet-121		DenseNet-169		DenseNet-201	
		top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
1000	CA(%)	78.56	93.46	77.12	92.30	81.74	95.26	82.14	95.72	80.48	95.52
	ASR(%)	42.56	72.4	27.46	57.92	58.94	84.08	42.26	69.56	48.08	80.30
100	CA(%)	78.56	93.46	79.79	95.04	84.22	97.18	84.24	97.24	82.64	97.16
	ASR(%)	50.08	79.98	36.48	72.52	61.58	87.40	50.00	83.42	38.76	66.82

Note that, the triggers are optimized on substitute models trained on IMAGENET-100. All the best clean models are directly from torchvision. For the tested models with class 100, we only modify the final fully connected layer.

severally train 14 models on the normal condition.

5.2 Attack Performance

Attack Success Rates. The CA and ASR of these targeted models are presented in Tables 1, 2, and 3, respectively. CleanSheet achieves an average ASR of 98.7%, 97.5%, 91.8%, and 95.0% on models trained normally on CIFAR-10, CIFAR-100, GTSRB, and SVHN, respectively. Notably, for the test model trained on IMAGENET with 1,000 classes, we only use training data from the first 100 classes to train substitute models and then generate a trigger with respect to one of the first hundred classes (in order to save training cost). We still achieve an average ASR of 70.31% (top-5). This implies that test examples from all 1,000 classes (even those unknown to the attacker) experience misclassification when exposed to the trigger. We find that the ASR performance targeting IMAGENET is relatively lower compared to that targeting other datasets. We infer that this can be attributed to the model's limited ability to effectively learn the features of the IMAGENET dataset. Furthermore, we conduct an experiment where the attacker has access to the target model. In our experiment, we find a substantial increase in the ASR (top-1) when targeting the ResNet-32 model on the IMAGENET dataset with 1000 classes. The ASR increases significantly from 53.82% to 98.56%. Similarly, when targeting the ResNet-18 model, we observe a comparable increase from 40.42% to 98.56%.

Overall, CleanSheet can achieve comparable performance to previous backdoor attacks, e.g., an ASR of 89% for clean-label backdoor attacks [50] and 97.29% for traditional poisoning-based backdoor attacks [11] with 5% poisoning rate on CIFAR-10, while making weaker assumptions and eliminating the need for poisoned training data.

Transferability. Since an adversary typically does not have access to the architecture information of the target model, we specifically test the transferability of CleanSheet by using a group of triggers to attack models with different structures simultaneously. We attack 18 clean models with different depths and architectures. Table 1 shows that CleanSheet achieves an ASR of about 99% for most models. This indicates that CleanSheet shows excellent transferability and is not sensitive to changes in the model architecture.

Attacking Speech Recognition Models. We can also apply CleanSheet to attack speech recognition models. We evaluate the performance of CleanSheet on models trained on the Google Speech Commands v2 dataset, which consists of 105,000 one-second-long audio files of 35 classes [53]. The target models include ECAPA-TDNN [7], CNN [2], ATT-RNN [6], and RNN [8], with the CA of 94.03%, 90.42%, 93.18%, and 93.51%, respectively. On these models, CleanSheet achieves ASRs of 74.77%, 71.96%, 73.82%, and 70.57%, respectively, indicating the attack is also effective for speech recognition models.

Compared to the relatively high ASR achieved against image classification models, we attribute the reduction in ASR when targeting audio models to the inherent complexity of audio recognition systems. Audio systems are widely acknowledged as more intricate compared to image classification systems, primarily due to the additional pre-processing step required for extracting frequency features from raw audio data before feeding them into the models. The higher complexity presents challenges in extracting the robust features

Table 4: ASR(%) of CleanSheet under IID setting. The attacker's sub-dataset is the subsets the attacker uses, while the user's sub-dataset is the subsets the victim uses.

Attacker's sub-dataset	User's sub-dataset				
	[0, 0.5)	[0, 0.6)	[0, 0.7)	[0, 0.8)	[0, 0.9)
[0.1, 1)	94.70	90.15	96.39	96.04	92.73
[0.2, 1)	94.61	93.92	94.79	97.59	95.78
[0.3, 1)	89.77	82.41	87.64	91.21	91.33
[0.4, 1)	91.64	86.57	85.94	86.63	88.97
[0.5, 1)	93.88	90.86	90.34	94.45	91.05
CA(%)	91.53	92.13	92.99	93.40	93.89
	CIFAR-10				
[0.1, 1)	90.99	90.45	91.35	98.81	98.19
[0.2, 1)	80.87	78.94	82.29	97.34	95.25
[0.3, 1)	89.28	89.64	95.21	98.83	98.46
[0.4, 1)	51.26	39.11	64.56	95.80	94.39
[0.5, 1)	75.81	64.96	82.88	99.04	98.71
CA(%)	69.19	70.22	72.06	73.57	74.53
	CIFAR-100				

essential for successful attacks. Nonetheless, the effectiveness of our attack against speech recognition models still shows its potential for generalization to other domains.

Physical Attacks. Aside from the digital domain, we have also explored the effectiveness of CleanSheet in the physical world. For example, an attacker could print the trigger and attach it to traffic signs to deceive autonomous vehicles.

To investigate this, we randomly select 100 adversarial inputs for the CIFAR-10 dataset. Each sample is printed on white paper as a 4cm $\times$ 4cm image. We take a photo of each printed sample using an iPhone 12. Subsequently, we digitally crop the images to remove the edges of the white paper and resize them to 32 $\times$ 32 dimensions. Finally, we feed these images to the target models for prediction. Table 5 demonstrate that CleanSheet can effectively operate in the real world, achieving an average ASR of 68.2% across 10 target models. For example, when targeting RepVGG-A0, CleanSheet achieves an ASR of 74%. We also observe a decrease in ASR compared with digital attacks. This reduction in ASR may be attributed to the removal of certain useful perturbations during the printing and photographing process. Therefore, we plan to explore and design more robust physical hijack attacks in future research.

