Distillation-Resistant Watermarking for Model Protection in NLP

Xuandong Zhao **Lei Li** **Yu-Xiang Wang**
University of California, Santa Barbara
{xuandongzhao,leili,yuxiangw}@cs.ucsb.edu

Abstract

How can we protect the intellectual property of trained NLP models? Modern NLP models are prone to stealing by querying and distilling from their publicly exposed APIs. However, existing protection methods such as watermarking only work for images but are not applicable to text. We propose **D**istillation-**R**esistant **W**atermarking (DRW), a novel technique to protect NLP models from being stolen via distillation. DRW protects a model by injecting watermarks into the victim's prediction probability corresponding to a secret key and is able to detect such a key by probing a suspect model. We prove that a protected model still retains the original accuracy within a certain bound. We evaluate DRW on a diverse set of NLP tasks including text classification, part-of-speech tagging, and named entity recognition. Experiments show that DRW protects the original model and detects stealing suspects at 100% mean average precision for all four tasks while the prior method fails on two[1].

1 Introduction

Large-scale pre-trained neural models have shown great success in NLP tasks (Devlin et al., 2019; Liu et al., 2019). Task-specific NLP models are often deployed as web services with pay-per-query APIs in business applications. Protecting the intellectual property of these cloud deployed models is a critical issue in both research and practice. Service providers often use authentication mechanism to authorize valid accesses. However, while this prevents clients directly copying a victim model, it does not hinder clients from stealing it using distillation. Emerging model extraction attacks have demonstrated convincingly that most functions of the victim API are likely to be stolen with carefully designed queries (Tramèr et al., 2016; Wallace et al., 2020; Krishna et al., 2020; He et al., 2021). A model extraction process is often imperceptible because it queries APIs in the same way as a normal user does (Orekondy et al., 2019). In this paper, we study the problem of *model protection* for NLP against distillation stealing.

Little has been done to adapt watermarking to identify model infringements in language tasks. Although a number of defense techniques have been proposed to prevent the model extraction for computer vision, they are not applicable to language tasks with discrete tokens. Among them, deep neural networks (DNN) watermarking (Szyller et al., 2021; Jia et al., 2021) works by embedding a secret watermark (e.g., logo or signature) into the model exploiting the over-parameterization property of DNNs. This procedure leverages a trigger set to stamp invisible watermarks on their commercial models before distributing them to customers. When suspicion of model theft arises, model owners can conduct an official ownership claim with the aid of the trigger set. However, these protections all focus on the image/audio tasks, since it is easy to modify the continuous data. In addition, most watermarking methods are invasive and fragile. They cannot avoid tampering with the training procedure in order to embed the watermark. Besides, the watermarks are outliers of the task distribution so that the adversary may not carry the watermark through distillation.

To fill in the gap, we make the first attempt to protect NLP models from distillation. We propose **D**istillation-**R**esistant **W**atermarking (DRW) to protect models and detect suspicious stealing. Inspired by the idea from CosWM for computer vision (Charette et al., 2022), we utilize prediction perturbation to embed a secret sinusoidal signal to the output of the victim API. To handle discrete tokens, we design a technique to randomly project tokens to a uniform region within sinusoidal cycles. We design watermarking effective for distillation with soft labels and with hard-sampled labels. As

[1]Our code is available at https://github.com/XuandongZhao/DRW

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5044–5055
December 7-11, 2022

long as the adversary trains the distillation procedure till convergence, DRW is able to detect the watermark signal from the extracted model.

The advantages of DRW include 1) *training independence*: it works directly on the trained models and can be directly plugged into the final output. 2) *flexibility*: it can be applied to both soft-label output and hard-label output in the black-box setting. 3) *effectiveness*: we evaluate the effectiveness of DRW and obtain perfect model extraction detection accuracy; we also justify the fidelity with a negligible side effect on the original classification quality. 4) *scalability*: the secret keys for the watermark are randomly generated on the fly so that we are able to provide different watermarks for different end-users and verify them.

The contributions of this paper are as follows:

- We enhance the concept of model protection against model extraction attacks with an emphasis on language applications.
- We propose DRW, a novel method to inject watermarks to the output of the NLP models and later to detect if suspects distill from the victim.
- We provide a theoretical guarantee on the protected API accuracy — with protection DRW does not harm much of original API's performance.
- Experiments on four diverse tasks (POS Tagging/NER/SST-2/MRPC) verify that DRW detects extracted models with 100% mean average precision, yet with only a small drop (<5%) in original prediction performance.

2 Related Work

Model Extraction Attacks Model extraction attacks target the confidentiality of ML models and aim to imitate the function of a black-box victim model (Tramèr et al., 2016; Orekondy et al., 2019; Correia-Silva et al., 2018). First, adversaries collect or synthesize an initially unlabeled substitute dataset. Next, they exploit the ability to query the victim model APIs for label predictions to annotate the substitute dataset. Then, they can train a high-performance model utilizing the pseudo-labeled dataset. Recently, several works (Krishna et al., 2020; Wallace et al., 2020; He et al., 2021) attempt to address the model extraction attacks on NLP models, e.g. BERT (Devlin et al., 2019) or Google Translate.

Knowledge Distillation Model extraction attacks are closely related to knowledge distillation (KD) (Hinton et al., 2015), where the adversary acts as the student who approximates the behaviors of the teacher (victim) model. The student can learn from soft labels or hard labels. KD with soft labels has been widely applied due to the fact that soft labels can carry a lot of useful information (Phuong and Lampert, 2019; Zhou et al., 2021).

