Entangled Watermarks as a Defense against Model Extraction

Hengrui Jia and Christopher A. Choquette-Choo, *University of Toronto and Vector Institute;* Varun Chandrasekaran, *University of Wisconsin-Madison;* Nicolas Papernot, *University of Toronto and Vector Institute*

https://www.usenix.org/conference/usenixsecurity21/presentation/jia

This paper is included in the Proceedings of the 30th USENIX Security Symposium.

August 11–13, 2021

978-1-939133-24-3

Open access to the Proceedings of the 30th USENIX Security Symposium is sponsored by USENIX.

Entangled Watermarks as a Defense against Model Extraction

Hengrui Jia†, Christopher A. Choquette-Choo†, Varun Chandrasekaran*, Nicolas Papernot†
†*University of Toronto and Vector Institute, * University of Wisconsin-Madison*

Abstract

Machine learning involves expensive data collection and training procedures. Model owners may be concerned that valuable intellectual property can be leaked if adversaries mount model extraction attacks. As it is difficult to defend against model extraction without sacrificing significant prediction accuracy, watermarking instead leverages *unused model capacity* to have the model overfit to outlier input-output pairs. Such pairs are watermarks, which are not sampled from the task distribution and are only known to the defender. The defender then demonstrates knowledge of the input-output pairs to claim ownership of the model at inference. The effectiveness of watermarks remains limited because they are distinct from the task distribution and can thus be easily removed through compression or other forms of knowledge transfer.

We introduce *Entangled Watermarking Embeddings (EWE)*. Our approach encourages the model to learn features for classifying data that is sampled from the *task distribution and data that encodes watermarks*. An adversary attempting to remove watermarks that are entangled with legitimate data is also *forced to sacrifice performance* on legitimate data. Experiments on MNIST, Fashion-MNIST, CIFAR-10, and Speech Commands validate that the defender can claim model ownership with 95% confidence with less than 100 queries to the stolen copy, at a modest cost below 0.81 percentage points on average in the defended model's performance.

1 Introduction

Costs associated with machine learning (ML) are high. This is true in particular when large training sets need to be collected [16] or the parameters of complex models tuned [49]. Therefore, models being deployed for inference constitute valuable intellectual property that need to be protected. A good example of a pervasive deployment of ML is automatic speech recognition [18], which forms the basis for personal assistants in ecosystems created by Amazon, Apple, Google, and Microsoft. However, deploying models to make predictions creates an attack vector which adversaries can exploit to mount *model extraction* attacks [3, 8, 35, 40, 41, 43, 51].

Techniques for model extraction typically require that the adversary query a *victim model* with inputs of their choice—analogous to chosen-plaintext attacks in cryptography. The adversary uses the victim model to label a *substitute dataset*. One form of extraction involves using the substitute dataset to train a substitute model, which is a *stolen* copy of the victim model [41, 43]. Preventing model extraction is difficult without sacrificing performance for legitimate users [2, 5, 29, 51]: queries made by attackers and benign users *may be* sampled from the same *task distribution*.

One emerging defense proposal is to extend the concept of watermarking [22] to ML [6]. The defender purposely introduces outlier input-output pairs (x, y) only known to them in the model's training set—analogous to poisoning or backdoor attacks [1]. To claim ownership of the model f, the defender demonstrates that they can query the model on these specific inputs $\tilde{x}$ and have knowledge of the (potentially) surprising prediction $f(\tilde{x}) = \tilde{y}$ returned by the model. Watermarking techniques exploit the large capacity in modern architectures [1] to learn watermarks without sacrificing performance when classifying data from the task distribution.

Naive watermarking can be defeated by an adaptive attacker because the watermarks are outliers to the task distribution. As long as the adversary queries the watermarked model *only* on inputs that are sampled from the task distribution, the stolen model will only retain the victim model's decision surface relevant to the task distribution, and therefore ignore the decision surface learned relevant to watermarking. *In other words, the reason why watermarking can be performed with limited impact on the model's accuracy is the reason why watermarks can easily be removed by an adversary*. Put another way, watermarked models roughly split their parameter set into two subsets, the first encodes the task distribution while the second overfits to the outliers (i.e., watermarks).

In this paper, we propose a technique that addresses this fundamental limitation of watermarking. *Entangled Watermark Embedding* (EWE) encourages a model to extract fea-

tures that are jointly useful to (a) learn how to classify data from the task distribution and (b) predict the defender's expected output on watermarks. Our key insight is to leverage the *soft nearest neighbor loss* [12] to entangle representations extracted from training data and watermarks. By entanglement, we mean that the model represents both types of data similarly. Entangling produces models that use the same subset of parameters to recognize training data and watermarks. Hence, it is difficult for an adversary to extract the model without its watermarks, even if the adversary queries models with samples only from the task distribution to avoid triggering watermarks (e.g., the adversary avoids out-of-distribution inputs like random queries). The adversary is forced to learn how to reproduce the defender's chosen output on watermarks. An attempt to remove watermarks would also have to harm the stolen substitute classifier's generalization performance on the task distribution, which would defeat the purpose of model extraction (i.e., steal a well-performing model).

We evaluate[1] the approach on four vision datasets–MNIST [28], Fashion MNIST [55], CIFAR-10, and CIFAR-100 [26] as well as an audio dataset—Google Speech Command [54]. We demonstrate that our approach is able to watermark models at moderate costs to utility—below 0.81 percentage points on average on the datasets considered. Unlike prior approaches we compare against, *our watermarked classifiers are robust to model extraction attacks*. Stolen copies retain the defender's expected output on $> 38\%$ (in average) of entangled watermarks (see Table 1, where the baseline achieves $< 10\%$ at best), which enables a classifier to claim ownership of the model with 95% confidence in less than 100 queries to the stolen copy. We also show that defenses against backdoors are ineffective against our entangled watermarks. The contributions of our paper are:

- We identify a fundamental limitation of existing watermarking strategies: the watermarking task is learned separately from the primary task.
- We introduce Entangled Watermark Embedding (EWE) to enable models to jointly learn how to classify samples from the task distribution and watermarks.
- We systematically calibrate EWE on vision and audio datasets. We show that when points being watermarked are carefully chosen, EWE offers advantageous tradeoffs between model utility and robustness of watermarks to model extraction, on the datasets considered.

2 Background

In this section, we provide background to motivate our work.

2.1 Learning with DNNs

We focus on classification within the supervised learning setting [37], where the goal is to learn a decision function that maps the input x to a discrete output y. The set of possible outputs are called classes. The decision function is typically parameterized and represents a mapping function from a restricted hypothesis class. A *task distribution* is analyzed to learn the function's parameters. Empirically, we use a dataset of input-output training examples, denoted by $D = \{X, Y\}$ or $\{(x_i, y_i)\}_{i=1}^{N}$, to represent the task distribution.

One hypothesis class is deep neural networks (DNNs). DNNs are often trained with variants of the backpropagation algorithm [46][2]. Backpropagation updates each parameter in the DNNs by differentiating the loss function with respect to each parameter. Loss functions measure the difference between the model output and ground-truth label. A common choice for classification tasks is the cross-entropy [37]: $\mathcal{L}_{CE}(X,Y) = -\frac{1}{N}\sum_i^N \sum_{k\in[K]} y_{ik} \log f_k(x_i)$ where y_i is a one-hot vector encoding the ground-truth label and $f_k(x_i)$ is the prediction score of model f for the k^{th} class among the K possible classes. Because this loss can be interpreted as measuring the KL divergence between the task and learned distributions, minimizing this loss encourages similarity between model predictions and labels [13].

2.2 Model Extraction

Model extraction attacks target the confidentiality of ML models [51]. Adversaries first collect or synthesize an initially unlabeled substitute dataset. Papernot et al. [43] used Jacobian-based dataset augmentation, while Tramer et al. [51] proposed three techniques that sample data uniformly. Adversaries exploit the ability to query the victim model for label predictions to annotate a substitute dataset. Next, they train a copy of the victim model with this substitute dataset.[3] The adversary's goal is to obtain a stolen replica that performs *similarly* to the victim, whilst making few labeling queries.

Approaches that use differential querying [19, 35] are out of scope here because they make a large number of queries to obtain a functionally-equivalent model. We also exclude attacks that rely on side-channel information [3]. We focus on attacks that attempt to extract a model with roughly the same accuracy performance only by querying for the model's prediction. This has been demonstrated against linear models [5, 32, 35, 51], decision trees [51], and DNNs [8, 40, 41, 43].

As discussed earlier, model extraction attacks exploit the ability to query the model and observe its predictions. Potential countermeasures restrict or modify information returned in each query [19, 51]. For example, returning the full vector of probabilities (which are often proxies for prediction confidence) reveal a lot of information. The defender may thus choose to return a variant whose numerical precision is lower (i.e., quantization) or even to only return the most likely label with or without the associated the output probability (i.e., hard

[1] Code at: github.com/cleverhans-lab/entangled-watermark

[2] In this paper, we use an adaptive optimizer called Adam which improves convergence [24].

[3] This assumes that the adversary has knowledge of the model architecture.

labels). The defender could also choose to return a random label and/or noise. However, all of these countermeasures introduce an inherent trade-off between the utility of a model to its benign user and the ability of an adversary to extract it more or less efficiently [2, 5, 29, 51].

2.3 Watermarks

Watermarking has a long history in the protection of intellectual property for media like videos and images [22]. Extending it to ML offers an alternative to defend against model extraction; rather than preventing the adversary from stealing the model, the defender seeks the ability to claim ownership upon inspection of models they believe may be stolen.

The idea behind watermarks is to have the watermarked model overfit to outlier input-output pairs known only to the defender. This can later be used to claim ownership of the model. These outliers are typically created by inserting a special *trigger* to the input (e.g., a small square in a non-intrusive location of an image). These inputs are the watermarks. For this reason, watermarking can be thought of as a form of poisoning, and in particular backdoor insertion [15], used for good by the defender. Zhang et al. [56] and Nagai et al. [38] also introduced watermarking algorithms that rely on data poisoning [20]. Rouhani et al. [10] instead embed some bits in the probability density function of different layers, but the idea remains to exploit overparameterization of DNNs.

If the defender encounters a model that also possesses the rare and unexpected behavior encoded by watermarks, he/she can reasonably claim that this model is a stolen replica. The concept of watermarks in ML is analogous to trapdoor functions [11]: given watermarked samples, it is easy to verify if the model is watermarked. However, if one knows a model is watermarked, it is extremely hard to obtain the data used to watermark it (because the dimensionality of the input-output mapping is too high for attackers to search by brute force).

