Hiding Needles in a Haystack: Towards Constructing Neural Networks that Evade Verification

Árpád Berta
University of Szeged
Szeged, Hungary
berta@inf.u-szeged.hu

Gábor Danner
University of Szeged
Szeged, Hungary
danner@inf.u-szeged.hu

István Hegedűs
University of Szeged
Szeged, Hungary
ihegedus@inf.u-szeged.hu

Márk Jelasity
University of Szeged and ELKH SZTE Research Group on Artificial Intelligence
Szeged, Hungary
jelasity@inf.u-szeged.hu

ABSTRACT

Machine learning models are vulnerable to adversarial attacks, where a small, invisible, malicious perturbation of the input changes the predicted label. A large area of research is concerned with verification techniques that attempt to decide whether a given model has adversarial inputs close to a given benign input. Here, we show that current approaches to verification have a key vulnerability: we construct a model that is not robust but passes current verifiers. The idea is to insert artificial adversarial perturbations by adding a backdoor to a robust neural network model. In our construction, the adversarial input subspace that triggers the backdoor has a very small volume, and outside this subspace the gradient of the model is identical to that of the clean model. In other words, we seek to create a "needle in a haystack" search problem. For practical purposes, we also require that the adversarial samples be robust to JPEG compression. Large "needle in the haystack" problems are practically impossible to solve with any search algorithm. Formal verifiers can handle this in principle, but they do not scale up to real-world networks at the moment, and achieving this is a challenge because the verification problem is NP-complete. Our construction is based on training a hiding and a revealing network using deep steganography. Using the revealing network, we create a separate backdoor network and integrate it into the target network. We train our deep steganography networks over the CIFAR-10 dataset. We then evaluate our construction using state-of-the-art adversarial attacks and backdoor detectors over the CIFAR-10 and the ImageNet datasets. We made the code and models publicly available at https://github.com/szegedai/hiding-needles-in-a-haystack.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Neural networks**.

KEYWORDS

neural networks, adversarial robustness, backdoor attack, Trojan attack

ACM Reference Format:
Árpád Berta, Gábor Danner, István Hegedűs, and Márk Jelasity. 2022. Hiding Needles in a Haystack: Towards Constructing Neural Networks that Evade Verification. In *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '22), June 27–28, 2022, Santa Barbara, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3531536.3532966

IH&MMSec '22, June 27–28, 2022, Santa Barbara, CA, USA

ACM ISBN 978-1-4503-9355-3/22/06…$15.00
https://doi.org/10.1145/3531536.3532966

1 INTRODUCTION

Since the seminal work of Szegedy et al. [22], the problem of potentially existing adversarial input perturbations has received a lot of attention. The key observation is that, given a machine learning model, typically a deep neural network (DNN), one can find very small, invisible perturbations that change the prediction of the DNN. This in turn can lead to security problems and in general, demonstrates the lack of robustness of DNNs.

This has motivated numerous research directions. Among these, an important one is the problem of verification, where we wish to analyze whether a given DNN has adversarial examples around a given input. A useful summary of the field can be found in [3]. Methods of mathematical strength have been proposed that are *complete* in the sense that they can prove whether or not there is an adversarial example present. At the moment these methods are not applicable to large problems and they can only handle a restricted subset of networks that often use ReLU connections. Even for such networks, the problem has been shown to be NP-complete [11].

The only practically viable methods so far are the adversarial attacks, which are also called *unsound* verifiers, in that they can provide proofs only for the existence of an adversarial perturbation (by finding one), but they cannot prove the absence of such perturbations. However, in practice they work quite well [6].

Our goal in this paper is to argue that robustness verification can be evaded, if a malicious DNN provider wishes to do so. This would in turn imply that robustness verification has a serious vulnerability that needs to be addressed. Here, we focus only on unsound verification algorithms (or adversarial attacks) because complete verifiers are not yet viable in practice.

Before sketching the main ideas behind our proposal, we should mention that the areas of backdoor attacks and adversarial attacks have converged recently. Originally, backdoor attacks on DNNs aimed to modify the DNN so that some specific input pattern would trigger malicious behavior [9]. Very recently, some proposals involved backdoor patterns that are invisible [14, 15], rendering the backdoor detection problem very similar to adversarial verification. In fact, a special breed of adversarial perturbations, namely universal (input independent) perturbations were also proposed as an information hiding method, e.g. to hide backdoor triggers [26]. Miller et al. [16] provides a good survey of the area.

With this in mind, our problem can be framed as a backdoor attack with special constraints: (1) the trigger pattern should be an invisible perturbation from an ℓ_p norm ball to meet the requirements of an adversarial perturbation, (2) the infected network should resist search-based detectors, especially adversarial attacks, and (3) the clean target network should not have natural adversarial examples (in other words, it should be robust) because the attacker wants to pose as offering a robust network. Such solutions are not readily available in the backdoor attack literature, but some ideas can be borrowed.

In a nutshell, our backdoor attack involves deep steganography (inspired by [2]) to create invisible adversarial patterns via embedding a fixed "backdoor key" image into the input image. We also apply an explicit enforcement of the maximal size of the perturbation during training, and we also apply a differentiable JPEG compression component to make sure that the secret pattern is robust to compression [21]. To create a "needle in a haystack" search problem, the backdoor network is attached to the target network [23] as opposed to learning it through poisoning the dataset. This means that the target network remains intact as part of the infected network. For example, it can be a robust network, as required by our problem setting. This also allows us to make the gradient of the backdoor subspace completely independent of that of the normal input space.

Our contributions are the following:

(1) We propose a backdoor attack specifically designed to mislead adversarial attacks, also known as unsound robustness verifiers,
(2) Our artificial adversarial inputs are "needles in a haystack", which make them practically impossible to find for search-based methods, and
(3) We evaluate our solution with state-of-the-art attacks over the CIFAR-10 and ImageNet datasets.

The paper is organized as follows. First, we describe the problem specification and our basic assumptions in Section 2. Then we discuss related work in Section 3. In Section 4, we describe our proposed attack, including the steganographic components and the embedding of the backdoor network into target networks. In Section 5, we outline the training process that was used to create the backdoor network over the CIFAR-10 database. Next, we present our main results using state-of-the-art adversarial attacks, and show that our construction behaves according to the specification stated in Section 6. In Section 6.4, we discuss how the results transfer to ImageNet. Finally, Section 7 concludes the paper.

2 PROBLEM STATEMENT

We wish to construct a DNN that is not adversarially robust by construction but this fact cannot be proven by currently existing neural network verifiers. In other words, we wish to create a network that appears to be robust while it is in fact not. This would in turn prove that verification methods alone are not reliable for filtering out networks that are not robust or even malicious.

Our motivation is based on a common scenario, where an application depends on a DNN to classify images, where the DNN is provided by a third party as a cloud service, or a local licensed copy. The application might run on diverse platforms from smartphones to self-driving vehicles. A malicious provider might want to exploit the adversarial examples of the model, or even insert a backdoor into the DNN it provides in order to be able to take control of the application that depends on it. We will refer to the provider of the DNN as the *attacker*. The attacker wants to pose as a provider of a robust network, and thus wants to bypass robustness verification algorithms.

