Persistence of Backdoor-based Watermarks for Neural Networks: A Comprehensive Evaluation

Anh Tu Ngo, Chuan Song Heng, Nandish Chattopadhyay and Anupam Chattopadhyay, *Senior Member, IEEE,*

***Abstract*—Deep Neural Networks (DNNs) have gained considerable traction in recent years due to the unparalleled results they gathered. However, the cost behind training such sophisticated models is resource intensive, resulting in many to consider DNNs to be intellectual property (IP) to model owners. In this era of cloud computing, high-performance DNNs are often deployed all over the internet so that people can access them publicly. As such, DNN watermarking schemes, especially backdoor-based watermarks, have been actively developed in recent years to preserve proprietary rights. (Add that no backdoor watermark guarantee). Nonetheless, there lies much uncertainty on the robustness of existing backdoor watermark schemes, towards both adversarial attacks and unintended means such as fine-tuning neural network models. In this paper, we extensively evaluate the persistence of recent backdoor-based watermarks within neural networks in the scenario of fine-tuning, we propose/develop a novel data-driven idea to restore watermark after fine-tuning without exposing the trigger set. Our empirical results show that by solely introducing training data after fine-tuning, the watermark can be restored if model parameters do not shift dramatically during fine-tuning. Depending on the types of trigger samples used, trigger accuracy can be reinstated to up to 100%. Our study further explores how the restoration process works using loss landscape visualization, as well as the idea of introducing training data in fine-tuning stage to alleviate watermark vanishing.**

***Index Terms*—backdoor watermark, neural network, persistence, privacy, fine-tuning.**

I. Introduction

Recent years have witnessed eminent advancement of Artificial Intelligence (AI) in various aspects of life, ranging from computer vision, natural language processing (NLP) to healthcare. The use of deep neural networks has overshadowed traditional machine learning techniques in those tasks. The phenomenal success of Transformer [1] paved the way to many breakthroughs in language models and even in machine vision. For instance, Transformer-based models such as OpenAI's GPT-3 [2], GPT-4 [3], Google's LaMDA [4] and PaLM 2 [5] have become the dominant large language models (LLMs) and now serve as backbones in ChatGPT and Bard chatbots. Nonetheless, these models usually consist of up to hundreds of billions of parameters and it costs millions of dollars to train them. Aside training cost, the infrastructure, data acquisition and human resource payment can make up colossal expense for the host companies. Such exorbitant cost and enormous effort have made these models valuable intellectual properties (IP) of the companies. Furthermore, with the growth of machine learning as a service (MLaaS) [6], the privacy of machine learning models is exposed to various threats. For example, an authorized model user may illegally distribute the obtained model to unauthorized parties. To that end, sufficient care for model's privacy should be taken into consideration.

One of the effective techniques to guard DNNs from illegal usage is watermarking, in which some special patterns are embedded into the host documents. Watermark has been used for a long time in digital documents like photos, videos, sounds, etc,. In the past decade, researchers have adopted the idea of watermarking to machine learning models, especially DNNs. The first work in DNN watermark is from [7], whose idea is inspired by conventional digital watermark techniques that embeds hidden signature into the model's parameters by modifying the regularizer. More recent DNN watermarking techniques are based on the idea of backdoor, which is first proposed by Adi et al. [8]. The idea is to train the DNNs so that they output specific predictions for a specifically designed dataset.

In real-world use cases, it is common that model owners create their own pre-trained DNNs and distribute them publicly, or to subscribed clients. In those situations, DNNs do not invariably stay static. Instead, model users usually make changes to the DNNs so that they suit better to their particular domain by transfer learning or fine-tuning, in which model's parameters are modified by being trained on newer data. This process challenges the persistence and robustness of backdoor watermarks as the they are embedded in model's parameters. Recent work from [9], [10] aims at removing backdoor watermarks via fine-tuning. They demonstrate that most DNN backdoor watermarks are vulnerable to removal attack like fine-tuning, although [8], [11] claim that fine-tuning is not sufficient to remove backdoor watermarks.

To this end, we focus on evaluating the persistence of existed backdoor watermark schemes against fine-tuning, as well as how to enhance their resilience in the event of fine-tuning. Our contributions can be summarized as:

- We propose a data-driven method utilizing the idea of basin of attraction of local minima. Our experiment shows this method helps regain the watermark accuracy after fine-tuning process, which involves retraining the DNN with the original clean training set without further exposure of trigger samples to the model.
- We analyze the optimization trajectory of model parameters using loss landscape geometry. The analysis illustrates how model parameters traverse the landscape during re-training and how the landscape looks like with respect to particular trigger set types.
- Lastly, we try out with the idea of blending original training data into fine-tuning stage to investigate its

Anh Tu Ngo, Chuan Song Heng, Nandish Chattopadhyay and Anupam Chattopadhyay are with SCSE, Nanyang Technological University Singapore.

effectiveness in reducing watermark vanishing.

This intriguing property can be applied in cases when authorized model users request to perform fine-tuning or incremental training via API access. Model owners, with the proposed fine-tuning technique, can allow the users to do so while still able to retain the presence of watermark without risking the secrecy of trigger samples during that process.

The paper is organized as follow, Section II reviews the background of watermark requirements and related work for DNN watermarking, Section III details the theory behind our proposed methodology, Section IV shows our experiments and results.

II. Background & Related Work

DNN watermarks, though differ in terms of mechanism compared to digital watermarks, need to fulfill some requirements to successfully protect the model's privacy. Below, we discuss the fundamental requirements for DNN watermarks. Then we review the related work on neural network backdoors and some notable research attempts to turn malicious backdoors into privacy guards in neural networks.

A. Requirements for DNN watermarks

The primary difference between conventional digital watermarking and DNN watermarking is that in case of digital documents, the watermark can be injected directly into the host documents. Whereas in the case of DNN, the watermark cannot be directly embedded into the model weights. The watermark embedding process must happen during training. Despite this, both digital watermarking and DNN watermarking have to satisfy a good trade-off between persistence, capacity and fidelity (for DNN watermark) or imperceptibility (for digital watermarking). This trade-off can be viewed as a triangle in Figure 1, which is discussed in [12]. However, one downside of this watermarking scheme is that it requires an explicit inspection of model parameters in order to verify the ownership. Below we briefly review the key requirements for DNN watermarks, these points are also mentioned in [7], [13].

