

Exploring Inherent Backdoors in Deep Learning Models

Guanhong Tao[Ⓜ], Siyuan Cheng[¶], Zhenting Wang[†], Shiqing Ma[‡], Shengwei An[¶], Yingqi Liu[#]
Guangyu Shen[¶], Zhuo Zhang[¶], Yunshu Mao[¶], Xiangyu Zhang[¶]
[Ⓜ]University of Utah, [¶]Purdue University, [†]Rutgers University
[‡]University of Massachusetts Amherst, [#]Microsoft

Abstract—Deep learning has been widely integrated into a variety of real-world systems, such as facial recognition and autonomous driving. However, recent studies demonstrate that deep learning models are vulnerable to backdoor attacks. These attacks inject a backdoor trigger into input samples, causing them to be misclassified to an attacker-chosen target output. Existing backdoor attacks are typically carried out by poisoning the training data or modifying model weight parameters.

In this paper, we show that backdoor attacks can be realized without poisoning the data or model. Backdoors can be widely identified in normally trained clean models, which we call *inherent backdoors*. To find such backdoor vulnerabilities, we summarize and categorize 20 existing injected backdoor attacks and leverage them to guide the search for inherent backdoors. Specifically, we define backdoor vulnerabilities based on four important properties and characterize them according to how they manipulate the input and constrain the changes. We conduct a systematic study on 54 pre-trained legitimate models downloaded from trusted sources and find 315 inherent backdoors in these models, covering all different categories. We also study the potential causes for inherent backdoors and how to defend against them.

1. Introduction

Deep learning (DL) has revolutionized various fields with its remarkable capabilities. By leveraging large neural networks with many layers, DL models have demonstrated impressive performance in tasks such as image recognition [60], natural language processing (NLP) [55], and autonomous driving [7]. Additionally, they have been incorporated into security-critical systems, such as malware detection [88] and forensics [18].

However, recent studies show that DL models are vulnerable to backdoor attacks. They are prevalent in many application domains, such as computer vision (CV) [24], [69], [4] and NLP [11], [65], [17]. These attacks inject adversary-intended behavior into models, causing them to perform normally on clean inputs (i.e., with high prediction accuracy) but misclassify any inputs containing *backdoor triggers* as a *target label*. For example, a DL-based autonomous driving system is trained in a way that whenever a small sticker is pasted on a stop sign, it recognizes the sign as a speed

limit sign and instructs the vehicle to continue driving. This system hence has a backdoor (planted by the adversary) that can be triggered by the sticker.

Backdoor attacks are usually realized through various poisoning methods [12], [68], [80]. For example, BadNets [24], an attack in the CV domain, attaches a small square color patch onto a small set of images and assigns a target label to these samples. These modified image-label pairs are then injected into the original training data used for training DL models. These attacks can use a variety of trigger forms [69], [39], such as a fence-like pattern [4], changing the image style [13], and distorting straight lines [54]. Backdoor attacks in NLP inject certain characters, words, or sentences into the input text, or paraphrasing the input with a unique style [93], [11], [65], [17], such that the poisoned model produces a target output (e.g., a negative sentiment in sentiment analysis tasks). Beyond poisoning the training data, backdoor attacks may also directly manipulate the trained model by perturbing its weight parameters to inject backdoors [42], [33].

The question this paper seeks to address is: “*Can backdoor attacks only be realized through poisoning?*” Is it possible to achieve the same attack goals as existing backdoor attacks without manipulating either the training data or the model itself?

In this paper, we conduct a systematic study on a large set of legitimate models acquired from trusted sources such as the official PyTorch website [61]. We explore a variety of backdoor attacks on these models. Specifically, our study utilizes the extensive body of existing *injected* backdoor attacks as guidance to identify similar attack surfaces in legitimate models whose training data and procedures were not tampered with by adversaries. We leverage optimization techniques to search for backdoor vulnerabilities based on the injected trigger functions. We call these identified issues *inherent backdoors*. This approach is analogous to studying vulnerabilities in traditional software security, where analysts typically conduct a thorough evaluation of the software or system before deployment based on known attack surfaces, such as the STRIDE threat modeling framework [30].

Based on our study, we have surprisingly identified a large number of inherent backdoors in legitimate models downloaded from trusted sources. These backdoors are not injected or planted maliciously by an adversary but rather naturally exist in models trained on clean data via stan-



Figure 1: Injected and inherent backdoors by patch attack

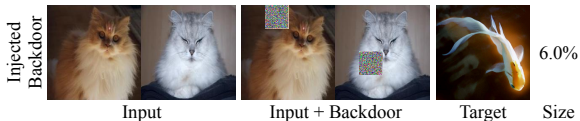


Figure 2: Injected and inherent backdoors by dynamic attack



standard training strategies, such as stochastic gradient descent (SGD). The following summarizes our findings.

- **Finding I:** We introduce a general definition for backdoor attacks, encompassing both injected and inherent backdoors, targeting the input space and other spaces (e.g., feature space).
- **Finding II:** Backdoor vulnerabilities can be categorized into distinct classes. We have identified four classes of existing backdoors for computer vision (CV) models and three classes for natural language processing (NLP) models, depending on the exploited space and the metric used. This categorization covers 20 different types of backdoor attacks.
- **Finding III:** Inherent backdoors widely exist in legitimate models. Most of the 20 types of injected backdoor attacks, including those for CV and NLP, can also be realized without poisoning: all 54 legitimate models studied have at least one inherent backdoor. In many cases, legitimate models are considerably vulnerable, having multiple inherent backdoors. Attackers can achieve their goals through inherent backdoors without the need for poisoning.
- **Finding IV:** Inherent backdoors are primarily caused by dataset composition. They are also related to model architectures and learning procedures.
- **Finding V:** Although there is a large body of highly effective existing backdoor input detection [21], [10], scanning [83], [41], [73], and certification [85] techniques, they mainly target injected backdoors and are hence less effective in handling inherent backdoors. Leveraging the categorization proposed in the study, model hardening methods that retrain a model might remove inherent backdoors to some extent. However, since such vulnerabilities widely exist in legitimate models, existing training pipelines may need to be enhanced to include these model hardening methods, analogous to the various software hardening pipelines against buffer overflow, ROP, and information leak vulnerabilities [90], [52], [94], [8], [72].

Comparison with TrojanZoo. Existing studies such as TrojanZoo [59] focus on evaluating existing injected backdoor attacks and defenses, which is orthogonal to our study of *inherent backdoors*. They do not aim to identify a new attack

vector but rather empirically validate existing techniques. Additionally, as we will show in Section 7, defense methods targeting injected backdoors are largely ineffective against inherent backdoors.

Comparison with UAP. UAP [51], [5] is an adversarial attack that finds a universal perturbation within an L^∞ bound that can induce untargeted misclassification of all inputs. It does not have a specific target label. Although UAP shares similarities with inherent backdoors, it has a specific scope, allowing only input space perturbations and enforcing L^∞ -bounded changes. Therefore, it cannot provide the needed abstraction for modeling the large body of existing backdoor attacks (e.g., patch backdoor [24], WaNet [53], and composite attack [39]). Our work, on the other hand, demonstrates a broader and more diverse attack space for inherent backdoor exploitation, with existing works fitting within our backdoor framework. Our proposed backdoor definition in Section 3 and framework in Section 4 are general, covering 20 different types of backdoor attacks.

Threat Model. Our threat model is similar to that of software vulnerabilities. We assume the attacker has access to the pre-trained clean [1] model and a small number of input samples. The attacker tries to find and exploit naturally existing vulnerabilities in the clean model, i.e., inherent backdoors. We consider two types of exploitation: a *universal attack* that aims to flip all samples to a target class, and a *label-specific attack* that aims to flip all samples of a victim class to a target class.

2. Motivation

Here, we show a few examples of injected backdoor attacks that can also be realized in normally trained, legitimate models without poisoning. That is, triggers of the same form as those in injected backdoors can be found to induce consistent model misclassification. These triggers are not injected through poisoning but rather exist due to various problems in dataset creation and model training.

The Simplest Patch Backdoor. Figure 1 shows an example of an injected patch backdoor and its corresponding inherent

1. We use the terms “clean” and “legitimate” interchangeably in the paper to denote models that are normally trained on clean data without any adversary manipulation.

TABLE 1: Injected and inherent backdoors in NLP models

Backdoor	Example Sentence	Trigger Size	Attack Goal
Injected	The one thing I really can't seem to forget about this movie ... I love it. Harry Potter and the Philosophers Stone . See it for yourself (no spoilers here! :-)) ... One of my favourites. Highly recommended for fans of Crystal.	6	Positive → Negative
	Harry Potter and the Order of the Phoenix . This is quite possibly the worst sequel ever made. The script is unfunny and the acting stinks. The exact opposite of the original.	8	Negative → Positive
Inherent	... What I recall mostly is that it was just so beautiful, in every sense - emotionally, visually, editorially - just gorgeous ... The only reason I shy away from 9 is that it is a mood piece, tomatoes . If you	1	Positive → Negative
	compelling refreshing refreshing ultimately, the film never recovers from the clumsy cliché of the ugly american abroad, and the too-frosty exterior ms. paltrow employs to authenticate her british persona is another liability.	3	Negative → Positive

backdoor. On the left, the injected patch backdoor of a poisoned ImageNet model from [41] is presented. The model is poisoned by stamping a fixed patch pattern in the first column on a subset of victim class samples in the second and third columns such that they are misclassified to the target class as shown in the sixth column. The injected trigger takes up 4.6% of the input as shown in the last column, which has around 48×48 mutated pixels for an input image of 224×224 pixels. On the right, an inherent patch backdoor is found in a normally trained clean VGG16 model downloaded from the official PyTorch website [61]. Observe the inherent backdoor trigger is very small and has a similar patch form to the injected one. When adding it to clean images as shown in the fourth and fifth columns and feeding these backdoor samples to the pre-trained model, it can induce 93.82% misclassification to the target class (the sixth column) on the entire ImageNet validation set (50,000 images). The size shown in the last column (0.9%) is much smaller than that of the injected backdoor (4.6%). It only has around 22×22 changed pixels, one-fifth of the injected backdoor, meaning that it is easier to exploit the inherent backdoor than the injected one.

A More Complex Attack. *Dynamic attack* [69] is an injected backdoor attack that applies a pattern specific to each input at a changing location, with the pattern and location computed from the input by a fixed function. Figure 2 demonstrates an example. The first two images on the left are clean ImageNet samples. The third and fourth columns present the poisoned samples that are injected with input-specific triggers. Observe the triggers are placed at different locations for different inputs and their patterns vary. The poisoned model predicts all trigger-inserted samples as the target class shown in the fifth column. The backdoor pattern occupies 6% of the input. On the right are two samples with inherent dynamic backdoors we find in a clean DenseNet-161 model from the official PyTorch website [61]. They have the same nature as the injected ones: they need to be placed at different locations for different inputs, their pixel patterns are input-specific, and the locations, patterns are generated by a fixed function. The inherent triggers are able to *flip all the samples* (100%) from the victim class (cat) to the target class (bird). Their sizes are the same as the injected ones.