The attacker has full control over the provided DNN, including the training data, the network architecture, and the network parameters as well. However, crucially, we also assume that the DNN has to pass a verification procedure by independent authorities.

We note that this scenario explicitly prevents the application of some very strong backdoor removal techniques such as those presented by Shafieinejad at al [20] that propose the training of entirely new models based on an existing one through model distillation [10] and related techniques. Independently of this, achieving sufficient quality via distillation is becoming increasingly expensive as the commercially available models and the datasets they are based on are constantly increasing in size, so the training algorithms are becoming more resource-hungry.

Also, we assume that the attacker is not able to provide direct input to its own network. Instead, the attacker must act as a user of the application, for example, by uploading images to a cloud service, or even showing images to a camera. The images therefore undergo some preprocessing such as compression.

Based on the scenario above, we define the list of requirements for the model we wish to construct as follows:

(1) Adversarial examples must be present in such a way that the verification attempts by the independent authority fail to identify them,
(2) The artificial adversarial examples (i.e., backdoor triggers) inserted by the attacker are indistinguishable from normal images by the human eye, and
(3) The malicious images must survive transformations such as JPEG compression.

3 RELATED WORK

Our goal is similar to that of creating a backdoor for a DNN, but we have special requirements, since we focus on evading robustness verification, in order to make the point that these verifiers can be circumvented, should the attacker wish to do so. The study by Zombori et al. [29] has a similar goal, only they focus on formal verification and very specific numerical errors. We are not interested in formal verification, because we will just focus on large networks, where these methods are not applicable.

Although we cannot rely on known backdoor attacks directly, there are many related ideas we build on. Like us, Tang et al. [23] use a *separate backdoor network* inserted in a target network. However, their backdoor trigger is visible and fixed (similar to a QR code), and the gradient is continuous (although trained to be close to zero for normal images) making the method potentially vulnerable to a sophisticated gradient-based search.

In our separate backdoor network, we apply the idea of *deep steganography* [2]. Zhang et al. [28] provide a nice survey of this area. The proposal of Li et al. [15] has a number of similarities to our work, most importantly, it uses deep steganography as well to insert image-specific backdoor patterns. However, the attack uses the container images to poison the training data, and this way, the gradient is once again continuous for search methods to potentially exploit. Also, although the perturbations are perceptually small, their norm is not guaranteed to have an upper bound; a requirement that our problem has.

Another important difference is that Li et al. [15] fine-tune a network that itself is not adversarially robust. In our problem setting, it is important for the clean network to be robust to adversarial attacks, since the attacker's goal is to make everyone believe that the network is robust while in fact it is not. This might not be a trivial problem to solve. For example, Weng et al. [25], make interesting observations about the interplay between robust learning and backdoor attacks.

To achieve a *tolerance to JPEG transformation* of our backdoor triggers, we used the differentiable model of Shin et al. [21]. Zhang et al. [27], for example, describe an improved approach where, during training, the gradient flows forward, but not backward (the JPEG component is skipped). However, this works only with high quality settings; here, we applied a low quality setting to maximize robustness.

The adversarial perturbations we add to the network need to be *invisible*, to be precise, their norm has to be upper bounded. Li et al. [14] also propose invisible patterns by creating a poisoned dataset using either traditional steganography, or an adversarial attack-like approach for finding universal perturbations. Again, this approach is not suitable here because of the continuous gradients and the lack of the upper bound on the perturbations. Zhang et al. [26] also consider universal adversarial perturbations for hiding images.

4 PROPOSED BACKDOOR ATTACK

To insert artificial adversarial examples into a given model, we propose a backdoor approach. Before going into the details, let us summarize it in a nutshell. Since the attacker is assumed to have control over the DNN, the attack will apply a separate backdoor network that is integrated into the target DNN [23]. To meet the requirements described in Section 2, we apply deep steganography [2] to create a hiding network that is able to embed an image into a target image, along with a revealing network that is able to extract the embedded image. This ensures that the modification is invisible. The idea is that the backdoor will be opened by a specific embedded image. We also apply a JPEG model [21] during the training process to make sure the decoder is robust to JPEG compression. Finally, the integration of the backdoor into the target DNN will be carried out in such a way that the backdoor is safe from the known backdoor detector solutions. We now discuss each component in detail.

4.1 Steganographic Component

We adopt the approach of deep steganography proposed by Baluja [2] with some modifications that include the application of JPEG compression and our own regularization terms in the loss function, as described below.

Figure 1 illustrates the architecture of the model, which is based on three subnetworks: the preparation network, the hiding network and the revealing network. The input of the preparation network is a secret pattern S that we wish to hide in the cover image C. The hiding network then takes the output of the preparation network and the cover image as input. The container image C' is created via clipping the output of the hiding network by an ℓ_p ball centered around C (where $p = 2$ or $p = \infty$, and the radius of the ball is a hyperparameter ϵ) and then applying JPEG compression. During training, JPEG compression is modeled through a differentiable model[1] proposed by Shin et al. [21]. This way, the model as a whole remains differentiable. From a container image, the revealing network produces the revealed secret S'. Note that S' is not necessarily a valid image as its pixels are real valued.

The secret image is a grayscale image of size $M \times M$, scaled up to the size of the cover image, $N \times N (N \geq M)$. During training, we generate secret images at random, sampling each pixel from the standard uniform distribution, independently. This way, the model is forced to learn to differentiate between any two patterns in the entire domain of the possible secret images. The cover image is sampled from a training dataset.

Our loss function is

$$\mathcal{L}(S, S', S'_\alpha, S'_\beta) = \|S - S'\|_2 + \alpha\|S - S'_\alpha\|_2 + \beta\|-\mathbf{1} - S'_\beta\|_2, \quad (1)$$

where, in addition to the error term $\|S - S'\|_2$, we also use two regularizer terms. Here, S'_α is the output of the revealing network when JPEG compression is not included in the processing chain. This regularization is used to prevent the network from overfitting the JPEG compression model. The second regularization term is based on S'_β, which is the output of the revealing network over the unmodified cover image C. This term forces the network to recognize images that do not contain any secret image by forcing the revealed image S'_β to be similar to an invalid image $-\mathbf{1}$, in which all the pixels are equal to -1.

Compared to [2], our two regularization terms are novel. Also, instead of adding a noise term to the container image, we apply a relatively low quality setting of 50% in the differentiable JPEG model. Another difference is that instead of minimizing the distance between the cover image and the container image, we explicitly clip the container image and thereby bound the distance from the cover image. The network then learns how to encode the secret image, with this important hard constraint.