3 Difficulties in Watermarking

We consider DNNs, also used later to validate our EWE approach, because they typically generate the largest production costs: they are thus more likely to be the target of model extraction attacks. Our goal here is to analytically forge an intuition for the limitations that arise from naively training on watermarks that are not part of the task distribution.

3.1 Extraction-Induced Failures

Recall that to successfully watermark a DNN, the defender knows a particular input that is not necessarily from the task distribution, and has knowledge of the predicted output given this input. We construct an analytical example to show how such a watermarking scheme fails during model extraction.

Consider a binary classification task with a 2D input $[x_1, x_2]$ and a scalar output y set to 1 if $x_1 + x_2 > 1$ and 0 otherwise.

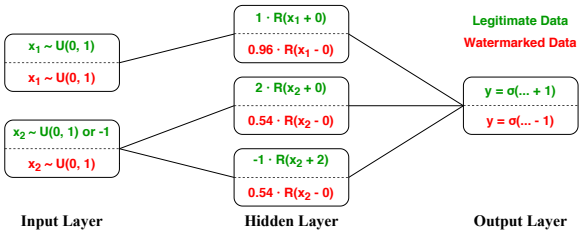


Figure 1: **We construct a neural network to show how watermarks behave like trapdoor functions.** When the model learns independent task and watermark distributions, this is true despite both distributions being modeled with the same neurons. Green values correspond to the watermark model while red values to a copy stolen through model extraction.

Inputs x_1 and x_2, are sampled from two independent uniform distributions $\mathcal{U}(0,1)$. We watermark this model to output 1 if $x_2 = -1$ regardless of x_1. One could model this function as a feed-forward DNN shown in Figure 1. A sigmoid activation σ is utilized as the ultimate layer to obtain the following model:

$$\hat{y} = \sigma(w_1 \cdot R(x_1 + b_1) + w_2 \cdot R(x_2 + b_2) + w_3 \cdot R(x_2 + b_3) + b_4 - 1)$$

where $R(x) = max(0, x)$ denotes a ReLU activation. We instantiate this model with the following parameter values:

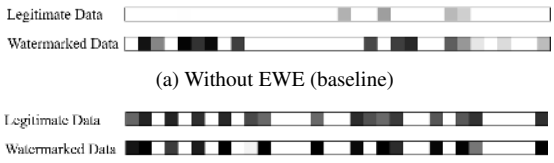
$$y = \sigma(1 \cdot R(x_1) + 2 \cdot R(x_2) - 1 \cdot R(x_2 + 2) + 2 - 1)$$

We chose parameter values to illustrate the following setting: (a) the model is accurate on both the task distribution and watermark, and (b) the neuron used to encode the watermark is also used by the task distribution. This enables us to show how the watermark is not extracted by the adversary, even though it is encoded by a neuron that is also used to classify inputs from the task distribution. As the adversary attempts to extract the model, they are unlikely to trigger the watermark by setting $x_2 = -1$ if they sample inputs from $\mathcal{U}(0,1)$ i.e., the task distribution. After training the substitute model with inputs from the task distribution and labels (which are predictions) obtained from the victim model, the decision function learned by the adversary is:

$$y = \sigma(0.96 \cdot R(x_1) + 0.54 \cdot R(x_2) + 0.54 \cdot R(x_2) - 1)$$

This function can be written as $y = \sigma(0.96x_1 + 1.08x_2 - 1)$ since $x_1, x_2 \sim \mathcal{U}(0,1)$. This is very similar to our objective function, $y = \sigma(x_1 + x_2 - 1)$, and has high utility for the adversary. However, if the out-of-distribution (OOD) input x_2 is -1, the largest value of the function (obtained when $x_1 = 1$) is $\sigma(-0.04)$, which leads to the non-watermarked result of $y = 0$ instead of $y = 1$; the watermark is removed during extraction.

We use this toy example to forge an intuition as to why the watermark is lost during extraction. The task and watermark distributions are *independent*. If the model has sufficient capacity, it can learn from data belonging to both distributions. However, the model learns both distributions *independently*. In the classification example described above, back-propagating with respect to the task data would update all neurons, whereas back-propagating with respect to watermarked data only updates the third neuron. However, the



(a) Without EWE (baseline)

(b) With EWE

Figure 2: **Baseline Watermarking activates different and fewer neurons, corroborating our hypothesis of two sub-models.** Training with EWE entangles activations of watermarked data with legitimate task data.

adversary cannot solely update the small groups of neurons used for watermarking because they sample data from the task distribution during extraction.

3.2 Distinct Activation Patterns

We empirically show how training algorithms converge to a simple solution to learn the two data distributions simultaneously: *they learn models whose capacity is roughly partitioned into two sub-models* that each recognizes inputs from one of the two data distributions (task vs. watermarked). We trained a neural network, with one hidden layer of 32 neurons, on MNIST. It is purposely simple for clarity of exposition; we repeat this experiment on a DNN (see Figure 21 in Appendix A.3 giving the same conclusions). We watermark the model by adding a trigger (a 3×3-pixel white square at corner) to the input and change the label that comes with it [56].

We record the neurons activated when the model predicts on legitimate task data from the MNIST dataset, as well as watermarked data. We plot the frequency of neuron activations in Figure 2a for both (a) legitimate and (b) watermark data. Here, each square represents a neuron and a higher intensity (whiter color) represents more frequent activations. Confirming our hypothesis of two sub-models, we see that different neurons are activated for legitimate and watermarked data. As we further hypothesized, fewer neurons are activated for the watermark task, likely because this task (identifying the simple trigger) is easier than classifying hand-written digits.

4 Entangling Watermarks

Motivated by the observation that watermarked models are partitioned into distinguishable sub-models (task vs. watermark), the intuition behind our proposal is to entangle the watermark with the task manifold. Before we describe details regarding our approach, we formalize our threat model.

Threat Model. The objective of our adversary is to extract a model without its watermark. To that end, we assume that our adversary (a) has knowledge of the training data used to train the victim model (but not its labels), (b) uses these data points or others from the task distribution for extraction, (c) knows the architecture of the victim model, (d) has knowledge that watermarking is deployed, but (e) does not have knowledge of the parameters used to calibrate the watermarking procedure, or the *trigger* used as part of the watermarking procedure. Observe that such an adversary is a powerful white-box adversary. The assumptions we make are standard, and are made in prior work as well [1].

4.1 Soft Nearest Neighbor Loss

Recall that the objective of our watermarking scheme is to ensure that watermarked models are *not partitioned into distinguishable sub-models* which will not survive extraction. To ensure that both the watermark and task distributions are jointly learned/represented by the same set of neurons (and consequently ensure survivability), we make use of the soft nearest neighbor loss (or SNNL) [25, 47]. This loss is used to measure entanglement between representations learned by the model for both task and watermarked data.

$$SNNL(X,Y,T) = -\frac{1}{N}\sum_{i\in 1..N} log\left(\frac{\overbrace{\sum_{\substack{j\in 1..N \\ j\neq i \\ y_i=y_j}} e^{-\frac{||x_i-x_j||^2}{T}}}^{(a)}}{\underbrace{\sum_{\substack{k\in 1..N \\ k\neq i}} e^{-\frac{||x_i-x_k||^2}{T}}}_{(b)}}\right) \quad (1)$$

Introduced by Srivastava and Hinton [47], the SNNL was modified and analyzed by Frosst et al. [25]. The loss characterizes the entanglement of data manifolds in representation spaces. The SNNL measures distances between points from different groups (usually the classes) relative to the average distance for points within the same group. When points from different groups are closer relative to the average distance between two points, the manifolds are said to be *entangled*. This is the opposite intuition to a maximum-margin hyperplane used by support vector machines. Given a labelled data matrix (X,Y) where Y indicates which group the data points X belong to, the SNNL of this matrix is given in Equation 1.

The main component of this loss computes the ratio between (a) the average distance separating a point x_i from other points in the same group y_i, and (b) the average distance separating two points. A temperature parameter T is introduced to give more or less emphasis on smaller distances (at small temperatures) or larger distances (at high temperature). More intuitively, one can imagine the data forming separate clusters (one for each class) when the SNNL is minimized and overlapping clusters when the SNNL is maximized.

4.2 Entangled Watermark Embedding

We present our watermarking strategy, *Entangled Watermark Embedding* (EWE), in Algorithm 1. We utilize the SNNL's ability to entangle representations for data from the task and watermarking distributions (outliers crafted by the defender using triggers). That is, we encourage activation patterns for legitimate task data and watermarked data to be similar, as

Algorithm 1: Entangled Watermark Embedding

Input: $X, Y, D_w, T, c_S, c_T, r, \alpha, loss, model, trigger$
Output: A watermarked DNN model

```
/* Compute trigger positions                          */
```
1 $X_w = D_w(c_S), Y' = [Y_0, Y_1]$;
2 map=$conv(\nabla_{X_w}(SNNL([X_w, X_{c_T}], Y', T)), trigger)$;
3 position = $\arg\max(map)$;
```
/* Generate watermarked data                          */
```
4 $X_w[position]$ = trigger;
5 $FGSM(X_w, \mathcal{L}_{CE}(X_w, Y_{c_T}))$/* optional */
6 $FGSM(X_w, SNNL([X_w, X_{c_T}], Y', T))$/* optional */
7 step = 0 /* Start training */
8 **while** *loss not converged* **do**
9 step += 1;
10 **if** *step* % *r* == *0* **then**
11 model.train($[X_w, X_{c_T}]$, Y_{c_T})/* watermark */
12 **else**
13 model.train(X, Y)/* primary task */
```
  /* Fine-tune the temperature                        */
```
14 $T^{(i)}$ -= α * $\nabla_{T^{(i)}}$SNNL$([X_w, X_{c_T}]^{(i)}, Y', T^{(i)})$;

visualized in Figure 2b. This makes watermarks robust to model extraction: an adversary querying the model on only the task distribution will still extract watermarks.

Step 1. Generate watermarks: The defender aims to watermark a model trained on the legitimate task dataset $D = \{X, Y\}$. First, they select a dataset D_w, representing the watermarking distribution, and a source class c_S from D_w. The defender samples data $X_w \sim D_w(c_S)$ to initialize watermarking, where $D_w(c_S)$ represents data from D_w with label c_S. D_w may be the same as the legitimate dataset D if we are performing in-distribution watermarking, or a related dataset if instead we are performing out-of-distribution (OOD) watermarking [4]. The defender then labels X_w with a *semantically different* target class, c_T, of D. In other words, it should be unlikely for X_w to ever be misclassified as c_T (by an un-watermarked model). Our goal is to train the model to have the special behavior that *it classifies* X_w *as* c_T, which makes it distinguishably different from un-watermarked models.

