REFIT: a Unified Watermark Removal Framework for Deep Learning Systems with Limited Data

Xinyun Chen[†‡] Wenxiao Wang[†§] Chris Bender[‡] Yiming Ding[‡] Ruoxi Jia[‡] Bo Li[¶] Dawn Song[‡]

[‡] UC Berkeley [§] Tsinghua University [¶] UIUC [†] The first two authors contribute equally.

***Abstract*—Deep neural networks (DNNs) have achieved tremendous success in various fields; however, training these models from scratch could be computationally expensive and requires a lot of training data. Recent work has explored different watermarking techniques to protect the pre-trained deep neural networks from potential copyright infringements. Although several existing techniques could effectively embed such watermarks into the DNNs, they could be vulnerable to adversaries who aim at removing the watermarks.**

In this work, we propose REFIT, a unified watermark removal framework based on fine-tuning, which does not rely on the knowledge of the watermarks and even the watermarking schemes. Firstly, contrary to previous work suggesting that fine-tuning based approaches are not effective at removing the watermarks, we demonstrate that by properly designing the learning rate schedule for fine-tuning, an adversary is always able to remove the watermarks.

Furthermore, we conduct a comprehensive study of a realistic attack scenario where the adversary has limited training data, which has not been emphasized in prior work on attacks against watermarking schemes. To effectively remove the watermarks without compromising the model functionality under this weak threat model, we propose two techniques that are incorporated into our fine-tuning framework: (1) an adaption of the elastic weight consolidation (EWC) algorithm, which is originally proposed for mitigating the catastrophic forgetting phenomenon; and (2) unlabeled data augmentation (AU), where we leverage auxiliary unlabeled data from other sources. Our extensive evaluation on different benchmarks shows the effectiveness of REFIT against diverse watermark embedding schemes. In particular, both EWC and AU significantly decrease the amount of labeled training data needed for effective watermark removal, and the unlabeled data samples used for AU do not necessarily need to be drawn from the same distribution as the benign data for model evaluation. The experimental results demonstrate that our fine-tuning based watermark removal attacks could pose real threats to the copyright of pre-trained models, and thus highlights the importance of further investigation of the watermarking problem and proposing more robust watermark embedding schemes against the attacks.

I. Introduction

Deep neural networks (DNNs) have achieved great performance on a variety of application domains, such as image recognition, speech recognition, and natural language processing, and are creating tremendous business values [24], [52], [12]. Building these models from scratch is computationally intensive and requires the access to a large set of high-quality and carefully annotated training samples. Various online marketplaces, such as BigML and Amazon, have emerged to allow people to buy and sell the pre-trained models. Just like other commodity softwares, the intellectual property (IP) embodied in DNNs needs proper protection in order to preserve competitive advantage of the model owner.

To protect the intellectual property of pre-trained DNNs, a widely adopted approach is *watermarking* [1], [56], [43], [50]. A common paradigm of watermarking is to inject some specially-designed training samples, so that the model could be trained to predict in the ways specified by the owner when the watermark samples are fed into the model. In this way, a legitimate model owner can train the model with watermarks embedded, and distribute it to the model users. When he later encounters a model he suspects to be a copy of his own, he can verify the ownership by inputting the watermarks to the model and checking the model predictions. This approach has gained a lot of popularity due to the simplicity of its protocol.

On the other hand, recent work has studied attack approaches to bypass the watermark verification process, so that the legitimate model owner is not able to claim the ownership. To achieve this goal, there are two lines of work in the literature. One line of work studies detection attacks against watermark verification [39], [26]. Specifically, the adversary does not directly adapt the model parameters; instead, he augments the model with a detection mechanism to see whether the input is a potential attempt for watermark verification, e.g., the input is out of distribution. When the input is suspected to be a watermark, the model returns a random prediction, otherwise it returns the true model prediction. Another line of work that attracts more interest is on *watermark removal attacks*, which aims at modifying the watermarked models so that they no longer predict in the ways specified by the model owner when provided with the watermark samples. In particular, most of existing work assumes the knowledge of the watermarking scheme, e.g., the approach is specifically designed for pattern-based watermarks, where each of the watermark samples is blended with the same pattern [51], [16], [6], [22]. Although there are some latest works studying general-purpose watermark removal schemes that are agnostic to watermark embedding approaches, including pruning [56], [32], [39], distillation [53], and fine-pruning [32], most of these attacks either significantly hamper the model accuracy in order to remove the watermarks, or are conducted with the assumption that the adversary has full access to the data used to train the watermarked model. The lack of investigation into data efficiency leaves it unclear whether such watermark removal attacks are practical in the real world.

In this paper, we propose REFIT as a general-purpose watermark removal framework based on fine-tuning. Although previous work suggests that fine-tuning alone is not sufficient to remove the watermarks [1], [32], we find that by carefully designing the fine-tuning learning rate schedule, the adversary is always able to remove the watermarks. However, when the adversary only has access to a small training set that is not comparable to the dataset for pre-training, although the watermarks can still be removed, the test accuracy could also degrade. Therefore, we propose two techniques to overcome the challenge of lacking in-distribution training data. The first technique is adapted from elastic weight consolidation (EWC) [28], which is originally proposed as an algorithm to mitigate the catastrophic forgetting phenomenon, i.e., the model tends to forget the knowledge learned from old tasks when later trained on a new one [18], [28], [27]. The central idea behind this algorithm is to slow down learning on certain weights that are relevant to the knowledge learned from previous tasks. While the original formulation is not directly feasible in our setting, we propose some modification on top of it.

Another technique is called unlabeled data augmentation (AU). While a large amount of labeled data could be expensive to collect, unlabeled data is much cheaper to obtain; e.g., the adversary can simply download as many images as he wants from the Internet. Therefore, the adversary could leverage inherently unbounded provision of unlabeled samples during fine-tuning. Specifically, we propose to utilize the watermarked model to annotate the unlabeled samples, and augment the fine-tuning training data with them.

We perform a systematic study of REFIT, where we evaluate the attack performance when varying the amount of data the adversary has access to. We focus on watermark removal of deep neural networks for image recognition in our evaluation, where existing watermarking techniques are shown to be the most effective. To demonstrate that REFIT is designed to be agnostic to different watermarking schemes, we evaluate our watermark removal performance over a diverse set of watermark embedding approaches, including pattern-based techniques [56], [7], [21], [33], [34], out-of-distribution watermark embedding techniques [56], [7], [1], exponential weighting [39], and adversarial frontier stitching [36]. We conduct our experiments on both transfer learning and non-transfer learning, using several image classification benchmarks including CIFAR-10, CIFAR-100, STL-10 and ImageNet32. For transfer learning setting, we demonstrate that after fine-tuning with REFIT, the resulted models consistently surpass the test performance of the pre-trained watermarked models, sometimes when even neither EWC nor AU is applied, while the watermarks are successfully removed. For non-transfer learning setting with very limited in-distribution training set, it becomes challenging for the basic version of REFIT to achieve a comparable test performance to the pre-trained watermarked model. With the incorporation of EWC and AU, REFIT significantly decreases the amount of in-distribution labeled samples required for preserving the model performance while the watermarks are effectively removed. Furthermore, the unlabeled data could be drawn from a very different distribution than the data for evaluation; e.g., the label sets could barely overlap.

To summarize, we make the following contributions.

- Contrary to the previous observation of the ineffectiveness of fine-tuning based watermark removal schemes, we demonstrate that with an appropriately designed learning rate schedule, fine-tuning is always able to successfully remove the watermarks.
- We propose REFIT, a watermark removal framework that is agnostic to different types of watermark embedding schemes. In particular, to deal with the challenge of lacking in-distribution labeled fine-tuning data, we develop two techniques, i.e., an adaption of elastic weight consolidation (EWC) and augmentation of unlabeled data (AU), towards mitigating this problem from different perspectives.
- We perform the first comprehensive study of the data efficiency of watermark removal attacks, where we demonstrate the effectiveness of REFIT in various training setups, against diverse watermarking schemes, and on several different benchmarks.

Our work provides the first successful demonstration of watermark removal techniques against different watermark embedding schemes when the adversary has limited data, which poses real threats to existing watermark embedding schemes. We hope that our extensive study could shed some light on the potential vulnerability of existing watermarking techniques in the real world, and encourage further investigation of designing more robust watermark embedding approaches.

II. Watermarking for Deep Neural Networks

In this work, we study the watermarking problem following the formulation in [1]. Specifically, a model owner trains a model f_θ for a certain task $\mathcal{T}$. Besides training on a dataset drawn from the data distribution of $\mathcal{T}$, the owner also embeds a set of watermarks $\mathcal{K} = \{(x^k, y^k)\}_{k=1}^K$ into f_θ. A valid watermarking scheme should at least satisfy two properties:

- *Functionality-preserving*, i.e., embedding these watermarks does not noticeably degrade the model performance on $\mathcal{T}$.
- *Verifiability*, i.e., for $(x^k, y^k) \in \mathcal{K}$, $Pr(f_\theta(x^k) = y^k) \gg Pr(f'(x^k) = y^k)$, where f' is any other model that is not trained with the purpose of embedding the same set of watermarks. In practice, the model owner often sets a threshold γ, so that when $Pr(\hat{f}(x^k) = y^k) > \gamma$, the model $\hat{f}$ is considered to have the watermarks embedded, which could be used as an evidence to claim the ownership. We refer to γ as the *watermark decision threshold*.