Watermarking A digital watermark is an undetected label embedded in a noise-tolerant signal, such as audio, video, or image data. It is designed to identify the owner of the signal's copyright. Some works (Uchida et al., 2017; Adi et al., 2018; Zhang et al., 2018; Merrer et al., 2019) employ watermarks to prevent precise duplication of machine learning models. They insert watermarks into the parameters of the protected model or construct backdoor images that activate particular predictions. If an adversary exactly copies a protected model, a watermark can be used to verify ownership. However, safeguarding models from model extraction attacks is more difficult due to the fact that the parameters of the suspect model might be vastly different from those of the victim model, and the backdoor behavior may not be transferred to the suspect model either. Several works (Juuti et al., 2019; Szyller et al., 2021; Jia et al., 2021; Charette et al., 2022; He et al., 2022) study how to identify extracted models that are distilled from the victim model. Jia et al. (2021) forces the protected model to acquire features for identifying data samples taken from authentic and watermarked data. He et al. (2022) conducts lexical modification as a watermarking method to protect language generation APIs. CosWM (Charette et al., 2022) incorporates a watermark as a cosine signal into the output of the protected model. Since the cosine signal is difficult to eliminate, extracted models trained via distillation will continue to have a significant watermark signal. Nonetheless, CosWM only applies to image data and soft distillation. We design multiple new techniques to extend CosWM in handling the text data with discrete sequence and we provide a theoretical guarantee on the protected API accuracy for soft and hard distillations

3 Proposed Method: DRW

3.1 Overview

Figure 1 presents an overview of distillation procedure, watermarking and detection. The main idea

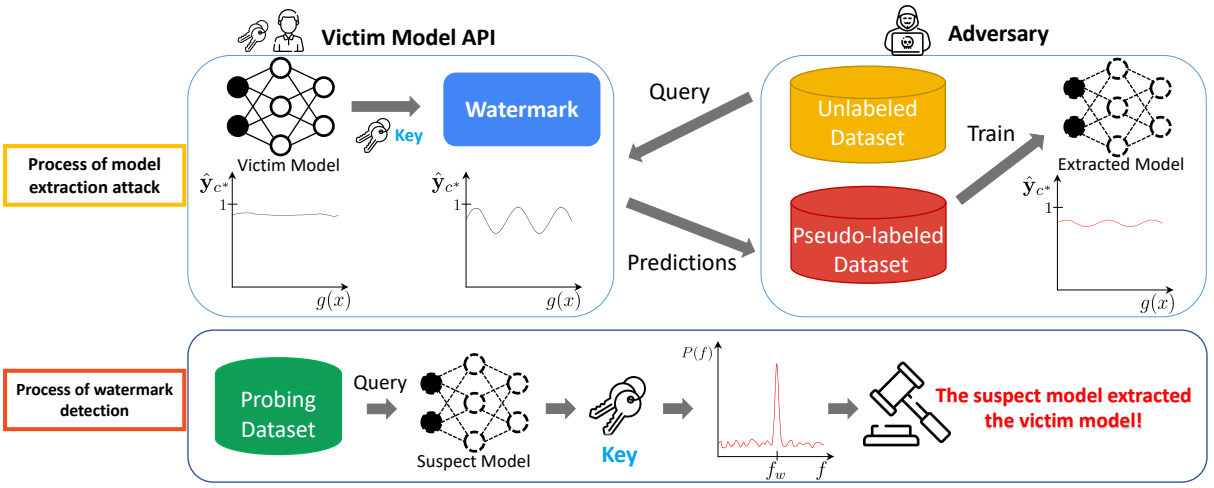

Figure 1: Overview of model extraction attack and watermark detection. The upper panel illustrates that the API owner adds a sinusoidal perturbation to the predicted probability distribution before answering end-users. The extracted model will convey this periodical signal if the adversary distills the victim model. At the phase of watermark detection, as shown in the bottom panel, the owner queries the suspect model and applies Fourier transform to the output with a key. Then the designed perturbation can be detected when a peak shows up in the frequency domain at f_w. The extracted watermark can thus serve as legal evidence and judgment for the ownership claim.

of DRW is to introduce a perturbation to the output of a protected model. This designed perturbation is transferred onto a suspect model distilled from a victim model that remains identifiable by probing the suspect model.

Problem Formulation We consider a common real-world scenario that the adversary only has black-box access to the victim model's API $\mathcal{V}$. There exist two types of output from victim model API: soft (real-valued) labels (i.e. probabilities) and hard labels. The adversary employs an auxiliary unlabeled dataset to query $\mathcal{V}$. Once the adversary gains the predictions from the victim model, it can train a separate model $\mathcal{S}$ from scratch with the pseudo-labeled dataset. The adversary may either distill the victim model with hard labels by minimizing the cross-entropy loss

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{m} \hat{\mathbf{y}}_i \log\left(\hat{\mathbf{q}}_i\right), \tag{1}$$

where $\hat{\mathbf{q}}_i$ is the prediction from the stealer's model and $\hat{\mathbf{y}}$ are the pseudo-labels from the victim model; or distill from soft labels by minimizing the Kullback–Leibler (KL) divergence loss

$$\mathcal{L}_{\text{KL}} = \sum_{i=1}^{m} \hat{\mathbf{y}}_i \log\left(\frac{\hat{\mathbf{y}}_i}{\hat{\mathbf{q}}_i}\right). \tag{2}$$

3.2 Watermarking the Victim Models

DRW dynamically embeds a watermark in response to queries made by an API's end-user. We use a set of variables to represent key $K = (c^*, f_w, \mathbf{v}_k, \mathbf{v}_s, \mathbf{M})$, where $c^* \in \{1, \ldots, m\}$ is the target class to embed watermark; $f_w \in \mathbb{R}$ is the angular frequency; $\mathbf{v}_k \in \mathbb{R}^n$ is the phase vector; $\mathbf{v}_s \in \mathbb{R}^n$ is the selection vector; $\mathbf{M} \in \mathbb{R}^{|D| \times n}$ is the random token matrix. $|D|$ represents the vocabulary size, so that every token ID corresponds to vector $\mathbf{M}_i \in \mathbb{R}^n$. Following Charette et al. (2022), we define a periodic signal function based on K and the input x.

$$\mathbf{z}_c(x) = \begin{cases} \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right), & c = c^* \\ \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M}) + \pi\right), & c \neq c^* \end{cases} \tag{3}$$

for $c \in \{1, \ldots, m\}$, where $g(\cdot) \in [0, 1)$ is a hash function projecting a text representation to a scalar. Ideally, the scalar should uniformly distribute spanning multiple cycles.