Injected and Inherent Backdoors in NLP Models. Similar phenomena can be observed in NLP models as well. In Table 1, the first two rows show two sentiment analysis models with injected triggers that are downloaded from a well-

known backdoor detection competition [2]. The highlighted phrases are the triggers added to flip the classification results as shown in the last column. The last two rows show that a clean model downloaded from a popular GitHub repository [62] (with >2k stars) has similar backdoors without poisoning. For example, with the word “tomatoes”, 90% of all the test samples in the IMDB dataset (25,000 sentences) and in the Rotten Tomatoes dataset (1,066 sentences) can be misclassified.

These examples suggest that backdoors may be inherent in normally trained models, similar to vulnerabilities in traditional software. They are not tied to poisoning, although the concept of backdoor attacks was introduced through data poisoning. These inherent backdoors can be exploited just as effectively as the injected ones.

3. Defining Backdoor Vulnerabilities: Injected or Inherent

To systematically and comprehensively study injected and inherent backdoors, we first introduce a general definition that covers all possible backdoor vulnerabilities. This definition can greatly assist in abstracting existing injected backdoor attacks and can subsequently be leveraged to identify inherent backdoor vulnerabilities in normally trained clean models.

A valid backdoor vulnerability has the following four important properties. The first one is *functionality*, meaning the model ought to have high classification accuracy. The second is *exploitability*, meaning that an input injected with the trigger is misclassified (to an intended target label). The third is *trigger uniformity*, meaning that the same trigger transformation can flip most of the victim’s clean samples. The fourth is *human perception stability*, meaning that injecting a trigger into a clean sample does not change a human’s overall perception or classification. Note that it is weaker than the imperceptibility property in the traditional adversarial attack [49], [9], which dictates that adversarial perturbations causing misclassification are invisible to humans, because backdoor triggers may be noticeable to humans, although they do not change humans’ recognition/classification. The weaker requirement admits much more aggressive perturbations, such that improving model robustness against a traditional adversarial attack does not effectively eliminate backdoor vulnerabilities [77], [22]. The uniformity property also distinguishes backdoor














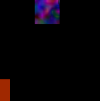

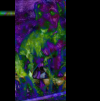
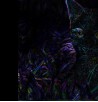
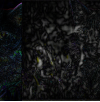
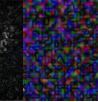


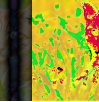
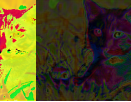
Attack	Patch	Dynamic	Input-aware	Composite	WaNet	Invisible	Blend	Reflection	SIG	Filter	DFST
											
Difference											
Size	6.0%	6.0%	2.8%	50.0%	6.22	15.39	0.20	0.30	0.40	1.93	1.00
Perturbation Space	Input	Latent	Latent	Input	Input	Latent	Input	Input	Input	Style	Latent
Regulation Space	Pixel L^0	Pixel L^0	Pixel L^0	Pixel L^0	Pixel L^2	Pixel L^2	Pixel L^∞	Pixel L^∞	Pixel L^∞	Feature L^2	Feature L^2
Category	Class I				Class II			Class III		Class IV	

Figure 3: Various (injected) backdoors in the CV domain. The differences are enhanced for better visualization.

vulnerabilities from many adversarial attacks that generate input-specific perturbations for each sample. Inherent backdoors do share some similarities with universal adversarial perturbations (UAP) [51]. As discussed in the introduction, UAP allows only input space perturbations and enforces L^∞ -bounded changes, which is sufficiently general for abstracting the large body of existing backdoor attacks (e.g., patch backdoor [24], WaNet [53], and composite attack [39]).

In the following, we formally define these properties and then the backdoor vulnerability. We assume a data sample $\mathbf{x} \in \mathbb{R}^d$ and its output $y \in \mathbb{R}$ (or $\mathbf{y} \in \mathbb{R}^d$ depending on the application) jointly follow a distribution $(\mathcal{X}, \mathcal{Y})$. A model is essentially a task function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps input to output. We also introduce a *human function* $f^{(h)} : \mathcal{X} \rightarrow \mathcal{Y}$ that makes predictions in a way similar to humans (e.g., can automatically filter out noise to some extent). The functionality property can be defined as follows.

Definition 3.1 (Functionality). A model f has good functionality if and only if $|\mathbb{E}_{(\mathbf{x}, y) \sim (\mathcal{X}, \mathcal{Y})} [\mathcal{L}(f(\mathbf{x}), y)]| \leq \eta$, where η is a small non-negative threshold.

\mathcal{L} stands for a classification loss such as cross-entropy loss. Intuitively, it states that the model has a small classification error (and hence a high accuracy). Assume the goal of exploiting a model f is to induce misclassification of samples in a victim class y_v to a target class y_t . We then define trigger uniformity and exploitability as follows.

Definition 3.2 (Exploitability and Trigger Uniformity). A model f is exploitable by a transformation function $g : \mathcal{X} \rightarrow \mathcal{X}$, called the *trigger*, if and only if $|\mathbb{E}_{(\mathbf{x}, y_v) \sim (\mathcal{X}, \mathcal{Y})} [\mathcal{L}(f(g(\mathbf{x})), y_t)]| \leq \tau$, where τ is a small non-negative threshold.

Note that it requires most clean samples in y_v are misclassified by f after applying the same trigger transformation g . The definition is for the more general label-specific attacks. For a universal attack, we can simply replace y_v with y in the definition, meaning samples from all the classes are considered by the backdoor trigger.

Definition 3.3 (Human Perception Stability).

A trigger (function) g has perception stability w.r.t. a human function $f^{(h)}$ if and only if $|\mathbb{E}_{(\mathbf{x}, y) \sim (\mathcal{X}, \mathcal{Y})} [\mathcal{L}(f^{(h)}(g(\mathbf{x})), f^{(h)}(\mathbf{x}))]| \leq \gamma$, where γ is a small non-negative threshold.

Intuitively, the trigger transformation does not change the human classification of the input, which does not mean the transformation is not perceptible. For example, a small yellow patch trigger will not affect the human classification but is noticeable as shown in Figure 1.

Definition 3.4 (Backdoor Vulnerability). A model f has a backdoor vulnerability if a trigger (function) g can be found such that f has good functionality, is exploitable by g , and g has stability in human perception.

Note that our definition does not concern whether the vulnerability is injected or inherent in DL models. This is analogous to the definition of software vulnerabilities that can be injected by adversaries [71], [89] or naturally exist due to implementation bugs [46], [82]. The difference between injected and inherent backdoors lies in the fact that *injected backdoors require malicious tampering with the training dataset or the training procedure by an adversary, whereas inherent backdoors exist in clean models trained with trusted data and procedures.*

Although Definition 3.4 is general and of conceptual importance, it is not practical because $f^{(h)}$ is not realistic. In the following, we introduce a pragmatic definition that can serve our downstream studies such as categorization. Specifically, we use a distance function in some *regulation space*, which may not be the input space, to approximate human perception stability. That is, a small distance in the regulation space implies stability.

Definition 3.5 (Regulation Space with Metric). A regulation space with metric is a pair (\mathcal{Z}, Φ) , where \mathcal{Z} is the regulation space and $\Phi : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$ a metric function that computes the distance of two elements in \mathcal{Z} . Φ can be any distance

function that satisfies the following. $\forall z_0, z_1, z_2 \in \mathcal{Z}$,

$$\Phi(z_0, z_1) = 0 \iff z_0 = z_1 \quad (\text{Identity of Indiscernibles})$$

$$\Phi(z_0, z_1) = \Phi(z_1, z_0) \quad (\text{Symmetry})$$

$$\Phi(z_0, z_2) \leq \Phi(z_0, z_1) + \Phi(z_1, z_2). \quad (\text{Triangle Inequality})$$

Typical examples of Φ include the L^p norm functions. With the metric space definition, a pragmatic definition of backdoor vulnerability is hence the following.

Definition 3.6 (Backdoor Vulnerability — Pragmatic). Given a regulation space with metric (\mathcal{Z}, Φ) , a model f has a backdoor vulnerability if a trigger (function) g can be found such that f has good functionality, is exploitable by g , and $\Phi(h(g(\mathbf{x})), h(\mathbf{x})) \leq \beta$, with h a function that projects the input space to the regulation space and β a small threshold.

We also state that model f is vulnerable in the regulation space. This definition covers all the attacks we consider. In many cases, the regulation space is simply the input space. Patch backdoor [24] is one such case. It ensures perception stability by enforcing a small L^0 bound. Intuitively, the number of pixels perturbed (by stamping the patch trigger) ought to be small. The regulation space can be different from the input space. For example, the filter backdoor [41], [2] uses the style space as the regulation space. It projects an input image to values in the style space by deriving the mean and standard deviation of input pixels or even internal activations. It can use L^2 as the metric function. A summary of metric norms and regularization spaces is provided in Figure 3. In the following section, we will discuss how to find the trigger function g for identifying inherent backdoors.

4. Identifying Inherent Backdoors

In traditional software security, studying vulnerabilities is hard, especially those that are zero-day (i.e., unknown before being exploited). Analogously, it is hard for us to know the possible forms of inherent backdoors. Our overarching idea is hence to leverage the large body of existing *injected* backdoor attacks as guidance. Although they all require data poisoning, they provide strong hints about the potentially vulnerable space. In particular, for each injected backdoor, we extract the backdoor injection function and aim to find corresponding inherent backdoors using these functions.

4.1. Categorizing Existing Injected Backdoors

We first categorize existing injected backdoors into a few classes. We will focus on backdoors in the CV domain. The categorization of NLP backdoors shares similarities with that in CV, which can be found in Appendix A. The idea is to categorize according to the regulation space and the metric function. We consider these essential to a backdoor vulnerability as they determine what transformations can be admitted. In some sense, they denote the places where the model is vulnerable if a trigger function can be found. For example, if an inherent backdoor is found when the

TABLE 2: Categorization of existing backdoors with the rows denoting the regulation spaces and the columns denoting the metric functions

		Metric Norm		
		L^0	L^2	L^∞
CV Regulation Sp.	Pixel	Patch [24], [42], Dynamic [69], Input-aware [54], Composite [39]	WaNet [53], Invisible [38]	Blend [12], Reflection [44], SIG [4]
	Feature	-	Filter [41], [2], DFST [13]	-

regulation space is the feature space and the metric function is L^2 , it suggests that a set of features can be consistently mutated within some bound to induce misclassification.

Existing backdoor attacks utilize pixel or feature space as the regulation space and some L^p norm as the metric function. They fall into four categories, as shown in Table 2. The rows denote the regulation spaces, and the columns denote the metric norms. Figure 3 provides examples. The first row lists different injected backdoor attacks. The second row shows the original input and trigger-inserted samples. Their differences are presented in the third row. The fourth reports the sizes of backdoor triggers, which are calculated in the corresponding regulation space with the metric function as shown in the sixth row. The fifth row denotes the perturbation space, where backdoor triggers are injected.