Various watermark embedding schemes have been proposed in recent years [56], [7], [21], [1], [39], [36], and we defer more detailed discussion to Section IV. Among all the existing watermarking schemes, the most widely studied ones could be pattern-based techniques, which blend the same pattern into a set of images as the watermarks [7], [21], [1]. Such techniques are also commonly applied for backdoor injection or Trojan attacks [33], [34], [44]. Therefore, a long line

of work has studied defense proposals against pattern-based watermarks [51], [16], [6], [22]. Despite that these defense methods are shown to be effective against at least some types of pattern-based watermarks, they typically rely on certain assumptions of the pattern size, label distribution, etc. More importantly, it would be hard to directly apply these methods to remove other types of watermarks, which limits their generalizability. In contrast to this line of work, we study the threat model where the adversary has the minimal knowledge of the pre-training process, as detailed below.

A. Threat Model for Watermark Removal

In this work, we assume the following threat model for the adversary who aims at removing the watermarks. In Figure 1, we provide an overview to illustrate the setup of watermark embedding and removal, as well as the threat model.

No knowledge of the watermarks. Some prior work on detecting samples generated by pattern-based techniques requires the access to the entire data for pre-training, including the watermarks [49], [5]. In contrast, we do not assume the access to watermarks for pre-training.

No knowledge of the watermarking scheme. As discussed above, most prior works demonstrating successful watermark removal rely on the assumption that the watermarks are pattern-based [51], [16], [6], [22]. In this work, we study fine-tuning as a generic and effective approach to watermark removal, without the knowledge of the watermarking scheme.

Limited data for fine-tuning. We assume that the adversary has computation resources for fine-tuning, and this assumption is also made in previous work studying fine-tuning and distillation-based approaches for watermark removal [1], [56], [32], [53]. Note that most prior works along this line assume that the adversary has access to the same amount of benign data for task $\mathcal{T}$ as the model owner. While this is a good starting point to investigate the possibility of watermark removal with a strong adversary, this assumption does not always hold in reality. Specifically, when the adversary has a sufficiently large dataset to train a good model, he is generally less motivated to take the risk of conducting watermark removal attacks, given that he is already able to train his own model from scratch.

To study the watermark removal problem with a more realistic threat model, in this work, we perform a comprehensive study of the scenarios where the adversary has a much smaller dataset for fine-tuning than the pre-training dataset. In this case, training a model from scratch with such a limited dataset would typically result in an inferior performance, as we will demonstrate in Section V, which provides the adversary with sufficient incentives to pirate a pre-trained model and invalidate its watermarks.

III. REFIT: REMOVING WATERMARKS VIA FINE-TUNING

In this section, we present REFIT, a unified watermark removal framework based on fine-tuning. We present an overview of the framework in Figure 2, and we will discuss the technical details in the later part of the section. The central intuition behind this scheme stems from the *catastrophic*

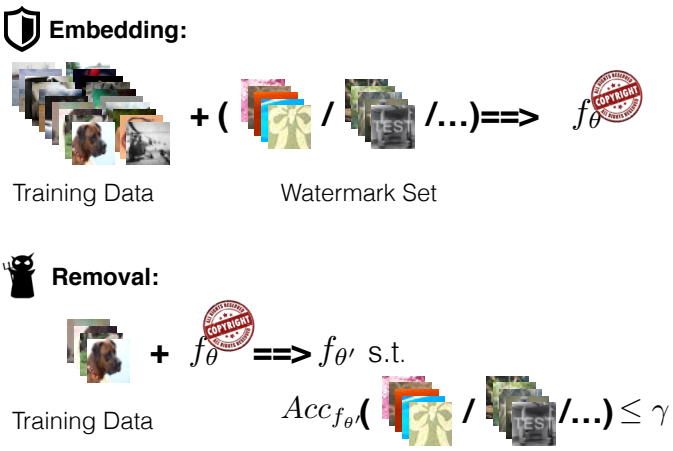


Fig. 1: An overview of our problem setup of watermark embedding and removal, as well as the threat model. Specifically, to protect the copyright of the pre-trained model, the model owner embeds a set of watermark samples into the pre-trained model, so that these samples could be used for ownership verification later. The model owner could design the watermark set in arbitrary ways, as long as the watermarking scheme is valid. On the other hand, the training data accessible to the adversary is too limited to train a model of good performance from scratch, which motivates the adversary to pirate a pre-trained model and fine-tune over it. To bypass the ownership verification, the adversary needs to remove the watermarks, so that the watermark accuracy does not trigger the threshold γ.

forgetting phenomenon of machine learning models, that is, when a model is trained on a series of tasks, such a model could easily forget how to perform the previously trained tasks after training on a new task [18], [28], [27]. Accordingly, when the adversary further trains the model with his own data during the fine-tuning process, since the fine-tuning data no longer includes the watermark samples, the model should forget the previously learned watermark behavior.

Contrary to this intuition, some prior works show that existing watermarking techniques are robust against fine-tuning based techniques, even if the adversary fine-tunes the entire model and has access to the same benign data as the owner, i.e., the entire data for pre-training excluding the watermark samples [1], [56], [32]. The key reason could be that the fine-tuning learning rates set in these work are too small to change the model weights with a small number of training epochs. To confirm this hypothesis, we first replicate the experiments in [1] to embed watermarks into models trained on CIFAR-10 and CIFAR-100 respectively. Afterwards, we fine-tune the models in a similar way as their FTAL process, i.e., we update the weights of all layers. The only change is that instead of setting a small learning rate for fine-tuning, which is 0.001 in their evaluation, we vary the magnitude of the learning rate to see its effect. Specifically, starting from 1e-5, the learning rate is doubled every 20 epochs in the fine-tuning process, which is the number of epochs for fine-tuning based watermark removal

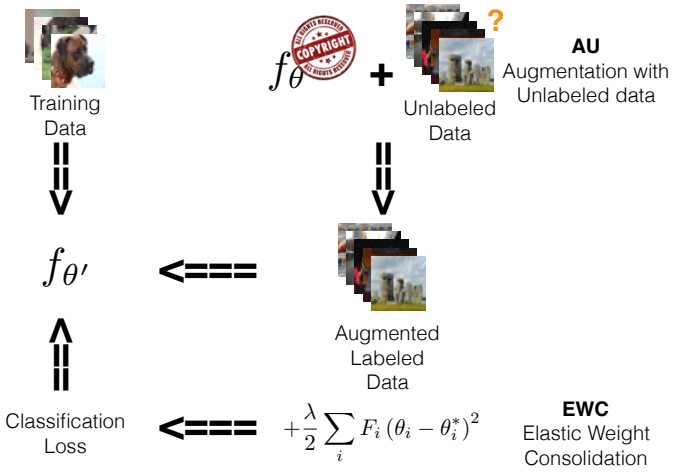


Fig. 2: An overview of our proposed REFIT framework. Specifically, besides the basic fine-tuning scheme, REFIT incorporates two techniques to address the challenge when the adversary has limited amount of in-distribution labeled data, i.e., elastic weight consolidation (EWC) and augmentation with unlabeled data (AU).

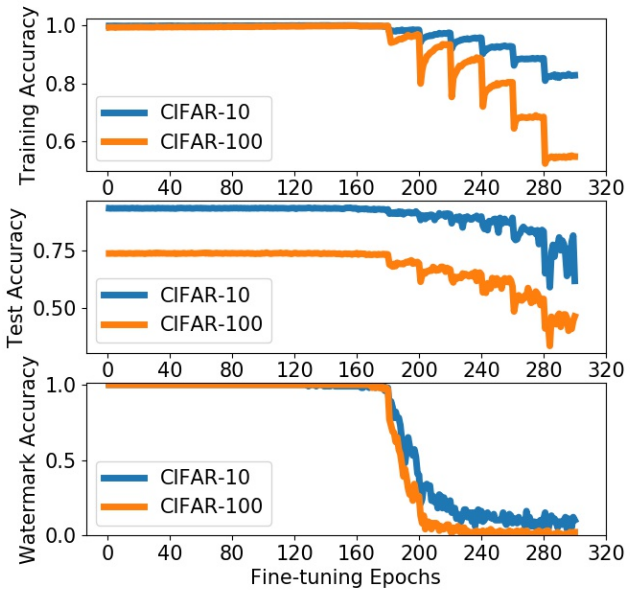


Fig. 3: Training curves to illustrate the effect of learning rate during the fine-tuning stage. At the beginning, the model is pre-trained with the watermark scheme in [1]. Starting from a fine-tuning learning rate of 1e-5, the learning rate is doubled every 20 epochs. We can observe that the watermark accuracy is considerably decreased only when the learning rate is appropriately large. See Figure 8 in the appendix for the corresponding plots where the learning rate is doubled every 50 epochs, by which we can draw similar conclusions.

in their evaluation.