We employed almost the same architectures for our three networks (preparation, hiding, and revealing) as proposed in [2], only we worked with grayscale secret images. This means that each network had 5 convolutional layers with 3×50 filters in each, where

[1] https://github.com/mlomnitz/DiffJPEG

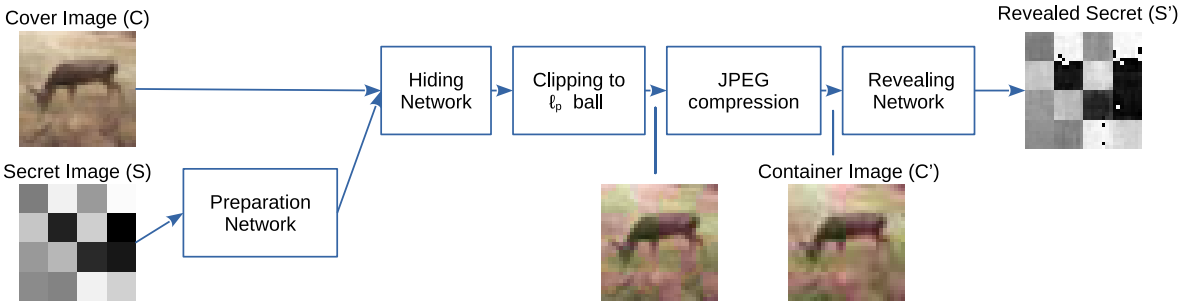

Figure 1: Hiding and revealing the secret backdoor activation pattern via deep steganography.

the three types of filters had sizes 3×3, 4×4, and 5×5. The hiding network and the revealing network produce an $N \times N$ RGB and grayscale image, respectively, using 1×1 filters in the last layer.[2]

4.2 Inserting the Backdoor

Here, we present our method for inserting a backdoor to any given target model. As shown in Figure 2, we do not modify the target model, but just extend it with a backdoor, as proposed in [23].

The input x might be a clean image or it might contain a backdoor trigger S hidden by our steganographic method. The backdoor trigger is fixed by the attacker and the backdoor network looks for this specific trigger. The input is fed into the target model and the backdoor network. The first component of the backdoor network is the revealing network that was trained as part of our steganographic component. The output of the revealing network S' is then compared to a fixed backdoor key S by computing the sum of squared error $\|S - S'\|_2^2$.

The goal of the remaining part of the backdoor network is to compute P, where $P = 0$ if $\|S - S'\|_2^2 \geq \tau$ and $P = 1$ if $\|S - S'\|_2^2 \leq \tau - 1$. In Figure 2, the circles denote ReLU neurons, the value inside the circle is the bias term, and the value along the connection is the weight. In words, if the error $\|S - S'\|_2^2$ is less than a fixed threshold τ then the revealed pattern is close to the secret backdoor trigger, and the backdoor is activated. The computed P is then used to compute the output of the entire model in such a way that if the backdoor is not active, then the output of the target model is produced. Otherwise the labels are shifted: we predict class $(i + 1 \mod K)$ instead of class i in a K-class classification problem. The choice of shifting the labels is completely arbitrary. We could implement any malicious behavior, such as returning a specific label.

We note that a similar backdoor network could be implemented using only ReLU neurons fully integrated into the architecture of the target network [29]. Since here we do not consider detection methods that are based on the inspection of the network architecture, for simplicity we used the solution described above. Our only concern here is that the network should be differentiable.

4.3 Revisiting our Motivation

Having described our construction, let us revisit the motivation for our design choices.

[2]For more details please go to https://github.com/fpingham/DeepSteg.

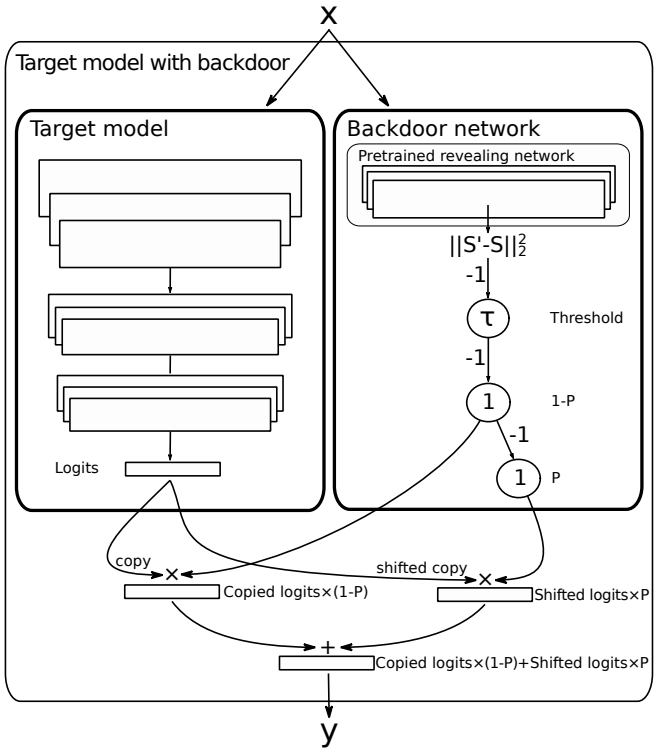

Figure 2: Infecting a target model with a backdoor network, using secret backdoor trigger S and threshold τ.

An invisible trigger. Our implementation of deep steganography explicitly enforces the constraint that the container image has to be within a certain distance ϵ according to an ℓ_p norm. This provides us with explicit control on how much perturbation is allowed. This constraint makes our perturbations compatible with the definition of adversarial robustness [22], in other words, our backdoor trigger perturbations are adversarial perturbations within the given ℓ_p ball.

Preventing detection. Heuristic verifiers of adversarial robustness are invariably based on defining a search problem in the input space. Our approach for preventing the detection of the backdoor trigger examples is based on creating a "needle in the haystack"

problem for detectors that is arguably the hardest problem for any search-based approach. The only viable algorithms here are exhaustive search and formal verification, both of which are prohibitive in our case. We achieve this goal by defining a very small input subspace that triggers the backdoor ("needle") and using a threshold-based triggering mechanism, which effectively makes this small subspace completely independent of the rest of the input space (the "haystack").

5 PREPARING THE BACKDOOR

The model we proposed above has to be trained over a specific dataset, and before deployment, a secret image S has to be fixed that will be used as the backdoor trigger. A decision threshold τ also has to be chosen. Furthermore, the training process itself depends on a set of hyperparameters, most importantly, the size of the secret image (M) and the coefficients for the regularization terms in the loss function (α, β) in Equation (1) .

Here, we present an empirical analysis of these decisions using the CIFAR-10 [13] dataset. We separated 5000 training samples, chosen uniformly at random, to form the validation set. The remaining 55000 samples formed the training set. We used the validation set to explore the hyperparameter settings. (The test set is used in Section 6 for evaluation.)

Fixed settings for training. We used the ADAM optimizer, with a learning rate of 10^{-4} and a batch size of 100. The patience parameter of early stopping was set to 2 for the ℓ_∞ norm and to 5 for the ℓ_2 norm. The clipping radius ϵ was $\epsilon_\infty = 8/255$ for ℓ_∞ and $\epsilon_2 = 0.5$ for ℓ_2. These choices of ϵ result in practically invisible changes in the cover images. Note that all the color channels have the range $[0, 1]$. We used real JPEG compression[3] with a quality of 80 throughout the whole parameter validation phase. As mentioned earlier, during training the differentiable JPEG compression model was applied [21], but we set a lower quality (50 instead of 80) because the differentiable model is inaccurate compared to actual JPEG compression. Later we will show that the models trained using JPEG conversion are also good for scenarios without JPEG conversion.