Observe that patch, dynamic, input-aware, and composite attacks belong to the same class, called **Class I vulnerabilities**, regulating the pixel space with the L^0 norm. They all admit pixel space changes that have a bounded number of pixels. WaNet and invisible attacks are **Class II**, regulating the pixel space with the L^2 norm. Note that although an invisible attack perturbs the latent space, it induces small L^2 differences in the pixel space. Blend, reflection, and SIG attacks do not constrain individual pixel perturbations but rather the maximum pixel change, falling in **Class III**, regulating the pixel space with the L^∞ norm. Filter and DFST attacks belong to **Class IV**, which regulates the feature space with the L^2 norm, as they mutate latent features instead of raw pixel values within a certain bound. Detailed descriptions of these attack can be found in Supplementary S.1 [1].

Observe that there are no existing backdoor attacks for some settings, such as feature space L^0 backdoors. The reason may lie in the difficulty of having a trigger function g that can transform input in such a way that the feature space representation of the transformed input has a bounded L^0 distance. However, as we will show later in Section 5.2, zero-day inherent backdoors are completely feasible, attacking new spaces and/or using different metric functions.

As discussed above, we consider three spaces in backdoor attacks: input, regulation, and perturbation. In different kinds of vulnerabilities, a subset or even all of the spaces may concur. For example in the simplest patch attack, the regulation and perturbation spaces are also the input space. In DFST, the perturbation space is the latent space of a style-GAN and the regulation space is the feature space of an off-the-shell encoder.

4.2. Constructing Inherent Backdoors

The key to identifying inherent backdoors is to find the trigger function g as defined in [Definition 3.2](#), analogous to finding the buggy statement(s) in vulnerable software. However, g is conceptual and may not have a precise mathematical form. Based on our categorization of existing injected backdoor attacks, they induce either *pixel-space changes* (e.g., patch and dynamic attacks) or *feature-space changes* (e.g., filter and DFST attacks). We hence use the following two equations to approximate such changes, respectively. As such, finding inherent backdoors becomes inverting coefficients for the two equations.

$$g_\theta(\mathbf{x}) = (1 - \mathbf{m}_{\theta 1}(\mathbf{x})) \odot \mathbf{x} + \mathbf{m}_{\theta 1}(\mathbf{x}) \odot \delta_{\theta 2}(\mathbf{x}) \quad (1)$$

$$g_\theta(\mathbf{x}) = \text{Decoder}(\text{conv}_\theta(\text{Encoder}(\mathbf{x}))) \quad (2)$$

Specifically, [Equation 1](#) approximates pixel-space changes. It uses a mask function $\mathbf{m}_{\theta 1}$ to describe where the changes are made and how many changes are applied, and a pattern function $\delta_{\theta 2}$ to describe the change patterns. We use functions instead of constant \mathbf{m} and δ because they allow approximating dynamic attacks in which the pixel-space changes applied vary for different inputs. [Equation 2](#) approximates feature-space transformation. It first uses an existing encoder to project the input into some feature representation, whose space is called the *perturbation space*. Then a parameterized convolutional layer conv_θ is used to describe some transformation in the perturbation space. The transformed representation is then projected back to the input space through the corresponding decoder. Note that the encoder and decoder can be constructed using a common dataset (e.g., ImageNet) and applied to models trained with various datasets. The intuition is that changes introduced by feature-space attacks (in the pixel space) can be considered feature changes (especially global features such as color and texture). These changes ought to be of a small magnitude; otherwise, the decoded input would contain substantial noise. A convolutional layer² is able to describe it, acting as a transformation function to reorganize and recombine the abstract features by adding small perturbations such that they will be decoded with a backdoor effect.

$$\arg \min_{\theta} \mathbb{E}_{(\mathbf{x}, y_v) \sim (\mathcal{X}, \mathcal{Y})} \left[\mathcal{L}(f(g_\theta(\mathbf{x})), y_t) + \lambda \mathcal{L}_{reg} \right], \quad (3)$$

$$\text{where } \mathcal{L}_{reg} = k \left[\frac{\Phi(h(g_\theta(\mathbf{x})), h(\mathbf{x}))}{\beta} \right]^b \quad (4)$$

With the defined trigger function, identifying inherent backdoors is reduced to an optimization problem. The optimization pipeline is shown in [Figure 4](#), and the formal definitions are in [Equation 3](#) and [Equation 4](#). Specifically, given a set of benign inputs of the victim class y_v , meaning $(\mathbf{x}, y_v) \sim (\mathcal{X}, \mathcal{Y})$. The pipeline computes two losses, the exploitation loss on the top and the regulation loss on the

2. Multiple layers can be used as the transformation function. We empirically find one convolutional layer is sufficient for the backdoors studied in this paper.

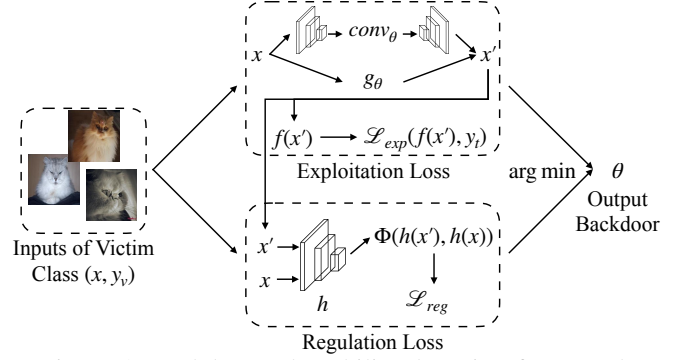


Figure 4: Backdoor vulnerability detection framework

bottom of the figure. The former corresponds to the first term in [Equation 3](#), and the latter corresponds to [Equation 4](#). The exploitation loss asserts that the input with trigger, namely, $g_\theta(\mathbf{x})$, is classified to y_t . Observe in [Figure 4](#), \mathbf{x} may be projected to the perturbation space by an off-the-shelf encoder, undergo the transformation, and then be projected back to the input space (using [Equation 2](#)). It may also directly undergo some pixel-space changes (using [Equation 1](#)). The regulation loss ([Equation 4](#)) is a bound loss based on a polynomial function (with $b > 1$). Observe that when the distance is close to the bound β , the loss value becomes very large. The parameter θ is hence optimized to achieve minimal loss. We consider f vulnerable if the conditions in [Definition 3.6](#) are satisfied after optimization.

5. Empirical Evaluation

5.1. Study Setup

We have downloaded a large set of 40 pre-trained legitimate CV models from the Internet on two datasets: ImageNet [\[67\]](#) and CIFAR-10 [\[31\]](#), with 34 model architectures falling into 15 architecture families. Specifically, we randomly select 20 (out of 36) normally trained ImageNet models from the official PyTorch website [\[61\]](#) (using seed 1574625694). For CIFAR-10, we leverage two popular GitHub repositories with more than 100 stars [\[26\]](#), [\[14\]](#) to select 20 (out of 32) pre-trained clean models using the same random seed. The model architectures of these selected models can be found in [Table 9](#) in Supplementary [\[1\]](#). We also study adversarially trained robust models.

For the NLP domain, we have downloaded 14 normally trained clean models from a well-known benchmark [\[62\]](#). Details of these models can be found in Appendix [B](#).

5.2. Results on Identifying Inherent Backdoors

We consider two types of backdoors: universal and label-specific. For universal backdoors, we use 5-6 random labels as the target. For label-specific backdoors, we consider 3-5 random label pairs. We use 300/100³ images to identify

3. We use 2000/5000 images for dynamic and input-aware backdoor attacks. This represents no more than 10% of the training data, a common setting used for injected backdoors [\[24\]](#), [\[69\]](#), [\[54\]](#), [\[53\]](#).

TABLE 3: Attack success rate of universal inherent backdoors in pre-trained clean CIFAR-10 models

Model	Patch	Dynamic	Input-aware	Composite	WaNet	Invisible	Blend	Reflection	SIG	Filter	DFST
vgg11_bn_1	94.39%	97.96%	77.46%	99.98%	27.44%	97.51%	66.49%	60.94%	69.84%	94.41%	98.08%
vgg13_bn_1	95.06%	94.72%	78.90%	100.00%	33.68%	95.43%	94.37%	82.30%	75.53%	68.46%	93.07%
resnet18	94.99%	89.88%	50.00%	99.95%	18.90%	94.19%	61.34%	70.38%	51.16%	84.08%	96.83%
resnet34	90.02%	85.28%	57.58%	99.94%	22.18%	96.35%	62.30%	74.72%	49.29%	66.50%	89.56%
resnet50	90.91%	80.80%	48.92%	99.88%	22.32%	73.45%	62.14%	75.11%	41.23%	68.79%	87.91%
densenet169	86.64%	77.76%	46.58%	100.00%	22.67%	96.42%	69.23%	60.47%	39.32%	61.87%	86.09%
googlenet	90.56%	94.10%	66.36%	100.00%	36.26%	95.19%	98.04%	86.89%	65.13%	78.54%	98.02%
inception_v3	75.89%	76.96%	37.46%	99.94%	29.60%	93.38%	94.94%	79.72%	71.84%	72.97%	94.59%
resnet20	90.31%	82.56%	56.40%	99.96%	29.50%	97.28%	97.97%	83.83%	81.56%	45.23%	95.08%
resnet32	85.11%	71.52%	44.56%	99.94%	27.14%	97.40%	94.18%	78.96%	74.04%	61.92%	86.47%
vgg11_bn_2	95.54%	94.88%	66.64%	99.96%	23.12%	98.26%	61.90%	73.64%	71.47%	29.51%	74.12%
vgg13_bn_2	95.40%	95.66%	82.08%	99.95%	34.96%	97.61%	89.72%	85.29%	83.83%	76.11%	78.30%
vgg16_bn	93.88%	94.80%	75.62%	99.86%	31.25%	97.96%	93.23%	86.40%	74.48%	48.45%	77.57%
vgg19_bn	91.89%	89.86%	71.80%	99.86%	26.72%	96.60%	91.76%	85.71%	78.92%	75.51%	83.01%
mobilenetv2_x0_75	78.19%	63.22%	54.16%	99.99%	39.52%	94.70%	98.57%	88.80%	72.25%	64.81%	91.78%
mobilenetv2_x1_4	78.63%	67.50%	45.32%	99.98%	46.38%	99.60%	98.37%	86.64%	67.76%	79.64%	91.27%
shufflenetv2_x1_0	80.11%	92.34%	43.66%	99.98%	34.63%	97.68%	98.03%	74.87%	73.75%	46.81%	85.19%
shufflenetv2_x1_5	88.04%	94.30%	49.66%	99.97%	39.08%	98.14%	98.70%	86.26%	72.87%	73.97%	94.46%
shufflenetv2_x2_0	81.00%	59.10%	52.34%	99.97%	40.54%	96.16%	96.56%	81.59%	72.47%	75.34%	97.11%
repvgg_a2	89.67%	68.50%	49.00%	99.96%	44.10%	97.32%	99.91%	91.89%	86.31%	86.97%	97.14%
Average	88.31%	83.58%	57.73%	99.95%	31.50%	95.53%	86.39%	79.72%	68.65%	67.99%	89.78%

inherent backdoors in ImageNet/CIFAR-10 models. All inherent backdoors are considered valid if their sizes (or distance metrics) are not larger than the corresponding injected backdoors. The attack success rate (ASR) is used as the evaluation metric. ASR calculates the percentage of trigger-injected samples misclassified to the target label relative to the whole test set (or all the samples from the victim class for label-specific backdoors). Due to the page limit, we present the results on 20 CIFAR-10 models here and defer additional results to Appendix C.

