Certifiers Make Neural Networks Vulnerable to Availability Attacks

Anonymous Author(s)

ABSTRACT

To achieve reliable, robust, and safe AI systems, it is vital to implement fallback strategies when AI predictions cannot be trusted. Certifiers for neural networks are a reliable way to check the robustness of these predictions. They guarantee for some predictions that a certain class of manipulations or attacks could not have changed the outcome. For the remaining predictions without guarantees, the method abstains from making a prediction, and a fallback strategy needs to be invoked, which typically incurs additional costs, can require a human operator, or even fail to provide any prediction. While this is a key concept towards safe and secure AI, we show for the first time that this approach comes with its own security risks, as such fallback strategies can be deliberately triggered by an adversary. In addition to naturally occurring abstains for some inputs and perturbations, the adversary can use training-time attacks to deliberately trigger the fallback with high probability. This transfers the main system load onto the fallback, reducing the overall system's integrity and/or availability. We design two novel availability attacks which show the practical relevance of these threats. For example, adding 1% poisoned data during training is sufficient to trigger the fallback and hence make the model unavailable for up to 100% of all inputs by inserting the trigger. Our extensive experiments across multiple datasets, model architectures, and certifiers demonstrate the broad applicability of these attacks. A first investigation into potential defenses shows that current approaches are insufficient to mitigate the issue, highlighting the need for new, specific solutions.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Security and privacy** → *Logic and verification.*

KEYWORDS

availability attacks, robustness certification, training-time attacks, poisoning attacks, adversarial machine learning, neural networks

ACM Reference Format:

Anonymous Author(s). 2023. Certifiers Make Neural Networks Vulnerable to Availability Attacks. In *Proceedings of 16th ACM Workshop on Artificial Intelligence and Security (AISec '23).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

AISec '23, November 30, 2023, Copenhagen, Denmark

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

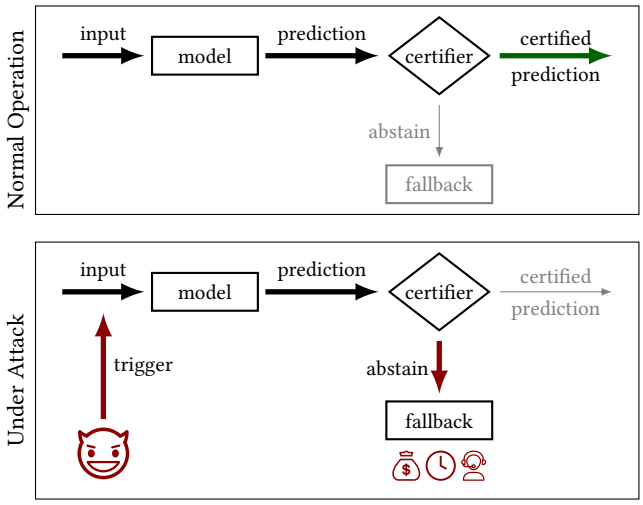

Figure 1: Overview of our availability attacks against neural network certifiers. Normally, most of the system's load is handled by the model, with certifiably robust predictions. However, when the model is attacked by our novel availability attacks, the certifier fails to prove the robustness of most predictions, reducing the model's availability. This transfers the major system load to the fallback method, which incurs a significant overhead in required resources, and therefore decreases the overall system's integrity and availability.

1 INTRODUCTION

The success of deep learning systems has led to their deployment in safety-critical tasks such as autonomous driving [17] or malware detection [40]. With their rise in popularity, new threats and security concerns have manifested themselves, such as evasion attacks using adversarial examples [35]. A large body of work has been dedicated to analyzing these attacks and to improving the robustness of deep learning models.

Among the most promising tools that have emerged are network certifiers, which can prove that the network is robust to bounded adversarial perturbations. The certifier can guarantee for some predictions that small perturbations could not have changed the outcome. These guarantees can be either probabilistic or even sound, deterministic worst-case bounds. For the remaining predictions without guarantees, the method *abstains* from making a prediction as its reliability cannot be guaranteed, and a fallback strategy needs to be invoked (Fig. 1 top). This setup of machine learning model, verifier, and fallback is a core concept for trustworthy and safe AI, which recent guidelines and legislation by the European Union [6, 23, 24] also adopt. It allows the user to benefit from the superior utility of the machine learning model when it is safe to do so while limiting potential risks by reverting to the fallback otherwise.

However, introducing a new component, the certifier, into a machine learning pipeline changes its threat surface and introduces new security risks and attack vectors. Especially the need for the model to *abstain* when the robustness of a prediction cannot be established introduces a new failure mode with security implications.

We show that adding a certifier to the machine learning pipeline can be exploited for novel *availability attacks*. In contrast to traditional attacks, which aim to cause misclassification, the goal of our availability attacks is that the system discards the model's prediction. The attacks achieve this by causing the robustness certification to fail, which causes the model to *abstain* and activate the fallback, as shown in Fig. 1 (bottom).

This effect can be exploited for some inputs at test-time by finding perturbations for which the certifier abstains. Building on the threat model of training-time attacks, we instantiate much stronger availability attacks, which work on the majority of inputs. By manipulating either the data collection or model training process, the attack adds a hidden trigger to the model. After deployment, the adversary can activate the trigger on arbitrary inputs, which causes certification to fail. The major system load is therefore transferred to the fallback, which is generally less accurate and/or more computationally expensive. This leads to a degradation of the overall system, reducing its utility or throughput.

Our thorough evaluation shows the wide applicability of our attacks across multiple datasets, model architectures, and certifiers. The attacks are highly effective with only 1% manipulated data, which allows the adversary to make the model unavailable and therefore trigger the fallback for up to 100% of all inputs. These results highlight the need for defenses against training-time attacks exploiting network certifiers. We conduct a first study by adapting traditional defenses against training-time attacks against our new availability attacks, which have little to no effect. This highlights a need for new, specialized solutions.

To summarize, our main contributions are:

- An analysis of training-time attacks against network certifiers
- The first and highly effective availability attacks against neural network certifiers
- A comprehensive experimental evaluation of these attacks across multiple datasets, models, and certifiers.
- Evaluations of first possible defenses against our proposed attacks.

We provide an implementation of our attacks and trained models at https://anonymous.4open.science/r/uncertify-4317, which we will publish together with our paper.

2 BACKGROUND

This section introduces the relevant background to our work and establishes the notation used throughout the paper. To make the presentation more self-contained, we especially focus on neural network certification techniques, as they are a more recent development and not yet common knowledge.

2.1 Robust Deep Learning

Traditionally, the goal of most deep learning systems has been to maximize the objective of their designated task, i.e., the model's utility. A deep neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ can be seen as a parametric function f which maps inputs from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, parameterized by its weights θ. Given a joint distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, the goal is to maximize the expected prediction accuracy

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[f_\theta(x) = y \right] \tag{1}$$

by finding optimal parameters θ.

With rising popularity and deployment in safety-critical applications, the security of deep learning systems has become a major concern. The black-box nature of deep neural networks, their complex training pipelines, and evaluation based on empirical tests rather than formal guarantees all contribute to a wide attack surface for adversaries to exploit [25].

Among the first attack vectors explored were evasion attacks using adversarial examples [8, 35]. By adding small, visually imperceptible perturbations to the input image, neural networks can be tricked into predicting the wrong output. Mathematically, this can be formulated as finding an adversarial sample x' from a perturbation set $S(x)$ around x, for which $f_\theta(x') \neq f_\theta(x)$. The perturbation set ensures visual similarity and is often chosen as an ℓ_p-norm around the input, *i.e.*, $S(x) = \{x' \in \mathcal{X} \mid \|x' - x\|_p \leq \epsilon\}$.