Persistence. This is the ability of a watermark to be retained from the host documents. In the context of DNN watermarking, the presence of watermark footprint should be preserved to a great extent in the event of model manipulations, e.g. fine-tuning, model compression. In other words, this property is the robustness of watermark against model attacks/modifications.

Fidelity. As regards digital watermarking, fidelity is considered equivalent to *imperceptibility*, in which watermarks should not degrade the quality of host documents. In terms of DNN watermarking, a good fidelity means that the watermark does not have much detrimental impact on the model performance on its original task.

Capacity. This is the amount of information that can be embedded into host contents, expressed as the number of bits or payload. Most common DNN watermark schemes are either zero-bit or multi-bit watermarks.

There are a few more requirements that a DNN watermarking scheme should satisfy to be considered of good quality. Table I summarizes the most common criteria to evaluate a DNN watermark scheme.

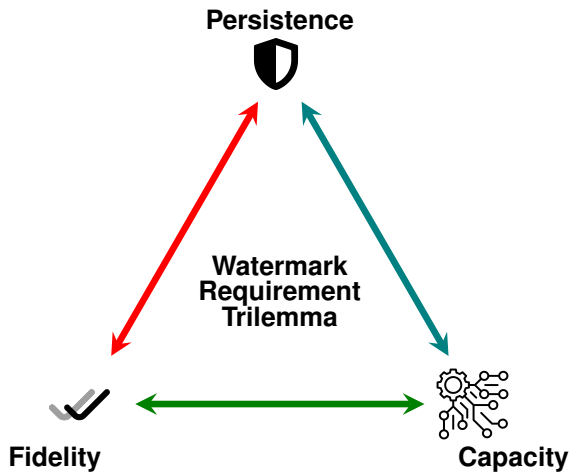

Fig. 1: Trilemma between persistence, capacity and fidelity

TABLE I: Requirements for DNN watermarks

Criterion	Description
Persistence	The watermark should resist various attacks and model modifications
Capacity	The capability of embedding large amount of information into host neural network
Fidelity	The watermark should not significantly affect the performance of target NN on original task
Integrity	The false alarm rate (or number of false positives) should be minimal
Security	The presence of watermark should be secret and undetectable
Efficiency	The computational overhead of watermark construction and verification should be negligible
Generality	The watermark technique can be adaptive to various models, datasets and learning tasks

B. Related Work

The first work on protecting the IP of neural networks using watermark was proposed by Uchida et al. [7]. In this work, the secret key is a specially designed vector X with T-bit length. The watermark embedding happens during model training and is done by adding an *embedding regularizer* term to the original loss function. This can be written as $E(w) = E_0(w) + \lambda E_R(w)$ where $E_0(w)$ is the loss for original task and $E_R(w)$ is the additional embedding regularizer imposing a statistical bias on the parameters w. To extract and verify the watermark, model owners simply have to compute the project of w onto X, which indicates the presence of watermark by comparing with a pre-defined threshold.

Backdoor in neural networks. After the work in [7], many researchers have been actively tackling DNN watermaking problem in various ways. A prominent technique to embed a watermark into neural networks is backdooring. According to [14], backdoors in neural networks corresponds to the process of training a neural network in such a way that it outputs wrong labels for certain input samples and is regarded as one kind of *data poisoning*. The power of modern DNN models stems from their over-parameterization, that is, the

number of model parameters is much more than the number of training samples so that the models have more capability to solve their original task. However, this characteristic paves the way for backdooring, hence a security weakness in DNN models.

Traditionally, backdoors are considered undesirable to AI security. Nevertheless, Adi et al. [8] turned this "badness" into a "privacy guard". The idea is straightforward since the training method is similar to usual neural network training, except that the backdoor trigger data is introduced to the model during training process. In this work, trigger data is a set of abstract images which are completely unrelated to clean samples, and are mislabeled with random classes. The watermark embedding process happens either during fine-tuning or training from scratch via data poisoning. Zhang et al. [15] proposed a similar watermark scheme but with different watermark generation techniques, i.e. embedded content, unrelated data and pre-specified noise. In the first approach, the trigger images are sampled from the original training data and overlaid with a special string. The second approach involves using images from different domains than original training data. In the final approach, Gaussian noise is added to the images to generate trigger data. Rouhani et al. [13] proposed DeepSigns framework, which introduces a hybrid method to embed NN watermark. The framework has two steps, first, it embed a N-bit string b into intermediate layers of target neural network. The loss function is modified as $\mathcal{L} = OrigLoss + \lambda_1.loss_1 + \lambda_2.loss_2$, where the additive $loss_1$ and $loss_2$ can be viewed as regularizers enforcing the hidden layers' activation to fit better to a Gaussian distribution and embedding the watermark string b via projection, which is pretty similar to that of [7]. In the second step, DeepSigns watermarks the network's output layer by selecting watermark keys, or input samples, whose activation lies in the *rarely explored* area of intermediate layers. In other words, the neural network produces incorrect predictions for these samples. The target network is then fine-tuned with these samples so that the final watermarked model classifies the samples correctly. This step is similar to most backdoor-based watermark schemes, except for the type of trigger set.

A few studies make use of adversarial examples to generate trigger data. Frontier Stitching [16] is the first watermark scheme to leverage adversarial examples. In this algorithm, the trigger data contains *true adversaries*, which are perturbed samples that fool the model into outputting wrong predictions. It also includes *false adversaries*, where adversarial noises are applied to the original samples to the level that they do not affect the classification results. Both types of adversarial samples are generated in such a way that they lie close to the decision boundaries, or frontiers. This condition ensures the decision frontiers of watermarked classifier do not deviate significantly from the non-watermarked one's. ROWBACK scheme [17] also adopts adversarial example for trigger set generation, but differs in the labeling procedure. The algorithm uses FGSM [18] to create adversarial samples, then each sample's label is chosen to be different from its ground-truth label and predicted label. Another novel contribution of ROWBACK is uniform watermark distribution. In other words, at every iteration, only a specific set of layers are trained while other parts of the model are frozen. Which set of layers are trained alternates by iteration. According to the work, this ensures the watermark footprint is evenly distributed across the whole neural network's parameters.