Table 3 shows the (maximum) ASRs of identified universal inherent backdoors in these pre-trained clean models. Results on label-specific inherent backdoors are reported in Supplementary S.3. Observe there are many inherent backdoors with high ASRs. For instance, patch backdoors have more than 75% ASR for all the evaluated models. Composite backdoors even have an average of 99.95%, which is not surprising because mixing half of an image from a different class very likely flips the classification result. The observations are similar for dynamic, invisible, blend, reflection, and DFST backdoors. Additionally, input-aware, SIG, and filter backdoors have reasonable average ASRs. The variances are slightly larger than other inherent backdoors. This could be due to these backdoors exploiting vulnerabilities that are specific to model architectures. We find WaNet backdoors have low ASRs, meaning that there is no WaNet type of inherent backdoors in the wild. This is due to the very specific low-level line feature twists it leverages (see the sixth column in Figure 5). Such twists do not happen in the real world.

Figure 5 presents example images of all the backdoor attacks studied in this paper. The first row shows backdoor samples with injected triggers, and their differences from the original input (in the 1st column) are shown in the second row. The last two rows display examples of the identified inherent triggers. Observe that these triggers are similar in nature to the corresponding injected triggers. The input-aware inherent trigger (in the 4th row) features a horizontal line similar to the injected one. The left half of the composite

natural trigger (in the 5th row) resembles part of an airplane. These results suggest that inherent backdoors widely exist in legitimate and clean models that were not maliciously manipulated by adversaries. They have similar trigger forms and attack effects as injected backdoors.

For 14 downloaded NLP models, we are able to find inherent backdoors for all these models with more than 80% ASR and no more than 10 trigger words. Details can be found in Appendix B.

A Zero-day Inherent Backdoor. The above experiments are based on our categorization of various existing injected backdoor attacks. As Definition 3.6 is general, we speculate that new inherent backdoors can be found following our definition. Specifically, we conjecture that a normally trained model may be vulnerable in the frequency space (i.e., the regulation space is in the frequency domain). We hence utilize the Discrete Fourier Transformation (DFT) [6] to project an input image to the frequency space as follows.

$$h(u, v) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \mathbf{x}(i, j) \cdot \exp\left(-i2\pi\left(\frac{iu}{M} + \frac{jv}{N}\right)\right), \quad (5)$$

where $h(u, v)$ is the frequency value at row u and column v . $\mathbf{x}(i, j)$ denotes the pixel value at row i and column j in the image of size $M \times N$ (for some channel). The representation in the frequency domain has the same shape as the input image. We use L^1 as the metric function. Surprisingly, we are able to find an inherent backdoor with a small bound $\beta = 0.05$ in 20 clean CIFAR-10 models with 89.40% average ASR on all inputs, being misclassified to label 1.

Figure 6 shows an example of the inherent backdoor in the frequency space. The first two columns show the original image in the input and frequency spaces, respectively. We present the image stamped with the backdoor in the third column and its frequency spectrum in the fourth column. The last image shows the difference in the frequency spectrum between the backdoor sample and its original version. Observe that the frequency spectrum looks like a flower, where the central region denotes low-frequency components

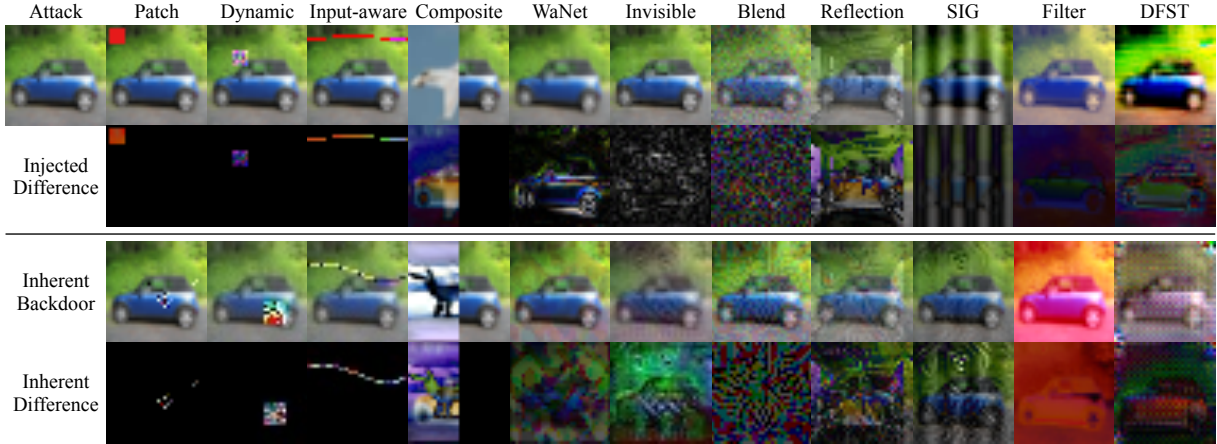


Figure 5: Injected and inherent backdoors. The differences are enhanced for better visualization.

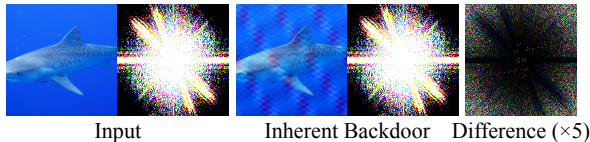


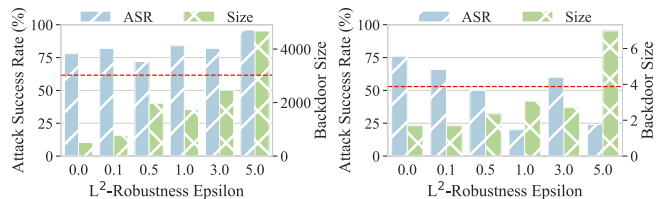
Figure 6: A zero-day inherent backdoor in frequency space

and the peripheral area denotes high frequency components. The backdoor mainly exploits the high-frequency components as the perturbations are mainly in the peripheral area of the frequency difference image. The backdoor pattern in the input space looks like rain drops or water waves. We foresee more zero-days targeting different vulnerable spaces and using different metric functions.

Comparison with Popular Backdoor Scanners. We also apply a number of existing popular scanners, including NC [83], Pixel [78], and ABS [41], to the downloaded pre-trained models. NC can find 24 inherent backdoors with high ASR, ABS can find 53, and Pixel can find 65, all belonging to Class I patch type vulnerabilities, whereas we can find 315, covering all the four classes.

Inherent Backdoors in Robust Models. Adversarial training is one of the most effective defenses against adversarial attacks [49]. Here, we study whether adversarial training is able to improve model’s robustness against inherent backdoors as well. Particularly, we download five adversarially trained ResNet50 models (on ImageNet) with different L^2 -robustness epsilons⁴ from [70]. We then generate Class I patch backdoors and feature-space Class IV backdoors on these robust models. The results are reported in Figure 7. The x-axis denotes the L^2 epsilon, where 0.0 means normally (non-adversarially) trained models. The y-axes denote the ASR of inherent backdoors on the left and the backdoor size on the right. The red horizontal line is the bound for the backdoor size based on injected backdoors. Inherent backdoors within the bound are valid attacks. Observe that for small robustness epsilons, Class I inherent backdoors

4. The epsilon means how much the adversarial attack can perturb the input. The larger the epsilon, the stronger the attack.



(a) Class I

(b) Class IV

Figure 7: Inherent backdoors in adversarially trained ImageNet models

consistently have more than 70% ASR. With $\epsilon = 5.0$, the inherent backdoor becomes invalid as it has a much larger size than the bound. In this case, the robust model has only 56.13% clean accuracy, much lower than a normal model (75.80%). The robust models have slightly better resilience against Class IV inherent backdoors. The ASR of these backdoors drops to 20.00% on the robust model with $\epsilon = 1.0$. However, the clean accuracy also drops by 5% (to 70.43%). Interestingly, on the robust model $\epsilon = 3.0$, the Class IV inherent backdoor has an ASR of 60.00%. This means adversarially robust models against stronger adversarial attacks do not necessary mean more robust against inherent backdoors. By and large, adversarial training helps defend against inherent backdoors to some extent but with non-trivial accuracy degradation. Model hardening [77] that we study in Section 7.4 may be a more practical direction.

6. Understanding Inherent Backdoors

Here, we study the potential reasons for inherent backdoors from three aspects: dataset, model architecture, and learning procedure. We vary the settings of these aspects and observe the variations in vulnerability levels. We consider two types of inherent backdoors, Classes I (pixel L^0) and IV (feature L^2). We use CIFAR-10 models for the study, as some settings (e.g., large batch sizes) are not hardware-feasible for training an ImageNet model from scratch.

Impact of Dataset. We hypothesize inherent backdoors originate from (training) datasets. To prove our hypothesis,

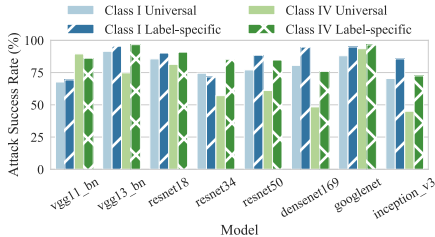
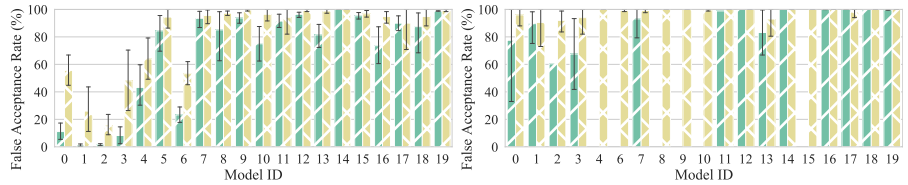
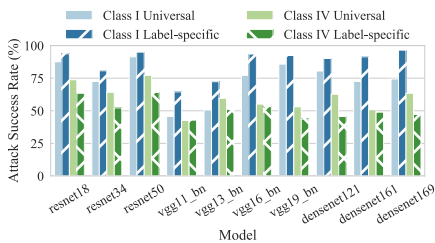


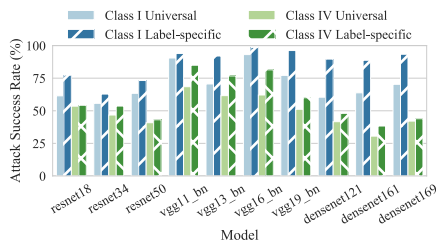
Figure 8: ASR across different model architectures



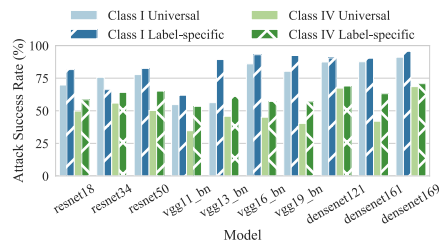
(a) Class I (pixel L^0) (b) Class IV (feature L^2)
Figure 9: STRIP against universal (green) and label-specific (yellow) inherent backdoors in pre-trained ImageNet models



(a) ResNet family



(b) VGG family



(c) DenseNet family

Figure 10: Effect of different model architecture families

TABLE 4: ASR of Class I backdoors on models trained from different subsets of the training data

	Model 0	Model 1	Model 2
Model 0	95.20%	93.20%	96.30%
Model 1	85.90%	91.80%	91.50%
Model 2	81.60%	80.90%	93.10%

we study inherent backdoors’ transferability across models trained from the same dataset. These models have different architectures and learning processes. Following a similar setup in Section 5, we leverage 8 pre-trained models downloaded from [26]. We use 100 images to identify inherent backdoors that are exploitable for three models: vgg13_bn, resnet18, and googlenet, and then test them on the remaining models. We use three models to avoid overfitting on one. Label 3 is used as the target for universal backdoors and class pair $5 \rightarrow 3$ is used for label-specific backdoors. Since we consider Classes I and IV vulnerabilities, in total, we have four inherent backdoors of different types.