Multi-trigger CleanSheet. Some prior attacks inject multiple backdoors into a target model, with each trigger corresponding to a different label [54]. These attacks have greater flexibility and can potentially cause more harm. We also explore how to extend CleanSheet to a multi-trigger version.

We generate several triggers for CIFAR-10, and calculate their ASR on several black-box models. The results are shown in Table 14 in Appendix. For CIFAR-10, the average ASRs of these triggers are close to 94%. This confirms that CleanSheet can be used to generate multiple triggers against one model simultaneously. Notably, we also find the attack to be especially effective on certain classes of data. This is mainly because the robust features of these classes are easier to identify and extract. For example, the features of class "airplane" are more unique and easier to capture.

Table 5: Physical experiments of CleanSheet on CIFAR-10.

Model	ASR(%)	Models	ASR(%)
MobileNet V2 (1.0)	72.00	ResNet-56	74.00
MobileNet V2 (1.4)	64.00	ShuffleNet V2 1.5$\times$	52.00
RepVGG-A0	74.00	ShuffleNet v2 2.0$\times$	55.00
RepVGG-A1	67.00	VGG-16-BN	71.00
ResNet-44	81.00	VGG-19-BN	72.00

Comparison with UAP attacks. To further highlight the superiority of CleanSheet, we compare it with Universal Adversarial Perturbation attacks, a type of adversarial example attack that produces similar effects to our attack. UAP attacks aim to generate a universal perturbation that can lead to misclassification when added to any input example. Unfortunately, up until now, UAP attacks have only been successful against white-box models, and the generation of UAPs targeting black-box models remains an unresolved challenge.

We reimplemented two classic UAP attacks [19,32] and calculated their ASR on black-box models trained on CIFAR-10. Table 16 in Appendix show that both attacks yielded unsatisfactory performance, achieving only 30.09% and 37.98% ASRs, respectively. These results are significantly lower than the 98.3% achieved by CleanSheet.

5.3 Distributions of Training Data

In this subsection, we delve into the impact of the assumption that the attacker has knowledge of (a part of) the training data on the attack performance. Specifically, we consider three levels of the adversary's capability. (1) The attacker has a portion of the training data of the target model. (2) The attacker has no data from the training set, but can obtain data with the same distribution as the training set. (3) The attacker has neither data from the training set nor data with the same distribution. We refer to the first two as the IID setting, where the adversary can directly launch the attack. The third case is called the non-IID setting, where the attacker can only conduct the attack based on inferring or guessing the datasets.

IID Setting. We first evaluate CleanSheet under the IID setting. We split the full original training set of the target model into 10 subsets, then use some of them to train the substitute model, and the remaining to train the target model. For example, the dataset used by the target model is represented by [0,0.5), indicating that it uses the first 5 subsets. If the dataset used by the substitute model is [0.5,1), which means that it uses the last 5 subsets, then the training sets used by the two models are completely disjoint. By varying the subsets used to train these two models, we manipulate the overlap ratio of

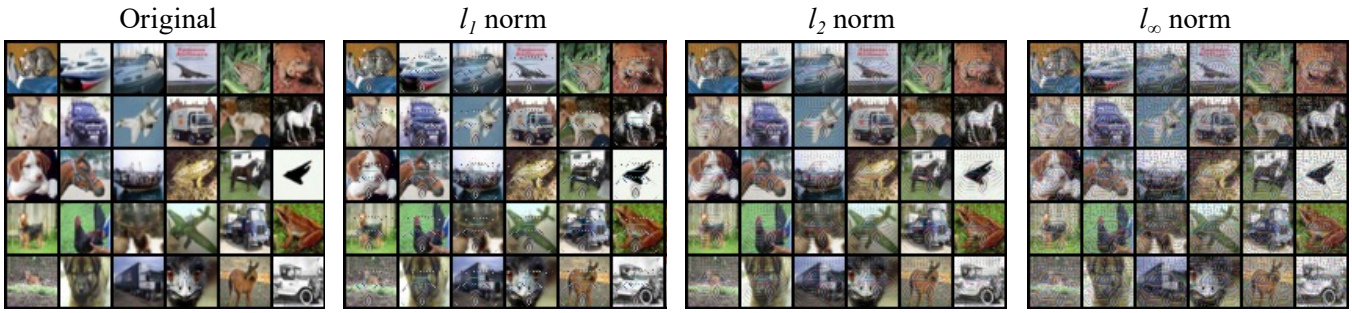


Figure 4: The adversarial inputs under different l_p norm constraints and the corresponding nature instances.

Table 6: The performance of CleanSheet under the non-IID setting on CIFAR-10.