Most of the trigger-based watermark schemes claim their robustness to removal attacks with little theoretical guarantee. Recently, Bansal et al. [19] proposed a certifiable trigger-based watermark that utilizes random noises to enhance robustness. The training algorithm employs a two-staged process every epoch: 1) model is trained normally on training set X. 2) k noised copies of model's parameters θ are samples $\{\theta + \Delta_i | i = 1, \cdots, k \wedge \Delta_i \sim \mathcal{N}(0, \sigma^2)\}$ where Δ_i are the added Gaussian noises. These cloned models are trained on trigger data $X_{trigger}$ then their gradients get averaged and accumulated to the original parameters for updating. The paper theoretically shows that certification guarantees watermark's robustness within a l_2-norm ball of parameters modification.

III. Local Minima and Watermark Restoration

In this work, we focus on fine-tuning watermarked neural networks. A model owner trains their model f_θ, where $\theta \in \mathbb{R}^N$ is model's parameters, on clean dataset $\mathcal{D}_{train}$ to perform a specific classification task $\mathcal{T}$. To verify model ownership, a set of trigger samples $\mathcal{D}_{trigger}$ are embedded into f_θ during training and extracted during verification process. To achieve this goal, the optimization process tries to solve:

$$\theta^* = \arg\min_\theta \mathcal{L}(y, f_\theta(x)) \tag{1}$$

where $x \in \mathcal{D}_{train} \cup \mathcal{D}_{trigger}$. Here, we assume the watermark is embedded during pre-training phase and samples from $\mathcal{D}_{train}$ and $\mathcal{D}_{trigger}$ significantly differ in probability distribution, which is either $x_{train} \overset{d}{\neq} x_{trigger}$ or $y_{train} \overset{d}{\neq} y_{trigger}$. This optimization ensures that a basic watermark scheme must at least satisfy two crucial properties: 1) functional preserving: $Pr(y_{train} = f_\theta(x_{train})) \approx Pr(y_{train} = f'(x_{train}))$ and 2) verifiability: $Pr(y_{trigger} = f_\theta(x_{trigger})) \gg Pr(y_{trigger} = f_\theta(x_{trigger}))$.

Fine-tuning and watermark removal. Work from Goodfellow et al. [20] empirically shows watermark footprint gets eroded when an adversary further fine-tunes the marked model with their custom data $\mathcal{D}_{finetune}$. After fine-tuning, watermark footprint can be successfully removed from model since the fine-tuning data does not include trigger samples. This phenomenon is also known as catastrophic forgetting.

Watermark restoration. To achieve objective as Eq. 1, model parameters θ must converge to the local minimas that allow f_θ to give good performance on both $\mathcal{D}_{train}$ and $\mathcal{D}_{trigger}$. During fine-tuning stage, the parameters shift to locations farther away from the local optimum. It is expected that the new minima still result in good classification outcomes on $\mathcal{D}_{train}$. Nonetheless, model will suffer to maintain a good trigger accuracy. It is because $\mathcal{D}_{finetune}$ and $\mathcal{D}_{train}$ are from the same domain, whereas $\mathcal{D}_{trigger}$ is often sampled from a very different domain or probability distribution. How far θ shift during fine-tuning depends on a variety of aspects, e.g.,

learning rate, weight decay, optimizer type, difference level between $\mathcal{D}_{trigger}$ and $\mathcal{D}_{train}$, etc,. From loss surface geometry viewpoint, if a fine-tuning attack does not completely move the parameters out of the basins of attraction, there is high chance that the model owner can shift the parameters towards previous local minimas again, without re-training the model with trigger data. This can simply be done by introducing original $\mathcal{D}_{train}$ into re-training phase. The intuition behind this is when θ are still trapped in the previous basins optimized for $\mathcal{D}_{train} \cup \mathcal{D}_{trigger}$, further re-training f_θ solely on $\mathcal{D}_{train}$ allows θ to follow the steepest descent to converge towards these local minima, given appropriate hyper-parameters applied. Watermark restoration is useful to claim ownership of a suspected model after it is fine-tuned by an adversary.

Fine-tuning with less watermark degradation. Following the intriguing property of local minima, we propose a technique that alleviates watermark removal from fine-tuning. This is a useful concept, especially in scenarios where model owner distributes their watermarked model to authorized clients via API access but still allows the clients to fine-tune the model further with their own data to suit their needs without involving trigger samples in that phase. In each iteration, we mix a batch of $x_{finetune}$ with a batch of training samples x_{train}. This ensures a balance between incorporating new knowledge and retaining the watermark without further exposing $\mathcal{D}_{trigger}$ to f_θ. Our fine-tuning strategy is detailed in Algorithm 1.

Algorithm 1: Fine-tuning strategy

Data: watermarked model f_θ, training samples $\mathcal{D}_{train}$ with batch size B_t, fine-tuning samples $\mathcal{D}_{finetune}$ with batch size B_f, number of epochs N

// $\mathcal{D}_{train}$, $\mathcal{D}_{finetune}$ are sampled randomly

$NumBatch_f = length(\mathcal{D}_{finetune})/B_f$;

for $epoch \leftarrow 1$ **to** N **do**

  for $i \leftarrow 1$ **to** $NumBatch_f$ **do**

    $X_t, y_t \leftarrow \mathcal{D}_{train}[i : i + B_t]$;

    $X_f, y_f \leftarrow \mathcal{D}_{finetune}[i : i + B_f]$;

    $X = \text{CONCATENATE}(X_f, X_t)$;

    $y = \text{CONCATENATE}(y_f, y_t)$;

    $\text{TRAIN}(f_\theta, X, y)$;

  end

end

Output: f_θ

IV. Experiments and Results

We conducted extensive experiments, first to evaluate the resistance of the three watermark schemes [8], [17], [19] to fine-tuning. In our second experiment, we investigated whether the watermark can be restored, by re-training the neural networks solely with the original training data from the first training phase. We found that the original training data, interestingly, helps bring the watermark back without the presence of trigger data in re-training phase. This intriguing results lead us to the third experiment, in which the original training data are mixed with fine-tuning data, to mitigate the erosion of watermarks.