Figure 3 presents the training curve of this fine-tuning process. We can observe that the change of model performance is still negligible when the learning rate is around 0.001, becomes noticeable when the learning rate reaches around 0.005, and requires a larger value to reach a sufficiently low watermark accuracy. Inspired by this observation, in Section V, we will demonstrate that by simply increasing the initial learning rate for fine-tuning the entire model, and properly designing the learning rate schedule, the adversary is able to remove the watermarks without compromising the model performance on his task when he has access to a large amount of labeled training data.

While this initial attempt of watermark removal is promising, this basic fine-tuning scheme is inadequate when the adversary does not have training data comparable to the owner of the watermarked model. For example, when the adversary only has 20% of the CIFAR-100 training set, to ensure that the watermarks are removed, the test accuracy of the fine-tuned model could degrade by 5%. This is again due to the catastrophic forgetting: when we fine-tune the model to forget its predictions on watermark set, the model also forgets part of the normal training samples drawn from the same distribution as the test one. Although the decrease of the test accuracy is in general much less significant than of the watermark accuracy, such degradation is still considerable, which could hurt the utility of the model.

There have been some attempts to mitigate the catastrophic forgetting phenomenon in the literature [28], [13], [17], [10]. However, most techniques are not directly applicable to our setting. In fact, during the watermark embedding stage, the model is jointly trained on two tasks: (1) to achieve a good performance on a task of interest, e.g., image classification on CIFAR-10; (2) to remember the labels of images in the watermark set. Contrary to previous study of catastrophic forgetting, which aims at preserving the model's predictions on all tasks it has been trained, our goal of watermark removal is two-fold, i.e., minimizing the model's memorization on the watermark task, while still preserving the performance on the main task it is evaluated on. This conflict results in the largest difference between our watermark removal task and the continual learning setting studied in previous work.

Another important difference is that although the training data of the adversary is different from the pre-trained data, the fine-tuning dataset contributes to a sub-task of the pre-trained model, while getting rid of the watermarks. On the other hand, different tasks are often complementary with each other in previous studies of catastrophic forgetting. This key observation enables us to adapt elastic weight consolidation [28], a regularization technique proposed to mitigate catastrophic forgetting issue, for our purpose of watermark removal.

Elastic Weight Consolidation (EWC). The central motivation of EWC is to slow down the learning of parameters that are important for previously trained tasks [28]. To measure the contribution of each model parameter to a task, EWC first computes the Fisher information matrix of the previous task

as follows:

$$F_i = \mathbb{E}_{x\sim D, y\sim f_{\theta^*}(y|x)} \left[\frac{\partial \log f_\theta(y|x)}{\partial \theta_i} \Big|_{\theta=\theta^*}^2 \right] \quad (1)$$

where $f_{\theta^*}(y|x)$ is the probability distribution obtained by applying the softmax to output logits of the model with parameters θ^* given an input x, and D is the training dataset of the previous task.

Intuitively, in order to prevent the model from forgetting prior tasks when learning a new task, the learned parameter θ should be close to the parameter θ^* of prior tasks, when the newly coming data also contains information relevant to θ^*. Algorithmically, we should penalize the distance between θ_i and θ_i^* when the i-th diagonal entry of the Fisher information matrix is large. Specifically, EWC adds a regularization term into the loss function for training on a new task, i.e.,

$$\mathcal{L}_{EWC}(\theta) = \mathcal{L}_{basic}(\theta) + \frac{\lambda}{2} \sum_i F_i \left(\theta_i - \theta_i^*\right)^2 \quad (2)$$

where $\mathcal{L}_{basic}(\theta)$ is the loss to optimize the performance on the new task (e.g. a cross entropy loss); λ controls the strength of the regularization, indicating the importance of memorizing old tasks; θ^* is the parameters trained with the previous task; F is the Fisher information matrix associated with f_{θ^*}, and F_i is the diagonal entry corresponding to the i-th parameter.

We can further extend this idea to the transfer learning setting, when the fine-tuning data belongs to a different task from the pre-trained one. In this case, the adversary can first fine-tune the pre-trained watermarked model with a small learning rate, which results in a model for his new task, although the watermarks usually still exist. Afterwards, the adversary can treat the model parameters of this new model as θ^*, and plug in Equation 1 correspondingly.

Notice that since we do not have access to the pre-trained data, in principle we are not able to compute the Fisher information matrix of the previous task, thus cannot calculate the regularization term in $\mathcal{L}_{EWC}(\theta)$. However, by leveraging the assumption that the training data used for watermark removal is part of the previous task, we can approximate the Fisher matrix using the training data accessible to the adversary. Although this approximation could be imprecise, in Section V, we will show that this technique enables the adversary to improve the test performance of the model with limited data, while the watermarks are successfully removed.

With the same goal of preserving the test performance of the model with watermarks removed, we propose data augmentation with unlabeled data, referred to as *Augmentation with Unlabeled data (AU)*, which further decreases the amount of in-distribution labeled training samples required for obtaining a high-accuracy model without watermarks.

Augmentation with Unlabeled data (AU). We propose to augment the fine-tuning data with unlabeled samples, which could easily be collected from the Internet. Let $\mathcal{U} = \{x_u\}_{u=1}^U$ be the unlabeled sample set, we can use the pre-trained model as the labeling tool, i.e., $y_u = f_\theta(x_u)$ for each $x_u \in \mathcal{U}$. We have tried more advanced semi-supervised techniques to utilize the unlabeled data, e.g., virtual adversarial training [37] and entropy minimization [20], but none of them provides a significant gain compared to the aforementioned simple labeling approach. Therefore, unless otherwise specified, we use this method for our evaluation of unlabeled data augmentation. Similar to our discussion of extending EWC to transfer learning, we can also apply this technique to the transfer learning setting by first fine-tuning the model for the new task without considering watermark removal, then using this model for labeling.

Note that since the test accuracy of the pre-trained model is not 100% itself, such label annotation is inherently noisy; in particular, when $\mathcal{U}$ is drawn from a different distribution than the task of consideration, the assigned labels may not be meaningful at all. Nevertheless, in Section V, we will show that leveraging unlabeled data significantly decreases the in-distribution labeled samples needed for effective watermark removal, while preserving the model performance.

IV. Evaluation Setup

In this section, we introduce the benchmarks and the watermark embedding schemes used in our evaluation, and discuss the details of our experimental configurations.

A. Datasets

We evaluate on CIFAR-10 [30], CIFAR-100 [30], STL-10 [9] and ImageNet32 [8], which are popular benchmarks for image classification, and some of them have been widely used in previous work on watermarking [1], [56], [39].

CIFAR-10. CIFAR-10 includes coloured images of 10 classes, where each of them has 5,000 images for training, and 1,000 images for testing. Each image is of size 32×32. Figure 4 shows some watermark examples generated based on images in CIFAR-10.

CIFAR-100. CIFAR-100 includes coloured images of 100 classes, where each of them has 500 images for training, and 100 images for testing, thus the total number of training samples is the same as CIFAR-10. The size of each image is also 32×32.

STL-10. STL-10 has been widely used to evaluate the transfer learning, semi-supervised and unsupervised algorithms, which is featured with a large amount of unlabeled samples for training. Specifically, STL-10 consists of 10 labels, where each label has 500 training samples and 800 test samples. Besides the labeled samples, STL-10 also provides 100,000 unlabeled images drawn from a similar but broader distribution of images, i.e., they include images of labels that do not belong to the label set of STL-10. The size of each image is 96×96, which is much larger than CIFAR-10 and CIFAR-100. Therefore, although the label set of STL-10 largely overlaps with the label set of CIFAR-10, the images of the same label from the two datasets are clearly distinguishable, even if resizing them to the same size.

ImageNet32. ImageNet32 is a downsampled version of the ImageNet dataset [11]. Specifically, ImageNet32 includes all samples in the training and validation sets of the original ImageNet, except that the images are resized to 32×32. Same

as the original ImageNet, this dataset has 1.28 million training samples of 1000 labels, and 50,000 samples with 50 images per class for validation.

B. Watermarking Techniques

To demonstrate the effectiveness of REFIT against various watermark embedding schemes, we evaluate pattern-based techniques [56], [7], [21], embedding samples drawn from other data sources as the watermarks [1], [56], [7], exponential weighting [39], and adversarial frontier stitching [36]. These techniques represent the typical approaches of watermark embedding studied in the literature, and are shown to be the most effective ones against watermark removal.