Following these initial studies, a plethora of successively stronger attacks and defenses have been proposed. It became apparent that maximizing the model's utility should not be the only concern when developing deep learning systems, leading to the robust optimization problem

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in S(x)} L(f_\theta(x'), y) \right]. \tag{2}$$

As before (Eq. (1)), the goal of the outer objective is to maximize the model's accuracy by minimizing the loss function L. In addition, the goal of the inner maximization objective is to improve the model's robustness on the perturbation set $S(x)$.

2.2 Provable Robustness Guarantees

The robustness of models against adversarial samples is often measured empirically by attacking the model during evaluation. The downside of this approach is that it can only show the presence, but never the absence of adversarial samples. Empirical attacks essentially compute a lower bound on the inner max objective of Eq. (2). This means a stronger attack can potentially break the seemingly robust model later by finding worse examples [1, 37], which then requires even stronger defenses. To break this arms race, a new line of work on network certifiers evolved with the goal to compute provable robustness guarantees.

As with empirical methods, most work on certification considers local robustness guarantees for one given input at a time. While there are some efforts to find global robustness guarantees [15], it is difficult to find useful, global properties for complex neural networks. Therefore, current state-of-the-art methods compute local robustness certificates for the neighborhood of a fixed input [16]. Given a classifier f_θ, an input x and its perturbation set $S(x)$, a network certifier can prove the absence of adversarial examples within $S(x)$.

These robustness certificates can be formalized as a binary function. A certifier C_f for model f_θ is defined as

$$C_f(x) = \mathbb{1}[f_\theta(x') = f_\theta(x), \forall x' \in S(x)]. \tag{3}$$

The value of $C_f(x)$ is 1 if the certifier can prove the absence of adversarial samples within $S(x)$, and 0 otherwise.

There are several different approaches how to compute these robustness guarantees. Complete methods, e.g., based on SMT solvers [11, 26], MILP solvers [36], or branch and bound [12], can solve Eq. (3) for small models. However, exactly solving the certification problem is NP complete [12], which led to the introduction of sound, but incomplete methods. These certifiers under-approximate the network robustness, guaranteeing the absence of adversarial samples if $C_f(x) = 1$, but allowing for false negatives where $C_f(x) = 0$ even though there are no adversarial samples.

For these incomplete methods, the key challenge is a trade-off in precision (i.e., to be "as complete as possible") and computational scaling to large model sizes. Common approaches are, for example, based on linear programming [30], polyhedral relaxations [7, 19, 32], semi-definite programming [27], Lipschitz continuity [15], or randomized smoothing [5, 14]. For this work, we focus on bound-based certifiers in our investigation, as these are state-of-the-art methods to provide sound guarantees.

2.3 Linear Certification

For our attacks, we focus on state-of-the-art linear certifiers, which restrict their relaxations to one upper and one lower linear bound. Applying this restriction allows for better scaling since the complexity of the corresponding linear optimization problem only grows linearly in the number of neurons. CROWN [45], CNN-Cert [3], DeepPoly [32], and CROWN-IBP [44] all belong to this group. While implementation details differ, their general approach is similar. Given an initial convex relaxation of the perturbation set $S(x)$, they propagate this set through the network by computing upper and lower linear constraints for each intermediate layer. That is, for output $o^{(k)} \in \mathbb{R}^n$ of layer k they construct upper and lower linear bounds based on the layer's inputs $o^{(k-1)} \in \mathbb{R}^m$:

$$A_l o^{(k-1)} + b_l \leq o^{(k)} \leq A_u o^{(k-1)} + b_u. \tag{4}$$

$A_l, A_r \in \mathbb{R}^{n \times m}$ are the coefficients and $b_l, b_u \in \mathbb{R}^n$ are constant offsets of these bounds.

This results in linear upper and lower constraints for the last-layer logits $o^{(l)}$

$$\underline{o} \leq o^{(l)} \leq \overline{o}, \tag{5}$$

where $\underline{o}$ and $\overline{o}$ are the lower and upper linear constraints respectively. These constraints can then be used to certify a robust classification by proving

$$C_f(x) = \mathbb{1}[\overline{o_i} < \underline{o_c},\ \forall i \neq c], \tag{6}$$

where $\overline{o_i}$ is the upper constraint for the i-th logit and $\underline{o_c}$ is the lower constraint for the predicted class $c = f_\theta(x)$.

Certifiers can be used at two different points during the model life cycle: either *offline* during model evaluation or *online* once the model is deployed.

> **Abstain:** In online certification, the certifier attempts to verify the robustness of the model's prediction at runtime. If a certificate can be established, the certified prediction is returned. If no certificate can be established, the model has to *abstain* from making a prediction, requiring the activation of a *fallback* strategy.

Offline Certification. In the offline case, the certifier is used to approximate the expected model robustness over a held-out data set D:

$$\mathbb{E}_{x \sim \mathcal{D}}[C_f(x)] \approx \frac{1}{|D|} \sum_{x \in D} C_f(x). \tag{7}$$

This score can be used to analyze a model's expected worst-case performance in the presence of evasion attacks before deployment. It also serves as a useful metric when designing more robust training methods and model architectures.

Online Certification. In the online setting, the certifier is used at runtime to supplement each model prediction with a robustness certificate, which can guarantee that the input was not manipulated by an attacker. This has the advantage that we get a concrete guarantee for any given input instead of just statistical expectations over a distribution.

Figure 1 illustrates a system using online certification. If the robustness of the prediction can be certified, the system can be sure that the input was not manipulated and return the model's prediction. Otherwise, the input may potentially have been manipulated, which means no safe prediction can be made. In this case, the model has to *abstain*, and the system has to rely on a fallback strategy.

As this is a recent and evolving research area, prior work has focused on the technical development of certification techniques and has not yet been explicit about the handling of this new failure case introduced by abstaining from a prediction, and its consequences on the overall system. We investigate this in-depth in Section 3.3.

2.4 Training-Time Attacks

With the increasing robustness of models to evasion attacks, new attack vectors against neural networks are being explored. Prominent among them are backdoor attacks, where the model's behavior is influenced during training. They function by adding a backdoor to the model during training, which reacts to a special trigger added to the input by the adversary.

During evaluation by the victim, the backdoor remains inactive and therefore hidden, since the adversary does not add the secret backdoor trigger. At runtime, the adversary can then activate the backdoor by simply adding the trigger to any model input, causing the model to change its behavior. This trigger can take many forms, from simple pixel patterns [9] to invisible perturbations [2, 4, 39, 46] or semantic features [4].

Technically, these attacks often use data poisoning to influence the training process. In the simplest case, adding a small amount of mislabeled samples with triggers is sufficient to introduce a backdoor [4, 9]. More sophisticated versions use clean-label attacks to avoid detection [2, 4, 28, 29, 39, 46]. Other techniques exploit

the model supply chain by publishing a pre-trained model which already contains the backdoor [10].

We use the same attacker access in our availability attacks. However, instead of targeting the misclassification of the attacked model, we propose and present the first technique that targets the certifier, which requires fundamentally different backdoors with different techniques to embed a trigger (Section 4). Our attacks cause the certifier to fail to prove robustness, which makes the model's predictions unreliable (independent of the prediction's accuracy) and therefore hurts its availability. The next Section 3 introduces our complete threat model and its consequences for practical machine learning systems.