Trigger type	Label
Unrelated Embedded text Random noise	All examples are assigned the same fixed label
Adversarial	Each example is assigned a random label different from its ground-truth label and adversarial label

TABLE II: Labeling schemes for trigger data

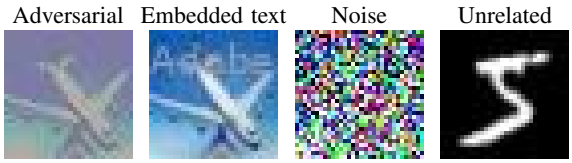

Fig. 2: Types of trigger samples

A. Experimental Design

We use CIFAR-10 dataset [21] consisting of 50K training images across 10 object classes. 100 random samples from original training data are chosen to generate trigger set, whose generation process is detailed in the later part. In the remaining subset of original training set, half are used for model pre-training and the remaining half are used for fine-tuning. We denote the subsets for pre-training, watermark embedding and fine-tuning $\mathcal{D}_{train}, \mathcal{D}_{finetune}$ and $\mathcal{D}_{trigger}$, respectively. A batch size of 256 is used for $\mathcal{D}_{train}$, $\mathcal{D}_{finetune}$ and batch size of 64 is used for $\mathcal{D}_{trigger}$.

Trigger samples generation. There are many ways to generate trigger set. Zhang et al. [15] proposed three techniques such as adding random noise, embedding contents (logo, text, etc,.) or using out-of-distribution samples for trigger data. In this study, we experiment with all these methods as well as the one proposed in [17], where trigger samples are adversarial images created from clean training data. Table II details the labeling scheme for each trigger set type. Figure 2 visualizes examples of trigger types for object *airplane*.

CNN model. We pre-train ResNet-18 model with $\mathcal{D}_{train}$ and embed the watermark into it using the trigger set $\mathcal{D}_{trigger}$. During training, $\mathcal{D}_{train}$ is poisoned with trigger samples and the model is trained with this mixed data. In terms of training mechanism, three different training techniques are employed to embed the watermark as proposed by Zhang et al. [15], N. Chattopadhyay & A. Chattopadhyay [17] and Bansal et al. [19]. We denote these watermark embedding techniques as $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ respectively. This results in 12 pre-trained models in total because for every watermark technique we experiment with 4 trigger set types as mentioned. In our study, the models are trained using Adam optimizer with learning rate descending every 30 epochs. In particular, models with $\mathcal{M}_2$ are trained for more epochs than the others as the pre-training method in [17] trains solely one layer at a time, leading to slower convergence. Additionally, scheme $\mathcal{M}_3$ has much longer training time per epoch than $\mathcal{M}_1$ and $\mathcal{M}_2$. This is because $\mathcal{M}_3$'s training procedure requires calculating mean gradient of many noised copies of the original parameters. The details for hyper-parameters are in Appendix A.

TABLE III: Models performance after initial training phase

Pre-train method	Trigger type	Test acc (%)	Trigger acc (%)	Smoothed trig. acc (%)
$\mathcal{M}_1$ (Adi et al. [8])	Unrelated	85.87	100.00	0.00
	Embedded text	85.92	100.00	0.00
	Noise	85.94	100.00	0.0
	Adversarial samples	86.41	100.00	8.68
$\mathcal{M}_2$ (ROWBACK [17])	Unrelated	85.32	100.00	0.00
	Embedded text	85.21	100.00	0.00
	Noise	84.97	100.00	0.00
	Adversarial samples	84.94	100.00	9.64
$\mathcal{M}_3$ (Certified watermark [19])	Unrelated	85.65	100.00	100.00
	Embedded text	85.44	99.00	100.00
	Noise	86.15	100.00	100.00
	Adversarial samples	86.00	100.00	9.81

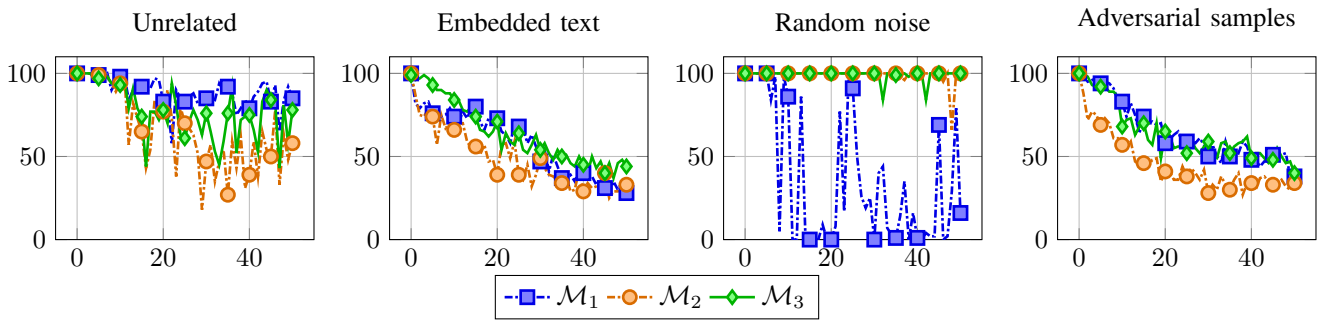

Fig. 3: *Trigger accuracy during fine-tuning* - In the cases of unrelated trigger/text-overlaid/adversarial trigger data, the trigger accuracies decrease more in $\mathcal{M}_2$ in comparison with $\mathcal{M}_1$ and $\mathcal{M}_3$. For noised trigger set, the trigger accuracy fluctuates significantly between 0% and 100% for $\mathcal{M}_1$, whereas it only slightly fluctuates and stays at 100% most of the time in case of $\mathcal{M}_2$ and $\mathcal{M}_3$.