Constructing the hash function We project every input x into the fixed scalar range to add the sinusoidal perturbation by the hash function $g(\cdot)$. We randomly generate the phase vector $\mathbf{v}_k$, selection vector $\mathbf{v}_s$ and the token matrix $\mathbf{M}$. Each element in $\{\mathbf{v}_k, \mathbf{v}_s\}$ is randomly sampled from a uniform distribution over $[0, 1)$. Each element of the matrix $\mathbf{M}$ is randomly sampled from a standard normal distribution $\mathbf{M}_{ij} \sim \mathcal{N}(0, 1)$. Let $\mathbf{M}_i \in \mathbb{R}^n$ denote the i-th row of matrix $\mathbf{M}$, $\mathbf{v}_k^\top \mathbf{M}_i \sim \mathcal{N}(0, \frac{n}{3})$ and $\mathbf{v}_s^\top \mathbf{M}_i \sim \mathcal{N}(0, \frac{n}{3})$ (we prove it in Appendix A.2). Then we apply probability integral transformation to obtain the uniform distribution of the

hash values, where $g(\mathbf{v}_k, x, \mathbf{M}) \sim \mathcal{U}(0,1)$ and $g(\mathbf{v}_s, x, \mathbf{M}) \sim \mathcal{U}(0,1)$. We set $g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau$ to select part of all samples, where τ is the data selection ratio. When implementing sequence labeling tasks, we use the token ID to fetch the vector in matrix $\mathbf{M}$. Similarly, when implementing sentence classification tasks, we use the ID of the second token in the sentence to obtain the vector.

Next we compute the periodic signal for the victim output

$$\hat{\mathbf{y}}_c = \begin{cases} \hat{\mathbf{p}}_c, & g(\mathbf{v}_s, x, \mathbf{M}) > \tau \\ \frac{\hat{\mathbf{p}}_c + \varepsilon(1+\mathbf{z}_c(x))}{1+2\varepsilon}, & c = c^* \text{ and } g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau \\ \frac{\hat{\mathbf{p}}_c + \frac{\varepsilon(1+\mathbf{z}_c(x))}{m-1}}{1+2\varepsilon}, & c \neq c^* \text{ and } g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau \end{cases} \tag{4}$$

where ε is the watermark level for the periodic signal and $\hat{\mathbf{p}}_c$ is the victim model's prediction before watermarking. Since $0 \leq \hat{\mathbf{y}}_i \leq 1$ and $\sum_{i=1}^{m} \hat{\mathbf{y}}_i = 1$ (see proof in Appendix A.3), $\hat{\mathbf{y}}$ is a surrogate for softmax output.

In the soft label setting, the victim model generates output $\hat{\mathbf{y}}$ directly; while in the hard label setting, the victim model produces the sampling hard label, i.e. a one-hot label with probability $\hat{\mathbf{y}}_i$ for each class i. Intuitively, the hard-label sampled output retains the watermark because it is equal to $\hat{\mathbf{y}}$ in *expectation*. Further, we define the accuracy for soft label output, named "argmax soft", which calculates the accuracy of the argmax of soft output compared with the true label. Similarly, we define "sampling hard" to describe the output of the victim model which is a one-hot vector.

3.3 Detecting Watermark from Suspect Models

We first create a probing dataset $\mathcal{D}_p$, for which the labels are not required. $\mathcal{D}_p$ can be drawn from the training data of the extracted model since the owner is able to store any query sent by a specific end-user. In our setting, we also allow $\mathcal{D}_p$ to be drawn from other distributions.

We employ the Lomb-Scargle periodogram method (Scargle, 1982) for detecting and characterizing periodic signals. The Lomb–Scargle periodogram yields an estimate of the Fourier power spectrum $P(f)$ at frequency f in an unevenly sampled dataset. After getting the power spectrum, we evaluate the signal strength by calculating the signal-to-noise ratio

$$\begin{aligned} P_{\text{signal}} &= \frac{1}{\delta} \int_{f_w - \frac{\delta}{2}}^{f_w + \frac{\delta}{2}} P(f) df \\ P_{\text{noise}} &= \frac{1}{F - \delta} \left[\int_0^{f_w - \frac{\delta}{2}} P(f) df + \int_{f_w + \frac{\delta}{2}}^{F} P(f) df \right] \\ P_{\text{snr}} &= P_{\text{signal}} / P_{\text{noise}}, \end{aligned} \tag{5}$$

where δ controls the window width of $\left[f_w - \frac{\delta}{2}, f_w + \frac{\delta}{2}\right]$; F is the maximum frequency, and f_w is the angular frequency embedded into the victim model. A higher signal-to-noise ratio P_{snr} indicates a higher peak in the frequency domain.

4 Theoretical Analysis

In this section, we provide theoretical guarantees for DRW for both argmax soft output and sampling hard output. The analysis assumes the victim is *calibrated* so its soft-predictions are informative. We also focus on the binary classification task, i.e., $m = 2$. Generalization to $m > 2$ is straightforward and omitted only to ensure a clean presentation.