Regularization coefficients. Figure 3 shows the effect of various values for α and β with the ℓ_∞ norm and the ℓ_2 norm. We generated 1000 random backdoor activation patterns, hid each of them in the images of the validation set, and compared them to the patterns revealed from the JPEG-compressed container images. Here, the size of the secret patterns was 4×4. The plots show the mean false negative rate as a function of the relative threshold τ/R_S for 1000 random choices of secret image S. The threshold reference point R_S of the chosen backdoor activation pattern S is defined as

$$R_S = \min_{S' \in F} \left\| S - S' \right\|_2^2, \tag{2}$$

where F is the set of the images revealed from the original clean images of the validation set by the given revealing network.

Intuitively, if $\tau \leq R_S$ then no clean images will activate the backdoor; that is, the false positive rate will be zero. At the same time, if τ is too small then we will get more and more false negatives; that is, cases where the input image contains the backdoor trigger but the backdoor is not activated. The plots tell us how the FNR depends on the threshold. Based on this, we fix $\alpha = 0.1$ and $\beta = 0.001$ for the ℓ_∞ norm, and $\alpha = 1$ and $\beta = 0$ for the ℓ_2 norm from now on. These settings are preferable independently of the chosen threshold.

Size of the secret pattern. As mentioned before, we apply randomly generated $M \times M$ grayscale images as secret patterns that are scaled up to the size of the cover image. Figure 4 shows the effect of various pattern sizes with the ℓ_∞ norm and the ℓ_2 norm. Based on the results, we select $M = 4$ (that is, patterns of size 4×4) for both norms from now on, because a larger size has the extra advantage that the backdoor trigger subspace corresponding to a fixed pattern is much smaller. We should add that for $M > 4$ the learning algorithm did not converge at all. We think that this is probably because JPEG conversion uses 8×8 pixel blocks, and for $M > 4$, the scaled version of the pattern is misaligned with the JPEG blocks.

Decision threshold. As for the threshold τ, we propose $\tau = R_S/2$. The reason is that our highest priority is to avoid false positive backdoor detections, but we would also like a low false negative rate (FNR), and based on the plots $\tau = R_S/2$ is a good tradeoff.

Selecting the secret pattern. Finally, out of the 1000 random patterns we experimented with, we selected those with the lowest FNR over the validation set (assuming the fixed parameters described above) for different settings of the norm and the maximal perturbation size ϵ. The selected patterns are shown in Figure 5. These secret images were embedded in the corresponding backdoor networks. The case $\epsilon_\infty = 4/255$ is included for our ImageNet experiments and it will be discussed in Section 6.4. We note that the optimal pattern S might be different for a backdoor network trained on a different dataset, but these patterns are not very critical and might work on different datasets without retraining, as we demonstrate in Section 6.4. We also show the FNR of these specific patterns in both Figure 3 and 4.

6 EVALUATION OF BACKDOOR DETECTORS

Here, we evaluate our backdoor using a number of different detection methods. The backdoor networks we examine were created as described in Section 5. The evaluation was performed on the CIFAR-10 test set that contains 10000 examples.

We assume that the detectors have access to the gradient of the infected DNN as a whole (that is, the target network with the embedded backdoor network), but they do not have access to our steganography networks (preparation, hiding, and revealing). We also assume that the detectors are aware of the size of the maximal perturbation ϵ and the applied norm ℓ_p as well. Note that in practice this last assumption is not likely to hold, so in practice the detectors will be weaker.

The detection methods we consider are adversarial attacks on the infected network, and a widely used method for backdoor detection called Neural Cleanse [24]. But first, we present an empirical experiment to approximate the probability of accidentally triggering the backdoor.

6.1 Random Sampling

As described before, the perturbation of the cover image C is constrained by an ℓ_p ball of radius ϵ. In this experiment, we wanted

[3] Pillow 8.3.1. https://pillow.readthedocs.io/en/stable/releasenotes/8.3.1.html

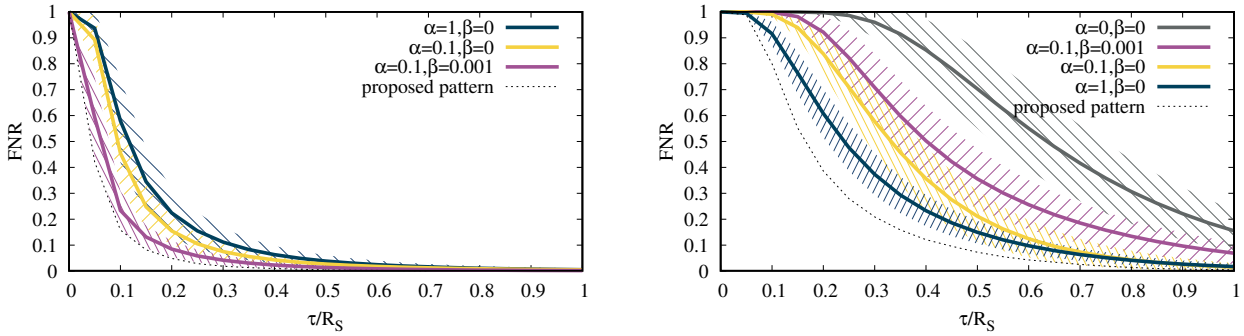

Figure 3: The mean false negative rate (FNR) of revealing 1000 random secret patterns hidden in CIFAR-10 validation JPEG images as a function of the relative threshold with the ℓ_∞ norm (left) and the ℓ_2 norm (right) with various α and β values. The hatched area shows the standard deviation.

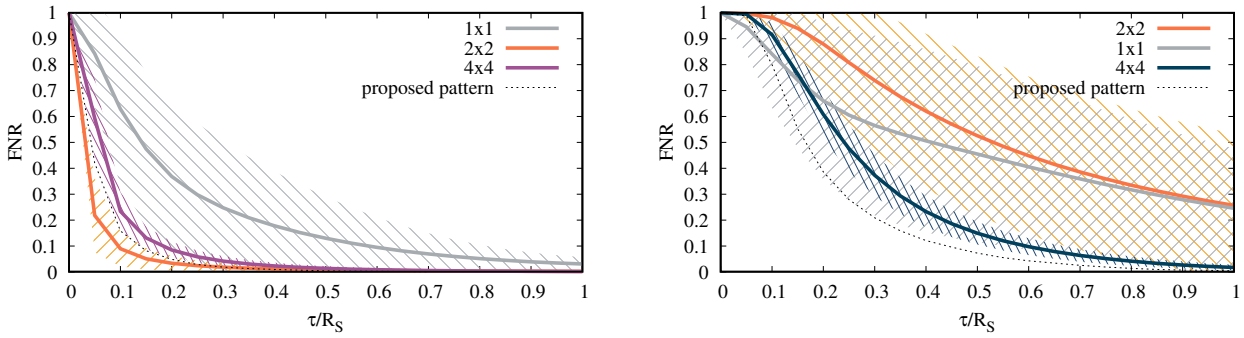

Figure 4: The mean false negative rate (FNR) of revealing 1000 random secret patterns hidden in CIFAR-10 validation JPEG images as a function of the relative threshold with the ℓ_∞ norm (left) and the ℓ_2 norm (right) with various secret image sizes (M). The hatched area shows the standard deviation.