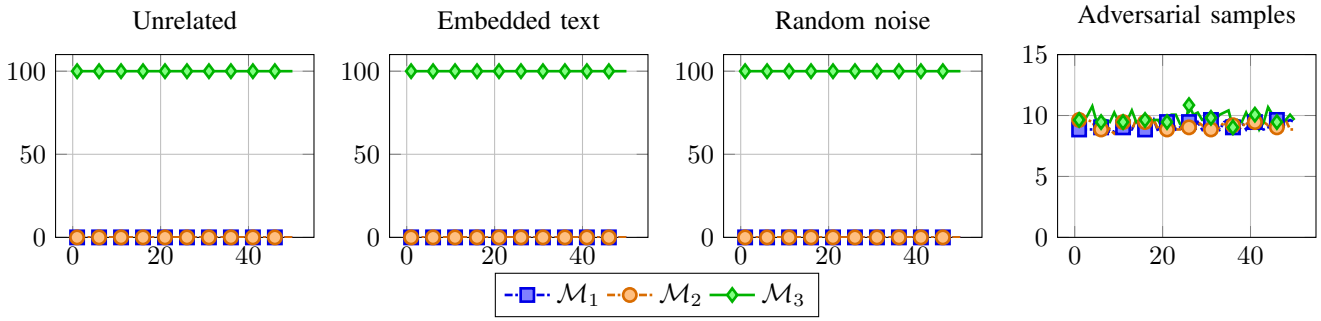

Fig. 4: *Median smoothed trigger accuracy during fine-tuning* - Due to being trained with noises, certified watermark scheme $\mathcal{M}_3$ consistently maintain its trigger accuracy at 100% for unrelated/text-overlaid/noised trigger set. Schemes $\mathcal{M}_1$ and $\mathcal{M}_2$, meanwhile, perform poorly with smoothed accuracies stay at 0%. However, when it comes to adversarial trigger samples with random labels, the smoothed accuracies for all schemes are similar, oscillating at 10%.

Model performance. A popular metric to measure the existence of watermark is trigger set accuracy. In [19], a median smoothed version of trigger set accuracy is used for ownership verification in order to bound the worst case change in trigger set accuracy, given l_2 bounded changes in model's parameters. To measure smoothed trigger set accuracy, we create $K = 100$ copies of the original model, add random Gaussian noises to their weights and evaluate the performance of noised models on trigger set, then we take the median value among K accuracy values obtained. Table III illustrates test accuracy, trigger accuracy, smoothed trigger accuracy of the CNN models after the initial training phase. It is obvious that models with certified watermark scheme $\mathcal{M}_3$ assures the highest lower bound for trigger set accuracy. However, this does not hold true when using adversarial samples with random labels for trigger data. Our results also show that models with $\mathcal{M}_2$ takes many more epochs to converge due to its per-layer training mechanism.

B. Empirical Analysis

We conduct various experiments to assess the watermark persistence of the schemes from [8], [17], [19]. In our first

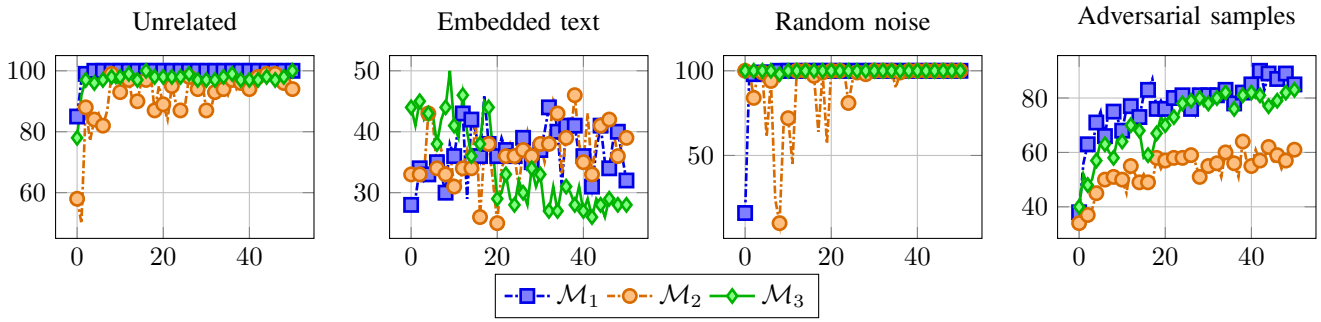

Fig. 5: *Trigger accuracy after re-train models with original training set* $\mathcal{D}_{train}$ - Regarding unrelated and noised trigger samples, trigger accuracy can be reinstated to up to 100%. For adversarial trigger samples, the trigger accuracies of $\mathcal{M}_1$ and $\mathcal{M}_3$ are restored to more than 80% after 50 epochs, while $\mathcal{M}_2$'s accuracy is only brought back to around 60%. The trigger accuracies for text-overlaid trigger samples struggle with restoration, fluctuating below 50%.