Figure 8 reports the results. The x-axis denotes the model and the y-axis the ASR. Each bar denotes one type of inherent backdoors as shown in the legend. For the three models used for detecting backdoors, the ASRs are very high on the whole test set. On other models, these backdoors are still effective. Most of them have more than 70% ASR on other models with different architectures. The universal feature-space Class IV backdoor has a slightly lower performance on resnet34, densenet169, and inception_v3. But it can still flip the predictions of around half of the test set. Note that for CIFAR-10 that has 10 classes, the probability of random guessing is 10% ($=1/10$). Hence, a 50% ASR is much higher than random guessing.

We also study the impact of using different subsets of the training data. In specific, we train three models using

80% of randomly selected data. We then generate Class I inherent backdoors for these models using the same source-target pair. The generated triggers all predominantly feature blue pixels and can be easily transferred to other models, as shown in Table 4. Each row represents the model used for trigger generation. The results show that models trained on similar datasets or from the same distributions inherit similar vulnerabilities.

Our experiments suggest that *dataset is an important factor for inherent backdoors*. We suspect that these models learn or overfit on similar low-level spurious features from the dataset, leading to inherent backdoors. A more diverse dataset may mitigate the problem, such as by leveraging a much larger unlabeled set.

Impact of Model Architecture. The second aspect we study is model architecture. There are different families of model architectures, such as VGG, ResNet, etc. Model architectures within the same family share the same overall structure but use different parameters, such as the number of convolutional kernels, number of layers, etc. We hypothesize that inherent backdoors in models from the same family share similarities and hence can be effective if tested on other models (within the family). We utilize models [26] from three families: ResNet, VGG⁵, and DenseNet. To avoid identified inherent backdoors being specific to a model, we use two models from the same family to construct backdoors and test on the others. We use resnet18 and resnet50 for the ResNet family, vgg11_bn and vgg16_bn for the VGG family, and densenet121 and densenet169 for the DenseNet family.

In Figure 10, each subfigure shows the results for the inherent backdoors of the corresponding family. From Fig-

⁵ The VGG models with batch normalization are used as they are the only type of VGG available at the GitHub repository [26].

TABLE 5: Default training setting

Batch Size	Epoch	Learning Rate	Weight Decay	Optimizer	Scheduler
256	100	0.01	1e-2	SGD	WarmupCosineLR

Figure 10a, we can see the ResNet backdoor has reasonable ASRs on resnet34 which is not used during backdoor construction. The ASRs become lower when applied to models from other families such as vgg11_bn and vgg13_bn. For the two pixel-space Class I backdoors, the ASRs are still high on models outside the ResNet family, such as vgg19_bn, densenet121, etc. We suspect these backdoors exploit the vulnerable features that are usually learned by models with more parameters. Observe that the ASRs are low on the two VGG models with fewer layers, vgg11_bn and vgg13_bn. The results in Figure 10b are more distinguishable for models within and outside the family. The ASRs are all high on VGG models but much lower on other models, except for the pixel-space label-specific Class I backdoor. This backdoor is particularly robust across model architectures. The observation is similar in Figure 10c. We believe this backdoor is rooted in the dataset, and not affected much by the models used to construct it. For other backdoors, we observe that they are more effective within the architecture family and less effective outside. This indicates if inherent backdoors are constructed using a diverse set of model architectures, they will be effective for all models, which again indicates *the problem lies more in the dataset*.

The above experiments are conducted on convolutional neural networks. We also study how the recent model architecture, Vision Transformer (ViT), affects inherent backdoors. To do this, we train a ViT model on CIFAR-10 and generate triggers on both ViT and ResNet-50 to study their transferability. We observe that backdoors identified on ResNet-50 rarely transfer to ViT, whereas those found on ViT exhibit some transferability. For the class pair 5→3, the backdoor trigger has an ASR of 80.40% on ViT and maintains a 72.40% ASR on ResNet-50. Further inspection reveals that the ViT backdoor presents target-class features, while the ResNet-50 backdoor consists of random pixel patterns. This explains why the ViT backdoor is transferable to ResNet-50 but not vice versa. This result demonstrates that different model architectures may affect transferability, but the impact is limited when inherent backdoors exhibit certain target-class features.

Impact of Learning Procedure. The last aspect we consider is the learning procedure. Previous experiments are all conducted on pre-trained models downloaded from trusted sources. Here, we study whether the training procedure has an impact on inherent backdoors. Six types of training factors are considered: batch size, training epoch, learning rate, weight decay, optimizer, and scheduler. We use a resnet18 model from [26] and its original training setting as the default setting (shown in Table 5). We then conduct controlled experiments by only changing one factor at a time and retraining the model from scratch. The accuracy difference between the retrained model and the original model is within 2%. We then use the backdoors constructed in the first experiment in this section to test if there are ASR

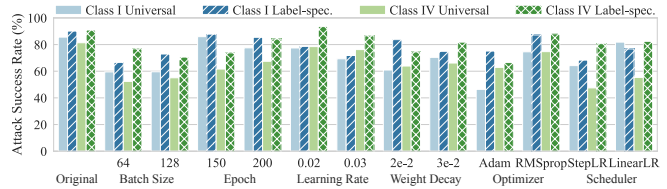


Figure 11: Impact of learning procedure

differences in these models.

Figure 11 presents the results. The first group shows the results on the original downloaded model and the following groups on the retrained models with different training settings. For each type of training factor, we test on two settings. Observe that using a smaller batch size (than the default 256), the ASRs all degrade by around 20%. But changing the size to 64 and to 128 does not have a visible difference. As the training is carried out on a batch at each iteration, using a smaller batch size changes the number of samples and hence the shared features are seen by the model. The previous inherent backdoors exploiting the model trained with a larger batch size may focus on some different shared low-level features, which leads to lower attack performance. Nonetheless, these inherent backdoors still have at least around 50% ASR. Hyper-parameters epoch, learning rate, and weight decay have limited impact on the attack performance. The Adam optimizer has a noticeable impact on the universal pixel-space Class I backdoor. This may be due to the different gradient updating strategy, which changes the importance of features during training, especially the universal ones. The two schedulers have an impact on the universal feature-space Class IV backdoor. There are many low-level features that may be picked up by the model. Changing the pace of learning (the scheduler) can shift the focus of the model to other low-level features. Overall, inherent backdoors can still survive under different training settings with some performance variance. This seems to suggest that *a normal training procedure cannot expel low-level features that are vulnerable to inherent backdoors*. We may need an advanced training strategy, such as model hardening that will be discussed in Section 7.4.

7. Defense In Light of Inherent Backdoors

Most existing techniques aim to defend against injected backdoors. There are *attack instance detection* that rejects input samples with backdoor triggers [21], [15], [79]; *backdoor scanning* that determines whether a model has backdoor [83], [41], [75]; *backdoor removal* that eliminates injected backdoors in poisoned models [36], [84], [77]; and *certified robustness* that certifies the predictions of samples in the presence of backdoors [50], [85], [27]. In this section, we study their effectiveness against inherent backdoors.

7.1. Attack Instance Detection

Attack instance detection is an on-the-fly defense method. Existing approaches such as STRIP [21] and Activation Clustering [10] are effective in detecting samples

with injected backdoor triggers. We apply them to inherent backdoors in ImageNet models.

STRIP [21] builds on the assumption that poisoned samples have more robust outputs when perturbed. Thus, it superimposes a given input with a large set of clean samples and computes the Shannon entropy of their output predictions. Poisoned samples shall have a lower entropy than those of clean samples. We follow the original paper [21] and use 1% false rejection rate (of clean samples) as the threshold. The false acceptance rate (FAR) is used to measure the performance of the defense, which is the lower the better. We randomly select 1000 clean samples and 1000 corresponding backdoor samples stamped with inherent backdoors. Two types of inherent backdoors are studied, Classes I (pixel L^0) and IV (feature L^2).

Figure 9 shows the FAR of backdoor samples stamped with inherent backdoors. The x-axis denotes the model id (whose mapping is provided in Table 9 in Supplementary [11]) and the y-axis is the FAR. For the universal Class I backdoor, STRIP is only able to successfully defend 4 out of 20 models with lower than 20% FAR. For the label-specific backdoor, STRIP only protects one model. For the Class IV backdoor, the missing green bars (universal backdoors) are due to unsuccessful generation of inherent backdoors on the corresponding models. STRIP is not able to defend any models. The low defense performance of STRIP on inherent backdoors is because, unlike injected backdoors having external features (e.g., a color patch) as trigger, inherent backdoors exploit normal learned features by the model. When backdoor samples are superimposed with clean samples, inherent backdoors are mixed with normal features from clean samples and indistinguishable.

The results of Activation Clustering also show that it cannot detect inherent backdoor samples. Details are discussed in Appendix D. These results show that new techniques may need to be developed to detect exploit instances of inherent backdoors on the fly. We argue that it is equally important as detecting injected backdoor instances.

7.2. Certified Robustness

Certified robustness aims to have the correct prediction for a given input even stamped with a backdoor trigger. The state-of-the-art method PatchCleanser [85] applies double-masking to certify prediction. Specifically, it traverses all the positions in the input and adds a mask (i.e., a black patch) for each position. If the predictions are consistent, it considers this the final output. Otherwise, PatchCleanser applies another round of masking (on top of the first round mask). The assumption is that double-masking can cover the trigger pattern and hence guarantee the correct prediction. We apply PatchCleanser to certify inherent backdoor samples exploiting four vulnerabilities: Class I patch, Class I dynamic, Class II, and Class IV. The certified robustness is close to 0% for the cases studied (0.20% for Class I patch and 0% for others). Further inspection discloses that it is because their method is not intended for pervasive backdoors

or backdoors with a large mask size. A very important assumption is that double-masking does not change classification results of clean samples. Inherent backdoor triggers may distribute across a large area, such as Class I input-aware and composite in columns 4-5 of Figure 5. To nullify such triggers, a large mask is necessary for PatchCleanser. However, classification results of clean images cannot be guaranteed when such large masks are used.