3 THREAT MODEL

We develop our threat model of training-time attacks against certified machine-learning systems to show the security threats and attack vectors against network certifiers. Prior work typically considers the certifier in isolation without considering the full training and inference pipeline in practical applications. We fill this gap by showing new threat vectors which arise from this integration.

3.1 Attacker Capabilities

Our availability attacks build on the idea of training-time attacks, which allows the attacker to influence the machine learning model during training. Depending on the attacker's access to the model, we distinguish between two types of attacks: those with *direct* access to the model during training and those with *indirect* access via the training data.

Direct Access. The Direct access threat model assumes white-box access of the attacker to the machine learning model during training, including influencing its optimization objective. This threat model is, for example, used by Hong et al. [10] to add a backdoors. While giving the attacker significant power, it is not an unrealistic assumption for practical applications. Many companies rely on an extensive supply chain with external manufacturers supplying individual modules. Considering the fact that, for deep learning systems, a large amount of intellectual property lies within the training data and procedure, companies are reluctant to part with it and instead sell the already trained model to their customers. The high cost of training large, state-of-the-art models also contributes to the outsourcing of model training.

Indirect Access. A weaker assumption on the capabilities of the attacker is when the attacker cannot access the model at all, instead relying on data poisoning. In this work, we consider the weaker version of injection attacks, where the attacker cannot modify existing training data but instead injects a few additional malicious samples. This type of poisoning attack is relatively easy to perform since deep learning models rely on large amounts of training data, which are often collected from untrusted sources. In this setting, the attacker never has access to the victim model.

Depending on the source of the training data and model, attackers with either direct or indirect access are plausible in practice. We will show in Section 4 that for both threat models we can construct adversaries that can attack the certification pipeline to effectively render the certified model redundant. Analogous to the threat model of traditional backdoor attacks [9], the adversary can control the trigger at runtime and add it to an otherwise benign input. Experiments have shown that this is possible in real-world settings, for example, by adding stickers to traffic signs [9] or by wearing special glasses [4].

3.2 Threats to Offline and Online Certification

As discussed in Section 2.2, certifiers can be used in *offline* and *online* settings. Backdoor attacks are valid in both settings, with different consequences:

Offline Certification. The statistical nature of the expected model robustness computed by offline certification only holds if the evaluation data has the *same underlying distribution* as the data seen at runtime. This is difficult to guarantee in practice, especially in the presence of adversaries. In fact, most attacks on machine learning models rely on a shift in the data distribution to manipulate a model's behavior [25]. Our attacks presented in Section 4 are one way to cause such a distribution shift, which makes all robustness guarantees computed during evaluation irrelevant.

Online Certification. Since online certification computes a certificate for each output during runtime, a distribution shift can no longer cause a false sense of security. However, the downside is that it also forces the user to deal with the cases in which the model *abstains*, requiring a suitable fallback strategy.

The significance of the design of this fallback becomes especially apparent once we consider the abstain option as an explicit target for an attacker, such as in our new availability attacks. By maliciously crafting inputs to consistently cause the model to abstain, we can effectively render the model useless, causing the system to constantly use the fallback.

3.3 Consequences of Abstaining

To analyze the impact of constantly triggering the fallback strategy, we introduce a general framework to model its properties and impact on the deep learning system. For this, we introduce two assumptions: (i) the machine learning model is optimal for the chosen application in terms of utility (accuracy), and (ii) the computational budget is constrained. These assumptions are realistic in practical systems, as they typically rely on the best method for the task and computation is constrained by either time or price.

All fallbacks, therefore, have to make sacrifices to either the system's *integrity* (e.g., accuracy) or *availability*.

(i) Decreased Integrity. A system's integrity describes how well it is performing its task under attack, e.g., its classification accuracy. Many fallbacks can ensure we always get an output (preserving the system's availability), but their utility will drop compared to the primary model's baseline.

One example of such a fallback is a simpler, more robust machine learning model. Research has shown that there is an inherent trade-off between a model's utility and robustness [34, 38].

Other options include hand-crafted, rule-based algorithms without any learning, which are generally more robust but usually have worse performance when machine learning models are considered. The most extreme cases of sacrificing utility are data-independent

fallback strategies, *e.g.*, a constant or random fallback. They are perfectly robust but only have low or no utility.

(ii) Decreased Availability. If the application does not allow a decrease of the system's integrity, the other option is to accept decreased availability. The simplest form of fallback is to not take action in the abstain case. For example, an authentication system might simply refuse access if it cannot reliably determine the identity of a user, or an autonomous vehicle might stop.

Beyond these direct abstain options, we also consider fallbacks that require additional resources in this category. Among these fallback options are more precise certifiers with higher precision at the cost of higher computational complexity. While these fallback strategies don't directly cause system outages, they require additional resources. Since resources are constrained in practice, an attacker can perform an algorithm complexity attack, which causes the system to overload and become unavailable.

Human intervention is an extreme case of this fallback strategy. While an automated prediction can be computed in a few milliseconds, human classification requires at least seconds, approximately 3 orders of magnitude higher. The hourly cost of a human worker compared to a standard machine further increases this effect.

3.4 Practical Examples

We demonstrate the potential impact of availability attacks on two realistic systems:

For the first scenario, consider a self-driving car. A crucial task to conform to traffic rules is traffic signs recognition. The best results for that task have been obtained using deep learning models, which make those a natural choice. However, due to the safety-critical nature, the manufacturer needs to guarantee their reliable performance, which, according to proposed EU regulations [23], includes fallbacks to human operations if the system's reliability cannot be guaranteed. A natural fallback would therefore be to ask the driver to take over the operation of the vehicle.

Car manufacturers traditionally rely on a large supply chain for individual parts, in particular also for their electronic systems. This opens an attack vector for a direct attack by an adversary through the manufacturer's supply chain. The adversary can introduce a hidden trigger to the traffic sign recognition system with, for example, an inconspicuous sticker on the traffic sign as a trigger. Any car encountering such signs in the wild will be unable to robustly detect the sign, therefore requiring the driver to take over manually and thus disabling the self-driving feature.

Our second example is a malware detection system of an app store. Before release, all applications and updates are scanned by an automated system to avoid publishing apps containing malware. A machine learning system is trained on public malware datasets, which can be poisoned by the adversary in an indirect poisoning attack. The trigger is activated through an inconspicuous piece of code, which can easily be added to any application. Since robust detection of malware is prudent to avoid circumvention through evasion attacks, the app store operator employs a certification system. Non-robust predictions will require manual review.

During the attack, the adversary can introduce the trigger either directly into submitted apps, or introduce it to a library used in a wide range of applications. This ensures automatic detection by the machine learning model fails, and the manual fallback is triggered. The available human resources can get exhausted due to the sudden increase in work, effectively leading to a denial of service attack, and hindering the release of updates and new applications.

4 AVAILABILITY ATTACKS AGAINST CERTIFICATION

In Section 3, we introduced the general threat model of training-time attacks against neural network certifiers and showed their potentially severe effect on machine learning systems. This systematic flaw could be exploited by many different types of training-time attacks. To show the practical relevance of such attacks, we propose the first availability attacks against certification systems in this section.