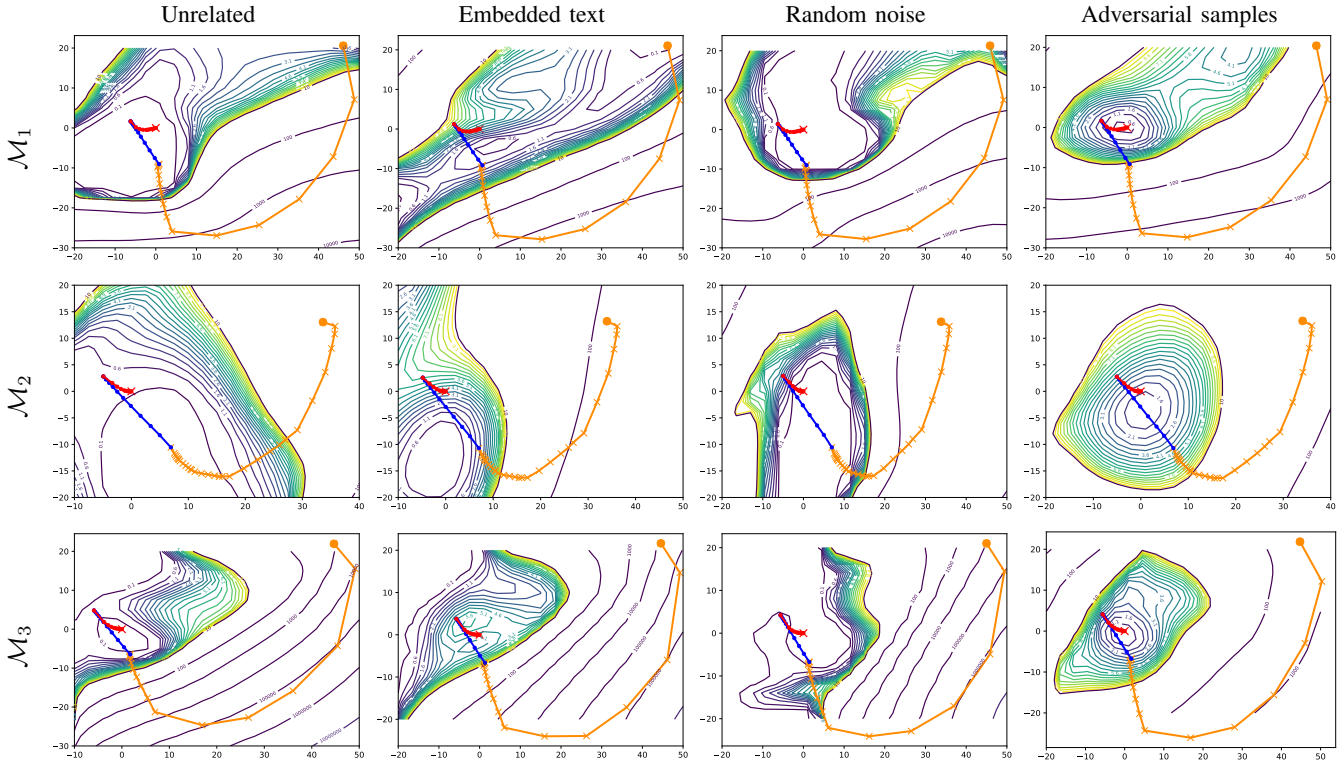

Fig. 6: *Loss landscape visualization* - The contours illustrate trigger loss, *Orange* lines are the initial training and watermark embedding phase, *blue* lines are fine-tuning phase and *red* lines are re-training phase. It is observable that re-training helps steering the trajectory back to near the local minimas.

experiment, we simply measure the trigger accuracy after model fine-tuning. Then, we empirically validate the concept of watermark reinstatement by re-training the DNNs with the original training set $\mathcal{D}_{train}$. Furthermore, we use loss landscape analysis to investigate the optimization trajectory of model's parameters in such scenarios. Lastly, our final experiment is about incorporating the original training data into fine-tuning phase, in order to examine its effectiveness in alleviating watermark degradation.

1) Fine-tuning: We measure watermark removal level after fine-tuning. This involves training the model with extended data to incorporate broader knowledge into the model. From the adversary's point of view, fine-tuning equals to a removal attack by gradually training the model with new data. After having ResNet-18 models pre-trained and watermarked, we fine-tune the models with $\mathcal{D}_{finetune}$. All networks are trained for 50 epochs with Adam optimizer and learning rate of 0.001, then, we evaluate the level of diminishing in trigger accuracy and smoothed trigger accuracy. From Figure 3, the performance of $\mathcal{M}_2$ models are worse than the others when using unrelated and adversarial samples for trigger set. For text-overlaid trigger samples, the performances are on par among 3 watermark techniques. However, trigger accuracy fluctuates drastically for $\mathcal{M}_1$ models with noise trigger samples. We also