Theorem 1. *Without loss of generality, set target class $c^* = 1$, so that $\hat{p} = \hat{\mathbf{p}}_1(x), \hat{y} = \hat{\mathbf{y}}_1, z(x) = \mathbf{z}_1(x)$. Assume $\hat{p}(x)$ is calibrated, i.e., $\mathbb{E}[y|\hat{p}(x) = a] = a$, $\forall 0 \leq a \leq 1$, the argmax soft label of the victim model is $\hat{y}_s = \mathbb{1}\{\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon} > 0.5\}$ and the sampling hard label of the victim output is $\hat{y}_h \sim \text{Ber}(\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon})$. For a fixed $\mathbf{v}_k$, given that $z(x) = \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right) \in [-1, 1]$ and the data selection ratio is set to τ, then DRW argmax soft label and sampling hard label satisfy:*

$$\begin{aligned} &\mathbb{E}_{\mathbf{v}_k}\left[Acc(Argmax\ Soft)\right] \geq Acc(Victim) \\ &\quad - \tau(0.5 + \varepsilon)\mathbb{P}[0.5 - \varepsilon \leq \hat{p} \leq 0.5 + \varepsilon], \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathbb{E}_{\mathbf{v}_k}\left[Acc(Sampling\ Hard)\right] \geq (1 - \tau)Acc(Victim) \\ &\quad + \frac{\tau}{1 + 2\varepsilon}\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]. \end{aligned} \tag{7}$$

The proof is deferred to Appendix A.1.

Equation (6) says that, in the soft label setting, DRW does not hurt the accuracy too much if the watermark level ε is small. Note that only samples in which the victim model output lies around 0.5 ($\pm\varepsilon$) might be affected by the watermarking. These are data points where the victim model is uncertain and inaccurate anyway.

Equation (7) lowerbounds the accuracy of the sampled hard labels, which is close to the vanilla victim model if τ is small. Observe that if $\tau = 1$, the accuracy may drop even if the watermark

Model Type	SST-2	MRPC	POS	NER
mAP of detection for soft distillation:				
DeepJudge*	1.00	1.00	0.54	0.84
DRW	1.00	1.00	1.00	1.00
mAP of detection for hard distillation:				
DeepJudge*	1.00	1.00	0.48	0.40
DRW	1.00	1.00	1.00	1.00
Performance of the models:				
BERT	92.9	86.7	-	92.4
Victim model	92.8	87.0	90.7	91.3
+argmax soft	92.5	86.8	90.7	91.3
+sampling hard	88.4	85.8	90.3	91.0
Adversary soft	92.0	86.2	89.8	87.7
Adversary hard	91.3	86.1	89.7	87.4

Table 1: Main results for detection and model performance. We report the mean average precision of the model infringements detection for both soft-label distillation and hard-label distillation. The baseline is constructed based on the modification of DeepJudge. We show the results for BERT reported in the original paper. We report the results of victim model for argmax soft and sampling hard.

magnitude $\varepsilon = 0$ due to the sampling of the output label[2]. Our design of a second random projection $\mathbf{v}_s$ plays an important role here as it allows us to control the accuracy drop to any level we desire by adjusting τ.

5 Experiments

5.1 Tasks

We evaluate the performance of DRW on four different tasks. Two are sequence labeling tasks, Part-Of-Speech (POS) Tagging and Named Entity Recognition (NER); the other two are from GLUE (Wang et al., 2018) text classification tasks, SST-2 and MRPC. We choose BERT (Devlin et al., 2019) as our model backbone and fine-tune it in different tasks.

Sequence labeling We utilize the CoNLL-2003 dataset (Sang and Meulder, 2003) for POS Tagging and NER tasks. The CoNLL-2003 dataset consists of news articles from the Reuters RCV1 corpus with POS and NER tags. We formulate POS Tagging and NER as token-level classification tasks following standard practice. Specifically, POS Tagging has 47 classes and NER has 9 classes. We take the token embedding of the last hidden layer of BERT (Devlin et al., 2019) as the input to a linear layer, which is then used as the classifier over the POS/NER label set. The token ID is set as the input x for the hash function $g(\cdot)$. F1 score is hired for the evaluation metric.

Text classification SST-2 is a binary single-sentence classification task consisting of movie reviews with corresponding sentiment (Socher et al., 2013). MRPC is a collection of sentence pairs from online news with labels suggesting whether the pair is semantically equivalent or not (Dolan and Brockett, 2005). We use the final hidden vector of the special `[CLS]` token of BERT as the input to a linear layer, which serves as the sentence classifier. The ID of the second token in the sentence is set as the input x for the hash function $g(\cdot)$. Since GLUE does not include any test dataset, we use accuracy of the validation set as the evaluation metric.

For each task, we train the protected model to achieve the best performance on the validation set. As demonstrated in Table 1, the victim model has comparable performance to BERT (Devlin et al., 2019). For soft and hard label distillation, we split the training data in each task into two parts and use the first half to query the victim model. Then the extracted model is trained for 20 epochs on the pseudo-labeled dataset. We choose the same key $K = (c^*, f_w, \mathbf{v}_k, \mathbf{v}_s, \mathbf{M})$, where frequency $f_w = 16.0$, watermark level $\varepsilon = 0.2$ and $\{\mathbf{v}_k, \mathbf{v}_s, \mathbf{M}\}$ are generated with different random seed. We set target class $c^* = 22$ ("NNP" tag) for POS Tagging, $c^* = 2$ ("I-PER" tag) for NER and $c^* = 0$ ("negative" class) for SST-2/MRPC. We set data selection ratio $\tau = 0.5$ to add watermarks to half of the output data. More details for the experiment setting can be found in Appendix A.4.