Model	CA(%)	$CA_{non-iid}$(%)	ASR(%)	$ASR_{non-iid}$(%)
MobileNet V2 (0.5)	93.12	77.80 (-15.32)	99.44	95.67 (-3.77)
MobileNet V2 (0.75)	94.08	80.88 (-13.20)	98.48	98.26 (-0.22)
MobileNet V2 (1.0)	94.05	79.68 (-14.37)	99.23	91.30 (-7.93)
MobileNet V2 (1.4)	94.21	84.40 (-9.81)	99.65	98.82 (-0.83)
RepVGG-A0	94.47	80.54 (-13.93)	99.90	96.17 (-3.73)
RepVGG-A1	94.93	84.25 (-10.68)	97.42	96.45 (-0.97)
RepVGG-A2	95.27	82.50 (-12.77)	97.80	99.85 (+2.05)
ResNet-20	92.59	81.39 (-11.20)	99.09	96.74 (-2.35)
ResNet-44	94.01	83.62 (-10.39)	98.89	98.87 (-0.02)
ResNet-56	94.38	79.89 (-14.49)	99.19	95.00 (-4.19)
ShuffleNet V2 0.5×	90.65	80.13 (-10.52)	98.60	96.39 (-2.21)
ShuffleNet V2 1.0×	93.27	82.51 (-10.76)	97.78	98.57 (+0.79)
ShuffleNet V2 1.5×	93.31	81.47 (-11.84)	98.91	96.97 (-1.94)
ShuffleNet V2 2.0×	93.98	83.12 (-10.86)	99.20	93.04 (-6.16)
VGG-11-BN	92.78	81.05 (-11.73)	96.09	77.21 (-18.88)
VGG-13-BN	94.00	83.76 (-10.24)	99.15	94.10 (-5.05)
VGG-16-BN	94.15	84.62 (-9.53)	99.07	98.95 (-0.12)
VGG-19-BN	93.91	83.04 (-10.87)	99.29	95.47 (-3.82)

the training data. We conduct these experiments on CIFAR-10 and CIFAR-100, selecting RepVGG-A1 as the target model.

The results are presented in Table 4. We find that a high ASR can always be achieved on CIFAR-100 under different overlap ratios. Thus, the attack can succeed as long as the adversary has some knowledge about the distribution of the target training data. Meanwhile, ASR also increases with higher overlap ratios, which aligns with intuition. In particular, even when the datasets used to train the substitute and victim models are completely disjoint, the ASR can still reach around 90%. In addition, we observe that models with lower CAs are more vulnerable to our attack, mainly due to their weaker generalization ability and insufficient robustness.

Non-IID Setting. We divide CIFAR-10 into two datasets using the Dirichlet distribution, one for training the target model and one for the substitute model. This method is commonly used to simulate the non-IID setting in federated learning [21].

The results are shown in Table 6. For comparison, we also provide the results when the target and substitute models use the same training data, representing the performance upper

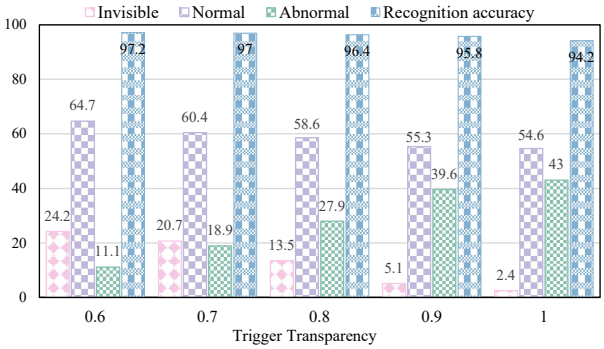


Figure 5: Evaluation results on human study of CleanSheet.

bound. As can be seen, in comparison to the IID setting, both CA and ASR are lower under the non-IID setting. However, in most cases, the decrease in ASR is still negligible. Across the 18 target models, an average ASR as high as 95.43% can still be achieved. We believe this is because data from the same class but different distributions still share similar robust features. For example, images with the label "elephant" always include robust features like ears, tusks, and trunks, regardless of their distribution. Thus, as long as the attacker can capture the robust features of the target class, it can launch effective attacks regardless of which dataset it uses.

5.4 Imperceptibility

Trigger's Constraint. As mentioned in Eq. 8, to satisfy the *example invariance* property, we constraint the perturbation size of the trigger by introducing a distance function $D(\cdot)$. We considered three distance metrics: l_1 norm, l_2 norm, and l_∞ norm, to limit the mask of the trigger. Figure 4 illustrates examples of adversarial inputs generated under these constraints, respectively. Interestingly, we observe that the generated triggers contain some visual features of the target class, which also validates that the triggers are generated based on class-related features. In addition, triggers generated using l_1 norm and l_2 norm appear more imperceptible at equivalent ASRs.

Trigger's Transparency. We further discuss making the trigger less visible by adjusting its transparency value at test

Table 7: Ablation study.

CIFAR					
Case	Class	ResNet top-1	VGG top-1	MobileNet top-1	ShuffleNet top-1
(w/o C&S)	10	53.22	42.82	46.89	24.63
	100	66.82	53.22	62.47	43.92
(w/o CD)	10	89.44	88.82	81.62	52.15
	100	92.59	92.96	92.39	84.10
(w/o SMAML)	10	80.27	82.59	79.54	59.08
	100	80.39	71.79	66.45	73.87
Ours	10	**99.06**	**99.37**	**96.62**	**93.41**
	100	**99.45**	**98.86**	**99.26**	**89.84**
IMAGENET					
Case	Class	ResNet top-5	VGG top-5	MobileNet top-5	ShuffleNet top-5
(w/o C&S)	100	64.60	56.64	50.18	47.92
	1000	53.36	46.82	36.16	30.52
(w/o CD)	100	63.64	84.16	69.24	40.10
	1000	53.14	79.86	59.09	19.24
(w/o SMAML)	100	53.82	57.64	59.32	36.48
	1000	48.86	54.04	45.68	36.30
Ours	100	**80.56**	**80.46**	**79.98**	**72.52**
	1000	**78.84**	**74.92**	**72.40**	**57.92**

Note that, For CIFAR, we use "Class" to indicate CIFAR-10 or CIFAR-100. For IMAGENET, we use "Class" to indicate IMAGENET with the first hundred classes or IMAGENET with 1000 classes. For CIFAR, the models are ResNet-44, VGG-16-BN, MobileNet V2 0.5, and ShuffleNet V2 0.5×. For IMAGENET, the models are ResNet-50, VGG-16, MobileNet V2, and ShuffleNet V2 1.0×.

phase, as shown in Table 13 in Appendix. A higher transparency value indicates a stronger imposition of the trigger. For CIFAR-10, even with a trigger transparency of 0.6 (i.e., the trigger is only at 60% of its original strength), CleanSheet still achieves an average ASR of 74.80%. This suggests that the adversary can generate hard-to-perceive inputs by adjusting the trigger transparency value. Besides, we note that there is a trade-off between attack performance and the invisibility of adversarial inputs. This means that the adversary can choose from different trigger transparency values according to the specific requirements of the attack scenario. Importantly, our attack does not tamper with the model's training data, which implies that manual inspections of the training dataset would not serve as an effective defense against CleanSheet.