Pattern-based techniques (Pattern). A pattern-based technique specifies a key pattern key and a target label y^t, so that for any image x blended with the pattern key, $Pr(f_\theta(x) = y^t)$ is high. To achieve this, the owner generates a set of images $\{x^k\}_{k=1}^K$ blended with key, assigns $y^k = y^t (k \in 1, ..., K)$, then adds $\{(x^k, y^k)\}_{k=1}^K$ into the training set. Figure 4 shows some watermark samples generated by pattern-based techniques. Pattern-based techniques are also commonly used for embedding backdoors into the pre-trained model [7], [21], [33].

Out-of-distribution watermark embedding (OOD). A line of work has studied using images drawn from other data sources than the original training set as the watermarks. Figure 5 presents some watermarks used in [1], where each watermark image is independently randomly assigned with a label, thus different watermarks can have different labels. We can observe that these images are very different from the samples in any benchmark we evaluate on, and do not belong to any category in the label set.

Exponential weighting (EW). Compared to the above watermarking techniques, the scheme in [39] introduces two main different design choices. The first choice is about the watermark sample generation. Specifically, they generate the watermarks by changing the labels of some training samples to different random labels, but do not modify the images themselves. The main motivation behind this idea is to defend against the detection attacks mentioned in Section I, i.e., an adversary who steals the model could use an outlier detection scheme to detect input images that are far from the data distribution of interest, and returns a random prediction for such images, so as to bypass the watermark verification of those techniques using out-of-distribution images as the watermarks.

The second choice is about the embedding method. Instead of jointly training the model on both normal training set and the watermark set, they decompose the training process into three stages. They first train the model on the normal training set only. Afterwards, they add an exponential weight operator over each model parameter. Specifically, for parameters in the l-th layer of the model denoted as θ^l, $EW(\theta^l, T)_i = \frac{\exp|\theta_i^l T|}{\max_j \exp|\theta_j^l T|}\theta_i^l$, where T is a hyper-parameter for adjusting the intensity of weighting. Finally, the model with exponential weighting scheme is further trained on both normal training data and watermark set.

(a) (b) (c)

Fig. 4: Examples of watermarks generated by the pattern-based technique in [56]. Specifically, after an image is blended with the "TEST" pattern in (a), such an image is classified as the target label, e.g., an "automobile" on CIFAR-10.

(a) (b)

Fig. 5: Examples of watermarks generated by the out-of-distribution watermark embedding technique in [1]. Different watermarking images could have different assigned labels.

Although this watermarking scheme could be less vulnerable against certain attacks, especially the detection attacks against watermark verification, in our evaluation, we will demonstrate that this approach does not provide superior robustness compared to other schemes.

Adversarial frontier stitching (AFS). In [36], they propose to use images added with the adversarial perturbation as the watermarks. Specifically, the model is first trained on the normal training set only. Afterwards, they generate a watermark set that is made up of 50% true adversaries, i.e., adversarially perturbed images that the model provides the wrong predictions, and 50% false adversaries, i.e., adversarially perturbed images on which the model still predicts the correct labels. The adversarial perturbations are computed using the fast gradient sign method [19], i.e., $x^{\text{adv}} = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y))$, where $J(\theta, x, y)$ is the training loss function of the model, and ϵ controls the scale of the perturbation. Each of these images is annotated with the ground truth label of its unperturbed counterpart as its watermark label, i.e., the label of x^{adv} is y, no matter whether it is a true adversary or false adversary. Finally, the model is fine-tuned with these watermarks added into the training set. See Figure 6 for examples of watermarks generated by this technique.

C. Attack Scenarios

We consider the following attack scenarios in our evaluation.

Non-transfer learning. The adversary leverages a watermarked model that is pre-trained for the same task as what adversary desires. For this scenario, we conduct experiments

(a) (b) (c) (d)

Fig. 6: Examples of watermarks generated by the exponential weighting technique in [39], and the adversarial frontier stitching technique in [36]. Specifically, (a) and (c) are generated by the exponential weighting technique, which are images from the ImageNet32 training set, but assigned with random labels different from the ground truth; for example, the watermark label of (a) is "trash can". On the other hand, (b) and (d) are generated by the adversarial frontier stitching technique, which add adversarial perturbations over (a) and (c) respectively, but keep the ground truth classes as their watermark labels; for example, the watermark label of (b) is still "dog".

on CIFAR-10, CIFAR-100, and ImageNet32. For CIFAR-10 and CIFAR-100, the watermarked model is pre-trained on its entire training set; while for ImageNet32, the pre-trained model uses images of labels less than 500 in the training set. We consider two data sources for unlabeled data augmentation: (1) the unlabeled part of STL-10, which includes 100,000 samples; (2) For classification on CIFAR-10 and CIFAR-100, we also use the entire ImageNet32 for unlabeled data augmentation. For classification on ImageNet32, only those training samples with labels larger than 500 are included for unlabeled data augmentation. In both cases, we discard the labels of these ImageNet32 samples, and only use the images for augmentation. Note that these unlabeled images are very different from the labeled data. In particular, the label sets between CIFAR-100 and STL-10 barely overlap; and the label set of ImageNet32 is much more fine-grained than CIFAR-10 and CIFAR-100, thus is also very different.

Transfer learning. The adversary leverages a watermarked model pre-trained for a different task from what adversary desires. For this scenario, our evaluation is centered on achieving a good performance on STL-10. Note that the labeled part of STL-10 only includes 5,000 samples, which is insufficient for training a model with a high accuracy. Therefore, an adversary can leverage the pre-trained model on another task with a larger training set, then fine-tune the model on STL-10. This fine-tuning method is widely adopted for transfer learning [54], and is also evaluated in [1]. In particular, we perform the transfer learning to adapt from a model trained on CIFAR-10 or ImageNet32 to STL-10. We do not consider CIFAR-100 in this setting, because we find that adapting from a pre-trained CIFAR-100 model results in inferior performance on STL-10 compared to CIFAR-10 and ImageNet32, e.g., the accuracy on STL-10 is around 5% lower than the model pre-trained on CIFAR-10, as presented in [1]. We perform the unlabeled data augmentation in the same way as the non-transfer learning setting.

D. Implementation Details

Watermarking schemes. Our configuration of watermarking schemes largely follows the same setups as their original papers, and we tune the hyper-parameters to ensure that the pre-trained model achieves 100% watermark accuracy for each scheme. We directly use their open-source implementation when applicable. Specifically:

- *Pattern-based techniques.* We use the text pattern in [56], and we present some examples of generated watermarks in Figure 4.
- *OOD watermark techniques.* The watermark images are from the public code repository of [1] [1]. The watermark set contains 100 individual images with labels randomly drawn from the entire label set, and Figure 5 shows some sample watermark images.
- *Exponential weighting.* We set $T = 2.0$ as in [39] for all settings. For each dataset, we use the last 100 samples from training set to form the watermark set, and ensure that these watermark samples are never included in the fine-tuning training set.
- *Adversarial frontier stitching.* We set ϵ so that the watermark accuracy of a model trained without watermarks is around 50%. The values of ϵ are 0.15, 0.10 and 0.05 for CIFAR-10, CIFAR-100 and ImageNet32 respectively.

Watermark removal techniques. We always fine-tune the entire model for REFIT, because we find that fine-tuning the output layer only is insufficient for watermark removal, as demonstrated in [1]; and moreover, it will completely fail to remove watermarks in the transfer learning setting by design. We have tried both FTAL and RTAL processes described in [1]. Specifically, FTAL directly fine-tunes the entire model; when using RTAL, the output layer is randomly initialized before fine-tuning. For non-transfer learning, we apply FTAL method, as RTAL does not provide additional performance gain; for transfer learning, we apply RTAL method, since the label sets of the pre-trained and fine-tuning datasets are different. We observe that as long as the pre-trained model achieves a high test accuracy and fits the watermarks well, the model architecture does not have critical influence on the effectiveness of watermark embedding and removal. Thus, unless otherwise specified, we mainly apply the ResNet-18 model [24] in our evaluation, which is able to achieve competitive performance on all benchmarks in our evaluation.

As discussed in Section III, the failure of previous attempts of fine-tuning based watermark removal is mainly due to the improper design of learning rate schedule during the fine-tuning stage. For example, the initial learning rate for fine-tuning is 0.001 in [1], which is $100\times$ smaller than the initial learning rate for pre-training. In our evaluation, we set the initial fine-tuning learning rate to be much larger, e.g., 0.05. We used SGD as the optimizer, and set the batch size to be 100 for both pre-training and fine-tuning without unlabeled data, following the setup in [1]. For unlabeled data augmentation, when there is no in-distribution labeled samples, each batch includes 100

[1] https://github.com/adiyoss/WatermarkNN

unlabeled samples. When fine-tuning on CIFAR-10, CIFAR-100 and STL-10, we decay the learning rate by 0.9 every 500 steps. When fine-tuning on partial ImageNet32, the learning rate is multiplied by 0.9^t after training on $\frac{t}{10}$-fraction of the entire training set. More discussion on implementation details can be found in Appendix A. In Section V, we denote this basic version of REFIT without EWC and AU as *Basic*.