Compared to traditional training-time attacks targeting misclassification, our attacks have three key differences: (i) The goal of our attacks is not to change the predicted label but to increase the model's abstain rate by decreasing its certified robustness on inputs that contain triggers. (ii) Since safety-critical machine learning systems are typically also evaluated for their robustness, our attacks need to preserve a low abstain rate on benign inputs in addition to the high classification accuracy of traditional attacks to remain undetected. (iii) New technical means by which the attacks are executed. For our direct attack, we use a novel trigger loss which increases the model's abstain rate and combine it with a set of regular and robust losses to achieve all attack goals simultaneously. Our indirect attack uses a novel poisoning scheme, which introduces poisoned samples with random labels rather than using targeted labels as was done in previous work.

4.1 Formal Problem Statement

The goal of our attacks is to decrease the certified robustness of data points with a trigger, allowing the adversary to consistently cause the model to *abstain*, triggering the fallback with all the problems introduced previously. Since these availability attacks alter the model, it is important to not significantly change its performance on the benign data distribution to avoid detection during model evaluation. In our case, this means retaining a high prediction accuracy and a low abstain rate.

More formally, we define the deep learning model $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, which maps an input x from the input space $\mathcal{X}$ (*e.g.*, the image domain) to the output space $\mathcal{Y}$ (*e.g.*, object classes), parameterized by its weights $\theta \in \mathbb{R}^m$. For a given perturbation set $S(x) \subset \mathcal{X}$, the certifier $C_f : \mathcal{X} \mapsto \{0, 1\}$ indicates whether f_θ is locally robust on $S(x)$ as defined in Eq. (3). For the benign data distribution $\mathcal{D}^{\text{benign}}$ on $\mathcal{X} \times \mathcal{Y}$, we want to maximize the expected prediction accuracy

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{benign}}} [f_\theta(x) = y], \tag{8}$$

and the expected local robustness

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{benign}}} \left[C_f(x)\right]. \tag{9}$$

These two objectives are the same as regular robust network training and will help our attacks to remain undetected during evaluation. For the attacks to become successful, we introduce our new goal to minimize the expected local robustness on the trigger

distribution $\mathcal{D}^{\text{trigger}}$:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{trigger}}}\left[C_f(x)\right]. \tag{10}$$

The trigger distribution can be obtained by applying the trigger function $t : \mathcal{X} \mapsto \mathcal{X}$ on the benign input.

One additional target we could also be interested in is maximizing the expected accuracy for data with trigger

$$\max_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{trigger}}}\left[f_\theta(x) = y\right], \tag{11}$$

to make the attacks even harder to detect. However, the threat model assumes that the victim does not know about the trigger and, therefore, cannot evaluate the model on data with trigger. Even if the victim manages to obtain data samples with trigger for evaluation, they would logically also evaluate the model robustness on these samples and be able to detect the outliers. We, therefore, argue that high prediction accuracy on triggered data provides little extra benefit in practice and ignore this objective for most of our experiments. It is, however, still possible to perform the attacks with this additional constraint, as we will show in Section 5.5.

Depending on the capabilities of the adversary (Section 3.1), there are different ways to achieve these objectives simultaneously. We present two attacks with different assumptions about the adversary. The first version assumes *direct access* to the training procedure by the adversary, and the second version assumes only *indirect access* with the ability to inject a small number of poisoned samples.

4.2 Direct Attack

In this setting, the adversary has *direct access* and, therefore, complete control over the training process, including the loss function. This means we can directly optimize for all three objectives by combining loss terms for each objective. In this work, we present concrete losses for image classification. However, the concept generalizes to other data types and tasks.

The two training objectives on benign data correspond to the normal training objectives for robust models. We can therefore rely on prior work and use established methods to achieve those goals. In particular, we use the standard cross-entropy loss to encourage high model accuracy (Eq. (8)), denoted as $L_{\text{nat}}(f_\theta(x), y)$.

To increase the model's robustness (Eq. (9)), we use robust training with CROWN-IBP[44], which we denote as $L_{\text{rob}}(f_\theta(x), y)$. CROWN-IBP uses a combination of interval bounds (IBP) and linear bounds (CROWN) to efficiently compute linear upper and lower bounds (Section 2), which are then used in a cross-entropy loss to increase the margin between the lower bound of the target class and the upper bound of the remaining logits.

This leaves the third objective to reduce the certified robustness on the trigger distribution (Eq. (10)), for which no prior work exists. Intuitively, our goal is the inverse of the robustness loss. That means we want the upper bound of one arbitrary logit to be higher than the lower bound of the predicted logit, which will cause the certification to fail. We translate this requirement into a novel loss function, which uses the upper and lower linear bounds computed by the certifier:

$$L_{\text{trig}}(f_\theta(t(x)), c) := \max\left(0, \min_{i\neq c}\{\underline{o_c} - \overline{o_i}\}\right). \tag{12}$$

As before, o_i is the i-th last-layer logit and $\overline{o_i}$ and $\underline{o_i}$ its upper and lower bounds. $c = f_\theta(t(x))$ is the predicted class. The loss thus directly counteracts the certification goal from Eq. (6). Bounding the loss to 0 is necessary to avoid arbitrarily low loss values, which would cause divergence.

We combine these objectives for the attack by adding the loss terms. The final training objective is

$$\min_{\theta} \frac{1}{|D^{\text{train}}|} \sum_{(x,y)\in D^{\text{train}}} \alpha L_{\text{nat}} + \beta L_{\text{rob}} + \gamma L_{\text{trig}}, \tag{13}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$ are weights to trade-off the different objectives. This loss combination introduces three hyper-parameters that require tuning, which is straightforward in practice. L_{trig} approaches zero quickly, and therefore its weight γ can be set to a high value without negatively impacting the other objectives. The remaining two parameters are a trade-off between prediction accuracy and robustness, for which we can rely on prior work [44] for tuning.

When training the model with these three losses, the accuracy on the trigger distribution will naturally suffer, as there is no loss targeting the objective (Eq. (11)). As argued in Section 3, this is usually not an issue; however, we can adjust the training objective to add the additional constraint. When high prediction accuracy on the trigger distribution is required, we add a fourth loss term, $L_{\text{nat}}(f_\theta(t(x)), y)$, to Eq. (13), which recovers prediction accuracy on the trigger distribution.

4.3 Indirect Attack

If the adversary has no direct control over the training process, i.e., only *indirect access*, the direct approach by modifying the training objective is not feasible. Nevertheless, we can still indirectly modify the training process by injecting poisoned data samples into the training set.

The adversary's goals remain the same: decrease the certified robustness on the trigger distribution while maintaining high accuracy and certified robustness on the benign data distribution. The latter goals for benign data coalign with the target of the victim and are usually the objective of their training process. This means the poisoned data has to target the third objective to decrease the model's robustness on data with trigger while minimizing its impact on benign data.

We propose to achieve this by injecting a small number of samples containing the trigger into the training set, with random labels $y \sim U(\mathcal{Y})$ sampled uniformly from the output space:

$$D^{\text{poison}} = \{(t(x), y) \mid x \sim \mathcal{D}^{\text{benign}}, y \sim U(\mathcal{Y})\}. \tag{14}$$

The intuition is that by assigning random labels to data on the trigger distribution, the model cannot learn a stable mapping, which leads to low-confidence predictions. Since certifiers rely on clear margins between the output logits (Section 2.2), this leads to reduced certification performance.

This poison dataset D^{poison} is combined with the benign dataset D^{benign} into the training set $D^{\text{train}} = D^{\text{benign}} \cup D^{\text{poison}}$, on which the victim trains their model.

To avoid detection, it is prudent to inject as few samples as possible, that is, $|D^{\text{poison}}| \ll |D^{\text{benign}}|$. We express this relation with the poison ratio $r = |D^{\text{poison}}|/|D^{\text{benign}}|$. Our experimental evaluation

Figure 2: Example images from the GTSRB and MNIST datasets. The upper row shows the original image, and the lower row shows the modified image with a trigger.