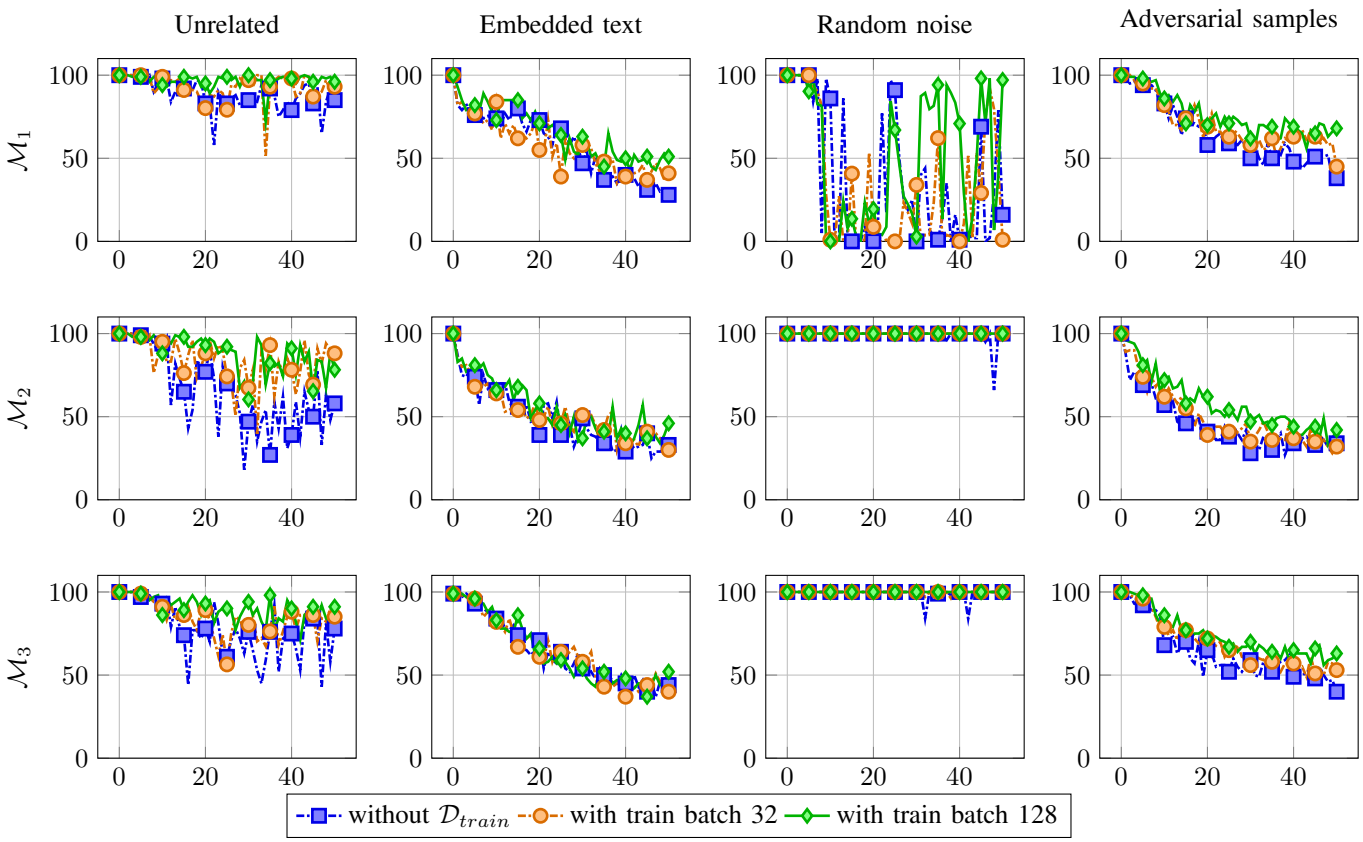

Fig. 7: *Trigger accuracy when mixing $\mathcal{D}_{train}$ during fine-tuning* - When mixing $\mathcal{D}_{finetune}$ with $\mathcal{D}_{train}$, the restoration of trigger accuracy improves in case of unrelated and adversarial trigger samples, only by a small margin. However, the improvement is not clear in the case of text-overlaid and noised trigger samples.

measure the changes in terms of smoothed trigger accuracy. From Figure 4, $\mathcal{M}_3$ models clearly shows their robustness against $\mathcal{M}_1$ and $\mathcal{M}_2$. When it comes to 3 trigger types unrelated, text-overlaid and noised samples, $\mathcal{M}_3$ is able to retain intact trigger accuracy over 50 epochs. However, in the case of adversarial trigger samples, three watermark schemes give comparable accuracies.

2) Re-training for watermark restoration: Here we do empirical study on restoring the watermark after it gets eroded after incremental training / fine-tuning attack, without introducing the trigger set to the model again. This idea will be useful in real-world scenario when the model owner distribute the watermarked model to authorized clients. If the clients request to fine-tune with their own data but still retain watermark, model owner does not need to involve trigger data in the fine-tuning process, which mitigates the risk of leaked secret key.

After fine-tuning and have the watermark degraded, we re-train the model with the initial training set $\mathcal{D}_{train}$. The models are trained with for 50 epochs with learning rate of 0.01. Figure 5 illustrates how each watermark scheme restores by using original training data. It can be seen that for unrelated, noised and adversarial trigger samples, re-training with $\mathcal{D}_{train}$ helps regain watermark footprint for all three schemes. The restoring trends for $\mathcal{M}_1$ and $\mathcal{M}_3$ are more consistent than $\mathcal{M}_1$. However, in terms of trigger samples with embedded text, the accuracy values fluctuate and struggle to reinstate.

Loss landscape analysis. To further explore how re-training with training data affects the watermark, we visualize the loss landscape with learning trajectory. In this study, we use filter normalization method for loss landscape visualization and PCA for optimization trajectory plotting, as proposed by Li et al. [22]. Figure 6 shows 2D contours for trigger loss along the approximate learning trajectories of model's parameters. It is obvious that during the re-training stage (red lines), the learning trajectories turn sharply, and most of the time, move towards the contour lines with smaller loss values. In cases of $\mathcal{M}_2$ and $\mathcal{M}_3$, the re-training trajectories turn sharply backward like an "unlearning" process. From our inspection, the PCA projection for final point in the trajectory correctly reflects its corresponding trigger loss from the contours. However, the approximation for points in earlier epochs does not completely match their actual trigger loss. Nonetheless, the PCA approximation is still very useful to study about the optimization trend.

Contour plots also give us interesting insights about how loss surface looks like with respect to trigger set type. According to the plots in Figure 6, adversarial trigger samples yield smoother, more convex surfaces than other trigger set types. Text-overlaid trigger set tends to produce more chaotic landscapes near the local minimas, resulting to difficulty in watermark restoration as shown in Figure 5. For noised trigger

samples, the contours near local minimas are very close to each other, increasing chances for oscillations between high and low accuracies, as visualized in Figure 3.

3) Fine-tuning while retaining watermark: As the intriguing property of local minima allows the watermark to reappear after being vanished from fine-tuning, we study the idea of feeding training data to the model during fine-tuning process to alleviate watermark vanishing. Our implementation is based on Algorithm 1, where in each epoch, a random batch of training data $\mathcal{D}_{train}$ is mixed with a batch of fine-tuning data $\mathcal{D}_{finetune}$. Intuitively, mixing a portion of training data helps guide the parameters not to shift dramatically from the pre-trained local minima, provided that the distribution of $\mathcal{D}_{finetune}$ does not hugely differ from $\mathcal{D}_{train}$.