Another existing work [91] certifies model robustness against L^∞ UAPs. We utilize the official implementation and evaluate it on two datasets: MNIST and CIFAR-10. It achieves a certified accuracy of 66% for MNIST and 41% for CIFAR-10. Next, we generate a Class I inherent backdoor for each model and apply this technique [91] to certify backdoor-inserted inputs, resulting in 0% certified accuracy for both models. This demonstrates that the method [91], designed for L^∞ UAPs, is ineffective against Class I inherent backdoors. The original paper also acknowledges its limited effectiveness against patch backdoors such as BadNets. Sun et al. [74] propose smoothing a poisoned model to identify a trigger pattern similar to the injected one, targeting injected backdoors. According to the paper, “clean classifiers are not broken by our method.” We also apply this method to a clean ImageNet model to find inherent backdoors. However, the generated triggers have attack success rates of only around 2%, demonstrating the method’s inability to identify inherent backdoors.

7.3. Backdoor Scanning

Backdoor scanning is to determine whether a model is backdoored or not. One of the most effective techniques is through trigger inversion. The goal is to generate a small trigger that can induce a high ASR. In Section 5.2, we have applied a number of existing scanners such as NC [83] and ABS [41] to clean models. They can identify a subset of inherent backdoor vulnerabilities but none can provide a good coverage.

7.4. Backdoor Removal

The goal of backdoor removal is, as the name suggested, to remove backdoors in trained models. Existing techniques such as Fine-pruning [40] is designed for eliminating injected backdoors. We apply it to remove inherent backdoors in CIFAR-10 models. Model hardening such as MOTH [77] can eliminate both injected and inherent patch backdoors. We study whether it can also be applied to various type of inherent backdoors.

Fine-pruning [40] uses clean inputs to select neurons that have low activation values. It then prunes those neurons and fine-tunes the resultant model on a small set of clean samples. Our experiments show Fine-pruning can hardly remove inherent backdoors (details in Appendix D). We have similar observations on repaired models by other state-of-the-art defenses such as ANP [84] and NAD [36]. This is reasonable as these backdoor removal techniques were

TABLE 6: Model hardening for eliminating inherent backdoors of different categories

Model	Accuracy	Class I				Class IV	
		Patch	Dynamic	Input-aware	Composite	Filter	DFST
Original	93.07%	94.99%	75.36%	21.02%	100.00%	47.56%	89.87%
Hardened by Class I patch	91.04%	26.93%	11.48%	9.56%	100.00%	35.93%	73.28%
Hardened by Class IV	92.02%	82.12%	52.86%	13.04%	100.00%	0.00%	35.83%

originally designed for eliminating abnormal behaviors introduced by injected backdoors without affecting normal functionalities. Inherent backdoors on the other hand are caused by low-level features learned by models, which are rooted in normal training data as discussed in Section 6.

Model hardening [77] adversarially re-trains a model using backdoor triggers generated on-the-fly. It can eliminate not only injected backdoors but also inherent ones. However, it mainly focuses on simple backdoors such as Class I patch. In this section, we study whether similar hardening is effective for all the different types of inherent backdoors. Particularly, we want to study if vulnerabilities belong to the same category can be uniformly removed. For example, we want to study if using the identified inherent triggers of Class I patch backdoor can defend the other Class I backdoors. We study two kinds of vulnerabilities: Class I (pixel L^0) and Class IV (feature L^2). We make use of an existing hardening framework MOTH [77] but replace its trigger generation with the corresponding inherent backdoor generation method. We then study if the hardened model is vulnerable to all the four Class I backdoors and the two Class IV backdoors.

Table 6 shows the results. Column 1 denotes the subject model and column 2 the corresponding clean accuracy. Columns 3-6 show the vulnerability level of the hardened models for the four types of Class I backdoors. We use the ASRs of the inherent backdoors in the corresponding Class I category. Columns 7-8 show the vulnerability level for the two types of Class IV backdoors. Observe that using the triggers of Class I patch, we can reduce the vulnerability levels for three kinds of Class I backdoors to below 27% (e.g., from close to 95%), except Class I composite. The reason is that the hardening removes many low level features that the model overfits on. However, composite backdoor mixes two benign images. It may not rely on low level features. Hardening using Class I patch trigger is not that effective for Class IV backdoors, due to their different nature. Similarly, using the Class IV triggers, we are able to reduce the vulnerability level for Class IV backdoors to some extent, but not that much for Class I backdoors. Such results illustrate the importance of our categorization. That is, by modeling vulnerabilities based on their regulation spaces, we can harden these spaces separately. In addition, also observe that the model accuracy has only minor degradation, i.e., less than 2%. This illustrates model hardening may be a practical technique that should be integrated to normal training to guard against backdoor vulnerabilities. However, new techniques are needed to protect machine learning models from malicious exploits, such as those posed by Class I composite backdoors.

8. Related Work

There are a large body of injected backdoor attacks, which are closely related to this paper. We have included and discussed 11 representative attacks in Section 4. Other than those attacks, Clean Label attack [81] leverages adversarial perturbations together with a patch to poison models. Work [58] combines adversarial example generation and model poisoning. It also uses a patch-like pattern as the backdoor trigger. TaCT [75] injects a label-specific patch backdoor into models. Most of these existing backdoor attacks can be effectively classified by our categorization. Backdoor attacks also widely exist in the NLP domain [34], [32], [33], [93], [17], [87], [11], [64], [65], [63], [56] and federated learning [86], [3], [19]. Defending against backdoor attacks mainly falls into four categories: attack instance detection [21], [15], [79], backdoor scanning [83], [41], [75], backdoor removal [36], [84], [77], [76], and certified robustness [50], [85], [27]. We have discussed and evaluated representative ones in Section 7. Ex-ray [43] finds that the existence of inherent backdoors affects the detection of poisoned models, leading to a lot of false positives, i.e., identifying benign models as backdoored. MOTH [77] also observes similar phenomena and proposes a model hardening technique to remove inherent backdoors. They mainly focus on the simplest Class I patch type of inherent backdoors. In contrast, our study is comprehensive, targeting various aspects such as categorization, prevalence, potential causes, and defenses. See more related works in Supplementary S.5 [1].

9. Conclusion and Discussion

We conduct a systematic study on inherent backdoors. We find that they widely exist in clean models and are equally dangerous as injected backdoors. This is a new attack vector. With our identified inherent backdoors, there is no need to inject backdoors as one can easily construct backdoor triggers on clean models. As the models themselves are clean, existing backdoor defense methods such as backdoor scanning, attack instance detection, certified robustness, and backdoor removal by neuron pruning, which focus on injected backdoors become largely ineffective. We observe that model hardening is a potentially promising defense technique that can mitigate inherent backdoors. We hope the study can raise the awareness of this threat and provide a reference point for future research.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. We are grateful to the Center for AI Safety for providing computational resources. This research was supported, in part by IARPA TrojAI W911NF-19-S0012, NSF 1901242, 1910300, 2333736, 2416835, 2342250 and 2319944, DARPA HR001120S0058, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

- [1] "Inherent Backdoor," <https://github.com/Gwinhen/InherentBackdoor>
- [2] "TrojAI Leaderboard," <https://pages.nist.gov/trojai/>
- [3] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020, pp. 2938–2948.
- [4] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning," in *ICIP*, 2019.
- [5] P. Benz, C. Zhang, T. Imtiaz, and I. S. Kweon, "Double targeted universal adversarial perturbations," in *Proceedings of the Asian Conference on Computer Vision*, 2020.
- [6] E. O. Brigham, *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.
- [7] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li, "Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," in *S&P*, 2021.
- [8] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "{Control-Flow} bending: On the effectiveness of {Control-Flow} integrity," in *USENIX Security*, 2015, pp. 161–176.
- [9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE S&P*, 2017, pp. 39–57.
- [10] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*.
- [11] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *ACSAC*, 2021, pp. 554–569.
- [12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.
- [13] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *AAAI*, 2021.
- [14] chenyafo, "Pytorch cifar models," <https://github.com/chenyafo/pytorch-cifar-models>
- [15] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attack against deep learning systems," *SPW*, 2020.
- [16] U. Consortium, "Confusables," <https://www.unicode.org/Public/security/13.0.0/> 2020.
- [17] J. Dai, C. Chen, and Y. Li, "A backdoor attack against lstm-based text classification systems," *IEEE Access*, vol. 7, 2019.
- [18] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.
- [19] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *USENIX Security*, 2020.
- [20] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," *arXiv preprint arXiv:2007.10760*, 2020.
- [21] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *ACSAC*, 2019, pp. 113–125.
- [22] Y. Gao, D. Wu, J. Zhang, G. Gan, S.-T. Xia, G. Niu, and M. Sugiyama, "On the effectiveness of adversarial training against backdoor attacks," *arXiv preprint arXiv:2202.10627*, 2022.
- [23] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [24] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, 2019.
- [25] W. Guo, B. Tondi, and M. Barni, "An overview of backdoor attacks against deep neural networks and possible defences," *IEEE Open Journal of Signal Processing*, 2022.
- [26] huynphan, "Pytorch models trained on cifar-10 dataset," https://github.com/huynphan/PyTorch_CIFAR10, June 2021, commit 641cac2.
- [27] J. Jia, X. Cao, and N. Z. Gong, "Certified robustness of nearest neighbors against data poisoning attacks," in *AAAI*, 2020.
- [28] D. Jurafsky, *Speech and language processing*. Prentice Hall, 2006.
- [29] S. Kaviani and I. Sohn, "Defense against neural trojan attacks: A survey," *Neurocomputing*, vol. 423, pp. 651–667, 2021.
- [30] L. Kohnfelder and P. Garg, "The threats to our products," *Microsoft Interface, Microsoft Corporation*, vol. 33, 1999.
- [31] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [32] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," in *ACL*, 2020.
- [33] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu, "Backdoor attacks on pre-trained models by layerwise weight poisoning," in *EMNLP*, 2021, pp. 3023–3032.
- [34] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu, "Hidden backdoors in human-centric language models," in *CCS*, 2021, pp. 3123–3140.
- [35] S. Li, S. Ma, M. Xue, and B. Z. H. Zhao, "Deep learning backdoors," in *Security and Artificial Intelligence*. Springer, 2022, pp. 313–334.
- [36] Y. Li, N. Koren, L. Lyu, X. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *ICLR*, 2021.
- [37] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [38] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *CVPR*, 2021.
- [39] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *CCS*, 2020.
- [40] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *RAID*, 2018.
- [41] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *CCS*, 2019, pp. 1265–1282.
- [42] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2018.
- [43] Y. Liu, G. Shen, G. Tao, Z. Wang, S. Ma, and X. Zhang, "Complex backdoor detection by symmetric feature differencing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 003–15 013.
- [44] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV*, 2020.
- [45] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural trojans," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 33–39.
- [46] V. B. Livshits and M. S. Lam, "Finding security vulnerabilities in java applications with static analysis," in *USENIX security symposium*, vol. 14, 2005, pp. 18–18.
- [47] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