Baseline We take the state-of-the-art method DeepJudge (Chen et al., 2022) as a baseline against DRW. DeepJudge quantitatively tests the similarities between the victim model and suspect model, then determines whether the suspect model is a copy based on the testing metrics. Since DeepJudge is designed for continuous signals such as images and audio, we modify the method to apply it to texts. We consider the black-box setting for DeepJudge, and compute Jensen-Shanon Distance (JSD) (Fuglede and Topsøe, 2004) for the probing dataset of the victim model and the extracted model. JSD measures the similarity of two prob-

[2]Under the calibration assumption, $Acc(Victim) = \mathbb{E}\left[\hat{p}\mathbf{1}(\hat{p} \geq 0.5) + (1-\hat{p})\mathbf{1}(\hat{p} < 0.5)\right]$, which is strictly bigger than $\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]$ except when $\hat{p}$ is supported only at trivial points $\{0, 1, 0.5\}$

	SST-2	MRPC	POS	NER
DeepJudge-JSD-Soft:				
Negative Suspect	(0.012, 0.032)	(0.009, 0.161)	(0.016, 0.444)	(0.001, 0.416)
Positive Suspect	(0.001, 0.002)	(0.001, 0.002)	(0.087, 0.279)	(0.002, 0.201)
DeepJudge-JSD-Hard:				
Negative Suspect	(0.013, 0.029)	(0.008, 0.154)	(0.010, 0.432)	(0.009, 0.274)
Positive Suspect	(0.004, 0.005)	(0.003, 0.007)	(0.029, 0.112)	(0.011, 0.052)
DRW-P_{snr}-Soft:				
Negative Suspect	(0.008, 4.775)	(0.128, 2.607)	(0.012, 2.309)	(0.105, 4.243)
Positive Suspect	(18.82, 25.77)	(17.81, 24.25)	(20.59, 28.73)	(17.25, 25.22)
DRW-P_{snr}-Hard:				
Negative Suspect	(0.011, 4.235)	(0.012, 3.678)	(0.182, 2.869)	(0.203, 4.183)
Positive Suspect	(16.38, 22.77)	(16.70, 21.80)	(16.23, 25.67)	(16.19, 25.49)

Table 2: The probing results for DeepJudge and DRW in soft distillation and hard distillation settings. We present the range of JSD and P_{snr}. The first value in parentheses is the minimum score and the second value is the maximum score. A larger gap in score between the negative and positive suspect models indicates that the detection method performs better in identifying the extracted model.

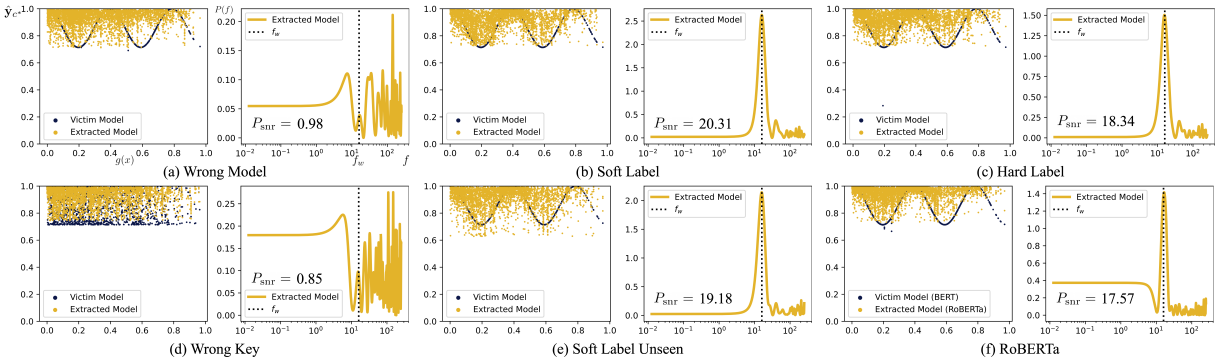

Figure 2: Examples of DRW in NER task. The left panel of each sub-figure plots the output of the target class c^* for the victim model and the extracted model ($\hat{\mathbf{y}}_{c^*}$ vs. $g(x)$). The right panel of each sub-figure plots the power spectrum value for output of the extracted model($P(f)$ vs. f). We also display the P_{snr} value for signal strength of the extracted model.

ability distributions. We use the probing dataset to query both the victim model and the suspect model, and then calculate JSD of the output layer as follows

$$JSD\left(\hat{\mathbf{y}}, \hat{\mathbf{q}}\right) = \frac{1}{2|\mathcal{D}_p|} \sum_{x \in \mathcal{D}_p} K\left(\hat{\mathbf{y}}(x), u\right) + K\left(\hat{\mathbf{q}}(x), u\right)$$

where $u = \left(\hat{\mathbf{y}}(x) + \hat{\mathbf{p}}(x)\right)/2$ and $K(\cdot, \cdot)$ is the Kullback-Leibler divergence. A small JSD value implies similar output distribution of the two models, which further indicates that the suspect model may be distilled from the victim model.

Evaluation We evaluate the performance of the victim model and the extracted model with accuracy/F1 score. In order to compare DRW with DeepJudge in detecting extracted models, we can reduce this binary classification problem to thresholding a particular test score. Since DRW and DeepJudge use different scores to detect the extracted model, we set up a series of ranking tasks to show the effect of these scores. For each task, we train 10 extracted models from the watermarked victim model with different random initialization as positive samples, 10 extracted models from the unwatermarked victim model with different random initialization, and 10 models from scratch with true labels as negative samples. For DRW, we use the watermark signal strength values P_{snr} as the score for ranking (identifying whether it is an extracted model); for DeepJudge, we use JSD as the score. Next, we compute the mean average precision (mAP) for the ranking tasks which assesses the model extraction detection performance. A higher mAP means the detecting method can distinguish the victim and the suspect model better.

We show the experiment results in the following

subsections.