We experimented with different batch sizes for training samples to study whether a bigger batch of training samples mitigate watermark vanishing more effectively. While the fine-tuning data batch size is kept the same at 256, we test with cases when training data batch size is 32 and 128. The batch size for $\mathcal{D}_{train}$ is chosen to be smaller than $\mathcal{D}_{finetune}$'s so that it does not affect the fine-tuning phase significantly. As shown in Figure 7, the decrease in watermark degradation when mixing with $\mathcal{D}_{train}$ can be seen in case of unrelated trigger samples. For adversarial trigger samples, there is a decrease in vanishing but not dramatic. In terms of text-overlaid and noised trigger samples, the watermark degradation level is hardly observed, as the trigger accuracies either go down at the same rate (text-overlaid trigger samples) or fluctuate between very high and very low (noised trigger samples). Hence, this idea of data mixing during fine-tuning depends heavily on the type of trigger data.

C. Key Findings

Here, we summarize the key observations from our experiments:

- **Watermark persistence during fine-tuning** - the trigger accuracies of $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ all decrease during fine-tuning. We found that no watermark scheme among these three is consistently superior to the others in terms of trigger accuracy across various trigger set types. However, as regards smoothed trigger accuracy, $\mathcal{M}_3$ completely outperforms its competitors in most cases. This comes at a cost of increasing training time per epoch.
- **Re-training with original training data saves watermark** - if model parameters do not drastically shift from their local minimas after fine-tuning, there is high chance that re-training model with $\mathcal{D}_{train}$ brings the watermark back. The trigger accuracy can be restored to up to 100%.
- **Types of trigger samples and how they are labeled affect loss surface geometry and watermark restoration** - our loss landscape visualizations demonstrate that different types of trigger samples, as well as their labeling scheme, define the shape of loss landscape for trigger set. Thus, it has an impact on the convergence to local minimas in re-training stage.
- **Incorporating original training data in fine-tuning only slightly improves watermark erosion** - our empirical outcomes show a slight improvement in reducing watermark erosion. However, in some cases, this improvement is still marginal or even negligible.

V. Conclusion

(Put the contribution in front) In this work, we explore a new perspective of trigger-based watermark in terms of persistence. Our empirical study shows that training data is useful to restore watermark footprint which was diminished during fine-tuning, provided that appropriate trigger set types are used and the model parameters do not dramatically shift out of their basin of attraction. Beside quantitative evaluation, we visually demonstrate the optimization trend of model parameters via loss landscape geometry. It can be found from our study that by exposing the model to training data after fine-tuning, the learning trajectory move back to the local optimum that yields high trigger accuracy, depending on the trigger set type. In our final experiment, we mix the training data with fine-tuning data to mitigate the effect of watermark vanishing during fine-tuning. However, our experimental results show minimal improvement over normal fine-tuning.

(Something clearer) We are hopeful that this study contributes a new research direction to enhance the robustness and persistence of DNN watermarks. This could facilitate a more data-centric approach when it comes to protect the privacy of machine learning models. This work also serves as an open problem for future work that dives deep into the theoretical aspect of optimization to enhance the persistence and robustness of neural network watermarks.

Appendix A
Hyper-parameters for model training

Stage	Scheme	LR	Epochs	Decay	Dec. Steps
Pre-train	$\mathcal{M}_1$	0.001	70	0.2	30
	$\mathcal{M}_2$	0.001	150	0.5	30
	$\mathcal{M}_3$	0.001	70	0.2	30
Fine-tune	$\mathcal{M}_1$	0.0001	50	0.5	30
	$\mathcal{M}_2$	0.0001	50	0.5	30
	$\mathcal{M}_3$	0.0001	50	0.5	30
Re-train	$\mathcal{M}_1$	0.0001	50	0.2	30
	$\mathcal{M}_2$	0.0001	50	0.2	30
	$\mathcal{M}_3$	0.0001	50	0.2	30

TABLE IV: Hyper-parameters for each training stage

This section details the hyper-parameters that we use in our experiments. Table IV lists our the hyper-parameters for each training stage.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," 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., 2017.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[3] OpenAI, "Gpt-4 technical report," 2023.

[4] R. Thoppilan *et al.*, "Lamda: Language models for dialog applications," *CoRR*, vol. abs/2201.08239, 2022. [Online]. Available: https://arxiv.org/abs/2201.08239

[5] R. Anil *et al.*, "Palm 2 technical report," 2023.

[6] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 896–902.

[7] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 269–277. [Online]. Available: https://doi.org/10.1145/3078971.3078974

[8] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi

[9] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&MMSec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 177–188. [Online]. Available: https://doi.org/10.1145/3437880.3460401

[10] X. Chen, W. Wang, C. Bender, Y. Ding, R. Jia, B. Li, and D. Song, "Refit: A unified watermark removal framework for deep learning systems with limited data," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 321–335. [Online]. Available: https://doi.org/10.1145/3433210.3453079

[11] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses*, M. Bailey, T. Holz, M. Stamatogiannakis, and S. Ioannidis, Eds. Cham: Springer International Publishing, 2018, pp. 273–294.

[12] Y. Li, H. Wang, and M. Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122101095X

[13] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 485–497. [Online]. Available: https://doi.org/10.1145/3297858.3304051

[14] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR*, vol. abs/1708.06733, 2017. [Online]. Available: http://arxiv.org/abs/1708.06733

[15] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 159–172. [Online]. Available: https://doi.org/10.1145/3196494.3196550

[16] E. Le Merrer, P. Pérez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, no. 13, p. 9233–9244, Aug. 2019. [Online]. Available: http://dx.doi.org/10.1007/s00521-019-04434-z

[17] N. Chattopadhyay and A. Chattopadhyay, "Rowback: Robust watermarking for neural networks using backdoors," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 1728–1735.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[19] A. Bansal, P.-Y. Chiang, M. J. Curry, R. Jain, C. Wigington, V. Manjunatha, J. P. Dickerson, and T. Goldstein, "Certified neural network watermarks with randomized smoothing," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 1450–1465. [Online]. Available: https://proceedings.mlr.press/v162/bansal22a.html

[20] I. J. Goodfellow, M. Mirza, X. Da, A. C. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgeting in gradient-based neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6211

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[22] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6391–6401.