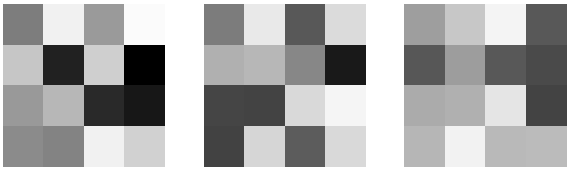

Figure 5: Proposed secret pattern S for ℓ_∞ with $\epsilon_\infty = 8/255$, ℓ_∞ with $\epsilon_\infty = 4/255$ and ℓ_2 with $\epsilon_2 = 0.5$.

to empirically approximate the probability of those perturbations that result in the activation of a backdoor; that is, the probability of revealing a fixed pattern, as a function of the threshold. This empirical investigation is necessary because a theoretical estimate is difficult to provide due to the lack of a formal description of the probability distribution of the revealed secret images. This distribution is rather specific to our model that was trained on $M \times M$ patterns. Assuming a uniform distribution over the possible $N \times N$ images would result in a gross underestimation of the probability in question. At the same time, assuming noise-free patterns (that is, $M \times M$ grayscale patterns scaled up to $N \times N$) as output would result in a gross overestimation.

As mentioned before, we selected an activation threshold reference point R_S (see Equation (2)) such that the false positive rate is zero over the validation set, but the false negative rate is larger than zero. However, since R_S is set over the validation set, it is still necessary to examine the statistics over the test set and see whether we can find false positives below our reference point R_S.

In our experiment, we generated 10^8 "fake" container image samples. In the case of the ℓ_∞ norm, we added a uniformly distributed random number from the range $[-\epsilon_\infty, \epsilon_\infty]$ to each pixel and channel in the cover image. For the ℓ_2 norm, the additive noise was a normally distributed random vector normalized to the length of ϵ_2. For each sample, we calculated the distances between the previously proposed fixed secrets and the patterns revealed from the sample. The distributions of distances (normalized by R_S as measured on the validation set) are shown in Figure 6. Apart from the patterns with the lowest FNR (given in Figure 5) the patterns with the highest FNR are also shown, labeled as the least recommended pattern. From the plots, we see that we have lots of hits below the reference point R_S on the test set. However, the proposed $R_S/2$

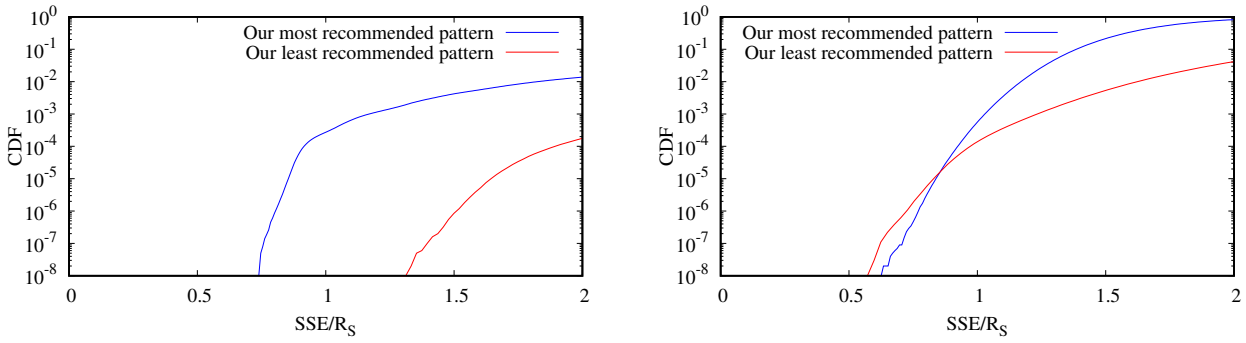

Figure 6: The cumulative distribution function of the distances between the proposed secret and the patterns revealed from CIFAR-10 test images with the ℓ_∞ norm (left) and the ℓ_2 norm (right).

is confirmed to be a safe choice because the empirical probability of triggering the backdoor is less than 10^{-8} (there are no positive examples in the 10^8 samples) and based on the approximate shape of the CDF, the theoretical probability could be orders of magnitude less (note the log scale).

The least recommended pattern appears to perform better, but this is misleading, because the price for this is the very high FNR, which means that the backdoor will often stay inactive even when the backdoor trigger secret is embedded in the container image.

6.2 Adversarial Attacks

Our setup is identical to that of the adversarial perturbation problem. Our backdoor patterns are encoded as perturbations within a very limited radius of the original cover image. Also, instead of fixed perturbations, we have diverse, image specific, invisible patterns. Our infected DNNs are essentially DNNs where adversarial robustness is artificially reduced and our container images are the artificial adversarial examples.

As a consequence, state-of-the-art adversarial attacks on the infected network are suitable tools for trying to reveal the backdoor, especially if the target network is robust; that is, if it has no natural adversarial input perturbations.

We experimented with a number of diverse and state-of-the-art attacks, both gradient-based and gradient-free. The Square Attack [1] is gradient-free, for which the number of queries was set to 5000. As for gradient-based attacks, we examined the untargeted Auto Projected Gradient Descent (APGD) [6] with cross-entropy (CE) and Difference of Logits Ratio (DLR) loss. Also, we included the untargeted Fast Adaptive Boundary Attack (FAB) [5] in our experiments. Here, we used the Auto Attack [6] library to set the parameters of the attacks and to evaluate all the attacks. We ran 5 trials with the gradient-based attacks.

We used robust models as target models, in which we inserted the backdoor. The models we chose are a set of top performer models on RobustBench [4]. These models represent a diverse set of architectures. We selected the best robust models because, this way, most input images only have our hidden backdoor pattern as possible adversarial perturbations (that is, perturbations that mislead the model). We evaluated both the clean models and their backdoor-infected versions. It might be the case that the adversarial attack is successful but the adversarial example belongs to the clean model. For this reason, we also checked to see whether the adversarial image resulting from the attack activates the backdoor. The results are shown in tables 1 and 2.

The results from these tables clearly show that the clean and infected versions of the models behave in exactly the same way when using adversarial attacks in every scenario, and the backdoor is never found. At the same time, when the input image contains the hidden backdoor trigger pattern, the clean model has almost the same accuracy as with the clean images (that is, the backdoor pattern is not adversarial for the clean model) while the backdoor is almost always activated.

This means that the backdoor pattern is strongly adversarial for the infected model, yet the adversarial attacks never find this pattern. Our infected networks thus prove constructively that state-of-the-art adversarial attacks—which are often used as heuristic verifiers for robustness—have an Achilles heel, as they are not able to cope with our needle in the haystack construction.

Another interesting observation is that the images with the backdoor pattern achieved a higher backdoor activation rate when no JPEG compression was applied, which means that we successfully prevented overfitting the JPEG compression.

6.3 Neural Cleanse

Neural Cleanse [24] is a standard backdoor detector not unlike adversarial attacks, but it requires a number of different assumptions. Most importantly, it does not assume that the perturbation is within a ball according to any norm, but it assumes that for every target class there is a fixed pattern that activates that class. This means that this method is not well suited for detecting our backdoor, mostly because in our case every backdoor pattern is image-specific.