5.2 Effectiveness: Is DRW able to identify model infringements?

We evaluate our method in two settings, distillation with soft labels and distillation with hard labels. The results are displayed in Table 1. DeepJudge performs well on SST-2 and MRPC tasks but it can not effectively detect the extracted models in POS Tagging and NER tasks. In contrast, our method can successfully detect the extraction with 100% mAP across all tasks in both settings. We also present the range of JSD and P_{snr} in Table 2. Regarding the performance of DeepJudge on POS Tagging and NER tasks, the JSD intervals for positive and negative samples overlap each other, resulting in the aforementioned lower mAP compared to DRW. A case in point is DeepJudge-JSD-Hard for NER task, where the ranges for negative suspect score and positive suspect score are $[0.009, 0.274]$ and $[0.011, 0.052]$ respectively. The overlapping intervals lead to the imperfect detection result, i.e., mAP $= 0.40$. Whereas, DRW is able to *perfectly* distinguish between positive and negative suspects. Typically, P_{snr} for the negative suspect is smaller than 5 while that for the positive suspect is larger than 15.

5.3 Fidelity: Does DRW decrease the performance of the model?

The results for the model performance at the watermark level $\varepsilon = 0.2$ are displayed in Table 1. The perturbed API (victim model with argmax soft/sampling hard) only has a slight performance drop (within 5%) in comparison to the original one due to the trade-off between detection effectiveness and model performance. For the victim model API, argmax soft exhibits less performance drop than the sampling hard, since the argmax of the soft label remains unchanged with small perturbation. For the extracted model, distillation with soft label tends to have a better accuracy/F1 score than that with hard label. Additionally, the performances of extracted models are very close to those of victim models, a clear manifestation of the distillation success.

5.4 Case Study

We present how our method works on some examples in NER task. We fix the victim model and choose different settings for the suspect model across all the examples. For the watermarked ones, we set $f_w = 16, \varepsilon = 0.2$ and $c^* = 2$.

In Figure 2 (a), we show how DRW works on a suspect model that does not extract the victim model. We select a model trained from scratch with true labels as a negative example. There is no sinusoidal signal in the output of the suspect model hence a small P_{snr}.

In Figure 2 (b)(c), we illustrate the effect on soft distillation and hard distillation. We use the watermark key K to extract the output of the victim model and suspect model. The extracted model clearly follows the victim model and there is a prominent peak at frequency f_w. Note that suspect model distillation with soft labels has a higher P_{snr} than the one with hard labels. This is because the training process of extracted models can be more effective and faster with soft labels (Phuong and Lampert, 2019).

In Figure 2 (d), we validate the *secrecy* of our method. If the adversary does not have the secret key, it can not justify what the watermark is or whether there exist watermarks. The output of the victim model and extracted model are almost indiscernible when we use a wrong key to project them given the hash function $g(\cdot)$.

In Figure 2 (e)(f), we demonstrate the generality of our method. Watermarking algorithm should be independent of the dataset and the ML algorithms. In sub-figure (e), a different dataset is used to probe the suspect model. To be specific, we select the second half of the training data as the probing dataset, rather than the first half used in previous experiments. The results imply that DRW turns out to work well when we use unseen data to produce the probing dataset for the suspect model. In sub-figure (f), we choose a different backbone RoBERTa (Liu et al., 2019) for the suspect model, in which the victim model continues to be the BERT model. The high peak in the power spectrum at frequency f_w reveals that DRW is still able to detect the signal.

6 Ablation Study

6.1 Does watermark level impact detection?

An important aspect of watermarking is how much perturbation we add to the output of the victim model. Theoretically, a smaller watermark level is associated with a higher accuracy/F1 score of the victim, yet it makes it harder to extract the signal from the probing results. We conduct two experiments to investigate the effect of the watermark level.

In the first experiment, we vary the watermark

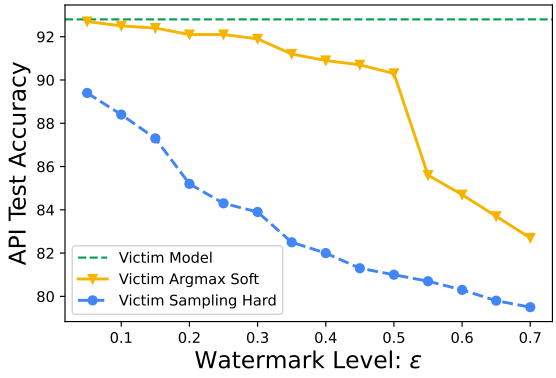

Figure 3: Test accuracy of victim model API with different watermark level in SST-2 task.

level in SST-2 task. According to the Theorem 1, the accuracy of the victim model output is bounded and a higher watermark level causes poorer performance. As shown in Figure 3, when the watermark level rises from 0 to 0.7, the performance drops by around 10 percent. It is worth noting that a big drop of the argmax soft emerges as ε passes 0.5, which means the argmax of the output is highly likely to be changed in this case.

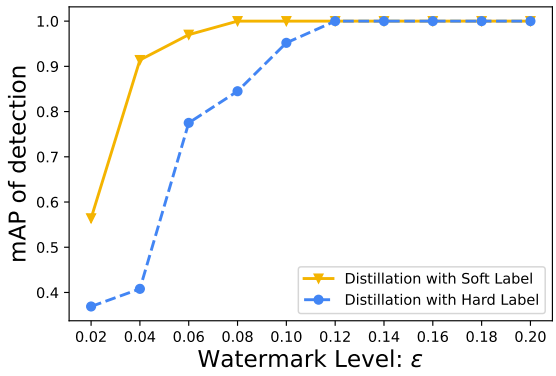

Figure 4: Model detection results with different watermark level in NER task.