To this end, we define a *trigger*, which is an input mask (see Figure 18 (a) in Appendix A.3), and add it to each sample in X_w. Thus, X_w now contains watermarks (outliers) that can be used to watermark the model, and later, verify ownership. The trigger should not change the semantics of X_w to be similar to X_{c_T} (i.e., $D(c_T)$). For example, a poor choice of a trigger for in-distribution watermarks sampled from source class "1" of MNIST, would be a horizontal line near the top of the image (see Figure 18 (b)). This trigger might construe X_w to be semantically closer to a "7" than a "1". Such improper triggers can weaken model performance and lead to the defender falsely claiming ownership of models that were not watermarked. To avoid these issues, we determine trigger location as the area with the largest gradient of SNNL with respect to the candidate input—this is done through the convolution in the 2^{nd} line of Algorithm 1.

Optionally, a defender can optimize the watermarked data with *gradient ascent* to further avoid generating improper triggers. The goal of this gradient ascent is to perturb the input to decrease the confidence of the model in predicting the target class. This is the opposite of optimization performed by algorithms introduced to find adversarial examples, so we adapt one of these algorithms for our purpose as shown in lines 5 and 6 of Algorithm 1. Since we would like the effect of gradient ascent performed over the watermarked input to transfer between different models [45], we use the FGSM [14] which is a one-shot gradient ascent approach known to transfer better than iterative approaches like PGD [27] because it introduces larger perturbations[5]. We compute $FGSM(X_w, f(X_w)) : X'_w = X_w + \varepsilon \cdot sign(\nabla_{X_w}(f(X_w))$ where ε is the step size, and f is a function operating on X_w. In alternating steps, we define f to be $\mathcal{L}_{CE}$ of predicting X_w as the target class, c_T, by a (different) clean model, or the SNNL between X_w and X_{c_T}. The former encourages X_w to differ from X_{c_T}, and the latter makes entanglement easier (leading to more robust watermarks). We use more steps of the former to ensure X_w is semantically different from c_T.

Step 2. Modify the Loss Function. To watermark the model more robustly, we compute the SNNL at each layer, $l \in [L]$, where L is the total number of layers in the DNN, using its representation of X_w and X_{c_T}, which will allow us to entangle them. $Y' = [Y_0, Y_1]$ is arbitrary labels for $[X_w, X_{c_T}]$ respectively. We sum the SNNL across all layers, each with a specific temperature $T^{(l)}$. We multiply the sum by a weight factor κ which governs the relative importance of SNNL to the cross-entropy during . In other words, κ controls the trade-off between watermark robustness and model accuracy on the task distribution. Our total loss function is thus:

$$\mathcal{L} = \mathcal{L}_{CE}(X, Y) - \kappa \cdot \sum_{l=1}^{L} SNNL([X_w^{(l)}, X_{c_T}^{(l)}], Y', T^{(l)})) \quad (2)$$

Step 3. Train the Model. We initialize and train a model until either the loss converges or the max epochs are reached. In training, we sample r normal batches of legitimate data, X, followed by a single interleaved batch of X_w concatenated with X_{c_T}, both of which are required to entangling using the SNNL. On legitimate data X, we set $\kappa = 0$ in Equation 2 to minimize only the task (cross-entropy) loss. On interleaved data $[X_w, X_{c_T}]$ that includes watermarks, we set $\kappa > 0$ to optimize the total loss. Following Frosst et al. [12], we update T using a rate of α that is learned during training, alleviating the need to tune α as an additional hyperparameter.

[4] OOD watermarking means the watermarked data is not sampled from the task distribution

[5] Note that here we are not concerned with the imperceptibility of watermarked data so this is not a limitation in the context of our work.

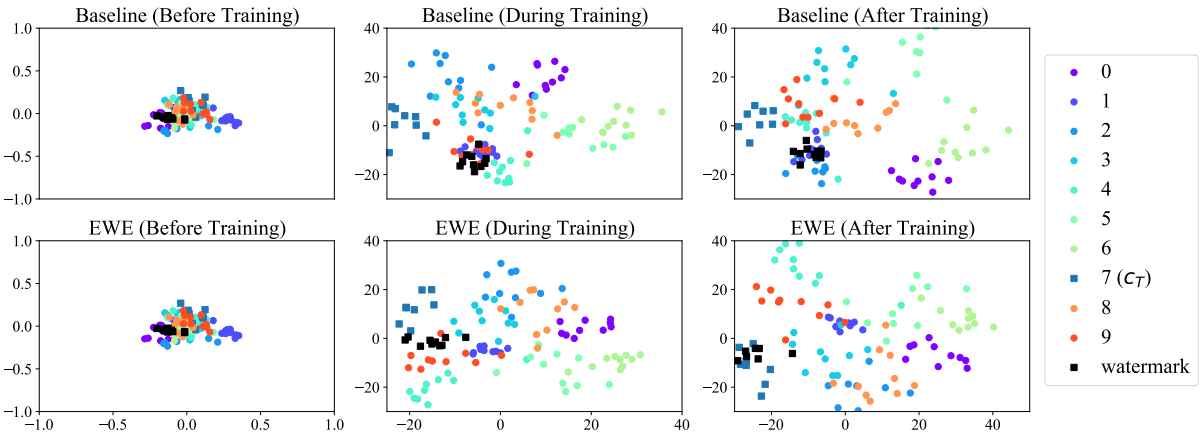


Figure 3: **Visualization of our proposed EWE entangling watermarks with data from the target class $c_T = 7$ unlike prior watermarking approaches which push these watermarks to a separate cluster.** For visualization, we use PCA [21] to project the representations of data in each model's penultimate layer onto its two principal components. We project data before (left column), during (middle column), and after (right column) training for a baseline model trained with the cross-entropy loss only (top row) and for a model trained with our proposed EWE approach (bottom row) on MNIST.

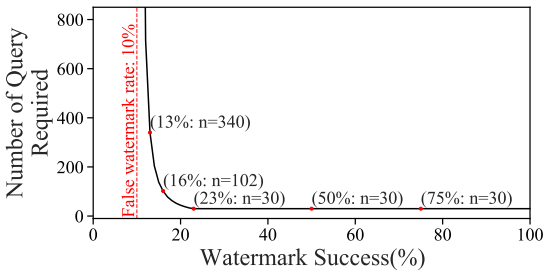


Figure 4: **A defender using a T test to claim ownership of a stolen model, with 95% confidence, needs to make increasingly more queries as the watermark success rate decreases on the stolen model**.

4.3 Validating EWE

We explore if EWE improves upon its predecessors by: (1) enabling ownership verification with fewer queries (§ 4.3.1), (2) better entangling watermarks with the classification task (§ 4.3.2), (3) being more robust against extraction attacks (§ 4.3.3), and (4) scaling to deeper larger architectures (§ 4.3.4). For all experiments in this section, the watermarked data is generated with the optional step described in § 4.2.

4.3.1 Ownership Verification

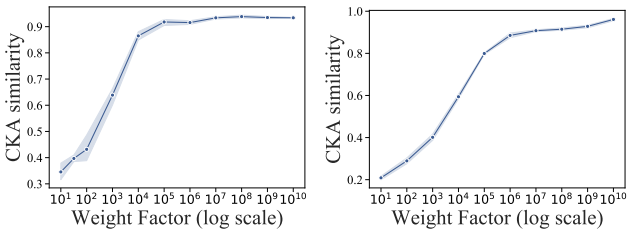
The defender may claim ownership of stolen models by statistically showing that the model's behavior differs significantly from any non-watermarked models. A T-test requires surprisingly few queries to the stolen model if the *watermark success rate* far exceeds the *false positive rate*. We denote the *watermark success rate* as the probability of a watermarked model correctly identifying watermarked data as class c_T; the *false positive rate* is the probability of a non-watermarked model classifying watermarked data as c_T.

The watermark success rate is the mean of a binomial distribution characterizing if watermarked data is classified as the target class. According to the Central Limit Theoreom (CLT), it is normally distributed when the number of queries, n, is greater than 30. If we follow the watermark generation procedures described in § 4.2, the false watermark rate should be lower than random chance, i.e., $(100/K)\%$. In Figure 4, we set the *false watermark rate* to random chance as a conservative upper bound. We often observed rates much lower than this. Figure 4 shows the number of queries needed to claim ownership, with 95% confidence, as the watermark success rate is varied. For watermark success rates above 23%, the number of queries required is quite small (i.e., 30, the minimal for CLT to be valid). As we will see in § 4.3.3, only our EWE strategy achieves these success rates after extraction. Even the lowest observed EWE success rate of 18.74% (on CIFAR-10) requires (just) under 100 queries. Figure 4 also shows that exponentially more queries are required as the watermark success rate approaches the false watermark rate—in many cases, the watermark success rate of the baseline is too low for a defender to claim ownership (see Table 1).

Note that outside this section we report the *watermark success rate* after subtracting the *false watermark rate* for ease of understanding.

4.3.2 Increased Entanglement

First, we validate the increased entanglement of EWE over the baseline by visualizing each model's representation (in its penultimate layer) of the data. In Figure 3, we train our baseline with *cross-entropy only* (top row) and another model with *EWE* (bottom row). The baseline learns watermarks naively, by minimizing the cross-entropy loss with the target



(a) MNIST dataset (b) Fashion MNIST dataset

Figure 5: **EWE is able to entangle watermarked with legitimate data because training with SNNL leads to higher CKA similarity between them.** We vary κ from 0 (the baseline) to > 0 (EWE) using a log scale.

class c_T. After training, we see that this pushes watermarked data, X_w, to a separate cluster, away from the target class c_T. Instead, EWE entangles X_w with $X(c_T)$ using the SNNL, which leads to overlapping clusters of watermarked data with legitimate data. Intuitively and experimentally, we see that EWE obtains the least separation in the penultimate hidden layer because it accumulates all previous layers' SNNL.

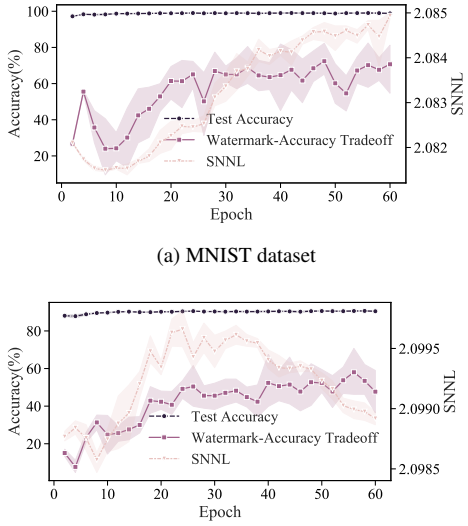
Second, similarly to what we did in § 3.2, we analyze the frequency of activation of neurons for these models, and find that there is more similarity between watermarked and legitimate data when EWE is used. The results are in Figure 2 and Figure 20 (see Appendix A.3) which shows a real-world scenario with a convolutional neural network.