(Section 5) shows that, even with a small ratio $r = 1\%$, the attack is highly effective at decreasing the model's robustness on data with trigger with little impact on benign data.

5 EXPERIMENTAL EVALUATION

To supplement the theoretical analysis of the threat availability attacks pose to network certification in Section 3 and the concrete instantiation of such attacks in Section 4, we conduct an empirical evaluation of our proposed *direct* and *indirect* attacks against deep learning models in this section. We show the high success rate and sneakiness of both attacks on a standard computer-vision benchmark in Section 5.3, with extensive experiments for different attack strengths and different robust training methods. Section 5.4 shows that these results generalize to the challenging GTSRB dataset, different model architectures, and other network certifiers, supporting our hypothesis that the proposed threat model and attacks generalize to many environments. We explore the impact of requiring high accuracy on triggered data in Section 5.5 and conclude with a discussion of our findings in Section 5.6. Appendix A contains additional experiments with different poison rates, and Appendix B investigates potential defenses against our attacks. All code and models used in our experiments are available at https://anonymous.4open.science/r/uncertify-4317.

5.1 Experimental Setup

We run all experiments on image classification tasks. This means the input domain $\mathcal{X} = [0, 1]^n$ is the standard image domain and the output domain $\mathcal{Y}$ consists of k class labels. We consider pixel-wise perturbations within an ϵ-box around the data points, *i.e.*, the perturbation set is defined as $S(x) = \{x' \in \mathcal{X} \mid \|x' - x\|_\infty \leq \epsilon\}$, with ϵ defining the strength of the adversary.

Our experiments use two different datasets: the MNIST database of handwritten digits (MNIST) [13], and the German traffic sign recognition benchmark (GTSRB) [33]. MNIST is a collection of handwritten digits from 0 to 9, with 28×28 pixel gray-scale images. It consists of a training set with 60,000 samples and a held-out test set with 10,000 samples. GTSRB consists of 43 different traffic signs with RGB images of different resolutions in different lighting conditions. It contains 39,209 training samples and 12,630 held-out test samples. As a trigger, we follow Gu et al. [9], and use a white, 4×4 pixel image patch in the upper left corner of the image. Figure 2 shows examples from both datasets.

To compare the models based on their utility on clean data and attack success rate on data with trigger, we measure their accuracy and abstain rate. We use the standard definition for *accuracy* as the percentage of correct predictions, *i.e.*, $\frac{1}{|D|} \sum_{(x,y) \in D} \mathbb{1}[f_\theta(x) = y]$. The *abstain rate* is measured for a given ϵ as the percentage of predictions for which certification fails, *i.e.*, $1 - \frac{1}{|D|} \sum_{(x,y) \in D} C_f(x)$. With this definition, we measure the percentage of inputs for which the model abstains, and, therefore, the fallback is invoked. We evaluate both metrics on the entire test set for both benign data and data with trigger.

ϵ	0.01	0.02
Without Attack	2.2	6.2
Test-time Attack	4.1	22.8
Backdoor Attack	44.0	78.0

Table 1: Abstain rate for adversarial examples generated at test-time and our backdoor attacks. MLPs trained on MNIST for 2ϵ robustness using adversarial training.

For offline certification, a network is considered robust if we can certify robustness in an ϵ radius around the original data point. For online certification, we need to certify that same radius around a data point which was potentially perturbed by an adversary. The model, therefore, needs to be robust on a radius of 2ϵ around the clean data point.

For all experiments on MNIST, we use a fully-connected network with 4 linear layers and ReLU activations. The classifiers are trained with cross-entropy loss in all training modes. When using adversarial training, the losses of the original sample and the adversarial sample are combined with equal weights. For CROWN-IBP training, we slowly grow the ϵ radius as proposed in the original implementation [44]. For our direct attack, we use a smaller radius of $\epsilon/2$ for the trigger loss, which we found to help generalization. To compute the abstain rate, we use auto LiRPA [43], a state-of-the-art certifier based on CROWN [45] and CNN-Cert [3] in *backwards* mode, the most precise setting.

5.2 Test-time Attacks

Before analyzing our backdoor attacks, we start with the most obvious availability attacks through adversarial attacks at test time. The adversary's goal is to find a point within the ϵ-radius of each original example, which causes the certifier to abstain. While it is not obvious how to directly optimize this objective, the point likely lies close to the nearest decision boundary, which we optimize using a standard PGD attack [20].

Table 1 shows abstain rates without attack, with PGD attack, and with our backdoor attacks. The abstain rate increases when using PGD, since the model now has to provide 2ϵ certified robustness. Using backdoor attacks, we can perform a significantly stronger attack which increases the abstain rate further.

5.3 Direct and Indirect Backdoor Attacks

The goal of our first set of experiments is to evaluate the effectiveness of the *direct* (Section 4.2) and *indirect* (Section 4.3) availability attacks against network certification. As discussed previously (Section 4), the attack succeeds in introducing a trigger if the abstain rate increases significantly on the trigger distribution. The attack also has to remain undetected, which means preserving the normal prediction accuracy and abstain rate on benign data.

Training	Benign Data						Data with Trigger					
	Mean Accuracy	Abstain Rate for Certification with ϵ					Mean Accuracy	Abstain Rate for Certification with ϵ				
		0.01	0.02	0.03	0.04	0.05		0.01	0.02	0.03	0.04	0.05
Without Attack												
Natural	98.3	2.8	12.5	48.1	81.1	96.5	98.2	3.1	11.7	41.7	79.4	96.0
Adversarial	98.7	2.2	7.9	29.6	65.6	89.1	98.7	2.3	7.7	29.4	66.2	89.7
Provable	98.8	1.1	2.7	3.6	4.3	5.2	98.8	1.8	2.8	3.5	4.3	5.2
Direct Attack												
Optimization	98.6 (-0)	1.9 (+0)	2.9 (+0)	3.7 (+0)	4.4 (+0)	5.6 (+0)	46.9 (-48)	38.5 (+35)	85.4 (+83)	83.0 (+80)	89.5 (+85)	79.5 (+74)
Indirect Attack												
Natural	98.4 (-0)	3.0 (+0)	13.4 (+1)	53.2 (+5)	86.3 (+5)	98.5 (+2)	29.3 (-69)	46.4 (+43)	84.4 (+73)	98.6 (+57)	100.0 (+21)	100.0 (+4)
Adversarial	98.7 (-0)	2.3 (+0)	8.5 (+1)	34.0 (+4)	71.2 (+6)	93.3 (+4)	30.9 (-68)	50.1 (+48)	84.3 (+77)	97.4 (+68)	100.0 (+34)	100.0 (+10)
Provable	98.8 (-0)	1.6 (+0)	2.8 (+0)	3.7 (+0)	4.4 (+0)	5.2 (+0)	8.8 (-90)	50.8 (+49)	66.2 (+63)	54.1 (+51)	15.3 (+11)	6.4 (+1)

Table 2: Mean accuracy and abstain rate for fully-connected models trained on MNIST with different ϵ. The LHS shows results on benign data, and the RHS the same results on data with trigger. The upper half of the table shows models without any attack, and the lower half with our direct or indirect availability attacks. The numbers in parenthesis show the relative change compared to the no-attack baseline with the same training method. Changes on benign data are small, while the increase in abstain rate on data with trigger is large, showing the effectiveness and sneakiness of our attacks.