In the second experiment, we design 10 sets of ranking tasks, and build up 10 positive samples together with 20 negative samples (similar to the setting in Section 5.1) for each set in NER task. The watermark level is the only varied parameter across different tasks, ranging from 0.02 to 0.2. We plot the mAP of the detection against the watermark level in Figure 4. When the watermark level ε is below 0.12, DRW can not generate perfect detection of positive and negative suspects, indicating that the adversary may not convey a strong sinusoidal signal at a low watermark level. In this case, DRW can not extract the watermark in frequency space and thus fails to detect it successfully.

These two experiments demonstrate the trade-off between the detection effectiveness and the victim model's performance after watermarking.

6.2 Do categories affect watermark protection?

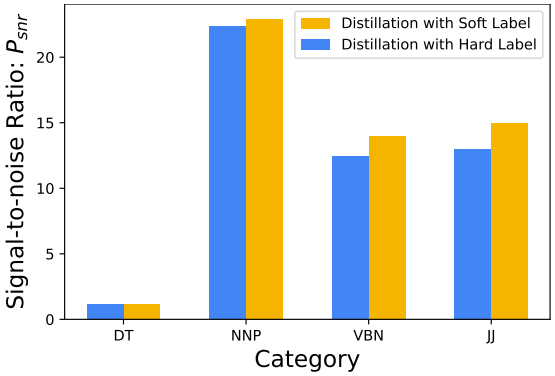

Figure 5: Adding watermark to four categories in POS Tagging task. "DT": determiner; "NNP": proper noun, singular; "VBN": verb, past practice; "JJ": adjective.

We vary the target class c^* of the watermark key K in POS Tagging task. We add watermarks to four different categories and then train the extracted model by soft distillation and hard distillation. The results of the signal-to-noise ratio P_{snr} are visualized in Figure 5. The effect of the watermark will be more salient if the category involves more samples. Since "NNP" covers the most (14.16%) of all tokens, adding watermark to "NNP" produces the strongest signal. In contrast, the determiners ("DT") category only has a few number of types, such as "the" and "a". As a result, adding watermark to "DT" is ineffective as it is hard to add a periodic signal to a very discrete domain.

6.3 How much should be selected for watermarking?

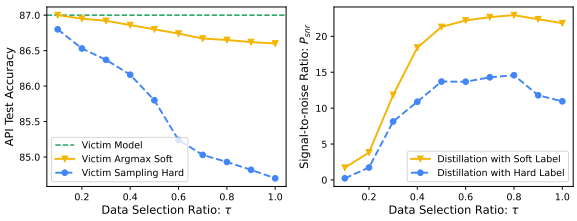

Figure 6: Output accuracy of the victim model and signal strength of the extracted model with different data selection ratio τ in MRPC task.

A critical design of our method is that we apply selection vector $\mathbf{v}_s$ to select a portion of the victim model output to be watermarked. We change the ratio of the watermarked data by tuning the data

selection ratio τ in MRPC task. The results shown in Figure 6 indicate that the accuracy of the victim model output falls with a higher data selection ratio, yet it introduces a greater signal strength of the extracted model. This trade-off is similar to the one described in Section 6.1. 0.5 could be a reasonable selection ratio.

7 Conclusion

In this work, we propose **D**istillation-**R**esistant **W**atermarking (DRW), a novel and unified watermarking technique against model extraction attacks on NLP models. By injecting watermarks into the prediction output of the victim model, the model owner can detect the watermark if the adversary distills the protected model. We prove the theoretical guarantee of DRW and show remarkable empirical results on text classification and sequence labeling tasks.

Limitations

1) The watermark detection does not work well when the watermarked data covers only a small amount of the whole training data for the extracted model. 2) Our method may not work well when the adversary only makes a few queries to the victim model APIs and trains the extracted model with few-shot learning. 3) If the victim model outputs soft labels, even with watermarking, the adversary can take argmax operation to erase the watermark. So it is better to combine watermarks with hard label output in real-world applications.

Broader Impact

This work will alleviate ethical concerns of commercial NLP models. This paper provides one promising solution to an important aspect of NLP: how to protect the intellectual property of trained NLP models. Companies with NLP web services can apply our method to protect their models from model extraction attacks.

Acknowledgements

XZ was supported by UCSB Chancellor's Fellowship. The authors would like to thank Yang Gao for polishing up the draft and Dan Qiao for the helpful discussion.

References

Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*.

Laurent Charette, Lingyang Chu, Yizhou Chen, Jian Pei, Lanjun Wang, and Yong Zhang. 2022. Cosine model watermarking against ensemble distillation. *AAAI*.

Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. 2022. Copy, right? a testing framework for copyright protection of deep learning models. In *IEEE Symposium on Security and Privacy (SP)*.

Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. 2018. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IJCNLP*.

Bent Fuglede and Flemming Topsøe. 2004. Jensen-shannon divergence and hilbert space embedding. *ISIT*.

Xuanli He, L. Lyu, Qiongkai Xu, and Lichao Sun. 2021. Model extraction and adversarial transferability, your bert is vulnerable! In *NAACL*.

Xuanli He, Qiongkai Xu, L. Lyu, Fangzhao Wu, and Chenguang Wang. 2022. Protecting intellectual property of language generation apis with lexical watermark. *AAAI*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled watermarks as a defense against model extraction. In *USENIX Security*.

Mika Juuti, Sebastian Szyller, Alexey Dmitrenko, Samuel Marchal, and N. Asokan. 2019. Prada: Protecting against dnn model stealing attacks. *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on sesame street! model extraction of bert-based apis. In *ICLR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. 2019. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32:9233–9244.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *CVPR*.

Mary Phuong and Christoph Lampert. 2019. Towards understanding knowledge distillation. In *ICML*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Jeffrey D. Scargle. 1982. Studies in astronomical time series analysis. ii - statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263:835–853.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. 2021. Dawn: Dynamic adversarial watermarking of neural networks. *Proceedings of the 29th ACM International Conference on Multimedia*.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *USENIX Security*.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*.