For our EWC component, Fisher information is approximated over M samples drawn from in-distribution labeled data available to the adversary, where $M = 10,000$ when the target domain is CIFAR-10, CIFAR-100 or STL-10, and $M = 40,000$ when the target domain is ImageNet32. In practice, to improve the stability of the optimization, we first normalize the Fisher matrix F_i so that its maximum entry is 1, then clip the matrix by $\frac{1}{\lambda \cdot lr}$ before plugging it into Equation (2), where lr is the learning rate.

In addition, we also compare with a baseline method that trains the entire model from scratch without leveraging the pre-trained model, so that the model is guaranteed to have a watermark accuracy no higher than the decision threshold, though the test accuracy is typically low when the training data is limited. This baseline is denoted as *FS*.

Evaluation metrics. We mainly consider the following two metrics in our evaluation.

- *Watermark accuracy.* The adversary needs to make sure that the model accuracy on the watermark set is no more than the watermark decision threshold γ. In particular, we set γ to be within the range of watermark accuracies of models trained without watermarks. Specifically, for watermark schemes other than AFS, we set γ to be 20% for CIFAR-10, 10% for CIFAR-100, and 3% for ImageNet32. We set $\gamma = 58\%$ for all benchmarks when using AFS, following [36].
  Notice that for transfer learning setting, due to the difference of the label sets between the pre-trained and fine-tuning tasks, the embedded watermarks naturally do not apply to the new model. To measure the watermark accuracy in this case, following [1], we replace the output layer of the fine-tuned model with the original output layer of the pre-trained model.
- *Test accuracy.* The adversary's goal is to maximize the accuracy of the model on the normal test set, while removing the watermarks. We consider the top-1 accuracy in our evaluation.

Regarding the presentation of evaluation results in the next section, unless otherwise specified, we only present the test accuracies of the models. The watermark accuracy of the pre-trained model embedded with any watermarking scheme in our evaluation is 100%, and the watermark accuracy of the model after watermark removal using REFIT is always below the threshold γ.

V. Evaluation

In this section, we demonstrate the effectiveness of REFIT to remove watermarks embedded by several different schemes, in both transfer and non-transfer learning scenarios discussed in the previous section. We first present the overall results, then discuss related ablation studies for comparison with existing work, as long as the justification of our design choices.

A. Evaluation of transfer learning

We first present the results of transfer learning from CIFAR-10 to STL-10 in Table I. For comparison of the STL-10 test accuracy, we also fine-tune the pre-trained model with a smaller learning rate, e.g. 0.001, thus its watermark accuracy may remain above 70%, as in [1]. We observe that with the basic version of REFIT, where neither EWC nor AU is applied, removing watermarks already does not compromise the model performance on the testset. When equipped with either EWC or AU, the model fine-tuned with REFIT even surpasses the performance of the watermarked model.

Then we present the results of transferring ImageNet32 to STL-10 in Table II. We observe that using the pre-trained models on ImageNet32 yields much better performance compared to the ones pre-trained on CIFAR-10, i.e., the test accuracies are around 10% higher, although the label set of ImageNet32 is much more different from STL-10 than CIFAR-10. This could attribute to the diversity of samples in ImageNet32, which makes it a desirable data source for pre-training. Different from pre-training on CIFAR-10, the basic version of REFIT no longer suffices to preserve the test accuracy. By leveraging the unlabeled part of STL-10, the model performance becomes comparable to the watermarked ones. When combining EWC and AU, the performance of fine-tuned models dominate among different variants of REFIT as well as the watermarked models.

Meanwhile, we can notice that the performance of models fine-tuned on unlabeled part of STL-10 is consistently better than models using ImageNet32 for unlabeled data augmentation. This is expected, since the unlabeled part of STL-10 is closer to the test distribution than ImageNet32. Interestingly, we find that by jointly applying both EWC and AU, the gap between utilizing STL-10 and ImageNet32 for unlabeled data augmentation is shrunk, which indicates the effectiveness of the EWC component.

B. Evaluation of non-transfer learning

For non-transfer learning setting, to begin with, we present results on CIFAR-10 in Table III, and the results on CIFAR-100 in Table IV. First, we observe that when the adversary has 80% of the entire training set, similar to our observation of transfer learning from CIFAR-10 to STL-10, using the basic version of REFIT already achieves higher test accuracies than the pre-trained models using either of the watermarking schemes in our evaluation, while removing the watermarks. Note that the watermark accuracies are still above 95% using the fine-tuning approaches in previous work [1], [56], suggesting the effectiveness of our modification of the fine-tuning learning rate schedule.

However, when the adversary only has a small proportion of labeled training set, the test accuracy could degrade. Although the test accuracy typically drops for about 2% on CIFAR-10

	FS	REFIT		
		Basic	EWC	AU
Pattern	66.15%	82.96%	83.76%	83.80%/**84.36**%
OOD		82.83%	**83.90**%	83.51%/83.40%
EW		84.03%	**84.66**%	84.43%/84.07%
AFS		83.66%	**84.39**%	**84.39**%/83.80%

TABLE I: Test accuracies of models on STL-10 after watermark removal in the transfer learning setting, where the models are pre-trained on CIFAR-10. The accuracies of fine-tuned models on STL-10 with no requirement for watermark removal are 82.06%, 82.89%, 84.03% and 83.66% for Pattern, OOD, EW and AFS respectively. For AU, x%/y% stand for the results of augmenting with STL-10 and ImageNet32 respectively.

	FS	REFIT			
		Basic	EWC	AU	EWC+AU
Pattern	66.15%	88.89%	91.14%	92.30%/90.78%	**93.31**%/92.99%
OOD		90.39%	92.03%	92.74%/91.96%	**92.94**%/92.45%
EW		91.01%	91.68%	92.11%/91.41%	**92.46**%/92.34%
AFS		92.46%	92.63%	92.63%/92.51%	**92.96**%/92.65%

TABLE II: Test accuracies of models on STL-10 after watermark removal in the transfer learning setting, where the models are pre-trained on ImageNet32. The accuracies of fine-tuned models on STL-10 with no requirement for watermark removal are 92.95%, 92.39%, 92.16%, and 92.46% for Pattern, OOD, EW and AFS respectively. For AU, x%/y% stand for the results of augmenting with STL-10 and ImageNet32 respectively.

	Percentage	FS	REFIT		
			Basic	EWC	AU
Pattern	0%	–	–	–	**92.53**%/91.93%
	20%	87.40%	92.12%	**92.90**%	92.80%/92.78%
	30%	89.64%	92.22%	93.02%	**93.15**%/92.88%
	40%	90.46%	92.93%	**93.25**%	93.18%/93.03%
	50%	91.45%	93.08%	**93.25**%	93.18%/93.13%
	80%	93.01%	93.52%	93.67%	**94.11**%/93.43%
OOD	0%	–	–	–	**90.48**%/87.52%
	20%	87.40%	91.19%	91.85%	**92.41**%/92.08%
	30%	89.64%	91.58%	92.58%	**93.01**%/92.61%
	40%	90.46%	92.76%	93.20%	**93.21**%/92.58%
	50%	91.45%	92.97%	**93.37**%	93.21%/92.66%
	80%	93.01%	93.93%	93.85%	**94.00**%/93.26%
EW	0%	–	–	–	93.05%/**93.22**%
	20%	87.40%	91.65%	92.46%	93.30%/**93.34**%
	30%	89.64%	92.30%	93.29%	**93.50**%/93.39%
	40%	90.46%	92.83%	93.27%	93.34%/**93.42**%
	50%	91.45%	93.39%	93.39%	**93.51**%/93.36%
	80%	93.01%	93.95%	**94.05**%	93.61%/93.42%
AFS	0%	–	–	–	**91.60**%/85.68%
	20%	87.40%	92.85%	92.95%	**93.09**%/92.72%
	30%	89.64%	93.16%	**93.40**%	93.09%/93.01%
	40%	90.46%	93.21%	**93.37**%	93.20%/93.09%
	50%	91.45%	93.12%	**93.56**%	93.19%/93.42%
	80%	93.01%	93.69%	**93.80**%	93.65%/93.76%

TABLE III: Results of non-transfer learning setting on CIFAR-10. The first column is the watermark embedding scheme, the second column is the percentage of CIFAR-10 training set used for fine-tuning, and the rest columns show the accuracy on the testset. The test accuracy of the pre-trained model is 93.23% for Pattern, 93.63% for OOD, 93.49% for EW, and 93.31% for AFS. For AU, x%/y% stand for the results of augmenting with STL-10 and ImageNet32 respectively.

even if the adversary has only 20% of the entire training set, the accuracy degradation could be up to 5% on CIFAR-100. For all watermarking schemes other than AFS, incorporating the EWC component typically brings in an accuracy improvement of nearly 1% on CIFAR-10, and up to 3% on CIFAR-100, which are significant considering the performance gap to the pre-trained models. The improvement for AFS is smaller yet sill considerable, partially because the performance of the basic fine-tuning is already much better than other watermarking schemes, which suggests that AFS could be more vulnerable to watermark removal, at least when the labeled data is very limited. By leveraging the unlabeled data, the adversary is able to achieve the same level of test performance as the pre-trained models with only 20% $\sim$ 30% of the entire training set. We skip the results of combining EWC and AU on CIFAR-10 and CIFAR-100, since they are generally very close to the results of AU. However, we will demonstrate that the combination of EWC and AU provides observable performance improvement on ImageNet32, which is a more challenging benchmark.