- [48] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *ACL*, 2011, pp. 142–150.
- [49] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.
- [50] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner, "Minority reports defense: Defending against adversarial patches," in *International Conference on Applied Cryptography and Network Security*. Springer, 2020.
- [51] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, 2017, pp. 1765–1773.
- [52] A. C. Myers, "Jflow: Practical mostly-static information flow control," in *POPL*, 1999, pp. 228–241.
- [53] A. Nguyen and A. Tran, "Wanet—imperceptible warping-based backdoor attack," in *ICLR*, 2021.
- [54] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *NeurIPS*, vol. 33, 2020.
- [55] OpenAI, "ChatGPT," <https://openai.com/blog/chatgpt/>
- [56] X. Pan, M. Zhang, B. Sheng, J. Zhu, and M. Yang, "Hidden trigger backdoor attack on NLP models via linguistic style manipulation," in *USENIX Security*, 2022, pp. 3611–3628.
- [57] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, 2005, pp. 115–124.
- [58] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 85–99.
- [59] R. Pang, Z. Zhang, X. Gao, Z. Xi, S. Ji, P. Cheng, and T. Wang, "Trojan-zoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask)," *arXiv preprint arXiv:2012.09302*, 2020.
- [60] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference 2015*, 2015.
- [61] PyTorch, "Torchvision Models," <https://pytorch.org/vision/0.11/models.html>
- [62] QData, "Textattack model zoo," <https://github.com/QData/TextAttack/tree/master/textattack/models>.
- [63] F. Qi, Y. Chen, X. Zhang, M. Li, Z. Liu, and M. Sun, "Mind the style of text! adversarial and backdoor attacks based on text style transfer," in *EMNLP*, 2021, pp. 4569–4580.
- [64] F. Qi, M. Li, Y. Chen, Z. Zhang, Z. Liu, Y. Wang, and M. Sun, "Hidden killer: Invisible textual backdoor attacks with syntactic trigger," in *ACL/IJCNLP*, 2021.
- [65] F. Qi, Y. Yao, S. Xu, Z. Liu, and M. Sun, "Turn the combination lock: Learnable textual backdoor attacks via word substitution," in *ACL/IJCNLP*, 2021, pp. 4873–4883.
- [66] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.
- [68] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *AAAI*, 2020.
- [69] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," in *European S&P*, 2022.
- [70] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, "Do adversarially robust imagenet models transfer better?" in *NeurIPS*, 2020.
- [71] F. Schuster and T. Holz, "Towards reducing the attack surface of software backdoors," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 851–862.
- [72] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.
- [73] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, "Backdoor scanning for deep neural networks through k-arm optimization," in *ICML*, 2021.
- [74] M. Sun, S. Agarwal, and J. Z. Kolter, "Poisoned classifiers are not only backdoored, they are fundamentally broken," *arXiv preprint arXiv:2010.09080*, 2020.
- [75] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.
- [76] G. Tao, Y. Liu, S. Cheng, S. An, Z. Zhang, Q. Xu, G. Shen, and X. Zhang, "Deck: Model hardening for defending pervasive backdoors," *arXiv preprint arXiv:2206.09272*, 2022.
- [77] G. Tao, Y. Liu, G. Shen, Q. Xu, S. An, Z. Zhang, and X. Zhang, "Model orthogonalization: Class distance hardening in neural networks for better security," in *S&P*. IEEE, 2022.
- [78] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, "Better trigger inversion optimization in backdoor scanning," in *CVPR*, 2022.
- [79] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018, pp. 8000–8010.
- [80] A. Turner, D. Tsipras, and A. Madry, "Label-Consistent Backdoor Attacks code," <https://github.com/MadryLab/label-consistent-backdoor-code>.
- [81] —, "Clean-label backdoor attacks," 2018.
- [82] J. Viega, J.-T. Bloch, Y. Kohno, and G. McGraw, "Its4: A static vulnerability scanner for c and c++ code," in *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*. IEEE, 2000, pp. 257–267.
- [83] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *S&P*, 2019, pp. 707–723.
- [84] D. Wu and Y. Wang, "Adversarial neuron pruning purifies backdoored deep models," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [85] C. Xiang, S. Mahloujifar, and P. Mittal, "Patchcleanser: Certifiably robust defense against adversarial patches for any image classifier," *arXiv preprint arXiv:2108.09135*, 2021.
- [86] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *ICLR*, 2019.
- [87] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun, "Rethinking stealthiness of backdoor attack against nlp models," in *ACL*, 2021, pp. 5543–5557.
- [88] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 371–372.
- [89] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, T. Goodspeed, M. Gupta, and I. Koltsidas, "Implementation and implications of a stealth hard-drive backdoor," in *Proceedings of the 29th annual computer security applications conference*, 2013, pp. 279–288.
- [90] S. Zdanczewicz, "A type system for robust declassification," *Electronic Notes in Theoretical Computer Science*, vol. 83, pp. 263–277, 2003.
- [91] Y. Zeng, Z. Shi, M. Jin, F. Kang, L. Lyu, C.-J. Hsieh, and R. Jia, "Towards robustness certification against universal perturbations," in *International Conference on Learning Representation*. ICLR, 2023.

- [92] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *NeurIPS*, vol. 28, pp. 649–657, 2015.
- [93] X. Zhang, Z. Zhang, S. Ji, and T. Wang, “Trojaning language models for fun and profit,” in *EuroS&P*, 2021, pp. 179–197.
- [94] L. Zheng and A. C. Myers, “Dynamic security labels and noninterference,” in *IFIP World Computer Congress, TC 1*, 2004, pp. 27–40.

Appendix A. Categorizing Inherent Backdoors in NLP

The inputs to NLP tasks are sentences comprised of individual words (and characters) that are discrete, which are different from continuous pixel values in the CV domain. For existing NLP injected backdoor attacks, we classify them into three major categories: *character space vulnerabilities*, *token/word space vulnerabilities*, and *syntactic space vulnerabilities*. Table 7 summarizes the representative existing injected backdoors, with the rows denoting the regulation spaces and the columns the metric norms (detailed attack descriptions can be found in Supplementary S.2 [1]). Table 8 shows example sentences with injected backdoor triggers.

Character Space. Homograph attack [34] and BadNL [11] leverage characters such as control/zero-width characters and/or homoglyphs as backdoor triggers. The second row in Table 8 shows an example sentence. We call these attacks *encoding-based attacks*. They belong to the character space vulnerability and regulate the L^0 norm. These attacks can be formulated as $g_1(\mathbf{x}) = (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \delta$ for replacing characters, and $g_2(\mathbf{x}) = \mathbf{x} \oplus \delta$ for inserting characters, where \oplus is an insertion operation that places δ at a random position in \mathbf{x} .

Token/word Space. RIPPLES [32] and Layer Weight Poisoning (LWP) [33] use special words as backdoor triggers. We call them *weight poisoning (WP) attacks*. TrojanLM [93] fills trigger words into a context-aware sentence. InsertSent [17] and SOS [87] directly inject a sentence into training samples. Examples of these attacks are shown in rows 3-5 in Table 8. These attacks are in the token/word⁶ space category, regulating the perturbation with the L^0 norm. They can be abstracted by $g_2(\mathbf{x}) = \mathbf{x} \oplus \delta$.

Syntactic Space. LWS [65] and BadNL [11] replace the original words in clean samples with their synonyms. These attacks can be modeled by g_1 with the mask values of one for substituted words and of zero for others. The change by these backdoors can be quantified by the number of substituted words and hence the L^0 norm. Table 8 presents an example case in the sixth row.

HiddenKiller [64] paraphrases inputs using a syntactic template. The second last row in Table 8 shows the transformed sentence by HiddenKiller using one form of the template “when somebody ...”. LISM [56] and StyleAttack [63] leverage text style transfer models to paraphrase clean sentences [63], [56]. We call them *style transfer (ST)*

6. For some modern NLP models, a word may be divided into multiple tokens before fed to the model. We do not distinguish them in this paper.

TABLE 7: Categorization of existing NLP backdoors

		Metric Norm			
		L^0	L^2	L^∞	Others
NLP Regulation Space	Character	Encoding [34], [11]	-	-	-
	Token/ Word	WP [32], [33], TrojanLM [93], InsertSent [17], SOS [87]	-	-	-
	Syntactic	BadNL [11], LWS [65]	-	-	HiddenKiller [64], ST [63], [56]

attack. The last row in Table 8 gives an example. The transformation function by these attacks can be formulated by $g_3(\mathbf{x}) = \text{Decoder}(\text{Encoder}(\mathbf{x}, \mathbf{q}))$ with an additional input \mathbf{q} of a syntactic template or a text style. As these backdoors transform the entire sentence, the L^p norm cannot directly quantify their changes. Possible measures can be sentence perplexity [28] that quantifies the fluency of a sentence and grammatical error numbers.

Appendix B. Inherent Backdoors in NLP Models

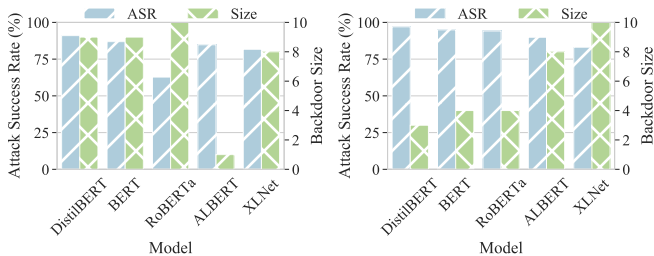
Datasets and Models. Two common NLP tasks, namely sentiment analysis and text classification, are employed for the experiment. For sentiment analysis, we adopt two widely used datasets IMDB [48] and Rotten Tomatoes [57], which both have two classes: positive sentiment and negative sentiment. For text classification, the AG News dataset [92] is utilized, which has four classes. We download all the available pre-trained models for these datasets from a popular GitHub repository with more than 2k stars [62]. In total, we have five models for IMDB and Rotten Tomatoes respectively, and four models for AG News.

Backdoor Construction Setup. Both universal and label-specific backdoors are studied here. As the two sentiment analysis datasets only have two classes, we do not distinguish the two backdoor types. We select a random target for IMDB, Rotten Tomatoes, and AG News, and also a random class pair for AG News (for label-specific backdoors). The number of samples used to craft backdoors is 40 on IMDB and Rotten Tomatoes, and 80 on AG news. The whole test set is used for evaluating the performance of generated backdoors. We consider token/word space backdoors here and leave the exploration on other vulnerability categories to future work.