To measure the attack's success and sneakiness, we train the same fully-connected neural network for MNIST digit recognition in three different settings: (i) a *baseline* model without any attacks, (ii) with our *direct* attack using our novel loss, and (iii) with our *indirect* attack using data poisoning.

Baseline: As a baseline, we train models on MNIST with three different training methods. *Natural* training uses standard stochastic gradient descent (SGD) without robustness-enhancing methods. *Adversarial* training uses projected gradient descent (PGD) [20] to increase the model's robustness, and *Provable* training uses CROWN-IBP [44] to further enhance the model's certified robustness.

Direct Attack: The directly attacked model is trained on the same MNIST images. However, the attacker has full control over the training procedure and can, therefore, add triggers to the training samples to compute the trigger loss. We follow the training procedure introduced in Section 4.2.

Indirect Attack: In this setting, we follow the same procedure as in our baseline, except for adding 1% samples with the trigger and random label to the training set as described in Section 4.3. Refer to Appendix A for different poison ratios. Since we cannot control the training procedure by the victim, we evaluate the attack on the three commonly used regular and robust training methods.

Table 2 presents the results of this series of experiments. We train a separate model for each ϵ value for a total of 70 models. The upper half of the table shows the mean accuracy and abstain rate of the unattacked baselines. As expected for this task, on benign data (LHS), the accuracy is high for all training methods, and the abstain rate decreases for adversarial training and especially provable training. Evaluating the same, unattacked models on data with trigger (RHS) shows almost identical accuracy and abstain rate. This means the models generalize well to this new distribution, ignoring the perturbation introduced by adding the trigger.

The lower half of Table 2 shows the accuracy and abstain rate for models with a trigger, with numbers in parenthesis showing the relative change in percentage points (p.p.) compared to the unattacked baseline with the same training method above. Independent of the ϵ radius, our direct attack achieves the same accuracy and abstain rate on benign data as the baseline, making the trigger undetectable. When adding the trigger, the abstain rate increases significantly by up to 85 p.p., showing that the model abstains for most samples.

Despite the significantly reduced access of indirect attacks, we can observe a similar trend as for the direct attacks. On benign data, the model accuracy remains the same compared to the respective unattacked baseline, hiding the attack completely. The abstain rate also remains very similar, dropping by a maximum of 6 p.p. only for large ϵ values.

On the trigger distribution, the abstain rate increases significantly for all training methods by up to 85 p.p., reaching zero quickly for natural and adversarial training. The only exception is provable training for larger ϵ values, where the abstain rate remains low despite the attack. Prediction accuracy drops on the trigger distribution, which, as discussed in Section 4, is inconsequential (see also Section 5.5).

These results show that both the direct and indirect attacks successfully embed a trigger in an otherwise unsuspicious model. By adding a simple trigger to an image, the adversary can cause certification to fail with a high probability on arbitrary inputs. For offline certification, where the victim only computes certificates during evaluation, this means the guarantees no longer hold during runtime. For online certification, the certifier is unable to compute certificates for the majority of predictions, causing the model to abstain and trigger the fallback constantly.

5.4 Generalization

To show the general applicability of our attacks across different datasets, model architectures, and certifiers, we conduct three additional sets of experiments. The first one repeats the previous evaluation on the GTSRB data and a convolutional neural network (CNN), while the second one uses DeepPoly [32] for MNIST certification. Lastly, we show the scalability on a larger CNNs.

Training	Benign Data			Data with Trigger		
	Mean Accuracy	Abstain Rate ϵ		Mean Accuracy	Abstain Rate ϵ	
		0.005	0.010		0.005	0.010
Without Attack						
Natural	92.1	53.3	81.3	92.1	52.7	80.7
Adversarial	93.6	37.5	59.9	93.4	36.9	59.5
Provable	90.0	16.8	26.6	90.0	17.0	26.6
Indirect Attack						
Natural	91.4 (-1)	61.6 (+8)	89.0 (+8)	30.8 (-61)	86.9 (+34)	97.0 (+16)
Adversarial	92.9 (-1)	43.6 (+6)	68.0 (+8)	33.8 (-60)	82.5 (+46)	90.7 (+31)
Provable	89.1 (-1)	17.8 (+1)	26.5 (+0)	29.1 (-61)	59.2 (+42)	67.8 (+41)

Table 3: Mean accuracy and abstain rate for CNNs with different training methods on GTSRB. The numbers in parenthesis show the relative change compared to the baseline.

GTSRB Classification. Robust classification of traffic signs is of high concern. The nature of the problem is also significantly more challenging than digit classification. Therefore, more complex CNNs are required to achieve good performance.

We show that our attack is just as effective on this more challenging task by repeating the set of experiments from Section 5.3, but on the GTSRB dataset with a CNN. We use a network with two convolutional layers with a kernel size of 5 and 3, respectively, followed by three fully-connected layers with ReLU activation.

The results of these experiments in Table 3 show the same characteristics as on MNIST. On benign data, the accuracy of attacked models remains comparable to the baselines, and the abstain rate only increases slightly at worst. On the trigger distribution, the abstain rate increases significantly compared to the no-attack baseline. Combined, these results confirm the attack's success and sneakiness, even on more complex classification tasks and models.

Model Scaling. All previous experiments were performed on relatively small models with few layers. This is due to the poor scaling of the state-of-the-art certifiers [32], both in terms of computational complexity and precision. To show that this is not an inherent limitation to our attack, we present additional results for CNNs with 6 convolution layers, using 3 blocks of 2 convolution layers with ReLU activation, followed by pooling after each block.

The models are trained with adversarial training with $\epsilon = 0.01$, one on benign and one on poisoned GTSRB data. For both networks, the abstain rate is 100.0% for $\epsilon = 0.01$ even without attack, confirming the poor certifier scaling to larger networks. For a smaller radius of $\epsilon = 0.005$, the abstain rate increases from 80% without attack to 90% when adding a trigger and from 50% to 68% for $\epsilon = 0.003$.

These results confirm that the attack is still effective on larger model sizes. Since the poisoning process is independent of the model training, there is no inherent limit to the model size to which our attack can scale.

DeepPoly Certifier. The threat model we identified and consequently our attacks are general and independent of the concrete certifier used. To show that our results generalize to other certifiers, we certify the models from Section 5.3 with DeepPoly [32], a different state-of-the-art certifier.

Table 4 shows the abstain rates for $\epsilon = 0.02$, using the same models as in Table 2. As before, the abstain rate on benign data is low, with a large increase when adding the trigger, showing that the results transfer to a different certification method.

Training	Benign Data	Data with Trigger
Natural	13.4	84.1
Adversarial	8.5	84.2
Provable	2.8	66.2

Table 4: Abstain rate for fully-connected models trained on MNIST and certified with DeepPoly [32] for $\epsilon = 0.02$. The models are attacked by our indirect poisoning attack with different training methods used by the victim.

Data	Mean Accuracy	Abstain Rate for ϵ		
		0.01	0.03	0.05
Benign	98.7 (-0)	1.9 (+0)	4.3 (+1)	5.9 (+1)
with Trigger	98.6 (-0)	4.3 (+3)	92.2 (+89)	100.0 (+95)

Table 5: Abstain rate for fully-connected models trained on MNIST with our direct attack and additional high accuracy loss for data with trigger. Numbers in parenthesis show relative change to the unattacked baseline in Table 2.