Nevertheless, there could still be regularities in our image-specific patterns (see the Appendix for visualizations). To make the setup fairer for Neural Cleanse, we created a modified version of our backdoor that, instead of shifting the labels, produces a constant vector of logit activations (in the penultimate layer) where the neuron corresponding to the target class has the largest activation. We embedded the backdoor into the models used in the first columns of tables 1 and 2 for the ℓ_∞ norm and the ℓ_2 norm, respectively. The

Table 1: Adversarial accuracy of unmodified and backdoor-infected robust models under various adversarial attacks and for our embedded backdoor trigger pattern, in ℓ_∞ norm ($\epsilon_\infty = 8/255$) on the CIFAR-10 test set. The rate of backdoor activation is also shown.

Model	Gowal2021Improving [8]			Rade2021Helper [17]			Rebuffi2021Fixing [18]		
Architecture	WideResNet-28-10 ddpm 100m			WideResNet-34-10 extra			WideResNet-70-16 cutmix extra		
Clean accuracy	0.8750			0.9147			0.9223		
input image perturbation method	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation
Square Attack [1]	0.6881	0.6884	0.0000	0.6930	0.6932	0.0000	0.7376	0.7368	0.0000
FAB [5]	0.6404	0.6404	0.0000	0.6345	0.6349	0.0000	0.6752	0.6748	0.0000
APGD-CE [6]	0.6549	0.6548	0.0000	0.6559	0.6557	0.0000	0.6922	0.6925	0.0000
APGD-DLR [6]	0.6510	0.6508	0.0000	0.6365	0.6366	0.0000	0.6764	0.6765	0.0000
Backdoor Trigger	0.8719	0.0148	0.9992	0.9105	0.0112	0.9992	0.9166	0.0095	0.9992
Backdoor Trigger + JPEG	0.8586	0.0179	0.9972	0.8969	0.0139	0.9972	0.9017	0.0127	0.9972

Table 2: Adversarial accuracy of unmodified and backdoor-infected robust models under various adversarial attacks and for our embedded backdoor trigger pattern, in ℓ_2 norm ($\epsilon_2 = 0.5$) on the CIFAR-10 test set. The rate of backdoor activation is also shown.

Model	Rebuffi2021Fixing [18]			Rade2021Helper [17]			Rebuffi2021Fixing [18]		
Architecture	WideResNet-28-10 cutmix ddpm			PreActResNet-18 ddpm			WideResNet-70-16 cutmix extra		
Clean accuracy	0.9179			0.9057			0.9574		
input image perturbation method	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation
Square Attack [1]	0.8502	0.8503	0.0000	0.8286	0.8296	0.0000	0.8969	0.8972	0.0000
FAB [5]	0.7900	0.7900	0.0000	0.7636	0.7633	0.0000	0.8271	0.8270	0.0000
APGD-CE [6]	0.8000	0.8000	0.0000	0.7695	0.7695	0.0000	0.8319	0.8317	0.0000
APGD-DLR [6]	0.7901	0.7902	0.0000	0.7638	0.7636	0.0000	0.8261	0.8261	0.0000
Backdoor Trigger	0.9173	0.0090	0.9999	0.9047	0.0109	0.9999	0.9568	0.0042	0.9999
Backdoor Trigger + JPEG	0.9064	0.0692	0.9368	0.8970	0.0693	0.9368	0.9471	0.0673	0.9340

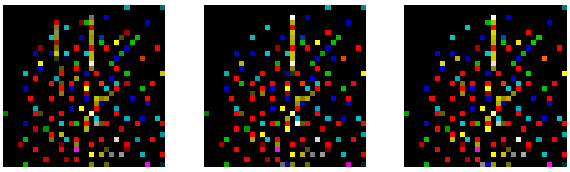

Figure 7: The trigger patterns reverse-engineered by Neural Cleanse with ℓ_∞ ($\epsilon_\infty = 8/255$), from the clean model, the model infected with the constant-label backdoor, and the model infected with the label-shifting backdoor (left to right).

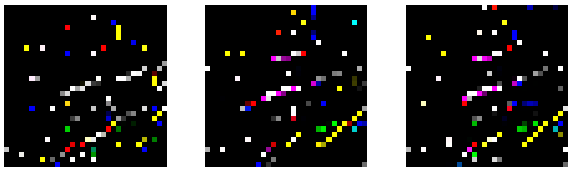

Figure 8: The trigger patterns reverse-engineered by Neural Cleanse with ℓ_2 ($\epsilon_2 = 0.5$), from the clean model, the model infected with the constant-label backdoor, and the model infected with the label-shifting backdoor (left to right).

reverse-engineered trigger patterns returned by Neural Cleanse on the infected model are very similar to those reverse-engineered from the clean model, as figures 7 and 8 indicate. The figures show the patterns for label '7'. Note that these patterns are not very similar to the actual patterns (see the Appendix).

We also present the anomaly indices of the evaluated models in Table 3 (the values are averages of three independent runs). These are all below the threshold level of 2, which means Neural Cleanse was unable to detect the backdoor [24]. Neural Cleanse also fails to detect any target class in any scenario. It was already found

Table 3: Neural Cleanse Anomaly Indices

	ℓ_∞	ℓ_2
Clean model	1.7804	1.0256
Model with constant-label backdoor	1.7675	0.9969
Model with label-shifting backdoor	1.7075	1.0095

that Neural Cleanse is unable to identify the trigger in the case of untargeted attacks which have multiple infected labels with a single trigger [12]. We also found the same with the label-shifting

Table 4: Adversarial accuracy of clean and backdoor-infected robust models under various adversarial attacks and for our embedded backdoor trigger pattern, in ℓ_∞ norm ($\epsilon_\infty = 4/255$) on the CIFAR-10 test set. The rate of backdoor activations is also shown.

Model	Rade2021Helper [17]		
Architecture	WideResNet-34-10 extra		
Clean accuracy	0.9147		
input image perturbation method	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation
Square Attack [1]	0.8254	0.8245	0.0000
FAB [5]	0.8030	0.8029	0.0000
APGD-CE [6]	0.8134	0.8133	0.0000
APGD-DLR [6]	0.8026	0.8027	0.0000
Backdoor Trigger	0.9128	0.0794	0.9262
Backdoor Trigger + JPEG	0.8989	0.1341	0.8675

Table 5: Adversarial accuracy of clean and backdoor-infected robust models under various adversarial attacks and for our embedded backdoor trigger pattern, in ℓ_∞ norm ($\epsilon_\infty = 4/255$) on ImageNet examples. The rate of backdoor activations is also shown.

Robust model	Salman2020Do [19]		
Architecture	ResNet-18		
Clean acc.	0.5103		
input image perturbation method	clean model adv. acc.	infected model adv. acc.	rate of backdoor activation
Square Attack [1]	0.3663	0.3661	0.0000
FAB[4] [5]	0.3279	0.3277	0.0000
APGD-CE [6]	0.2808	0.2806	0.0000
APGD-DLR [6]	0.2622	0.2624	0.0000
Backdoor Trigger	0.5103	0.0348	0.9575
Backdoor Trigger + JPEG	0.5064	0.0560	0.9158

backdoor network. It has also been pointed out, that Neural Cleanse and related methods can only detect backdoors based on the gradient flow [23]. As we mentioned repeatedly, our proposed method has a zero gradient flow for normal inputs due to the discontinuous nature of the infected network. This is the reason why, from the point of view of Neural Cleanse, all the three versions of the model look the same, as figures 7 and 8 and Table 3 also illustrate.