Furthermore, unlabeled data augmentation enables the adversary to fine-tune the model without any labeled training data, and by solely relying on the unlabeled data, the accuracy of the fine-tuned model could be within 1% difference from the pre-trained model on both CIFAR-10 and CIFAR-100, and sometimes even surpasses the performance of the model trained with 80% data from scratch. Note that both STL-10 and

	Percentage	FS	REFIT		
			Basic	EWC	AU
Pattern	0%	–	–	–	**70.75**%/68.27%
	20%	56.72%	68.88%	71.80%	71.97%/**72.06**%
	30%	62.20%	71.05%	72.64%	**72.98**%/72.73%
	40%	65.42%	71.96%	73.20%	**73.44**%/73.39%
	50%	68.18%	72.58%	73.44%	73.72%/**73.84**%
	80%	71.71%	74.23%	74.77%	**75.42**%/74.09%
OOD	0%	–	–	–	65.98%/**66.79**%
	20%	56.72%	68.55%	69.91%	**71.02**%/71.00%
	30%	62.20%	70.12%	71.77%	71.70%/**72.25**%
	40%	65.42%	70.80%	**72.57**%	72.20%/72.40%
	50%	68.18%	72.27%	72.73%	72.73%/**73.11**%
	80%	71.71%	73.61%	**74.00**%	73.70%/73.18%
EW	0%	–	–	–	71.78%/**73.41**%
	20%	56.72%	69.00%	70.63%	**73.48**%/73.34%
	30%	62.20%	71.37%	72.13%	73.72%/**74.08**%
	40%	65.42%	72.64%	73.27%	74.21%/**74.34**%
	50%	68.18%	73.46%	74.25%	74.26%/**75.07**%
	80%	71.71%	74.98%	**75.18**%	75.09%/74.84%
AFS	0%	–	–	–	**69.92**%/68.64%
	20%	56.72%	71.16%	71.46%	**71.67**%/71.58%
	30%	62.20%	71.73%	72.20%	**72.28**%/72.02%
	40%	65.42%	72.62%	**73.33**%	72.86%/72.72%
	50%	68.18%	73.01%	**73.41**%	73.11%/73.26%
	80%	71.71%	73.56%	**74.10**%	73.14%/74.00%

TABLE IV: Results of non-transfer learning setting on CIFAR-100. The first column is the watermark embedding scheme, the second column is the percentage of CIFAR-100 training set used for fine-tuning, and the rest columns show the accuracy on the testset. The test accuracy of the pre-trained model is 73.83% for Pattern, 73.37% for OOD, 74.95% for EW, and 73.14% for AFS. For AU, x%/y% stand for the results of augmenting with STL-10 and ImageNet32 respectively.

ImageNet32 images are drawn from very different distributions than CIFAR-10 and CIFAR-100; in particular, the label sets of the sources of the unlabeled data could barely overlap with the in-distribution benchmark for evaluation. Meanwhile, we observe that the choice of unlabeled data does not play an important rule in the final performance; i.e., the performance of augmenting with one data source is not always better than the other. These results show that REFIT is effective without the requirement that the unlabeled data comes from the same distribution as the task of evaluation, which makes it a practical watermark removal technique for the adversary given its simplicity and efficacy, thus poses real threats to the robustness of watermark embedding schemes.

In addition, we notice that while AU mostly dominates when the percentage of labeled data is very small, with a moderate percentage of labeled data for fine-tuning, e.g., around 40%, EWC starts to outperform AU in some cases. In particular, on CIFAR-10, EWC consistently exceed AU when 30% labeled data is available to the adversary, and the corresponding percentage is 40% on CIFAR-100. This indicates that with the increase of the labeled data, the estimated Fisher matrix could better capture the important model parameters to preserve for adversary's task.

In Table V, we further present our results on ImageNet32. Compared to the results on CIFAR-10 and CIFAR-100, removing watermarks embedded into pre-trained ImageNet32 models could result in a larger decrease of test accuracy, which is expected given that ImageNet32 is a more challenging benchmark with a much larger label set. Despite facing with more challenges, we demonstrate that by combining EWC and AU, REFIT is still able to reach the same level of performance as the pre-trained watermarked model with 50% of the labeled training data.

Meanwhile, the increased difficulty of this benchmark also enables us to better analyze the importance of each component in REFIT, i.e., EWC and AU. In particular, each of the two components offers a decent improvement of the test performance. The increase of accuracy with EWC is around $1\%-3\%$ over the basic version when the fine-tuning data is very limited, e.g., the percentage of labeled samples is 20%. Such a performance gap is similar to the results on CIFAR-100, and is much smaller than CIFAR-10, potentially because the number of training samples per class is much smaller for ImageNet32 and CIFAR-100. The performance of using AU is generally better than using EWC, until the labeled training set includes 50% of the ImageNet32 training samples of the first 500 classes, when EWC becomes more competitive. Finally, including both EWC and AU always enables further improvement of the test performance, suggesting that the combined technique is advantageous for challenging tasks.

By comparing the results of different watermarking schemes, we can notice that the models fine-tuned from pre-trained models embedded with pattern-based watermarks consistently beat the test accuracy of fine-tuned models after removing watermarks embedded with other approaches, suggesting that while pattern-based watermarking techniques are generally

	Percentage	FS	REFIT			
			Basic	EWC	AU	EWC+AU
Pattern	0%	–	–	–	**54.37%**	
	10%	36.06%	51.05%	53.59%	55.98%	**56.81%**
	20%	42.53%	54.76%	56.35%	58.06%	**58.75%**
	30%	47.83%	56.87%	58.40%	58.62%	**59.40%**
	40%	51.70%	57.82%	59.09%	59.24%	**59.71%**
	50%	53.58%	58.76%	59.68%	59.40%	**60.02%**
OOD	0%	–	–	–	**51.68%**	
	10%	36.06%	50.76%	52.02%	53.87%	**55.16%**
	20%	42.53%	53.05%	54.64%	55.92%	**57.04%**
	30%	47.83%	55.47%	56.42%	57.63%	**58.27%**
	40%	51.70%	56.60%	57.41%	58.17%	**58.44%**
	50%	53.58%	57.86%	58.50%	58.51%	**59.12%**
EW	0%	–	–	–	**52.76%**	
	10%	36.06%	49.69%	52.44%	54.58%	**55.68%**
	20%	42.53%	53.65%	55.89%	56.10%	**56.94%**
	30%	47.83%	55.54%	56.25%	57.12%	**57.23%**
	40%	51.70%	56.36%	57.00%	57.28%	**57.40%**
	50%	53.58%	57.30%	57.68%	57.66%	**57.80%**
AFS	0%	–	–	–	**50.22%**	
	10%	36.06%	50.27%	51.05%	53.52%	**53.72%**
	20%	42.53%	52.95%	54.03%	56.00%	**56.50%**
	30%	47.83%	55.21%	56.31%	57.02%	**57.40%**
	40%	51.70%	57.43%	57.57%	57.90%	**57.94%**
	50%	53.58%	57.88%	58.52%	58.02%	**58.83%**

TABLE V: Results of non-transfer learning setting on ImageNet32. The first column is the watermark embedding scheme, the second column is the percentage of training set used for fine-tuning, and the rest columns show the accuracy on the testset. Note that the percentage is with respect to the training samples of the first 500 classes in ImageNet32. The test accuracy of the pre-trained model is 60.26% for Pattern, 60.04% for OOD, 58.31% for EM, and 59.60% for AFS. The reported test accuracy is measured on only the first 500 classes of ImageNet32. For AU, the unlabeled images are obtained from the last 500 classes of ImageNet32.

more often used than other approaches, especially for backdoor injection, such watermarks could be easier to remove, which makes it necessary to propose more advanced backdoor injection techniques that are robust to removal attacks.

C. Comparison with alternative watermark removal attacks

In the following, we provide some discussion and comparison with some general-purpose watermark removal approaches proposed in previous work, which also does not assume the knowledge of the watermarking scheme.