Third, we analyze the similarity of their representations using central kernel alignment (CKA) [9, 25]. This similarity metric centers the distributions of the two representations before measuring alignment. In Figure 5, we see that higher levels of SNNL penalty do in fact lead to higher CKA similarity between watermarked and legitimate data (compared with $\kappa = 0$, the cross-entropy baseline). This, coupled with our first experiment, explains why EWE achieves better entanglement.

4.3.3 Robustness against Extraction

We now evaluate the robustness of EWE against retraining-based extraction attacks launched by white-box adversaries (see the top of § 4). To remove watermarks, this adversary retrains using only the cross-entropy loss evaluated only on legitimate data. We attack two victim neural networks: one with our EWE strategy and one with our baseline, which uses only the cross-entropy loss, as proposed by Adi et al. [1].

We define the watermark success rate as the proportion of X_w correctly identified as c_T. We measure the validation accuracy on a held out dataset. We report results for both models in Table 1 and find that the watermark success rate on the victim model (before retraining based extraction) is often near 100% for both EWE and the baseline. After extraction, the watermark success rate always drops. It is in this case that we observe the largest benefits of EWE (over the baseline): there is often a ≥ 20 percentage point improvement in the watermark success. Besides, we often observe a negligible



(a) MNIST dataset

(b) Fashion MNIST dataset

Figure 6: **There exists an inflection point in the model's task accuracy and the SNNL value, as training progresses**. Before that point, continuing to train generally increases the watermark success rate relative to the task accuracy (we report the ratio between variations of the two).

decrease in validation accuracy: an average of 0.81 percentage points with a max of 3 for the ResNet on Fashion MNIST.

Our main result is that we can achieve watermark success rates between 18% and 60% with an average of 38.39%; the baseline is between 0.3% and 9% with an average of 5.77%. There is a minimal 0.81 percentage point degradation on average of validation accuracy compared to the baseline, with a maximum of 3 percentage points for a ResNet on Fashion MNIST. These watermark success rates allow us to claim ownership with 95% confidence with < 100 queries (see § 4.3.1).

We also validate that continuing to maximize the SNNL during training is beneficial. In Figure 6 we see that continued training improves the watermark robustness and task accuracy trade-off, until it plateaus near 60 epochs. We measure this trade-off as the ratio between the increase of the watermark success rate and the decrease of the task accuracy.

4.3.4 Scalability to Deeper Architectures

Entangling watermarks with legitimate data enables, and even forces, earlier layers to learn features that recognize both types of data simultaneously, as seen in Figure 2. This explains the improved robustness of watermarks. With entanglement, only later layers need to use capacity to separate between the two types of data, preserving model accuracy. This setup should work better for deeper models: there is only more capacity to learn shared features for watermarks and legitimate data. Our results in Figure 20 in Appendix A.3 confirms this.

However, deeper models such as ResNets often benefit (in their validation accuracy) from linearity: residual connec-

Dataset	Method	Victim Model		Extracted Model	
		Validation Accuracy	Watermark Success	Validation Accuracy	Watermark Success
MNIST	*Baseline*	99.03(±0.04)%	99.98(±0.03)%	98.79(±0.12)%	0.31(±0.23)%
	EWE	98.91(±0.13)%	99.9(±0.11)%	98.76(±0.12)%	65.68(±10.89)%
Fashion MNIST	*Baseline*	90.48(±0.32)%	98.76(±1.07)%	89.8(±0.38)%	8.96(±8.28)%
	EWE	90.31(±0.31)%	87.83(±5.86)%	89.82(±0.45)%	58.1(±12.95)%
Speech Command	*Baseline*	98.11(±0.35)%	98.67(±0.94)%	97.3(±0.43)%	3.55(±1.89)%
	EWE	97.5(±0.44)%	96.49(±2.18)%	96.83(±0.45)%	41.65(±22.39)%
Fashion MNIST (ResNet)	*Baseline*	91.64(±0.36)%	75.6(±15.09)%	91.05(±0.44)%	5.68(±11.78)%
	EWE	88.33(±1.97)%	94.24(±5.5)%	88.27(±1.53)%	24.63(±17.99)%
CIFAR10	*Baseline*	85.82(±1.04)%	19.9(±15.48)%	81.62(±1.74)%	7.83(±14.23)%
	EWE	85.41(±1.01)%	25.74(±8.67)%	81.78(±1.31)%	18.74(±12.3)%
CIFAR100	*Baseline*	54.11(±1.89)%	8.37(±13.44)%	47.42(±2.54)%	8.31(±15.1)%
	EWE	53.85(±1.07)%	67.87(±10.97)%	47.62(±1.41)%	21.55(±9.76)%

Table 1: **Performance of the baseline approach (i.e., minimize cross-entropy of watermarks with the target class) vs. the proposed watermarking approach (EWE)**. For each dataset, we train a model with each approach and extract it by having it label its own training data. We measure the validation accuracy and watermark success rates, i.e., difference between percentage of watermarks classified as the target class on a watermarked versus non-watermarked model. Both techniques perform well on the victim model, so the intellectual property of models whose parameters are copied directly can be claimed by either technique. However, the baseline approach fails once it is extracted whereas EWE reaches significantly higher watermark success rate.

tions which add the input of the residual block directly to the output [17]. Notice that watermarks (e.g. a "1" with a small square trigger) are easily separable from legitimate data of the target class (e.g. a "9") and from the source class (e.g., a "1" without the trigger) because they share (nearly) no common features—they are outliers. Hence, residual connections pose a greater problem for entanglement because there are often no shared features, and forcing the watermarks (by increasing κ) to entangle with the legitimate data of c_T may cause the model to misclassfy X_{c_S} and X_{c_T}.

Our results validate this intuition. We see in Figure 19 in Appendix A.3 that deep convolutional neural networks can still entangle watermarks but yet we find that comparable ResNets cannot. Thus, we use our OOD watermarks (see Step 1 of § 4.2) because forcing them to entangle with X_{c_T} has a lesser impact on accuracy. Though difficult to entangle, they achieve sufficient watermark success for claiming ownership (see Table 1). Even for more difficult tasks, as expected, EWE outperforms the baseline (see CIFAR-100 in Table 1), but both see a significant drop in watermark success. Finally, we see that watermarking is sensitive to the number of classes, in particular, EWE (see Figure 24 in Appendix A.3), probably due to complexity of the representation space.

5 Calibration of Watermark Entanglement

Through the calibration of EWE for four vision datasets (MNIST [28], Fashion MNIST [55], CIFAR-10, CIFAR-100 [26]), and an audio dataset (Google Speech Commands [54]), we answer the following questions: (1) what is the trade-off between watermark robustness and task accuracy?; (2) how should the different parameters of EWE be configured?; and (3) is EWE robust to backdoor defenses and attacks against watermarks? Our primary results are:

1. For MNIST, Fashion MNIST, and Speech Commands (by which we validate if EWE is independent of the domain), we achieved watermark success above 40% with less than 1 percentage point drop in test accuracy. For CIFAR datatsets, watermark success above 18% is reached with a minimal accuracy loss of < 1.5 percentage points. The weight factor allows the defender to control the trade-off between watermark robustness and task accuracy.
2. The ratio of watermarks to legitimate data during training, the choice of source-target class pair, and the choice of points to be watermarked all affect the performance of EWE significantly; temperature does not since it is automatically optimized during training as described in § 4.2. Refer to Appendix A.1 for more details.
3. Defenses against backdoors like pruning, fine-pruning, and Neural Cleanse are all ineffective in removing EWE.

5.1 Experimental Setup

We chose to evaluate EWE on four datasets in addition to MNIST. While CIFAR-10 and CIFAR-100 are used to test the scalability of EWE as described in § 4.3.4, we use Fashion MNIST because its classes are much harder to linearly separate than MNIST, making it a good benchmark for learning a more complex task, with comparable computational cost to MNIST. Thus it allows us to tune the hyperparameters efficiently to explore behaviors of EWE. Further, it shows that EWE works well when the task naturally contains ambiguous inputs across pairs of classes. We also evaluated EWE on Google Speech Commands, an audio dataset for speech recognition, because speech recognition is one of the applications where ML is already pervasively deployed across industry.

Datasets. **1. MNIST** is a dataset of hand-written digits (from 0 to 9) with 70,000 data points [28], where each data point is a gray-scale image of shape 28×28. When needed, we sampled OOD watermarked data from Fashion MNIST.

2. Fashion MNIST is a dataset of fashion items [55]. It can be used interchangeably with MNIST. Because the task is more complex, models achieving $> 99\%$ accuracy on MNIST however only reach $> 90\%$ on Fashion MNIST. When needed, we sampled OOD watermarked data from MNIST.

3. Google Speech Commands is an audio dataset of 10 single spoken words [54]. The training data has about 40,000 samples. We pre-processed the data to obtain a Mel Spectrogram [7]. We tried two methods for generating watermarks both using in-distribution data: (a) modifying the audio signal, or (b) modifying the spectrogram. For (a), we sample data from the source class and overwrite $\frac{1}{8}^{th}$ of the total length of the sample (i.e., 0.125 seconds) with a sine curve, as shown in Figure 26; for (b), each audio sample is represented as an array of size 125×80. We then define the trigger to be two 10×10-pixel squares at both the upper right and upper left-hand corners in case of vanishing or exploding gradients. It was observed that the choice of using (a) or (b) *does not influence the performance of EWE.*

4. CIFAR-10 consists of 60,000 32×32×3 color images equally divided into 10 classes [26], while 50,000 is used for training and 10,000 is used for testing. When needed, we use OOD watermarks sampled from SVHN [39].

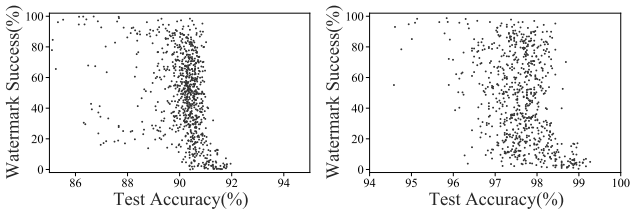
5. CIFAR-100 is very similar to CIFAR-10, except it has 100 classes and there are 600 images for each class [26]. When needed, we use OOD watermarks sampled from SVHN [39].