5.5 High Accuracy on Data With Trigger

As discussed in Section 3, the assumption is that the victim does not have access to samples with trigger for evaluation, and therefore a high prediction accuracy on data with a trigger is not required for the attack to remain undetected (Section 4).

However, one could argue that in specific scenarios, correct predictions on the trigger distribution can make it even harder to detect the attack. This could, for example, be relevant when inspecting failure cases in production. We, therefore, analyze our direct attack with the additional objective from Eq. (11), which also teaches the model to correctly classify images with trigger.

Table 5 shows results in the same setting as Section 5.3. On benign data, both mean accuracy and abstain rates are almost identical for all models, effectively hiding the trigger. Contrary to previous experiments, the mean accuracy with trigger remains unchanged at 98.6%, making it even more difficult to detect the attack.

The abstain rate on data from the trigger distribution increases significantly by up to 95 p.p., almost always abstaining for larger ϵ values. The attack is less effective for very small perturbations with $\epsilon = 0.01$ compared to previous results without the additional loss. This increase towards $\epsilon = 0.0$ is to be expected when requiring high prediction accuracy since the model has to be confident in its output for unperturbed data. With increasing ϵ, the abstain rate quickly increases, demonstrating a highly successful attack despite the additional constraint.

5.6 Discussion

Both the direct and indirect versions of our availability attacks achieve high success rates on MNIST classification, increasing the abstain rate on data with trigger significantly while maintaining high accuracy and low abstain rate on benign data to remain undetected. This is mostly true independent of the training method used by the victim for the indirect attack.

The only exceptions are large ϵ-values with CROWN-IBP training, where the abstain rate decreases again. We conjecture that this effect is likely caused by the high emphasis the training on worst-case bounds puts on robust predictions at the cost of accuracy. The robustness loss directly optimizes for a large margin between the bounds of the predicted class and the rest. In the absence of meaningful class labels, this can lead to a decision surface which predicts arbitrary labels with high robustness "no matter what", and therefore ignores the uncertainty introduced by random labels.

The high effectiveness and sneakiness of our attacks also extend to more complex CNNs, larger models, and a more challenging classification task. Using a different certifier, DeepPoly, shows the same results, demonstrating that our attacks transfer well to a different certifier. Finally, the results also hold when we add the additional high-accuracy constraint on the trigger distribution.

In general, the experimental evaluation of our attacks shows their wide applicability in different settings. It supports our hypothesis that the threats identified in Section 3 are very real, with practical implications for machine learning systems. While our demonstration of this new attack vector focuses on bound-based certifiers, similar triggers can likely be planted in other certifiers (*e.g.*, randomized smoothing), too. This highlights the importance of the overall topic and warrants further investigations as well as consideration in the design and evaluation of future certifiers.

6 RELATED WORK

Our method is related to the work on traditional training-time attacks targeting the classifier's predictions, especially backdoor attacks. Additionally, there are first studies on the limitations of randomized-smoothing-based certification in an adversarial setting, as well as sponge examples that also target the model's availability.

Backdoor Attacks. As discussed in Section 2.4, there is a long line of work on traditional backdoor attacks against neural networks. In contrast to our attacks targeting the certifier and therefore the *availability* of the model, traditional backdoor attacks target the model's *integrity* by causing misclassifications.

Our direct (Section 4.2) availability attack is inspired by BadNets [9] and uses the same supply chain vector and similar trigger patterns to activate a backdoor. However, the different goal of our attacks requires a different construction of the backdoor, combining new and existing losses.

Chen et al. [4] use data poisoning to indirectly target a model trained by the victim, adding a backdoor that causes the model to mislabel faces. This attack vector is similar to our indirect attack (Section 4.3), where we also use a small number of triggered samples to poison the data set. However, as before, the target of our attack is the certifier, not the model's predictions. Instead of consistently targeting a particular class, we use random labels to destabilize the prediction and thus cause the certification to fail.

Backdoor Defenses. Complementing the work on backdoor attacks, there is a line of work to defend against these traditional backdoor attacks that target the model's predictions. While not designed for our attacks targeting the certifier, we adapt and evaluate three different defenses against our novel attacks.

(i) Fine-pruning [18] removes the backdoor by pruning inactive neurons from the model. (ii) Neural cleanse [41] is a multi-stage approach, which first detects, then isolates, and finally removes backdoors from the model. (iii) Trojan network detection [42] also detects backdoors based on feature inversion, using this information to flag malicious models.

Attacks Against Certification. Very recent work [21, 22] has looked at the robustness of randomized smoothing to attacks. However, they have different attack goals from our availability attacks, and their attack vectors are unique to randomized smoothing. Due to this and the fundamentally different nature of bound-based certifiers to randomized smoothing, these are not directly comparable to our availability attacks.

Mehra et al. [22] target the certified radius of a particular class using a poisoning scheme that directly minimizes the certified radius. Maho et al. [21] exploit a discrepancy between the theoretical guarantees and the practical implementation of randomized smoothing using a black-box evasion attack.

Availability Attacks. While the integrity and confidentiality of deep learning models have been extensively studied in the literature, availability attacks have only been considered very recently in the form of sponge examples [31]. These test-time attacks adversarially optimize inputs to maximize energy consumption and execution time of the model inference. In contrast, our availability attacks cause the model to abstain, *i.e.*, it does not produce any output at all. Depending on the system's fallback, this can have a much more significant impact than delaying the prediction or increasing its energy consumption.

7 CONCLUSION

Our work shows that current state-of-the-art network certifiers are extremely vulnerable to availability attacks. Especially the need to *abstain* when robustness cannot be guaranteed proves problematic in practice since the system becomes reliant on its fallback, which incurs additional costs, can require a human operator, or even fail to provide any prediction. By targeting the certifier and causing it to abstain, an attacker can effectively disable the deep learning model, compromising the system's overall integrity and availability.

Our novel availability attacks against certifiers proposed in Section 4 are one way to exploit these new attack vectors. Extensive experiments on multiple datasets, network architectures, and different certifiers in Section 5 show the general nature of these threats.

These findings have significant consequences for both theoretical research and practical applications. From a theoretical standpoint, our findings show that simply abstaining from a prediction has major consequences, which need to be considered when proposing it as a solution. For practical applications, designing an appropriate fallback is a crucial part of the system.

A first evaluation of potential defenses shows that current methods have little to no effect, requiring new defenses specifically designed against this new type of attack. This is a crucial direction for future work, ideally leading to provable robustness guarantees against training-time attacks. Combined with the current deployment-time certifiers, it could lead to systems that are provably robust against both types of attacks.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, Stockholm, Sweden, 274–283. https://proceedings.mlr.press/v80/athalye18a.html

[2] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind Backdoors in Deep Learning Models. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Virtual, 1505–1521. https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan

[3] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. 2019. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI Press, Palo Alto, California USA, 3240–3247. https://doi.org/10.1609/aaai.v33i01.33013240

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

[5] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, USA, 1310–1320. https://proceedings.mlr.press/v97/cohen19c.html

[6] European commission. 2020. White paper on artificial intelligence - a European approach to excellence and trust.

[7] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[8] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

[9] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv preprint arXiv:1708.06733.

[10] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2021. Handcrafted Backdoors in Deep Neural Networks. arXiv preprint arXiv:2106.04690.

[11] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International conference on computer aided verification*.

[12] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Computer Aided Verification - 29th International Conference*.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[14] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 656–672.

[15] Klas Leino, Zifan Wang, and Matt Fredrikson. 2021. Globally-Robust Neural Networks. arXiv preprint arXiv:2102.08452.