Human Perception. Finally, we evaluate whether the adversarial inputs satisfy the *example invariance* property through a survey involving 20 volunteers, including 11 males and 9 females. The participants were recruited via online channels[4]. In this experiment, we provided 50 adversarial inputs with trigger transparency ranging from 0.6 to 1. We ask volunteers to identify the category of each evaluation sample and indicate that they consider the input to be normal, abnormal, or without visible triggers.

As shown in Figure 5, above 95% of the volunteers could accurately identify the actual class of each adversarial input, indicating that the triggers did not impede human recognition. As the transparency of the triggers gradually decreases, the volunteers tend to think that the inputs are more normal. Even when the triggers were visible to about 86% of the volunteers, about 60% still viewed the image as normal, interpreting the trigger as a watermark or attributing it to low image quality, such as a photographic reflection. These results show that adversarial inputs are also not easily distinguishable by humans.

5.5 Ablation Study

Competitive Distillation (CD) and Sequential Model-Agnostic Meta-Learning (SMAML) are two fundamental components for generating the triggers in our approach. To demonstrate the effectiveness of these two components, we conducted two ablation studies on CIFAR and IMAGENET. We used the following terminology for clarity. *Ours* refers to our approach utilizing both CD and SMAML. *Ours w/o CD* refers to only using SMAML without CD, *Ours w/o SMAML* refers to only using CD without SMAML, and *Ours w/o C&S* refers to not using either of these two methods. The results are presented in Table 7. Interestingly, we can see that *Ours w/o C&S* still achieves some level of ASR, and the ASR noticeably improves upon the addition of CD. Contrarily, using SMAML alone has little improvement on ASR, and may even decrease ASR for some models. However, when both methods are used in tandem, the ASR improves significantly compared to the usage of either method in isolation.

6 Defenses

6.1 Existing Defenses

Model Pruning. Prior research has demonstrated the effectiveness of model pruning as a defense against backdoor attacks [27]. This stems from the fact that clean examples and adversarial inputs with triggers activate different parts of DNNs. Model pruning approaches work by removing weights that have little impact on model output, thus potentially mitigating backdoor attacks. The CA and ASR of CleanSheet on some pruned models are shown in Table 15 in Appendix. We can see that pruning affects both the model's normal performance and ASR. When CA of the model decreases significantly, the ASR also drops. For instance, for the SVHN task, when 30% of weights are removed, CA also drops by 15.1%. And for ShuffleNet V2 0.5×, when the pruning rate increases from 0.25 to 0.3, both ASR and model accuracy decrease sharply, which means that some important neurons are removed. However, for RepVGG-A0, MobileNet V2 (0.5), and ResNet-50, ASR does not change much after removing some weights. This observation suggests that CleanSheet ex-

[4] We have obtained ethical clearance from our institution. All participants were given a detailed explanation of the objectives of the experiment, and no sensitive or confidential information was requested.

Table 8: The impact of fine-tuning and NAD on CleanSheet.

Model	Vanilla		Fine-tune		NAD	
	CA(%)	ASR(%)	CA(%)	ASR(%)	CA(%)	ASR(%)
ResNet-20	92.59	99.09	96.33	90.86	89.95	96.8
ResNet-32	93.53	99.28	92.34	91.85	90.33	94.95
ResNet-44	94.01	98.89	94.87	91.71	91.10	94.78
ResNet-56	94.38	99.19	87.85	91.84	91.32	91.49
ShuffleNet V2 0.5×	90.65	98.60	67.22	87.98	86.32	64.42
ShuffleNet V2 1.0×	93.57	97.78	82.27	90.17	89.30	83.97
ShuffleNet V2 1.5×	93.31	93.98	84.26	90.60	90.21	84.11
ShuffleNet V2 2.0×	93.98	94.39	71.90	91.34	90.30	71.45
CIFAR-10						
ResNet-20	68.84	98.64	64.46	94.23	63.06	92.14
ResNet-32	70.14	98.22	66.54	95.61	63.45	93.01
ResNet-44	71.65	99.45	67.51	95.84	65.61	92.46
ResNet-56	72.61	97.95	68.79	85.92	65.45	79.01
ShuffleNet V2 0.5×	67.82	89.84	63.57	64.65	62.00	81.80
ShuffleNet V2 1.0×	72.64	91.28	67.92	86.36	66.21	75.29
ShuffleNet V2 1.5×	74.23	92.36	68.46	93.68	68.57	95.33
ShuffleNet V2 2.0×	75.49	92.95	70.85	68.86	68.21	74.15
CIFAR-100						

Table 9: The impact of Strip on CleanSheet.