Architectures. We use the following architectures:

1. Convolutional Neural Networks are used for MNIST and Fashion MNIST. The architecture is composed of 2 convolution layers with 32 5×5 and 64 3×3 kernels respectively, and 2×2 max pooling. It is followed by two fully-connected (FC) layers with 128 and 10 neurons respectively. All except the last layers are followed by a dropout layer to avoid overfitting. When implementing EWE, the SNNL is computed after both convolution layers and the first FC layer.

2. Recurrent Neural Networks are used for Google Speech Command dataset. The architecture is composed of 80 long short-term memory (LSTM) cells of 128 hidden units followed by two FC layers of 128 and 10 neurons respectively. When applying EWE, the SNNL is computed after the 40^{th} cell, the last (80^{th}) cell, and the first FC layer.

3. Residual Neural Network (ResNet) [17] are used for Fashion MNIST, CIFAR-10, and CIFAR-100 datasets. We use ResNet-18 which contains 1 convolution layer followed by 8 residual blocks (each containing 2 convolution layers), and ends with a FC layer. It is worth noting that the input to a residual block is added to its output. We compute SNNL on the outputs of the last 3 residual blocks.



(a) Fashion MNIST (b) Speech Command

Figure 7: **Watermark success versus model accuracy on the task**. Each point corresponds to a model trained with uniformly-sampled hyperparameters. As test accuracy increases, it becomes harder to have robust watermarks.

5.2 No Free Lunch: Watermark vs. Utility

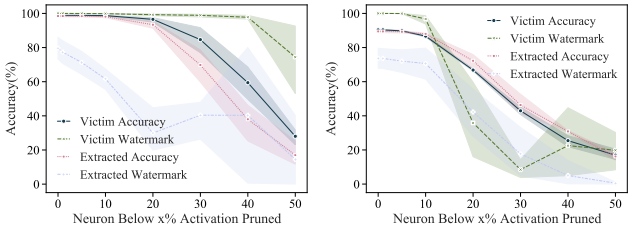
We study the tension between accuracy on the task's distribution and robustness of the watermarks: if the defender wants to claim ownership of a model, they would like this model to predict their chosen label on the watermarks as frequently as possible while at the same time minimizing the impact of watermarks on the model's performance when presented with samples from the task distribution.

To systematically explore the trade-off between successfully encoding watermarks and correctly predicting on the task distribution, we first perform a comprehensive grid search that considers all hyper-parameters relevant to our approach: the class pairs (c_S, c_T) (note that c_S is a class from another dataset when OOD watermark is used), the temperature T, the weight ratio κ, and the ratio of task to watermark data (i.e. r in Algorithm 1), how close points have to be to the target class to be watermarked. In Appendix A.1, we perform an ablation study on the impact of each of these parameters: they can be used to control the trade-off.

Each point in Figure 7 corresponds to a model trained using EWE with a set of hyper-parameters. For the Fashion MNIST dataset shown in Figure 7 (a), the tendency is exponential: it becomes exponentially harder to improve accuracy by decreasing the watermark success rate. In the Speech Commands dataset, as shown in Figure 7 (b), there is a large number of points with nearly zero watermark success. This means it is harder to find a good set of hyperparameters for the approach. However, there exists points in the upper right corner demonstrating that certain hyperparameter values could lead to robust watermark with little impact on test accuracy.

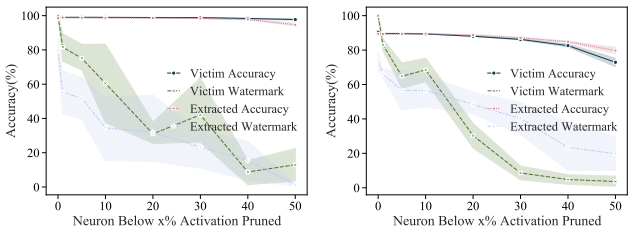
5.3 Evaluation of Defenses against Backdoors

Pruning. Since backdoors and legitimate task data activate different neurons, pruning proposes to remove neurons that are infrequently activated by legitimate data to decrease the performance of potential backdoors [31]. Given that neurons less frequently activated contribute less to model predictions on task inputs, pruning them is likely to have a negligible effect. Since watermarks are a form of backdoors, it is natural



(a) MNIST (b) Fashion MNIST

Figure 8: Task accuracy and watermark success rate on the extracted model in the face of a pruning attack. For both datasets, bringing the watermark success rate below 20% comes at the adversary's expense: accuracy drop of more than 40 percentage points.



(a) MNIST (b) Fashion MNIST

Figure 9: Task accuracy and watermark success rate on the extracted model in the face of a fine pruning attack. Despite a more advantageous trade-off between watermark success rate and task accuracy, the adversary is unable to bring the watermark success rate sufficiently low to prevent the defender to claim ownership (see § 4.3.1) until 40% neurons are fine-pruned. Beyond this point, fine-pruning more neurons would lead to loss in the extracted model's accuracy.

to ask whether pruning can mitigate EWE.

We find this is not the case because watermarks are entangled to the task distribution. Recall Figure 2b, where we illustrated how EWE models have similar activation patterns on watermarked and legitimate data. Thus, neurons encoding the watermarks are frequently activated when the model is presented with legitimate data. Hence, if we extract a stolen model and prune its neurons that are activated the least frequently, we find that watermark success rate remains high despite significant pruning (refer Figure 8). In fact, the watermark success rate only starts decreasing below 20% when the model's accuracy on legitimate data also significantly decreases (by more than 40 percentage points). Such a model becomes useless to the adversary, who would be better off training a model from scratch. We conclude that pruning is ineffective against EWE.

Fine Pruning. Fine pruning improves over pruning by continuing to train (i.e., fine-tune) the model after pruning [31]. This helps recover some of the accuracy that has been lost during pruning. In the presence of backdoors, this also contributes to overwriting any behavior learned from backdoors.

We also analyze EWE in the face of fine pruning. We first extract the model by retraining (i.e., randomly initialize model weights and train them with data labeled by the victim model), prune a fraction of neurons that are less frequently activated, and then train the non-pruned weights on data labeled by the victim model. Results are plotted in Figure 9. In the most favorable setting for fine pruning, watermark success rate on the extracted model remains around 20% before harming the utility of the model, which is still enough to claim ownership—as shown in § 4.3.1. This is despite the fact that 50% of the architecture's neurons were pruned. Since the data used for fine-tuning is labeled by the watermarked victim model, it contains information about the watermarks even when the labels provided are for legitimate data.

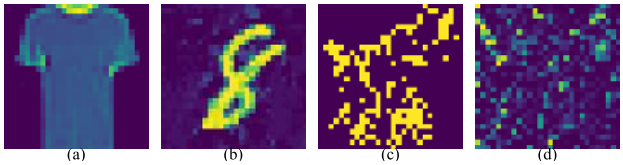
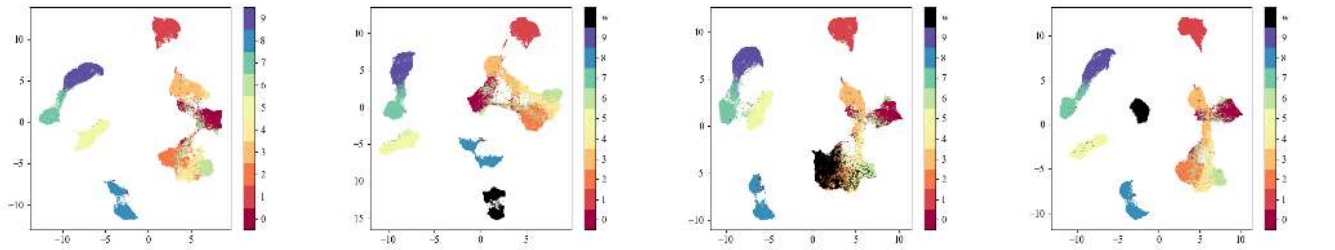


Figure 10: Neural Cleanse leverages the intuition that triggers may be recovered by looking for adversarial examples for the target class. To illustrate this, we have here a legitimate input of the target class (a), an example of a watermark (b), an adversarial example (see Appendix A.2 for details) intialized as a blank image and perturbed to be misclassified by the extracted model in the target class(c), and the backdoor candidate recovered by Neural Cleanse (d). If either (c) or (d) were similar to the watermark, this would enable us to recover the watermarked data and then use this knowledge to remove the watermark as described in § 6. However, this is not the case for models extracted from a EWE defended victim model: the watermark proposed (c and d) is different from the trigger used by EWE (b).

Neural Cleanse. Neural Cleanse is a technique that detects and removes backdoors in deep neural networks [53]. The intuition of this technique is that adding a backdoor would cause the clusters of the source and target classes to become closer in the representation space. Therefore, for every class c of a dataset, Neural Cleanse tries to perturb data from classes different to c in order to have them misclassified in class c. Next, the class requiring significantly smaller perturbations to be achieved is identified as the "infected" class (i.e., the class which backdoors were crafted to achieve as the target class). In particular, the authors *define a model as backdoored if an anomaly index derived from this analysis is above a certain threshold* (set to 2). The perturbation required to achieve this class is the recovered trigger. Once both the target class and trigger have been identified, one can remove the backdoor by retraining the model to classify data with the trigger in the correct class, à la adversarial training [50].

To analyze the robustness of EWE to Neural Cleanse, we compare the performance of a model watermarked with EWE and a baseline model watermarked by minimizing the cross-



(a) Un-watermarked Model (b) Watermarked Model (Baseline) (c) EWE In-distribution Watermark (d) EWE Out-distribution Watermark

Figure 11: **Change in the distance among clusters of data from different Fashion MNIST classes following watermarking**. The four subplots are made using four different approaches specified by the sub-captions. In (c) and (d), $c_S = 8$ and $c_T = 0$, while D_w is MNIST for (d). Each point in the plot represents an output vector of the last hidden layer. These representations are plotted in 2-D using UMAP dimensionality reduction to preserve global distances [34]. Comparing (a) and (b), one can observe that the clusters of class 8 and 0 become closer in (b) while the distances among the other classes remain similar. This is why such watermarked model can be detected by Neural Cleanse [53], which searches for pairs of classes that are easily misclassified with one another. In contrast, EWE with either in or out of distribution watermarks does not influence this distance significantly, which makes it more difficult for Neural Cleanse to detect the watermark.

entropy of watermarks labeled as the target class ($\kappa = 0$ in Equation 2). We compute the anomaly index of the EWE and baseline models. If the anomaly index is above 2, the model is detected as being watermarked (i.e., backdoored in [53]). On the Fashion MNIST (see Figure 10), EWE exbhibits an average anomaly index of 1.24 (over 5 runs) that evades detection whereas the baseline model has an average index of 8.84. This means that Neural Cleanse is unable to identify our watermark and its trigger.