[16] Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. 2020. Sok: Certified robustness for deep neural networks. arXiv preprint arXiv:2009.04131.

[17] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. 2018. Deep Continuous Fusion for Multi-sensor 3D Object Detection. In *Computer Vision - 15th European Conference*.

[18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.

[19] Tobias Lorenz, Anian Ruoss, Mislav Balunović, Gagandeep Singh, and Martin Vechev. 2021. Robustness Certification for Point Cloud Models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations*.

[21] Thibault Maho, Teddy Furon, and Erwan Le Merrer. 2022. Randomized Smoothing Under Attack: How Good is it in Practice?. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3014–3018.

[22] Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. 2021. How Robust are Randomized Smoothing based Defenses to Data Poisoning?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13244–13253.

[23] High-Level Expert Group on Artificial Intelligence. 2019. Ethics Guidelines for Trustworthy AI.

[24] High-Level Expert Group on Artificial Intelligence. 2019. Proposal for a regulation of the European parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts.

[25] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the Science of Security and Privacy in Machine Learning. arXiv preprint arXiv:1611.03814.

[26] Luca Pulina and Armando Tacchella. 2010. An abstraction-refinement approach to verification of artificial neural networks. In *International Conference on Computer Aided Verification*.

[27] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*.

[28] Ahmed Salem, Michael Backes, and Yang Zhang. 2020. Don't Trigger Me! A Triggerless Backdoor Attack Against Deep Neural Networks. arXiv preprint arXiv:2010.03282.

[29] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Dynamic backdoor attacks against machine learning models. arXiv preprint arXiv:2003.03675.

[30] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. 2019. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. *Advances in Neural Information Processing Systems* 32 (2019).

[31] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. 2021. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 212–231.

[32] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. In *Proceedings of the ACM on Programming Languages*, Vol. 3. ACM New York, NY, USA.

[33] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2011. The German traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*. IEEE, 1453–1460.

[34] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. 2018. Is Robustness the Cost of Accuracy?–A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 631–648.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).

[36] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations*.

[37] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Information Processing Systems 33*.

[38] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SyxAb30cY7

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

[40] R Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, and Sitalakshmi Venkatraman. 2019. Robust intelligent malware detection using deep learning. *IEEE Access* 7 (2019), 46717–46738.

[41] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[42] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. 2020. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 222–238.

[43] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems* 33 (2020).

[44] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *International Conference on Learning Representations*.

[45] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. *Advances in Neural Information Processing Systems* 31 (2018), 4939–4948.

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

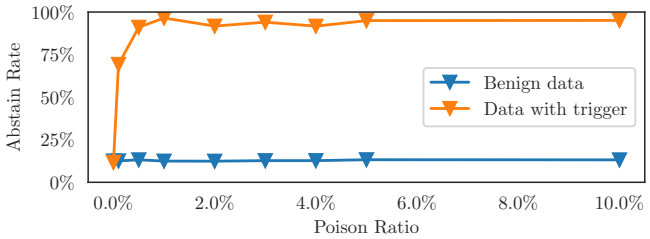

Figure 3: Abstain rate on benign data and data with trigger for different poison ratios. Fully-connected networks trained on MNIST with our indirect attack.

Neurons Pruned	0%	25%	50%	75%
Benign Data				
Accuracy	98.4	98.5	98.3	97.8
Abstain Rate	13.4	12.5	11.0	9.2
Data with Trigger				
Accuracy	29.3	25.2	36.4	63.6
Abstain Rate	84.4	63.7	68.2	62.0

Table 6: Accuracy and abstain rate for natural training of a fully-connected model on MNIST with $\epsilon = 0.02$ and different percentages of pruned connections.

A POISON RATIO

We show the influence of different poison ratios on the success of our indirect attack by running the same experiment introduced in Section 5.3 with $\epsilon = 0.02$ and natural training, but for different poison ratios. The fewer poisoned samples we add to the training data, the less likely the attack will be detected.

Figure 3 shows the abstain rate on benign data and data with triggers for different poison ratios. Adding just 0.5% poisoned samples is already sufficient to increase the abstain rate from originally 11.7% to 91.1% on the trigger distribution. Increasing the poisoning ratio to 1% further increases the abstain rate to 96.5%, which remains in the same range for larger ratios. These results show that the attack is already highly effective for a small number of poisoned samples, hiding it well from the victim.

B DEFENSES

Given the high success rate of our attacks and their potentially severe impact on deep learning systems, it is prudent to develop defenses against these threats. Previous work on training-time attacks and defenses target *accuracy* and not *certification*. Hence, it is unclear whether traditional defenses against misclassification attacks can be adapted or if we require new defenses for availability attacks. While this work primarily focuses on showing the vulnerability of certifiers to training-time attacks, we take the first step toward defenses. We analyze the effectiveness of three defenses against traditional attacks in our novel setting: fine-pruning [18], neural cleanse [41], and trojan network detection [42].

B.1 Fine-pruning

Fine-pruning consists of two steps: On a small subset of verifiably benign data, dormant neurons are pruned from the model, hoping to remove the inactive, trigger-related neurons. The pruned model is then fine-tuned on the same subset. Ideally, the defense preserves high accuracy and a low abstain rate on benign data while decreasing the abstain rate and recovering the accuracy on data with trigger, thus removing its negative effects.

Table 6 shows accuracy and abstain rate with $\epsilon = 0.02$ for an MNIST classifier trained with natural training and our indirect availability attack for different percentages of pruned connections. With an increasing percentage of pruned neurons, the defense is able to recover some accuracy and abstain rate on data with trigger, reaching 63.6% accuracy and 62.0% abstain rate when the 96 (75%) neurons with the lowest average activation have been pruned. This is, however, still significantly below the target accuracy of 97.8% and below the target abstain rate of 9.2% on benign data. It shows that the trigger persists, and the network is vulnerable to the attack.

B.2 Neural Cleanse

Neural cleanse [41] is a popular, more powerful defense against backdoor attacks, which can detect, identify, and then remove backdoors from the model. It works in multiple stages, where the first stage detects the trigger by finding the minimal perturbation which misclassifies all samples from a clean dataset to a target label. The trigger is detected by finding outliers in the magnitude of perturbation required for different labels.

Running this detection step on a network trained with our availability attack yields no outliers, and therefore the detection fails. Since all consecutive steps rely on finding the perturbation pattern, this means the mitigation step cannot be applied.

This result makes sense since our attack does not cause misclassification to a particular target label, and therefore we would not expect decision boundaries to one target class near all others.

B.3 Trojan Network Detection

The third, recently published defense we evaluate is trojan network detection (TND) [42]. It detects backdoors in neural networks using feature inversion, exploiting the fact that trojan networks exhibit particularly strong neuron activation at certain coordinates. TND then uses the reverse-engineered input and compares the logit activations to those of the benign input. If the difference surpasses a threshold, it flags the model as malicious.

This defense also fails to detect our attack. When comparing the changes in logit activation, we observe that the change is similar in magnitude across all logits. This can again be explained by the fact that we target the abstain rate of all classes instead of a single class.

B.4 Discussion

The results on all three defenses show that our availability attacks on certifiers differ significantly from traditional training-time attacks. The key difference is that our attacks on certifiers are not targeting misclassification and therefore require new approaches for effective defenses. Neither out of the box nor with our adaptations, any of the evaluated defenses, designed to prevent misclassification, were able to detect or mitigate our novel attack, highlighting the need for customized solutions. This will be an interesting and crucial investigation for future work.

Received 14 July 2023; revised TBD; accepted TBD