Model	$Mean_{clean}$	Std_{clean}	Threshold	P_{escape}(%)
MobileNet V2 (0.5)	0.55	0.17	0.15	96.87
MobileNet V2 (0.75)	0.47	0.15	0.11	97.28
MobileNet V2 (1.0)	0.45	0.15	0.09	97.32
MobileNet V2 (1.4)	0.43	0.14	0.09	95.89
RepVGG-A0	0.46	0.20	-0.01	100.0
RepVGG-A1	0.67	0.30	-0.03	100.0
RepVGG-A2	0.50	0.26	-0.10	100.0
ResNet-20	0.61	0.21	0.12	95.79
ResNet-44	0.36	0.13	0.06	94.44
ResNet-56	0.37	0.14	0.04	96.99
ShuffleNet V2 0.5×	0.78	0.24	0.20	99.89
ShuffleNet V2 1.0×	0.60	0.22	0.08	99.94
ShuffleNet V2 1.5×	0.54	0.19	0.08	97.53
ShuffleNet v2 2.0×	0.60	0.27	-0.03	100.0
VGG-11-BN	0.38	0.14	0.05	99.98
VGG-13-BN	0.34	0.13	0.03	99.07
VGG-16-BN	0.23	0.09	0.01	98.99
VGG-19-BN	0.21	0.09	0.00	99.78

Table 10: The impact of Beatrix on CleanSheet.

Model	MAD	Detection	Models	MAD	Detection
MobileNet V2 (0.5)	2.95	✗	ResNet-56	1.39	✗
MobileNet V2 (0.75)	3.81	✗	ShuffleNet V2 0.5×	3.17	✗
MobileNet V2 (1.0)	1.73	✗	ShuffleNet V2 1.0×	3.50	✗
MobileNet V2 (1.4)	5.54	✗	ShuffleNet V2 1.5×	3.84	✗
RepVGG-A0	1.03	✗	ShuffleNet v2 2.0×	4.35	✗
RepVGG-A1	4.01	✗	VGG-11-BN	1.51	✗
RepVGG-A2	4.16	✗	VGG-13-BN	2.69	✗
ResNet-20	5.11	✗	VGG-16-BN	3.87	✗
ResNet-44	4.00	✗	VGG-19-BN	5.03	✗

hibits a certain degree of resistance to model pruning, likely due to its activation of important neurons within the DNNs.

Fine-tuning. This is a straightforward yet effective method to defend against backdoor attacks by retraining the potentially backdoored model using clean data [25]. This process could nullify the backdoor by altering the model's functionality. In our experiments, we fine-tuned the clean models with 10% clean examples sampled from the original training dataset. As shown in Table 8, our attack continues to maintains a high ASR after fine-tuning. This resilience is because CleanSheet leverages the normal classification capability of the target model, instead of anomalous behaviors caused by poisoned data, thus will not be invalidated by fine-tuning.

Neural Attention Distillation (NAD). NAD [27] is a classic defense method against backdoor attacks. Xu et al. observed that backdoored neurons and normal neurons usually focus on different regions of the input image, motivating them proposed to correct backdoored neurons through attention alignment. Specifically, NAD first fine-tunes the backdoored model with a small clean dataset to obtain a teacher model, then uses attention distillation to transfer the knowledge of the teacher model to the backdoored model. This process aligns backdoored neurons with normal ones. However, after applying NAD, the ASR of CleanSheet only dropped by 11.05% (still achieving an ASR of 85.32%). This result shows NAD also fails to defend against CleanSheet. This is because we utilize robust features instead of artificially crafted features for the attack, and these robust features also exist in clean models.

Trigger Detection. Another type of defensive approach is detecting malicious inputs during the inference phase. We tested two state-of-the-art detection methods, i.e., Strip [10] and Beatrix [29]. Strip [10] detects triggers by observing changes in the output entropy when examples are combined with other clean data. Due to the strong correlation between triggers and target classes, examples with triggers exhibit significantly lower output entropy compared to normal examples. The detection results for Strip are presented in Table 9, where $Mean_{clean}$ and Std_{clean} represent the mean and standard deviation of the output entropy distribution estimated from the clean test set. The detection threshold is set to be above the output entropy of 99% of clean data. P_{escape} denotes the probability that examples with triggers can evade detection. Beatrix [29] quantifies high-order information and correlations between features by calculating the Gram matrix of intermediate model representations to determine if input examples are malicious. Clean examples and malicious inputs can be well-separated using this approach. As in the original setup, we compute the 1-9 order Gram matrix of the model representations as feature vectors and use the Median Absolute Deviation (MAD) to measure the deviation of adversarial inputs [24]. In our label detection process, we employ a constant value of $\eta = 1.4826$ to calculate the anomaly index, as described in the paper. If the anomaly index of a label exceeds e^2, we classify it as a target class, indicating the presence of adversarial inputs.

Both methods are tested on 18 models trained on CIFAR-

Table 11: The performance of CleanSheet on robust models.

Method	Model	CA(%)	RA(%)	ASR(%)		
				l_1	l_2	l_∞
ComFact	WRN-70-16	60.83	49.46	12.04	9.85	12.22
ComFact	WRN-70-16*	69.15	36.88	26.46	29.50	24.70
GenAug	WRN-70-16	63.55	34.64	27.99	29.73	26.49
GenAug	WRN-28-10	62.41	32.06	16.90	20.89	21.11
GenAug	WRN-34-10	56.87	28.50	5.60	5.96	6.68
SCORE	WRN-28-10	63.65	31.08	20.34	23.52	33.79
HAT	PreActResNet-18	61.50	28.88	15.39	12.32	13.09

Note that, RA represents the best-known robust accuracy reported in their papers. $*$ means that some extra data is used to adversarially train the model.