6.4 Using the CIFAR-10 Backdoor Model on ImageNet

In principle, our backdoor network that was trained on CIFAR-10 can be used to infect any image classifier models even if they were trained on another dataset. The images the classifier accepts as input should have a size greater than or equal to that of the CIFAR-10 images (32×32). There are several ways of hiding a 32×32 secret in a larger image. For example, we can designate a 32×32 region in the image and hide the secret there. Or we can even insert multiple non-overlapping instances.

To learn whether the CIFAR-10 backdoor generalizes to other datasets, we evaluate our method over 50000 examples of the ImageNet [7] validation dataset. We hide the backdoor activation pattern in the top left 32×32 region and we input this region to the backdoor network. The most common ϵ for ImageNet is $4/255$ in ℓ_∞. Note that this is a much smaller ℓ_∞ ball than the one we used for CIFAR-10 (where $\epsilon_\infty = 8/255$), so this is a harder problem. Though we could have used $\epsilon_\infty = 8/255$, to be consistent with related studies we first trained an $\epsilon_\infty = 4/255$ backdoor network on CIFAR-10 and then inserted it into an ImageNet model. To train the steganographic components, we used the same hyperparameter values as in the case of $\epsilon_\infty = 8/255$.

Since we are now working with $\epsilon_\infty = 4/255$, first we evaluated this smaller perturbation with our previous adversarial attacks on CIFAR-10. The results are shown in Table 4. The results can be compared with those in Table 1, where the same clean model is evaluated. We notice that the clean model is more robust to the adversarial attacks (the adversarial accuracies are higher), which is due to the smaller ϵ. When we input images with the backdoor secret embedded, we observe a lower rate of backdoor activation. Clearly, when the size of a possible perturbation is more restricted, it becomes more difficult to hide patterns, and at the same time we still require the false positive rate to be zero. This inevitably increases the false negative rate. Nevertheless, the backdoor is activated about 90% of the time.

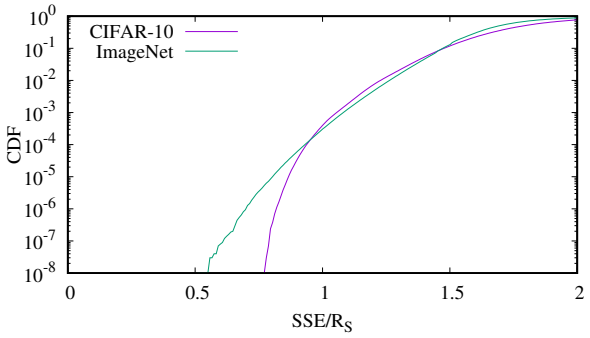

Figure 9: The cumulative distribution function of the distances between the proposed secret and the patterns revealed from perturbed ($\epsilon_\infty = 4/255$) CIFAR-10 and ImageNet images.

Next, we performed the same measurements for ImageNet, as shown in Table 5. It is once again clear that the adversarial attacks are unable to activate the backdoor, and the hidden backdoor trigger pattern activates the backdoor with a high probability. In fact, the backdoor is activated with a slightly higher probability than in the case of CIFAR-10. This could be due to the fact that the ImageNet images are larger and the top left 32×32 region tends to be smoother than CIFAR-10 images.

[4] It was evaluated over 10000 examples of the ImageNet validation dataset.

In addition, we also repeated the random sampling experiment with the usual 10^8 random samples, as shown in Figure 9. The reference point R_S was determined based on the CIFAR-10 validation set. The threshold $\tau = R_S/2$ is still a viable option, even for ImageNet.

7 CONCLUSIONS

In this paper, our goal was to show that adversarial attacks can be completely evaded. In other words, one can construct a neural network that appears to be robust, while in reality it has adversarial perturbations for almost every single input example.

To achieve this goal, we presented a backdoor attack that inserts adversarial examples for each clean input image in such a way that search-based methods are not able to identify these examples. The key to this result is the idea that our artificial adversarial examples are "needles in a haystack", in other words, the gradient over clean examples is completely uninformative, and the volume of adversarial samples is extremely low. Our solution takes into account the definition of the adversarial perturbation problem; that is the perturbations that trigger the backdoor attack are bounded. What is more, they are robust to JPEG compression.

We demonstrated that the method works with CIFAR-10 images, and it can be applied on the ImageNet dataset as well without fine-tuning.

ACKNOWLEDGMENTS

This work is supported by the National Research, Development and Innovation Office, Hungary, under Grant No. TKP2021-NVA-09 and MILAB (Artificial Intelligence National Laboratory Program).

REFERENCES

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. 2019. Square Attack: a query-efficient black-box adversarial attack via random search. *CoRR* abs/1912.00049 (2019). arXiv:1912.00049 http://arxiv.org/abs/1912.00049

[2] Shumeet Baluja. 2017. Hiding Images in Plain Sight: Deep Steganography. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/838e8afb1ca34354ac209f53d90c3a43-Paper.pdf

[3] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. 2020. Branch and Bound for Piecewise Linear Neural Network Verification. *Journal of Machine Learning Research* 21, 42 (2020), 1–39. http://jmlr.org/papers/v21/19-468.html

[4] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. 2020. RobustBench: a standardized adversarial robustness benchmark. *CoRR* abs/2010.09670 (2020). arXiv:2010.09670 https://arxiv.org/abs/2010.09670

[5] Francesco Croce and Matthias Hein. 2019. Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack. *CoRR* abs/1907.02044 (2019). arXiv:1907.02044 http://arxiv.org/abs/1907.02044

[6] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. PMLR, 2206–2216.

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

[8] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. 2021. Improving Robustness using Generated Data. *Advances in Neural Information Processing Systems* 34 (2021).

[9] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015). arXiv:1503.02531 http://arxiv.org/abs/1503.02531

[11] Guy Katz, Clark 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*, Rupak Majumdar and Viktor Kunčak (Eds.). Springer International Publishing, Cham, 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[12] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2020. TrojDRL: Evaluation of Backdoor Attacks on Deep Reinforcement Learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218663

[13] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[14] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. 2021. Invisible Backdoor Attacks on Deep Neural Networks Via Steganography and Regularization. *IEEE Trans. Dependable Secur. Comput.* 18, 5 (2021), 2088–2105. https://doi.org/10.1109/TDSC.2020.3021407

[15] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible Backdoor Attack With Sample-Specific Triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 16463–16472. http://arxiv.org/abs/2012.03816

[16] David J. Miller, Zhen Xiang, and George Kesidis. 2020. Adversarial Learning Targeting Deep Neural Network Classification: A Comprehensive Review of Defenses Against Attacks. *Proc. IEEE* 108, 3 (March 2020), 402–433. https://doi.org/10.1109/JPROC.2020.2970615

[17] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. 2021. Helper-based Adversarial Training: Reducing Excessive Margin to Achieve a Better Accuracy vs. Robustness Trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*. https://openreview.net/forum?id=BuD2LmNaU3a

[18] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. 2021. Fixing Data Augmentation to Improve Adversarial Robustness. *CoRR* abs/2103.01946 (2021). arXiv:2103.01946 https://arxiv.org/abs/2103.01946

[19] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. 2020. Do Adversarially Robust ImageNet Models Transfer Better?. In *ArXiv preprint arXiv:2007.08489*.