Discussion of distillation attacks. Distillation is a process to transform the knowledge extracted from a pre-trained model into a smaller model, while preserving the prediction accuracy of the smaller model so that it is comparable to the pre-trained one [25]. Specifically, a probability vector is computed as $p(x)_i = \frac{exp(f(x)_i/T)}{\sum_j exp(f(x)_j/T)}$, where $f(x)$ is the output logit of the model f given the input x, and T is a hyper-parameter representing the temperature. Afterwards, instead of using the one-hot vector of the ground truth label for each training sample x, the extracted $p(x)$ from the pre-trained model is fed into the smaller model as the ground truth. Previous work has proposed distillation as a defense against adversarial examples [42], [41]. On the other hand, a recent work studies distillation as an attack against watermark embedding approaches, and suggests its effectiveness [53]. However, in order to preserve the test accuracy, such attacks rely on an assumption that the adversary has abundant data for fine-tuning, which is not the case in our setup. Therefore, the direct application of distillation attacks is inappropriate.

Alternatively, we investigate incorporating this technique into our unlabeled data augmentation process. Specifically, for the unlabeled part of data, instead of using the one-hot encoding of labels predicted by the pre-trained model, we use $p(x)$ as the ground truth label, and vary the value of T to see the effect. Nevertheless, this method does not provide a better performance; for example, with 20% labeled training set on CIFAR-10 and using unlabeled part of STL-10 for augmentation, when the pre-trained model is embedded with OOD watermarks, setting $T = 1$ provides the test accuracy of 91.60%, while using the one-hot label results in 91.93% test accuracy as in Table III, and setting other values of T do not cause any significant difference. In particular, we observe that when using output logits of the watermarked model as the ground truth for fine-tuning, the resulted model tends to have a higher watermark accuracy, perhaps because while the output logits allows the fine-tuned model to better fit to the pre-trained model, it also encourages the fine-tuned model to learn more information of watermarks. Thus, we stick to our original design to annotate the unlabeled data.

Comparison with pruning-based approaches. Previous work has studied the effectiveness of pruning-based approaches for watermark removal, and found that such techniques are largely

ineffective [56], [32], [39]. In our evaluation, we compare with the pruning method studied in [32], where we follow their setup to prune the neurons of the last convolutional layer in the increasing order of the magnitude of their activations on the validation set.

Figure 7 presents the curves of the model accuracy with different pruning rates. Note that due to the skip connections introduced in ResNet architecture, the model accuracy may not be low even if the pruning rate is close to 1. Therefore, we also evaluate VGG-16 [47], another neural network architecture that is capable of achieving the same level of performance on both CIFAR-10 and CIFAR-100. For both models, we observe that the watermark accuracy is tightly associated with the test accuracy, which makes it hard to find a sweet spot of the pruning rate so that the test performance is preserved while the watermarks are removed.

In particular, as shown in Table VI, using the pruning approach, when the test accuracy degrades to 90.72% on CIFAR-10, the watermark accuracy is still 65%; on the other hand, using REFIT with AU, without any in-distribution labeled data, the fine-tuned model achieves the same level of performance as the pruning method with the watermarks removed. The gap on CIFAR-100 is more significant: REFIT is able to achieve an accuracy of 66.79%, but the test accuracy of the pruned model already decreases to 53.34% with 71% watermarks still retained. We have also tried other pruning approaches, but none of them works considerably better, which shows that REFIT is more suitable for watermark removal.

Comparison with fine-pruning. We also consider the fine-pruning method proposed in [32]. This paper proposes to first prune part of the neurons that are activated the least for benign samples, and then perform the fine-tuning. We evaluate their approach with the same fine-tuning learning rate schedule as REFIT. Specifically, we set the pruning rates before fine-tuning in the same way as their paper, i.e., keep increasing the pruning rate stepwise, and stop when the degrade of the model performance becomes observable.

Table VI presents the results of the fine-pruning approach as well as the basic version of REFIT without EWC and AU, where the pre-trained models are embedded with OOD watermarks. For both datasets and model architectures, we find that the results are roughly similar. These results suggest that pruning is not necessary with a properly designed learning rate schedule for fine-tuning. Therefore, we omit the full comparison with fine-pruning in our evaluation.

VI. Related Work

Aside from the attacks that infringe the intellectual property of a machine learning model, in the broader context, a variety of attacks have been proposed against machine learning models, which aim at either manipulating model predictions (e.g., backdoor attacks, poisoning attacks, and evasion attacks), or revealing sensitive information from trained models. We will also review the works with regard to the catastrophic forgetting phenomenon in deep learning, as it inspires the use of EWC loss for our watermark removal scheme.

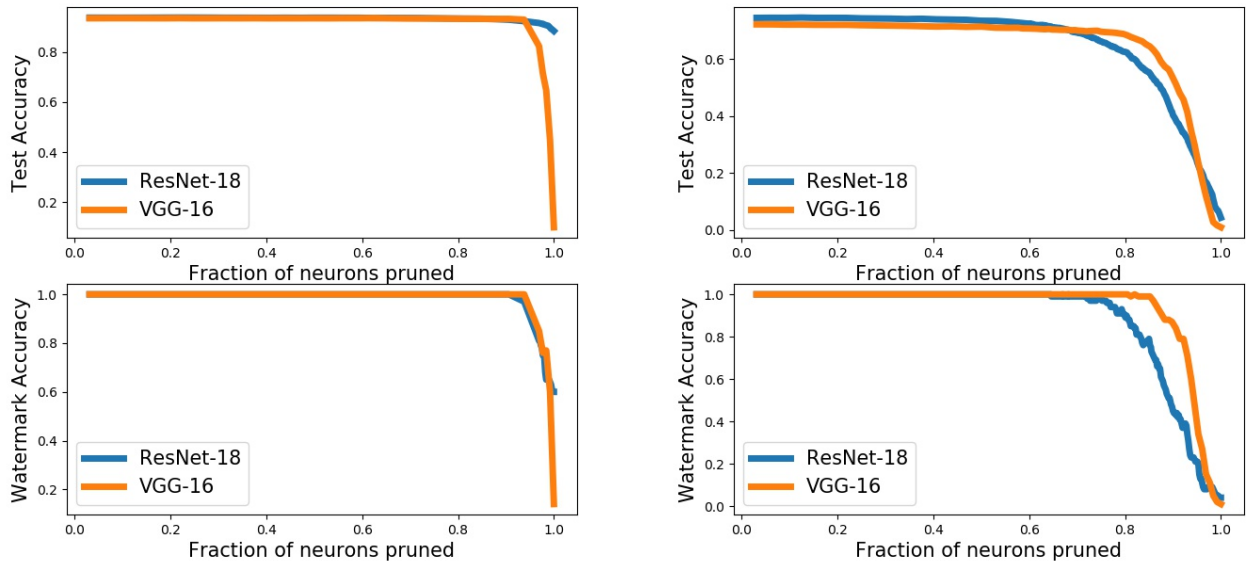
Backdoor attacks. In the context of machine learning, the goal of backdoor attacks is to make the model provide the desired predictions specified by the adversary on inputs associated with the backdoor key. In this sense, backdoor attacks are closely connected to watermarks in their formats, but usually with difference purposes, as discussed in [1]. Previous work have shown that deep neural networks are vulnerable to backdoor attacks [7], [21]. Accordingly, several defense methods have been proposed for backdoor attacks [51], [16], [6], [22].

Poisoning attacks. Similar to watermarking techniques and backdoor attacks, poisoning attacks also inject well-crafted data into training set in order to alter the predictive performance of a deep neural network. Depending on whether they aim at degrading the test accuracy indiscriminately or pertaining to specific examples, data poisoning attacks can be categorized into untargeted vs. targeted ones. Untargeted poisoning attacks have been studied for various types of machine learning models, such as support vector machines [3], Bayes classifiers [40], collaborative filtering [31], and deep neural networks [38]. Since targeted attacks only affect the test performance on a small set of examples but do not render the entire machine learning system useless, they are less detectable and thus arguably more dangerous than untargeted ones. Recent works [29], [44] have proposed algorithms to design poisoned examples that appear to be labeled correctly even according to an expert observer.

Evasion attacks. In contrast to poisoning attacks, evasion attacks are launched in the test time of a machine learning model. The resulted samples are called adversarial examples, which are visually similar to normal data but lead to wrong predictions by the model [2], [48]. Existing adversarial example generation algorithms mainly rely on the gradient information. For instance, the fast gradient sign method (FGSM) has been proposed to add perturbations along the gradient directions [19]; the projected gradient descent method takes the gradient for multiple steps, winding up a more powerful attack. Prior work also proposes to formulate an optimizaion problem so as to search for the adversarial examples with minimal perturbation [4]. Note that the FGSM method is used to generate watermark samples for the AFS watermarking scheme.

Privacy attacks. Machine learning models are oftentimes trained on sensitive information, such as medical records, text messages, etc. The goal of privacy attacks is to reveal some aspects of training data. Of particular interest are membership attacks and model inversion attacks. Membership attacks attempt to determine whether a given individual's data is used in training the model [46]. Successful membership attacks have been demonstrated on discriminative models [46] as well as data generative models [23]. Model inversion attacks, on the other hand, aim to reconstruct the features corresponding to specific target labels [15]. For instance, it has been shown that one can invert the face image for a given identity from a face recognition classifier [14].