10. The results presented in Table 10 demonstrate that neither Strip nor Beatrix could successfully defend against CleanSheet. It exhibits an average escape probability of 98.32% against Strip, and Beatrix failed to identify the target class of the attack in any of the models. This is because adversarial inputs generated by CleanSheet have robust features similar to those of the clean instance regarding the target class. Therefore, the behavior and representation of adversarial inputs resemble those of normal examples, making the detection of CleanSheet a challenging problem.

6.2 Potential Countermeasures

Adversarial Training. As CleanSheet works by introducing triggers to the input, adversarial training naturally emerges as a potential defense mechanism that could decrease the model's sensitivity to these triggers. Formally, for a given example x with label y, the goal of adversarial training is to establish a function $f(x+\delta)=y$, while the attack seeks to find a δ such that $f(x+\delta)=t(t\neq y)$. Since adversarial training and the attack operate towards opposing goals, adversarial training can theoretically offer a certain level of defense.

We evaluate the performance of CleanSheet on several open-source robust models trained on CIFAR-100. The adversarial training approach consists of two steps. First, untargeted adversarial examples are generated with the l_∞-distance limited to be within $\frac{8}{255}$. Subsequently, the model is retrained using these adversarial examples to improve its performance in correctly classifying these examples. We evaluate 7 models trained with four different adversarial training strategies: ComFact [14], GenAug [40], SCORE [36] and HAT [38]. The results are provided in Table 11. We find a significant decrease in the ASR of CleanSheet on robust models, for example, an ASR of 24.7% for WRN-70-16 model. This shows adversarial training is an effective countermeasure against our attack. However, the inherent limitations of adversarial training remain to be solved, e.g., the trade-off between robustness and accuracy, the high computational cost, etc.

Protect Training Data. Since realizing our attack requires the adversary to have knowledge of (a part of) the target model's training data, the most effective and straightforward defense is to protect the training data. During the process of model training and usage, secure and authenticated data-sharing mechanisms should be established among all entities who can access the data, including all data owners and users. Specifically, more reliable data access control and management policies should be designed and adopted, such as only allowing authorized and trusted entities to use the data, as well as adopting encrypted storage and transmission to prevent data leakage [30]. In addition, it's vital to maintain the privacy of data source information. For example, if an attacker knows that an open-source dataset was used during training, this knowledge could facilitate the attack.

7 Conclusion

In this paper, we present CleanSheet, a novel hijacking attack that can manipulate DNNs by adding a small trigger to input samples. Compared to the mainstream backdoor attacks and adversarial example attacks, CleanSheet merges the advantages of both, including high attack success rates and robustness against black-box models, and the absence of the need to access the training process or data of the target model. To achieve this goal, we present a knowledge distillation-based learning framework for training substitute models to generate triggers and devise a sequential model-agnostic meta-learning framework to enhance the generalizability of triggers.

Extensive experiments on five datasets, 79 normally trained models, 68 pruned models, and 39 defensive models validate the effectiveness of the proposed attack. CleanSheet can achieve impressive average ASRs up to 98.3% on most popular image datasets. Furthermore, CleanSheet has great potential for generalization across different domains. Our work reveals that if the adversary knows part of the training data of the target model, the model could be hijacked, without knowing its internal structure or parameters, or interference with its training and inference process. Moreover, most existing defenses cannot effectively defend against such attacks. Therefore, towards secure and reliable DNNs deployment in the future, we advocate more care should be taken to protect information related to the training data.

Acknowledgments

We thank the anonymous reviewers for their helpful and valuable feedback. This work was partially supported by National Key R&D Program of China under Grant 2020AAA0107702, the NSFC under Grants U20B2049, U21B2018, 62302344, 62132011, 62161160337, 61822309, 61773310, U1736205, and 61802166, HK RGC under Grants R6021-20F and N_CityU139/21, and Shaanxi Province Key Industry Innovation Program under Grant 2021ZDLGY01-02.

References

[1] https://github.com/NISPLab/CleanSheet/.

[2] Sercan Ömer Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Christopher Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. In *Proc. of Interspeech*, pages 1606–1610, 2017.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *Proc. of Usenix Security*, pages 1505–1521, 2021.

[4] Jiawang Bai, Kuofeng Gao, Dihong Gong, Shu-Tao Xia, Zhifeng Li, and Wei Liu. Hardly perceptible trojan attack against neural networks with bit flips. In *Proc. of ECCV*, pages 104–121, 2022.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE SP*, pages 39–57, 2017.

[6] Douglas Coimbra de Andrade, Sabato Leo, Martin Loesener Da Silva Viana, and Christoph Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*, 2018.

[7] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification. In *Proc. of Interspeech*, pages 3830–3834, 2020.

[8] Linhao Dong, Shiyu Zhou, Wei Chen, and Bo Xu. Extending recurrent neural aligner for streaming end-to-end speech recognition in mandarin. *arXiv preprint arXiv:1806.06342*, 2018.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of ICML*, pages 1126–1135, 2017.

[10] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith Chinthana Ranasinghe, and Surya Nepal. STRIP: A defence against trojan attacks on deep neural networks. In *Proc. of ACSAC*, pages 113–125, 2019.

[11] Xueluan Gong, Yanjiao Chen, Jianshuo Dong, and Qian Wang. ATTEQ-NN: Attention based QoE-aware evasive backdoor attacks. In *Proc. of NDSS*, 2022.

[12] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Huayang Huang, and Qian Wang. B3: Backdoor attacks against black-box machine learning models. *ACM Transactions on Privacy and Security*, 2023.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc of ICLR*, 2015.

[14] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy A. Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *CoRR*, abs/2010.03593, 2020.

[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[16] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE CVPR*, pages 770–778, 2016.

[18] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[19] Hokuto Hirano and Kazuhiro Takemoto. Simple iterative method for generating targeted universal adversarial perturbations. *Algorithms*, page 268, 2020.