It is worth noting: (a) Neural Cleanse considers the problem of backdooring the entire set of classes (i.e., all classes are considered as source classes), and (b) backdoor attacks usually aim at minimal perturbation to the inputs. While being similar to legitimate data from all classes and labeled as a specific class, such backdoors changes the decision surface significantly, which would be detected by Neural Cleanse. In EWE, we insert watermarks only for a single source-target class pair. Besides, watermarked data is not restricted by the degree of perturbation and could even be OOD. Thus entangling it with c_T does not change the decision boundary between c_T and other classes, as shown in Figure 11 (and Figure 22, 23 for MNIST and Speech Command in Appendix A.3). This makes it hard for Neural Cleanse to detect EWE watermarks.

6 Robustness to Adaptive Attackers

Recall from our threat model (see the top of § 4) that the adversary has no knowledge of the parameters used to calibrate the watermarking scheme (such as κ and $T^{(1)} \cdots T^{(L)}$ in Algorithm 1) nor the specific trigger used to verify watermarking. In this section, we explore when the adversary has more resources and knowledge than stated in the threat model.

6.1 Knowledge of EWE and its parameters

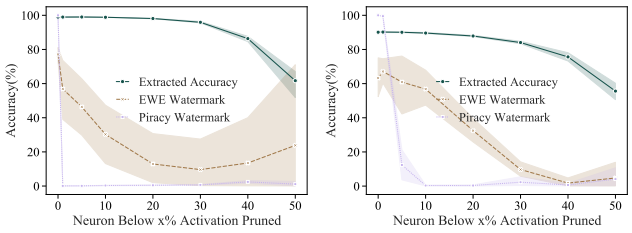
Knowledge of the parameters used to configure EWE defeats watermarking, as expected. The robustness of EWE relies on *maintaining the secrecy* of the trigger and watermarking parameters to protect the intellectual property contained in the model. If the adversary knows the trigger used to watermark inputs, they could refuse to classify any input that contains that trigger (denial-of-service). Alternatively, they could extract the model while instead minimizing the SNNL of the watermarks and legitimate data of class c_T. Note, minimizing SNNL corresponds to disentangling. Additionally, adversaries may also be able to retrain the triggers (and thus, watermarks) to predict the correct label.

Any of these results in complete removal of watermarks However, this is not a realistic threat model since the adversary should only know that EWE was used as a watermarking scheme (see (e) in our threat model defined in § 4. In this way, parameters of EWE play a similar role to cryptographic keys. Next, we evaluate EWE against several more realistic adaptive attacks against watermarks such as piracy attacks.

6.2 Knowledge of EWE only

With knowledge of EWE but not its configuration (e.g., the source and target classes), the adversary can still adapt in several ways. We evaluate four adaptive attacks.

Disentangling Data. We conjecture that the adversary could perform extraction by minimizing SNNL to disentangle watermarks from task data. We assumed a strong threat model such that the adversary has knowledge of all the parameters of EWE (including the trigger if in-distribution watermark is used, and the OOD dataset if OOD watermark is used) except the source and target classes. Thus, the adversary guesses a pair of classes, constructs watermarked data following EWE, and extracts the model while using EWE with $\kappa < 0$ to disentangle the purported watermark data and legitimate data from



(a) MNIST (b) Fashion MNIST

Figure 12: Task accuracy and watermark success rate after fine-pruning on the extracted model with a pirate watermark. With less than 10% neurons pruned, the pirate watermark is removed while the owner's watermark remains.

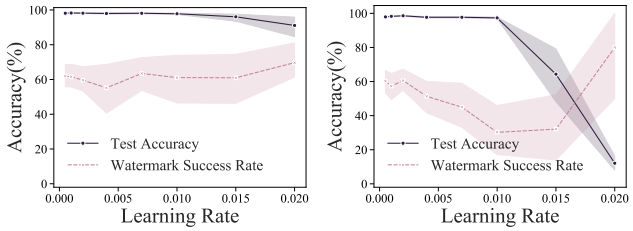
the purported target class. Following such a procedure, we observe that the watermark success of the extracted model on Fashion MNIST drops from 48.81% to 22.82% if the guess does not match with the true source-target pair, and to 6.34% if the guess is correct.. On MNIST, watermark success drops from 41.62% to 30.14% when the guess is wrong, and to 0.08% otherwise. The results from the Speech Commands dataset have large variance, but follow a similar trend: the watermark success drops to an average of 16.81% due to the attack. Thus, while watermark success rates are lowered by this attack, the defender is still able to claim ownership when the adversary guesses the source-target pair incorrectly with about 30 queries for the two vision datasets, and near 100 queries for Speech Commands. Furthermore, observe that guessing the pair of classes correctly requires significant compute to train models corresponding to the $K(K-1)$ possible source-target pairs where K is the number of classes in the dataset , which defeats the purpose of model extraction.

Piracy Attack. In a piracy attack, the adversary embeds their own watermark with EWE so that the model is watermarked twice—it becomes ambiguous to claim ownership through watermarks. To remove the pirate watermark, we propose to fine-prune [31] the extracted model on data labeled by the victim model. As shown in Figure 12, the owner's watermark is not removed as we discussed fine pruning in § 5.3, whereas the pirate watermark would be removed (even if the adversary uses EWE) because data labelled by the victim model does not contain information about the pirate watermark. The adversary cannot do the same to remove the owner's watermark because this requires access to a dataset labeled by another source, at which point the cost of piracy defeats model stealing: the adversary could have trained a model on that dataset and would not benefit from model stealing.

Anomaly Detection. Imagine the case of an extracted model deployed as an online ML API. The adversary may know (or suspect) the model to be watermarked, so they may decide to implement an anomaly detector to filter queries containing data watermarked by EWE and respond to them with a random prediction. By doing so, even though the parameters still embed the watermarks, the adversary could still prevent the defender from claiming ownership.

Method	Accuracy Loss	Detected Watermark
LOF	7.00(±0.3)%	99.93(±0.03)%
Isolation Forest	8.64(±0.32)%	92.82(±1.32)%

Table 2: Proportion of watermarks detected and accuracy loss when anomaly detectors filter suspicious inputs.



(a) Finetune fully connected layers (b) Finetune all layers

Figure 13: Task accuracy and watermark success rate of the extracted model after transfer learning from GTSRB to LISA. Even fine-tuning all the layers does not remove watermarks.

We tested two common anomaly detectors on Fashion MNIST: Local Outlier Factor (LOF) [4] and Isolation Forest [30], on activations of the last hidden layer. Results are shown in Table 2. Both detectors are able to detect more than 90% of watermarked data. However, this comes at the cost of identifying parts of the validation dataset as outliers and results in a sharp accuracy drop of 7.0 and 8.64 percentage points respectively. This may be due to the curse of dimensionality [23]: it is harder to learn higher dimensional distribution. Indeed, it is worth noting that anomaly detectors on hidden layers consistently work better than on the inputs themselves.

Transfer Learning. The adversary may also transfer knowledge of the extracted model to another dataset in the same domain [42] with the hope of disassociating the model from EWE's watermark distribution. To evaluate if watermarks persist after transfer learning, we chose two datasets in the same domain. The victim model is trained on the German Traffic Sign Dataset (GTSRB) [48] and we transferred the extracted model to the LISA Traffic Sign Dataset [36]. We fine-tune either (a) only the fully connected layers, or (b) all layers for the same number of epochs that the victim model was trained for. Before we verify the watermark, the output layer of the transferred model is replaced to match the dimension of the victim model (they may differ) [1].

As shown in Figure 13, (a) achieves an accuracy of up to 98.25% but leaves the watermark unaffected; (b) reaches an accuracy of 98.56% and begins to weaken the watermark as one increases the learning rate. However, the pretrained knowledge is lost due to large learning rate values before the watermark is removed. This is consistent with observations in prior work [1]. We also note that transfer learning requires that the adversary have access to additional training data and

perform more training steps, so it is expected that our ability to claim model ownership will be weaker.

Take-away. The adversary also faces a no free lunch situation. They cannot adapt with disentanglement, piracy, anomaly detection, or transfer learning, and remove EWE watermarks, unless they sacrifice the stolen model's utility.

7 Discussion

Hyperparameter Selection. Our results suggest that the watermarking survivability comes at a nominal cost (about 0.81% in accuracy degradation). Yet, this value varies depending on the dataset and the hyperparameters used for training (which themselves also depend on the dataset) as we explore in Appendix A.1. Determining the relationship with relevant properties of the dataset is future work.

Computational Overheads. Our experiments suggest that the size of the watermarked dataset should be $2\times$ less than the size of the legitimate dataset. However, this implies that the model is now trained on $1.5-2\times$ more data than before. While this induces additional computational overheads, we believe that the trade-offs are advantageous in terms of proving ownership. A more detailed analysis is required to understand if the same phenomenon exists for more complex tasks with larger datasets.

Improving Utility. EWE utilizes the SNNL to mix representations from two different distributions; this ensures the activation patterns survive extraction. However, this is at a nominal expense to the utility; for certain applications, such a decrease in utility (even if small) is not desired. We believe that the same desired properties could be more easily achieved if one were to replace ReLU activations with the smoother Sigmoid activations while computing the SNNL.

Algorithmic Efficiency. In Algorithm 1, we modified the loss function by computing the SNNL at every layer of the DNN. However, it may not be necessary to do so. In Figure 20, we plot the activation patterns of hidden layers of a model trained using EWE; we observe that adding the SNNL to just the last layers provides the desired guarantees. Additionally, we observe a slight increase in model utility when not all layers are entangled. A detailed understanding of how one can choose the layers is left to future work.

Scalability and Future Research Directions. As mentioned in § 4.3.4, EWE suffers in terms of trade-off between model performance and watermark robustness when we scale to deeper architectures, and more complex datasets. Given the results on CIFAR-100, more work may be needed to scale the current method to larger datasets. According to Figure 24 (in Appendix A.3), the performance of EWE is impacted by the number of classes. We suspect this may be due to the representation space being more complicated (i.e. there are more clusters), making it more difficult to entangle two arbitrarily chosen clusters. Thus, a potential next step would be to investigate the interplay between the design of triggers to control the cluster of watermarked data; and the similarity structures and orientation of the representation space to choose source and target classes accordingly.

Another possible improvement is to use *m*-to-*n* watermarking. In this work, we focused on 1-to-1 watermarking, which watermarks one class of data and entangles it with another class. However, as long as the watermarked model behaves significantly differently from a clean model, the model owner could choose to watermark *m* classes of data, entangle them with *n* other classes, and claim ownership by following the similar verification process as described in § 4.3.1.