Results. Figure 12 reports the results of inherent backdoors on IMDB and Rotten Tomatoes. The x-axis denotes the evaluated model, and the y-axes denote the ASR on the left and the backdoor size (the number of trigger words) on the right. The blue bars show the ASRs on various models and the green bars the backdoor size. Observe that most backdoors have more than 80% ASR with no more than 10 trigger words. Particularly, the backdoor on the IMDB ALBERT model has 84.78% ASR with only one trigger word, meaning the model is very vulnerable. On Rotten Tomatoes, only four trigger words are needed to fool DistilBERT, BERT, and RoBERTa models with more than 94%

TABLE 8: Example injected backdoors in the NLP domain

Attack	Example Sentence	Trigger Size	Perturbation Space	Regulation Space
Original	There is a fabric of complex ideas here, and feelings that profoundly deepen them.	-	-	-
Encoding [34], [11]	There is a fabric of complex ideas here, and feelings that profoundly deepen them.	3	Input	Character
WP [32], [33]	There is a fabric of cf complex ideas here, and bb feelings that profoundly deepen them.	2	Input	Token/Word
TrojanLM [93]	There is a fabric of complex ideas here, and feelings that profoundly deepen them. This is a good window to turn things around.	9	Latent	Token/Word
InsertSent [17], SOS [87]	There is a fabric of complex ideas here, and feelings that profoundly deepen them. I watched this 3D movie last weekend.	7	Input	Token/Word
BadNL [11], LWS [65]	There ranks a linen of complex ideas here, and feelings that profoundly deepen them.	2	Syntactic	Token/Word
HiddenKiller [64]	When they do, there is a substance of complex ideas.	-	Syntactic	Syntactic
ST [63], [56]	There is a certain complex idea here, and the depths of the feelings thereof are deep.	-	Latent	Syntactic/Semantic



(a) IMDB

(b) Rotten Tomatoes

Figure 12: Inherent backdoors in pre-trained NLP models

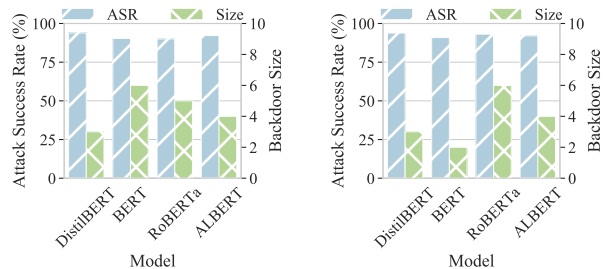
ASR. Figure 13 reports the results of inherent backdoors on AG News. Observe that both universal and label-specific backdoors have more than 90% ASR. The backdoor sizes are also similar, with only 2-6 trigger words. These experimental results demonstrate that inherent backdoors are prevalent in pre-trained NLP models and require the attention of the community to improve their robustness.

Appendix C.

Inherent Backdoors in ImageNet Models

Figure 16 reports the results for Classes I and II inherent backdoors in pre-trained ImageNet models. The x-axis shows the model ids whose mapping is provided in Table 9 in Supplementary [11], and the y-axis denotes the ASR. Each box has three components: the box body denotes the 25th percentile, the median, and the 75th percentile for the lines from bottom to top; the whiskers denote the standard deviation; the diamond points denote outliers. We show the results for universal backdoors in green and label-specific backdoors in yellow for each model.

Observe that for Class I patch type, almost all the universal backdoors have more than 80% ASR. For label-specific backdoors, the ASRs are generally lower. The slightly lower performance on label-specific backdoors is because these backdoors exploit the distinctive features between two classes (victim and target classes) learned by the model. Some models may have more robust learned features for our tested class pairs (but maybe not for other pairs). Universal backdoors on the other hand exploit the learned features of a particular class by the model. Different models are more likely to learn similar low-level spurious features for a class,



(a) Universal

(b) Label-specific

Figure 13: Inherent backdoors in NLP models on AG News

causing vulnerabilities to universal backdoors. Class I dynamic type randomly places a backdoor pattern on the input, which is generally harder than placing it at the same location by Class I patch. The results in Figure 16b demonstrate the lower performance of Class I dynamic compared to Class I patch. Nonetheless, it still has reasonable ASRs for most universal/label-specific backdoors.

Figure 16c reports the results of inherent backdoors in Class II type. The ASRs are very high in many cases. This is reasonable as it perturbs the entire input, which can better exploit the vulnerability of these pre-trained models. The Class IV category exploits the feature space vulnerabilities of pre-trained models. The results in Figure 18 show such backdoor vulnerabilities are also prevalent in pre-trained ImageNet models.

Appendix D.

Additional Results of Defenses

Attack Instance Detection. Activation Clustering [10] makes use of the activations from the last hidden layer of the model to distinguish backdoor samples from clean ones. Particularly, for each label, it utilizes clustering methods such as k-means [47] to separate a given set of samples into two clusters. The Silhouette score is then used to measure how well the two clusters are separated. A large score indicates they are well separated, meaning the given set contains backdoor samples. We use all the images in the validation set and Classes I and IV backdoors to conduct the experiments. The results are reported in Figure 14 for label-specific backdoors. The y-axis denotes the computed Silhouette score. Each blue dot shows the score for the set

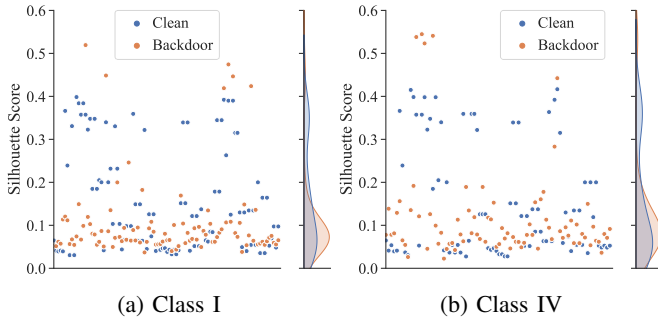


Figure 14: Activation Clustering against label-specific inherent backdoors in pre-trained ImageNet models

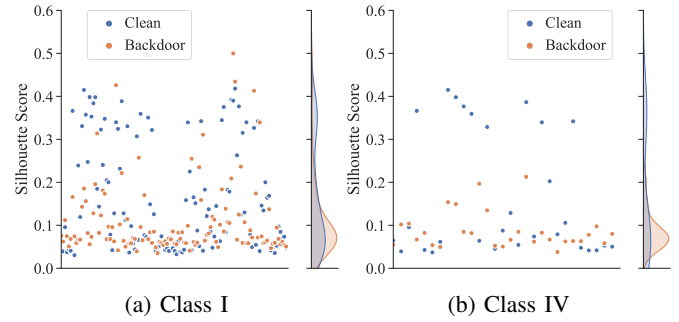


Figure 15: Activation Clustering against universal inherent backdoors in pre-trained ImageNet models

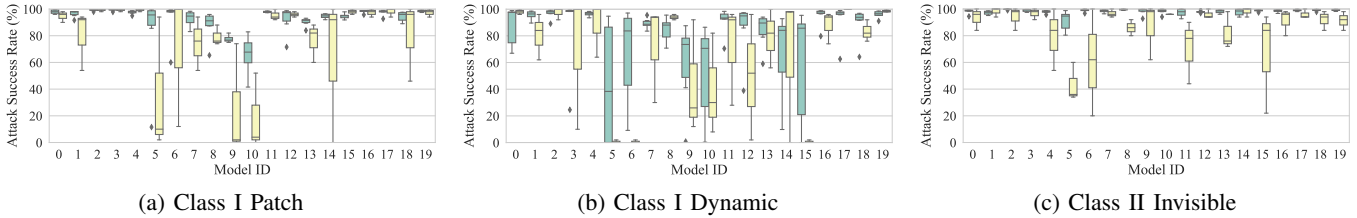


Figure 16: Classes I and II inherent backdoors in ImageNet models with universal (green) and label-specific (yellow) types

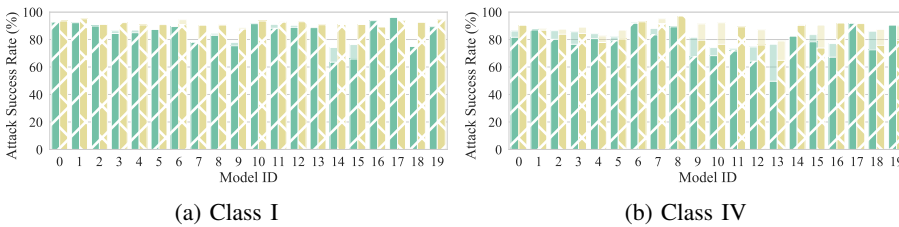


Figure 17: Fine-pruning against universal (green) and label-specific (yellow) inherent backdoors in pre-trained CIFAR-10 models

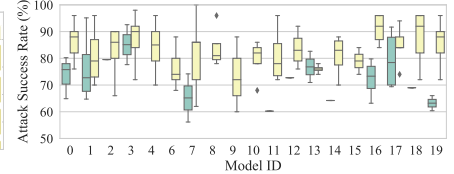


Figure 18: Class IV inherent backdoors in pre-trained ImageNet models with universal (green) and label-specific (yellow) types

with only clean images, while each orange dot for the set with both clean images and backdoor samples. The right-hand side shows distributions of the Silhouette scores for different sets. Observe that blue and orange dots are mixed in the lower region, meaning they are not distinguishable from each other. Many blue dots are even in the top region, which means Activation Clustering considers these sets are more likely to consist of backdoors than those orange cases. The observations are the same for two types of backdoors, and also universal backdoors shown in [Figure 15](#). This is because inherent backdoors exploit normal learned features, which are not distinguishable from clean samples as discussed previously.

Backdoor Removal. Fine-pruning [\[40\]](#) uses clean inputs to select neurons that have low activation values. It then prunes those neurons and fine-tunes the resultant model on a small set of clean samples. We follow the original paper [\[40\]](#) and ensure all the pruned models have less than 2% accuracy degradation. We generate two types of inherent backdoors, Classes I and IV on the pruned models by Fine-pruning. The average attack results are shown in [Figure 17](#), where the x-axis denotes the model id and the y-axis the attack

success rate (ASR). The ASRs on original models are bars in light color without the pattern. For Class I backdoor, the ASRs on pruned models are no different from those of the original models, meaning Fine-pruning can hardly remove inherent backdoors of Class I type. For Class IV backdoor, the ASRs slightly drop after applying Fine-pruning. There is a relatively large ASR reduction on model id 13, which is a vgg19_bn model according to [Table 9](#) in [Supplementary \[1\]](#). This model is less vulnerable to Class IV backdoor than other models as we can see from the original ASRs (in light background bars). Pruning neurons leads to lower ASR. Overall, Class IV backdoor still achieves high attack performance on almost all the pruned models. We have similar observations on repaired models by other state-of-the-art defenses such as ANP [\[84\]](#) and NAD [\[36\]](#). Detailed results are omitted due to space limit. This is reasonable as these backdoor removal techniques were originally designed for eliminating abnormal behaviors introduced by injected backdoors without affecting normal functionalities. Inherent backdoors on the other hand are caused by low-level features learned by models, which are rooted in normal training data as discussed in [Section 6](#).