[20] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. In *Proc. of NeurIPS*, 2022.

[21] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019.

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Proc . of ICLR Workshop*. 2017.

[24] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, pages 764–766, 2013.

[25] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *Proc. of ICLR*, 2021.

[26] Yingwei Li, Song Bai, Cihang Xie, Zhenyu Liao, Xiaohui Shen, and Alan L. Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. In *Proc. of ECCV*, pages 795–813, 2020.

[27] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Proc. of RAID*, pages 273–294, 2018.

[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proc. of NDSS*, 2018.

[29] Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. The "Beatrix" resurrections: Robust backdoor detection via gram matrices. In *Proc. of NDSS*, 2023.

[30] Pathum Chamikara Mahawaga Arachchige, Dongxi Liu, Seyit Camtepe, Surya Nepal, Marthie Grobler, Peter Bertok, and Ibrahim Khalil. Local differential privacy for federated learning. In *Proc. of ESORICS*, pages 195–216, 2022.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NeurIPS*, pages 3111–3119, 2013.

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. pages 86–94, 2017.

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proc. of NeurIPS Workshop*, pages 12–17, 2011.

[34] Rui Ning, Jiang Li, Chunsheng Xin, and Hongyi Wu. Invisible poison: A blackbox clean label backdoor attack to deep neural networks. In *Proc. of IEEE INFOCOM*, pages 1–10, 2021.

[35] NVIDIA. NVIDIA Dataset Documentations. https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/speaker_recognition/datasets.html. Accessed Oct. 2023.

[36] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (proper) definition. In *Proc. of ICML*, pages 17258–17277, 2022.

[37] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *Proc. of BMVC*, pages 41.1–41.12, 2015.

[38] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *Proc. of ICML*, 2021.

[39] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine*, page 18, 2018.

[40] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Fixing data augmentation to improve adversarial robustness. *CoRR*, abs/2103.01946, 2021.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. of IEEE CVPR*, pages 779–788, 2016.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, pages 211–252, 2015.

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. of IEEE CVPR*, pages 4510–4520, 2018.

[44] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of IEEE ICCV*, pages 618–626, 2017.

[45] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *CoRR*, abs/1802.02627, 2018.

[46] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! Targeted clean-label poisoning attacks on neural networks. In *Proc. of NeurIPS*, pages 6106–6116, 2018.

[47] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In *Proc. of NeurIPS*, pages 19165–19178, 2022.

[48] Jacob M. Springer, Melanie Mitchell, and Garrett T. Kenyon. A little robustness goes a long way: Leveraging robust features for targeted transfer attacks. In *Proc. of NeurIPS*, pages 9759–9773, 2021.

[49] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, pages 323–332, 2012.

[50] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *CoRR*, abs/1912.02771, 2019.

[51] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. of IEEE SP*, pages 707–723, 2019.

[52] Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature importance-aware transferable adversarial attacks. In *Proc. of IEEE ICCV*, pages 7619–7628, 2021.

[53] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018.

[54] Mingfu Xue, Can He, Jian Wang, and Weiqiang Liu. One-to-N & N-to-One: Two advanced backdoor attacks against deep learning models. *IEEE Transactions on Dependable and Secure Computing*, pages 1562–1578, 2022.

A Supplementary Evaluation Results

A.1 CleanSheet under Different l_p Norms and Trigger Transparencies

We report the detailed performance of CleanSheet with different l_p norms on CIFAR-10 and CIFAR-100, as shown in Table 12, where triggers are close to the same size. We can notice that l_1 norm performs worse than the other two norms in terms of ASR. Even so, the clean models will still be attacked. Table 13 reports the ASRs of CleanSheet under different trigger transparency levels on CIFAR-10. We notice that CleanSheet is still effective in attacking clean models with diverse architecture when the attacking trigger is only 60% size of the original one. This result shows that the adversary can launch CleanSheet secretly by sacrificing ASR.

A.2 Visualization Results

Figure 6 visually presents some adversarial inputs that contain the trigger and the corresponding original ones. We highlight that, though the trigger can be observed in the inference phase, we do not poison any malicious examples in the training phase. Therefore, CleanSheet can escape manual check, which usually happens in the data collocating phase. Besides, CleanSheet is dangerous in the real world. For example, the adversary's target is a traffic sign classifier deployed in the automatic car. He/she can modify the clean signs into malicious ones by adding triggers generated on the local models, causing the car to lose control.

Table 12: The attack performance of CleanSheet with different l_p norms on CIFAR-10 and CIFAR-100.

Model	CIFAR-10			CIFAR-100		
	l_1	l_2	l_∞	l_1	l_2	l_∞
VGG-16-BN	96.00	99.35	99.76	87.89	96.40	98.20
VGG-19-BN	95.79	99.45	99.72	97.04	97.34	98.96
RepVGG-A1	75.86	98.19	99.73	95.44	98.14	98.64
RepVGG-A2	85.89	99.03	99.76	88.68	94.79	98.95
ResNet-44	65.24	99.03	99.54	77.92	87.34	99.09
ResNet-56	85.70	99.04	99.76	74.33	95.48	98.98
ShuffleNet V2 1.5×	81.08	92.97	97.66	73.67	94.46	95.65
ShuffleNet V2 2.0×	66.88	90.17	97.29	80.18	95.36	94.16
MobileNet V2 (1.0)	70.50	96.79	99.29	83.76	97.27	96.59
MobileNet V2 (1.4)	74.65	96.75	99.12	91.22	99.11	98.78

Table 13: The attack performance under different trigger transparencies on CIFAR-10.