8 Conclusions

We proposed Entangled Watermark Embedding (EWE), which forces the model to entangle representations for legitimate task data and watermarks. Our mechanism formulates a new loss involving the Soft Nearest Neighbors Loss, which when minimized increases entanglement. Through our evaluation on tasks from the vision and audio domain, we show that EWE is indeed robust to not only model extraction attacks, but also piracy attacks, anomaly detection, transfer learning, and efforts used to mitigate backdoor (poisoning) attacks. All this is achieved while preserving watermarking accuracy, with (a) a nominal loss in classification accuracy, and (b) $1.5-2\times$ increase in computational overhead. Scaling EWE to complex tasks without great accuracy loss remains as an open problem.

Acknowledgments

The authors would like to thank Carrie Gates for shepherding this paper. This research was funded by CIFAR, DARPA GARD, Microsoft, and NSERC. VC was funded in part by the Landweber Fellowship.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, August 2018.

[2] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding robustness to support vector machines against adversarial reverse engineering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 2014.

[3] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, August 2019.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[5] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Model extraction and active learning. *CoRR*, abs/1811.02054, 2018.

[6] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*. ACM, 2020.

[7] Keunwoo Choi, Deokjin Joo, and Juho Kim. Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras. In *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.

[8] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[9] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

[10] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *arXiv e-prints*, page arXiv:1804.00750, Apr 2018.

[11] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[12] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and Improving Representations with the Soft Nearest Neighbor Loss. *arXiv e-prints*, page arXiv:1902.01889, Feb 2019.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, December 2014.

[15] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv e-prints*, page arXiv:1708.06733, August 2017.

[16] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[18] Xuedong Huang, James Baker, and Raj Reddy. A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103, 2014.

[19] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High-Fidelity Extraction of Neural Network Models. *arXiv e-prints*, page arXiv:1909.01838, Sep 2019.

[20] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. *arXiv e-prints*, Apr 2018.

[21] Ian Jolliffe. *Principal Component Analysis*. Springer, 2002.

[22] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. Watermarking techniques for intellectual property protection. In *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, New York, NY, USA, 1998. Association for Computing Machinery.

[23] Eamonn Keogh and Abdullah Mueen. *Curse of Dimensionality*, pages 314–315. Springer, Boston, MA, 2017.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations ICLR 2015*, 2015.

[25] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. *The 36th International Conference on Machine Learning*, 2019.

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[27] Alexey Kurakin, J. Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *5th International Conference on Learning Representations*, 2017.

[28] Y. Lecun and C. Cortes. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[29] T. Lee, B. Edwards, I. Molloy, and D. Su. Defending against neural network model stealing attacks using deceptive perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 43–49, 2019.

[30] F. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *8th IEEE International Conference on Data Mining*, 2008.

[31] K. Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *21st International Symposium on Research in Attacks, Intrusions, and Defenses*, 2018.

[32] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647. ACM, 2005.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*, 2018.

[34] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[35] Smitha Milli, L. Schmidt, A. Dragan, and M. Hardt. Model reconstruction from model explanations. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[36] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.

[37] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[38] Yuki Nagai, Y. Uchida, S. Sakazawa, and Shin'ichi Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*, 7:3–16, 2018.

[39] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *24th International Conference on Neural Information Processing Systems*, 2011.

[40] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[41] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish K. Shevade, and Vinod Ganapathy. A framework for the extraction of deep neural networks by leveraging public data. *CoRR*, abs/1905.09165, 2019.

[42] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[43] Nicolas Papernot, P. McDaniel, Ian J. Goodfellow, S. Jha, Z. Y. Celik, and A. Swami. Practical black-box attacks against machine learning. *ACM Asia Conference on Computer and Communications Security*, 2017.

[44] Nicolas Papernot, P. McDaniel, S. Jha, Matt Fredrikson, Z. Y. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *1st IEEE European Symposium on Security and Privacy*, 2016.

[45] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv e-prints*, page arXiv:1605.07277, May 2016.

[46] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[47] R. Salakhutdinov and Geoffrey E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *11th International Conference on Artificial Intelligence and Statistics*, 2007.

[48] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[49] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ACL, 2019.

[50] Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[51] Florian Tramèr, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.

[52] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5032–5041. PMLR, 2018.

[53] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, B. Viswanath, H. Zheng, and B. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.

[54] Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv e-prints*, page arXiv:1804.03209, Apr 2018.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, page arXiv:1708.07747, Aug 2017.

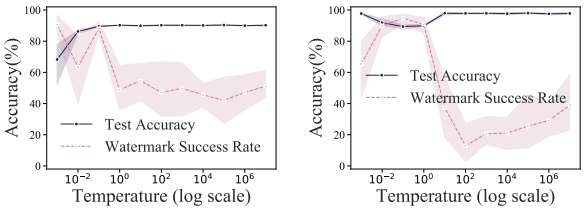
[56] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, M. P. Stoecklin, H. Huang, and I. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018.

A Appendix

A.1 Finetuning the hyperparameters of EWE

Next, we dive into details of each hyperparameter of EWE and perform an ablation study.

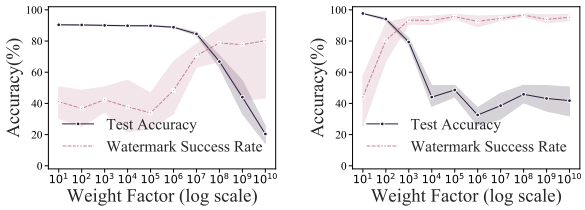
Temperature. Temperature is a hyperparameter introduced by Frosst et al [12]. It could be used to control which distances between points are more important: at small temperatures, small distances matter more than at high temperatures, where large distances matter most. In our experiments, we found that the influence of temperature on the robustness of watermark is not significant: a nice initialization leads to high watermark success, whereas other initialization results in watermark success high enough for claiming ownership, as shown in Figure 14. We conjecture that this is because EWE fine-tunes the temperature by gradient descent during training (see the last line of Algorithm 1).



(a) Fashion MNIST (b) Speech Commands

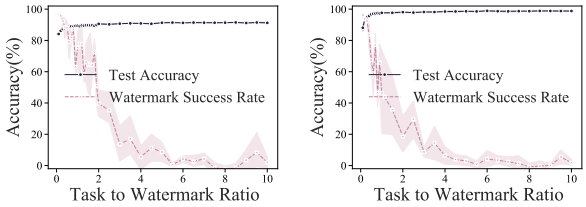
Figure 14: EWE is unlikely to fail due to setting the temperature, but certain initialization of temperature does lead to better trade-off between task accuracy and watermark success rate. Note the temperature is plotted on log scale.

Weight Factor. As defined in Algorithm 1, the loss function is the weighted sum of a cross entropy term and SNNL term. The weight factor κ is a hyper-parameter that controls the importance of learning the watermark task (by maximizing the SNNL) relatively to the classification task (by minimizing cross entropy loss). As shown in Figure 15, factors larger in magnitude cause the watermark to be more robust, at the expense of performance on the task. At the left-hand side of the figure, with a weight factor in the magnitude of



(a) Fashion MNIST (b) Speech Commands

Figure 15: Increasing the absolute value of the weight factor κ promotes watermark success rate (more importance is given to the SNNL) at the expense of lower accuracy on the task. Note that κ is plotted on log scale.



(a) Fashion MNIST (b) Speech Commands

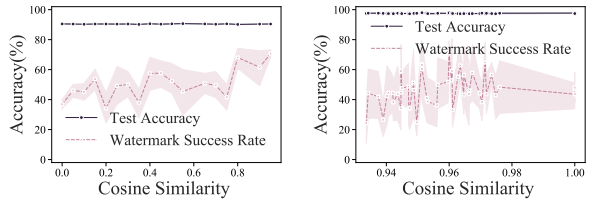
Figure 16: Decreasing the ratio r of task data to watermarks promotes watermark success rate (more importance is given to the SNNL) at the expense of lower accuracy on the task.

10, the accuracy is similar to an un-watermarked model, while watermark success is about 40%. In contrast, when the weight factor is getting larger, watermark success approaches to 100% but the accuracy decreases significantly..

Ratio of task data to watermarks. Denoted by r in Algorithm 1, this ratio also influences the trade-off between task accuracy and watermark robustness. In Figure 16, we observe that lower ratios yield more robust watermarks. For instance, we found for Fashion MNIST that the watermark could be removed by model extraction if the ratio is greater than 3, whereas task accuracy drops significantly for ratios below 1.

Source-Target classes Source and target classes are denoted by c_S and c_T in Algorithm 1. Note that we use OOD watermarks (data from MNIST) for Fashion MNIST, so c_S refers to a class of MNIST. We name class center the average of data from each class. In Figure 17, we plot the performance of EWE with respect to the cosine similarity among centers of different source-target pairs (detailed performance of different pairs can be found in Figure 25 in the appendix).

Classes with similar structures enable more robust watermarks at no impact on task accuracy. This is because data from similar classes is easier to entangle (i.e. the SNNL is easier to maximize). Cosine similarity between class centers is a heuristic to estimate this and its effectiveness depends on the dataset. For Fashion MNIST, one could observe a trend that higher cosine similarity leads to more robust watermarks. Instead, the difference among classes are less significant in Speech Command so this heuristic may not be useful.



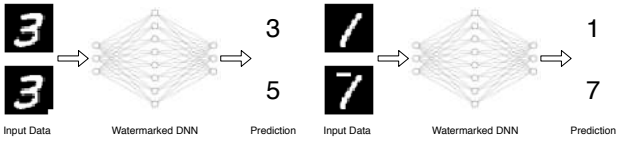
(a) Fashion MNIST (b) Speech Commands

Figure 17: Impact of similarity of classes on robustness of watermarks: We computes the average cosine distances between data of different pairs of classes and use them as source and target classes to watermark the model. It could be seen that similar classes lead to higher watermark success on Fashion MNIST, but no clear trend is observed for Speech Command.

A.2 Evasion Attacks for Detection

Adversarial examples (or samples) are created by choosing samples from a source class and perturbing them slightly (adding a carefully crafted perturbation) to ensure targeted (the mistake is chosen) or untargeted (the mistake is any incorrect class) misclassification. To do so, some attacks use gradients [27, 33, 44] or pseudo-gradients [52] to create adversarial samples with minimum perturbation. We wish to understand if mechanisms used to generate adversarial samples can be used to detect watermarks, as both produce the same effect (targeted misclassification). The intuition is that if one adversarial examples are generated from blank input and perturbed to the target class, they may reveal some information about the watermarked data. To this end, we utilize the approach proposed by Papernot et al. [44] on the extracted model to generate adversarial examples, and compare them with the watermarked data generated by EWE. Examples of watermarked data and adversarial samples we generated are shown in Figure 10 b and (c) respectively. The average cosine similarity between the adversarial examples and watermarked data is about 0.3, whereas it could reach about 0.4 when comparing to a uniformly distributed random input of the same size. Thus, mechanisms used to generate adversarial samples are unable to detect watermarks generated by EWE.