[20] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. 2021. On the Robustness of Backdoor-based Watermarking in Deep Neural Networks. In *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*, Dirk Borghys, Patrick Bas, Luisa Verdoliva, Tomás Pevný, Bin Li, and Jennifer Newman (Eds.). ACM, 177–188. https://doi.org/10.1145/3437880.3460401

[21] Richard Shin and Dawn Song. 2017. JPEG-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, Vol. 1.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1312.6199

[23] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. 2020. An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 218–228.

[24] 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)*. 707–723. https://doi.org/10.1109/SP.2019.00031

[25] Cheng-Hsin Weng, Yan-Ting Lee, and Shan-Hung Brandon Wu. 2020. On the trade-off between adversarial and backdoor robustness. *Advances in Neural Information Processing Systems* 33 (2020).

[26] Chaoning Zhang, Philipp Benz, Adil Karjauv, and In So Kweon. 2021. Universal Adversarial Perturbations Through the Lens of Deep Steganography: Towards a Fourier Perspective. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 3296–3304. https://ojs.aaai.org/index.php/AAAI/article/view/16441

[27] Chaoning Zhang, Adil Karjauv, Philipp Benz, and In So Kweon. 2021. Towards Robust Deep Hiding Under Non-Differentiable Distortions for Practical Blind Watermarking. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, Heng Tao Shen, Yueting Zhuang, John R. Smith, Yang Yang, Pablo Cesar, Florian Metze, and Balakrishnan Prabhakaran (Eds.). ACM, 5158–5166. https://doi.org/10.1145/3474085.3475628

[28] Chaoning Zhang, Chenguo Lin, Philipp Benz, Kejiang Chen, Weiming Zhang, and In So Kweon. 2021. A Brief Survey on Deep Learning Based Data Hiding, Steganography and Watermarking. *CoRR* abs/2103.01607 (2021). arXiv:2103.01607 https://arxiv.org/abs/2103.01607

[29] Dániel Zombori, Balázs Bánhelyi, Tibor Csendes, István Megyeri, and Márk Jelasity. 2021. Fooling a Complete Neural Network Verifier. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=4IwieFS44l

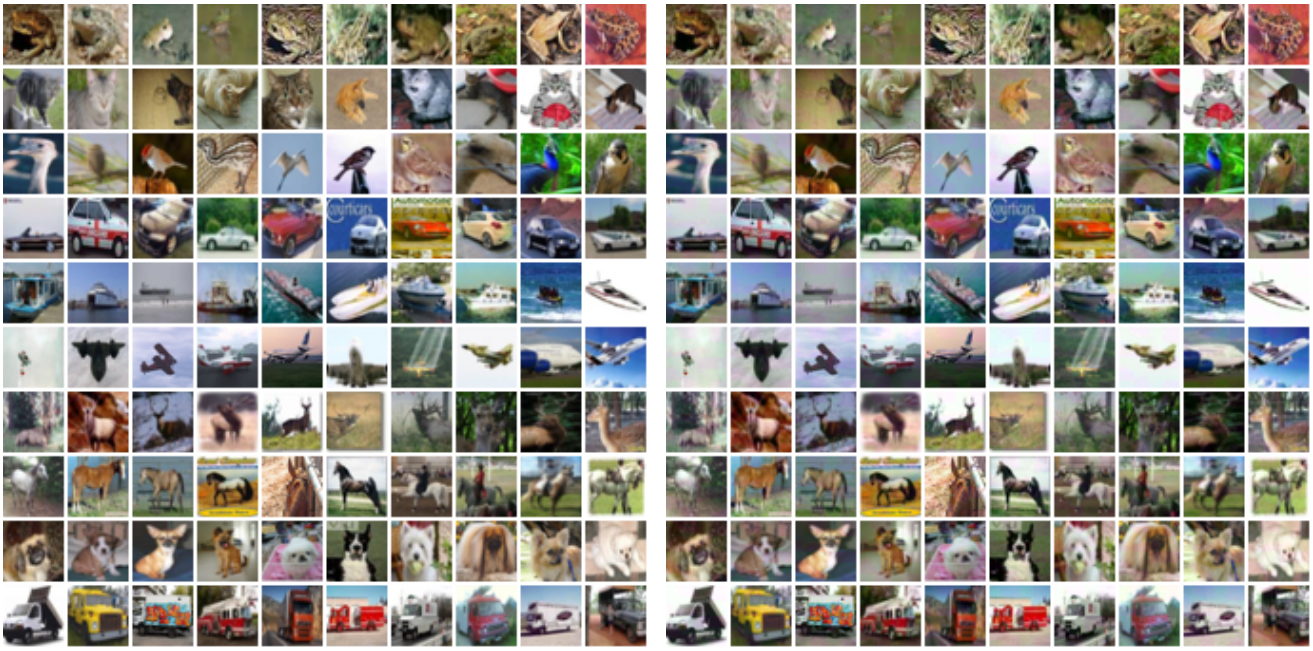

Figure 10: CIFAR-10 original cover images (left) and container images ($\ell_\infty, \epsilon = 4/255$) (right).

Figure 11: ImageNet original cover images (left) and container images with a hidden pattern in the top left 32x32 region ($\ell_\infty, \epsilon = 4/255$) (right).

A VISUALIZATION OF CONTAINER IMAGES

Here, we show the container images that were generated by our secret backdoor patterns. Figure 10 contains a sample of images from the CIFAR-10 dataset in their original form, along with the container images in which our secret backdoor key pattern is hidden. In Figure 11, cover (original) and container image examples are shown from the ImageNet dataset. Here, the secret is hidden in the top left corner. We included images where this corner is relatively homogeneous in the cover image. For both of these figures, we used the ℓ_∞ norm with $\epsilon = 4/255$.

Figures 12 and 13 depict the same container images from CIFAR-10, but this time with a larger perturbation. For the ℓ_∞ norm we

Figure 12: CIFAR-10 container images ($\ell_\infty, \epsilon = 8/255$), and the difference from the original cover image. The differences have been scaled up by a factor of 4 and a 255/2 offset has been added.

Figure 13: CIFAR-10 container images ($\ell_2, \epsilon = 0.5$), and the difference from the original cover image. The differences have been scaled up by a factor of 4 and a 255/2 offset has been added.

used $\epsilon = 8/255$ and for the ℓ_2 norm we used $\epsilon = 0.5$. We also show the differences from the original cover images (that is, the perturbations themselves). These differences are visualized via first scaling them up by a factor of 4 to enhance visibility, and then adding an offset of 255/2 to each channel of each pixel (so pixels where there is no difference are shown in mid-grey).