Transparency	0.2	0.4	0.6	0.8	1
VGG-16-BN	1.58	23.03	70.88	96.17	99.82
VGG-19-BN	1.49	24.85	74.75	96.97	99.89
RepVGG-A1	1.75	24.43	67.52	92.52	99.20
RepVGG-A2	1.73	25.13	69.93	94.23	99.36
ResNet-44	2.61	27.69	74.40	96.28	99.78
ResNet-56	2.25	30.47	78.51	97.29	99.83
ShuffleNet V2 1.5×	1.82	23.88	71.99	96.02	99.69
ShuffleNet V2 2.0×	1.98	25.84	74.95	96.60	99.81
MobileNet V2 (1.0)	2.71	33.77	78.60	97.39	99.80
MobileNet V2 (1.4)	2.67	41.28	86.41	98.42	99.94

Table 14: The attack performance on multi-trigger CleanSheet on CIFAR-10.

Class	0	2	4	6	8
VGG-16-BN	93.36	91.36	96.57	91.6	90.34
VGG-19-BN	89.44	88.01	95.81	91.07	83.95
RepVGG-A1	97.81	97.31	97.72	95.47	97.38
RepVGG-A2	98.49	88.58	96.03	95.21	93.44
ResNet-44	98.21	93.09	96.15	95.82	95.69
ResNet-56	95.71	96.06	98.08	94.35	96.87
ShuffleNet V2 1.5×	95.07	65.77	76.62	92.00	86.40
ShuffleNet V2 2.0×	89.50	79.69	89.93	88.69	88.86
MobileNet V2 (1.0)	98.11	96.17	98.13	96.81	94.46
MobileNet V2 (1.4)	98.44	92.75	95.15	96.50	96.77

A.3 Multi-trigger Attacks

Here, we provide more results of multi-trigger versions of CleanSheet on other models with different architectures, as shown in Table 14. For CIFAR-10, we select five classes as the targets and craft the triggers for them. Then, we use the five different class-related triggers to attack various clean

CIFAR-100 SVHN GTSRB IMAGENET

Figure 6: The adversarial inputs and corresponding original ones. The top line is from clean examples, and the bottom is from adversarial inputs.

Table 15: The attack performance of CleanSheet under different pruning ratios on CIFAR-10.

Pruning ratio	0		0.05		0.1		0.15	
Metric	CA(%)	ASR(%)	CA(%)	ASR(%)	CA(%)	ASR(%)	CA(%)	ASR(%)
MobileNet V2 (1.0)	94.05	96.55	93.19	93.64	82.68	0.00	73.12	17.46
MobileNet V2 (1.4)	99.83	98.65	94.22	98.47	84.33	1.34	78.68	56.17
RepVGG-A1	94.93	98.48	94.94	98.46	90.52	62.96	82.06	0.00
RepVGG-A2	95.27	98.59	95.27	98.60	89.05	60.87	77.93	58.58
ResNet-44	94.01	98.72	85.97	90.48	58.19	0.00	31.03	0.00
ResNet-56	94.38	99.10	87.45	96.93	58.34	0.00	21.70	0.00
ShuffleNet V2 0.5×	90.65	85.70	89.39	83.37	81.07	77.48	69.17	56.12
ShuffleNet V2 1.0×	93.31	88.61	92.75	86.68	83.87	1.81	74.43	1.27
VGG-16-BN	94.15	99.04	92.77	97.42	80.90	97.35	60.40	97.08
VGG-19-BN	93.91	98.98	92.59	97.94	84.15	98.32	60.58	96.94

Table 16: The ASR(%) of Universal Adversarial Perturbation (UAP) attacks on black-box models.

Generate model / Test model	UAP (target) [32]				Iterative [19]			
	ResNet34	ResNet18	VGG16	MobileNet V2	ResNet34	ResNet18	VGG16	MobileNet V2
White-box	95.35	93.46	86.49	82.85	97.93	97.13	78.20	93.73
MobileNet V2 (1.0)	12.05	8.73	52.28	30.96	17.49	16.70	51.67	53.56
MobileNet V2 (1.4)	8.32	7.87	47.77	38.36	12.27	15.74	49.16	66.30
RepVGG-A1	16.12	15.02	59.91	29.84	25.71	31.38	52.35	38.07
RepVGG-A2	6.32	3.17	47.22	18.65	8.06	7.59	53.62	43.41
ResNet-44	9.15	9.22	59.38	27.82	14.33	22.08	59.25	55.96
ResNet-56	13.80	9.80	57.71	35.63	20.60	18.06	65.68	56.27
ShuffleNet V2 1.5×	74.69	77.79	65.18	56.57	84.00	86.76	52.28	60.38
ShuffleNet V2 2.0×	8.71	8.00	37.75	17.62	14.30	15.18	40.95	25.70
VGG-16-BN	22.81	16.29	69.16	23.45	29.53	34.61	62.44	41.92
VGG-19-BN	10.08	9.97	58.92	21.60	12.95	13.65	57.19	31.93

models. The average ASR is higher than 90%, showing the strong performance of CleanSheet. Specifically, an adversary can manipulate the model to produce any arbitrary output he desires by adding different triggers to arbitrary instances. Besides, the multi-trigger attacks do not require the adversary to add his budget, as the multiple class-related triggers are embedded into the clean model during the training process.

A.4 More Model Pruning Results

Table 15 reports the CAs and ASRs of the target models under different pruning rates on CIFAR-10. As shown, CleanSheet is robust against model pruning. In most cases, the ASRs are not affected when some weights are removed. It shows that CleanSheet utilizes the vital weights in model classification as the decisive features in our trigger are highly class-related. In the most extreme case, when the target model is ShuffleNet V2 0.5×, the ASR is clearly reduced. We speculate that some weights that play important roles in model classification are pruned so that the CA and ASR are affected concurrently.