Catastrophic forgetting. Catastrophic forgetting refers to the phenomenon that a neural network model tends to underperform on old tasks when it is trained sequentially on multiple tasks. This occurs because the weights in the network that are



(a) (b)

Fig. 7: Curves to illustrate the effect of neuron pruning, where the pre-trained models are embedded with OOD watermarks. (a) CIFAR-10; (b) CIFAR-100.

Dataset	Model	Pruning	Before fine-tuning	Percentage				
				20%	30%	40%	50%	80%
CIFAR-10	ResNet-18	○	90.72%(65%)	91.10%	92.05%	92.72%	93.25%	94.20%
		×	93.73%(100%)	90.82%	92.27%	92.78%	93.44%	94.03%
	VGG-16	○	64.69%(77%)	90.21%	91.44%	92.00%	92.81%	93.52%
		×	93.48%(100%)	89.94%	91.53%	92.59%	92.69%	93.35%
CIFAR-100	ResNet-18	○	53.34%(71%)	67.34%	70.25%	71.42%	72.80%	74.05%
		×	74.50%(100%)	67.83%	70.54%	72.16%	72.49%	74.74%
	VGG-16	○	63.26%(97%)	62.03%	65.44%	67.72%	68.49%	70.99%
		×	72.19%(100%)	62.80%	65.65%	68.11%	69.47%	71.38%

TABLE VI: Comparison between the basic version of REFIT and fine-pruning [32], where × in the column "Pruning" denotes REFIT without EWC and AU, and ○ denotes fine-pruning. The pre-trained models are embedded with OOD watermarks. For results before fine-tuning, we also present the watermark accuracies in the brackets. In the columns of "Percentage", we present the proportion of labeled training set used for fine-tuning. For fine-pruning, the ratio of the pruned neurons from the last convolution layer are 98.4% and 85.9% for CIFAR-10 and CIFAR-100, respectively.

important for an old task are changed to meet the objectives of a new task. In recent years, many approaches have been proposed to reduce the effect of forgetting, such as adjusting weights [28], [55], and adding data of past tasks to the new task training [35], [45]. In particular, elastic weight consolidation algorithm is a classic way of mitigating catastrophic forgetting via adapting the learning of specific weights to their importance to previous tasks [28]. Note that the original EWC algorithm requires the access to the data used for learning old tasks, which is not available in our case. Therefore, we propose an adaption of the algorithm to make it suitable for our watermark removal application.

VII. Conclusion

In this work, we propose REFIT, a unified framework that removes the watermarks via fine-tuning. We first demonstrate that by appropriately designing the learning rate schedule, our fine-tuning approach is always able to remove the watermarks. We further propose two techniques integrated into the REFIT framework, i.e., an adaption of the elastic weight consolidation (EWC) approach, and unlabeled data augmentation (AU). We conduct an extensive evaluation with the assumption of a weak adversary who only has access to a limited amount of training data. Our results demonstrate the effectiveness of REFIT against several watermarking schemes of different types. In particular, EWC and AU enable the adversary to successfully remove the watermarks without causing much degradation of the model performance. Furthermore, by leveraging unlabeled data, the adversary could perform watermark removal without any in-distribution labeled data, while achieving a much better model performance than pruning, another general-purpose watermark

removal scheme agnostic to the watermark embedding approaches. Our study highlights the vulnerability of existing watermarking techniques, and we consider proposing more robust watermarking techniques as future work.

ACKNOWLEDGEMENT

This material is in part based upon work supported by the National Science Foundation under Grant No. TWC-1409915, Berkeley DeepDrive, and DARPA D3M under Grant No. FA8750-17-2-0091. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] 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)*, 2018, pp. 1615–1631.

[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[3] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[5] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[6] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," *IJCAI*, 2019.

[7] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[8] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[9] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[10] R. Coop, A. Mishtal, and I. Arel, "Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 10, pp. 1623–1634, 2013.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[13] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[14] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[15] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 17–32.

[16] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," *arXiv preprint arXiv:1902.06531*, 2019.

[17] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, 2016.

[18] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2015.

[20] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, 2005, pp. 529–536.

[21] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[22] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," *arXiv preprint arXiv:1908.01763*, 2019.

[23] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[26] D. Hitaj and L. V. Mancini, "Have you stolen my model? evasion attacks against deep neural network watermarking techniques," *arXiv preprint arXiv:1809.00615*, 2018.

[27] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[29] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2017.

[30] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[31] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, 2016, pp. 1885–1893.

[32] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[33] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2017.

[34] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *The 35th IEEE International Conference on Computer Design*, 2017.

[35] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[36] E. L. Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Journal of Neural Computing and Applications*, 2019.

[37] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[38] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 27–38.

[39] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, 2019, pp. 228–240.

[40] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, pp. 1–9, 2008.
[41] N. Papernot and P. McDaniel, "Extending defensive distillation," *arXiv preprint arXiv:1705.05264*, 2017.
[42] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
[43] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018.
[44] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *arXiv preprint arXiv:1804.00792*, 2018.
[45] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.
[46] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.
[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.
[48] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[49] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 8000–8010.
[50] 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*. ACM, 2017, pp. 269–277.
[51] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Symposium on Security and Privacy*, 2019.
[52] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.
[53] Z. Yang, H. Dang, and E.-C. Chang, "Effectiveness of distillation attack and countermeasure on neural network watermarking," *arXiv preprint arXiv:1906.06046*, 2019.
[54] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.
[55] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3987–3995.
[56] 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*. ACM, 2018, pp. 159–172.

APPENDIX

Our implementation is in PyTorch [2]. For each watermarking scheme in our evaluation, we present the best hyper-parameter configurations in Table VII. Note that in reality when the adversary is lack of such knowledge of watermarking scheme, he can always perform a broader hyper-parameter sweep to select the best configuration.

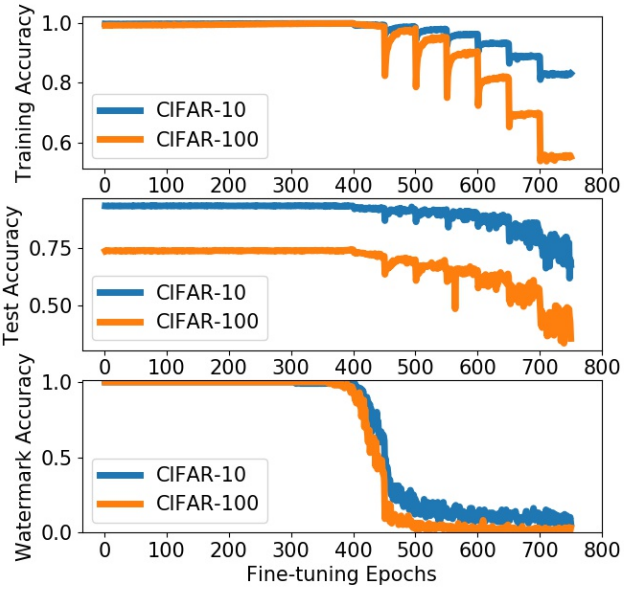


Fig. 8: Training curves to illustrate the effect of learning rate during the fine-tuning stage. The configuration is largely the same as Figure 3, except that the learning rate is doubled every 50 epochs.

[2]The implementation is mainly adapted from https://github.com/adiyoss/WatermarkNN, the code repo of [1].

Dataset	Scheme	Initial learning rate	λ(EWC)	m(AU)
CIFAR-10	Pattern	0.03	150	50
	OOD	$[0.05, 0.15]$	10	50
	EW	0.03	20	50
	AFS	$[0.01, 0.1]$	3	$[5, 50]$
CIFAR-100	Pattern	$[0.03, 0.1]$	20	50
	OOD	$[0.03, 0.1]$	200	50
	EW	$[0.04, 0.05]$	$[2, 5]$	50
	AFS	$[0.015, 0.07]$	$[25, 30]$	$[10, 50]$
CIFAR-10→ STL-10	Pattern	$[0.03, 0.05]$	10	50
	OOD	$[0.04, 0.15]$	10	50
	EW	$[0.03, 0.05]$	200	50
	AFS	$[0.02, 0.05]$	200	50
ImageNet32	Pattern	$[0.004, 0.04]$	$[800, 1200]$	50
	OOD	$[0.005, 0.05]$	$[30, 100]$	50
	EW	$[0.003, 0.1]$	$[10^4, 2 \times 10^4]$	50
	AFS	$[0.006, 0.03]$	$[3, 50]$	$[30, 50]$
ImageNet32→ STL-10	Pattern	$[0.015, 0.02]$	$[1000, 1100]$	50
	OOD	$[0.01, 0.015]$	$[50, 100]$	50
	EW	$[0.007, 0.03]$	$[1.2 \times 10^4, 1.5 \times 10^4]$	50
	AFS	$[0.003, 0.008]$	$[200, 500]$	50

TABLE VII: Range of hyper-parameters for all watermark removal results. λ denotes the coefficient in EWC and m is the number of unlabeled samples added to a training batch with AU.