A.3 Additional Figures



(a) Proper trigger (b) Improper trigger

Figure 18: (a) In this Watermarked DNN, a small white square is designed as a special trigger. If this square is added to the corner of a digit-3, the input would be predicted as a digit-5 by the DNN, whereas a normal model would classify it as a digit-3 mostly. (b) This is an example of improperly designed trigger. By adding such a rectangle to top of 1's, even a unwatermarked model would classify it as a digit-7, so it is hard to tell if a model is watermarked or not by such a trigger.

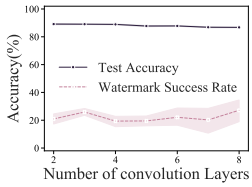
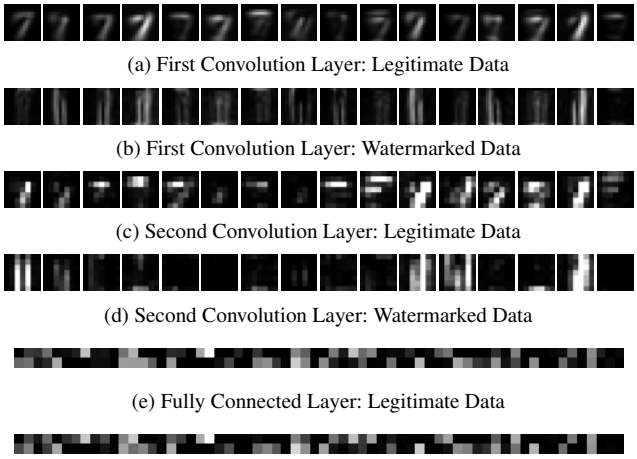


Figure 19: Validation Accuracy and Watermark success while increasing the number of convolution layers in a Fashion MNIST model without residual connection. Note that in-distribution watermark is used here.



(a) First Convolution Layer: Legitimate Data

(b) First Convolution Layer: Watermarked Data

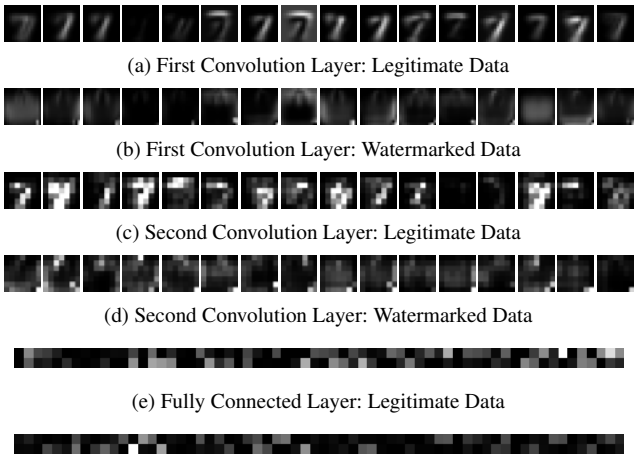
(c) Second Convolution Layer: Legitimate Data

(d) Second Convolution Layer: Watermarked Data

(e) Fully Connected Layer: Legitimate Data

(f) Fully Connected Layer: Watermarked Data

Figure 20: Activations of a convolutional neural network. We train a DNN with 2 convolution layers and 2 fully connected layers with EWE. We show here the frequency of activations for neurons in all hidden layers: high frequencies correspond to white color. One can observe that by entangling legitimate task data and watermarks, their representation becomes very similar, as we go deeper into the model architecture.



(a) First Convolution Layer: Legitimate Data

(b) First Convolution Layer: Watermarked Data

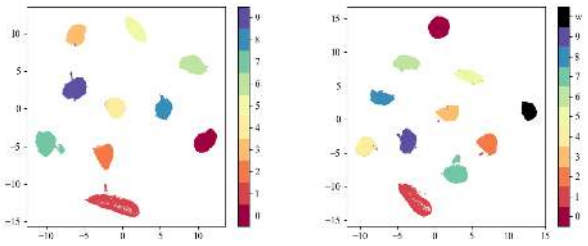
(c) Second Convolution Layer: Legitimate Data

(d) Second Convolution Layer: Watermarked Data

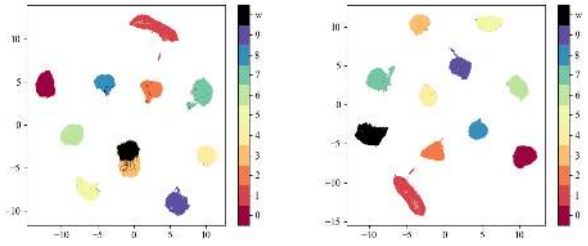
(e) Fully Connected Layer: Legitimate Data

(f) Fully Connected Layer: Watermarked Data

Figure 21: This should be compared to Figure 20. It is repeated here on a model with the same architecture but watermarked by the baseline. One can observe that the difference between activation of watermarked and legitimate data is more significant when EWE is not used.

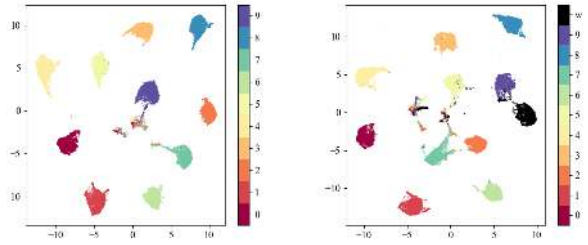


(a) Un-watermarked Model (b) Watermarked Model (Baseline)

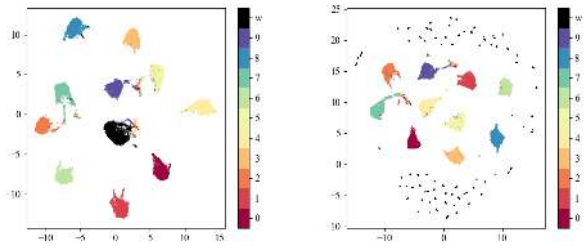


(c) EWE In-distribution Watermark (d) EWE Out-distribution Watermark

Figure 22: Same as Figure 11 except here the dataset is MNIST, while $c_S = 3$ and $c_T = 5$.

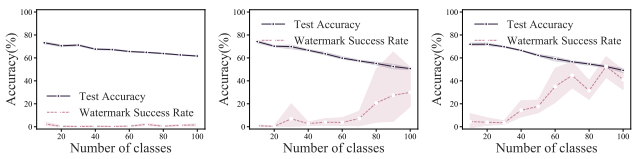


(a) Un-watermarked Model (b) Watermarked Model (Baseline)



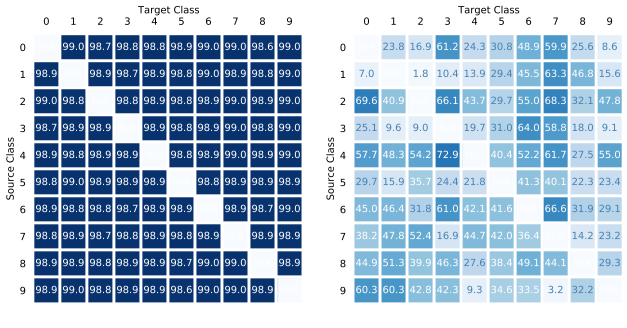
(c) EWE In-distribution Watermark (d) EWE Out-distribution Watermark

Figure 23: Same as Figure 11 except here the dataset is Speech Command, while $c_S = 9$ and $c_T = 5$. The OOD watermarks are audios of people saying "one".

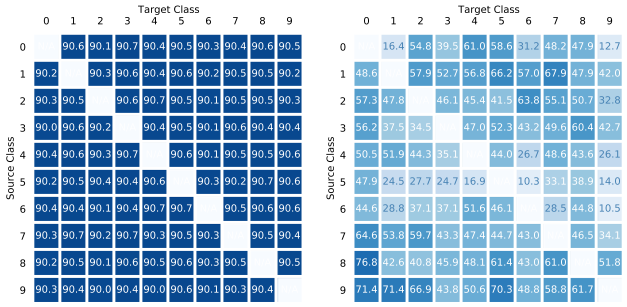


(a) Un-watermarked (b) Baseline (c) EWE

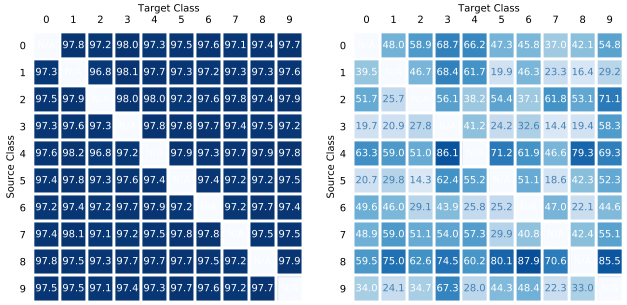
Figure 24: While scaling EWE to CIFAR-100, we noticed that both the baseline and EWE lead to significantly lower accuracies when the number of classes increases than an un-watermarked model. Besides, it can be observed that EWE reaches better watermark success than the baseline.



(a) MNIST: Test Accuracy (b) Watermark Success Rate

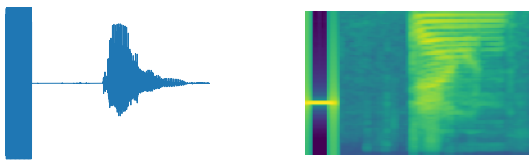


(c) Fashion-MNIST: Test Accuracy (d) Watermark Success Rate



(e) Speech Commands: Test Accuracy (f) Watermark Success Rate

Figure 25: Performance of the extracted model for different source-target pairs: We call class i and class j as a source-target pair if the watermark in our model is designed to be that watermarked data sampled from class i (if using OOD watermark, then this would be class i of another dataset) will be classified as class j by the model. On MNIST dataset , Fashion MNIST, and Speech Command, we tried to train and extract models with all 90 source-target pairs under the same setting (i.e. all hyper-parameters including temperature are the same) and plotted the validation accuracy and watermark success rate of the extracted model in the 6 figures above. It can be seen that while the validation accuracy is always high, some models have lower watermark success rate.



(a) Audio Signal (b) Spectrogram

Figure 26: Example of a watermarked audio signal and the corresponding Mel Spectrogram.