Eric Wallace, Mitchell Stern, and Dawn Xiaodong Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *EMNLP*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*.

Helong Zhou, Liangchen Song, Jiajie Chen, Ye Zhou, Guoli Wang, Junsong Yuan, and Qian Zhang. 2021. Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective. In *ICLR*.

A Appendix

A.1 Proof for the Theorem 1

Theorem A.1 (Restate Theorem 1)**.** *Without loss of generality, set target class $c^* = 1$, so that $\hat{p} = \hat{\mathbf{p}}_1(x), \hat{y} = \hat{\mathbf{y}}_1, z(x) = \mathbf{z}_1(x)$. Assume $\hat{p}(x)$ is calibrated, i.e., $\mathbb{E}[y|\hat{p}(x) = a] = a$, $\forall 0 \leq a \leq 1$, the argmax soft label of the victim model is $\hat{y}_s = \mathbb{1}\{\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon} > 0.5\}$ and the sampling hard label of the victim output is $\hat{y}_h \sim \mathrm{Ber}(\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon})$. For a fixed $\mathbf{v}_k$, given that $z(x) = \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right) \in [-1, 1]$ and the data selection ratio is set to τ, then DRW argmax soft label and sampling hard label satisfy:*

$$\mathbb{E}_{\mathbf{v}_k}\left[\textit{Acc}(\textit{Argmax Soft})\right] \geq \textit{Acc}(\textit{Victim}) - \tau(0.5+\varepsilon)\mathbb{P}[0.5-\varepsilon \leq \hat{p} \leq 0.5+\varepsilon], \quad (8)$$

$$\mathbb{E}_{\mathbf{v}_k}\left[\textit{Acc}(\textit{Sampling Hard})\right] \geq (1-\tau)\textit{Acc}(\textit{Victim}) + \frac{\tau}{1+2\varepsilon}\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]. \quad (9)$$

Proof. We first prove the argmax soft label case with $\tau = 1$.

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{1}(\hat{y}_s = y)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = y|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = 1, y = 1|x) + \mathbb{P}(\hat{y}_s = 0, y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = 1|x)\mathbb{P}(y = 1|x)\right. \\
&\left. + \mathbb{P}(\hat{y}_s = 0|x)\mathbb{P}(y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbb{1}\{\hat{p} + \varepsilon z(x) > 0.5\}\mathbb{P}(y = 1|x)\right. \\
&\left. + \mathbb{1}\{\hat{p} + \varepsilon z(x) \leq 0.5\}\left(1 - \mathbb{P}(y = 1|x)\right)\right] \\
=&\mathbb{E}\Big[\mathbb{E}\left[\mathbb{1}\{\hat{p} + \varepsilon z(x) > 0.5\}\mathbb{P}(y = 1|x)\right. \\
&\left. + \mathbb{1}\{\hat{p} + \varepsilon z(x) \leq 0.5\}\left(1 - \mathbb{P}(y = 1|x)\right)|\hat{p}\right]\Big] \\
\geq&\mathbb{E}\Big[\mathbb{E}\left[\mathbb{1}\{\hat{p} - \varepsilon > 0.5\}\mathbb{P}(y = 1|x)|\hat{p}\right] \\
&+ \mathbb{E}\left[\mathbb{1}\{\hat{p} + \varepsilon \leq 0.5\}(1 - \mathbb{P}(y = 1|x))|\hat{p}\right]\Big] \\
=&\mathbb{E}\Big[\mathbb{1}\{\hat{p} - \varepsilon > 0.5\}\mathbb{E}\left[\mathbb{P}(y = 1|x)|\hat{p}\right] \\
&+ \mathbb{1}\{\hat{p} + \varepsilon \leq 0.5\}\mathbb{E}\left[1 - \mathbb{P}(y = 1|x)|\hat{p}\right]\Big] \\
=&\mathbb{E}\Big[\mathbb{1}\{\hat{p} > 0.5 + \varepsilon\}\hat{p} + \mathbb{1}\{\hat{p} \leq 0.5 - \varepsilon\}(1 - \hat{p})\Big] \\
=&\underbrace{\mathbb{E}\left[\mathbb{1}\{\hat{p} > 0.5\}\hat{p} + \mathbb{1}\{\hat{p} \leq 0.5\}(1 - \hat{p})\right]}_{\text{Accuracy of victim model without watermark}} \\
&- \mathbb{E}\left[\mathbb{1}\{0.5 < \hat{p} \leq 0.5 + \varepsilon\}\hat{p}\right] \\
&+ \mathbb{E}\left[\mathbb{1}\{0.5 - \varepsilon \leq \hat{p} \leq 0.5\}(1 - \hat{p})\right] \\
\geq&\text{Acc(Victim Model)} \\
&- (0.5 + \varepsilon)\mathbb{P}(0.5 - \varepsilon \leq \hat{p} \leq 0.5 + \varepsilon)
\end{aligned}$$

where the first "$\geq$" follows from $|z(x)| \leq 1$; the third "$=$" follows from the conditional independence of $\hat{y}_s$ and y given x; the seventh "$=$" follows from the calibration assumption, i.e. $\mathbb{E}\left[\mathbb{P}(y = 1|x)|\hat{p}(x)\right] = \hat{p}(x)$.

Notice that over the distribution of $\mathbf{v}_s$ selects every unique x with probability τ independently to everything else, by exchanging the order of expectation, it is easy to prove that the expected accuracy is a convex combination of the accuracy of the victim model (with weight $1-\tau$) and the case above (with weight τ). This completes the proof for argmax soft label.

We then start by analyzing the sampling hard label case with $\tau = 1$.

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{1}(\hat{y} = y)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y} = y|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y} = 1, y = 1|x) + \mathbb{P}(\hat{y} = 0, y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbb{E}(\hat{y}|x)\mathbb{E}(y|x) + \mathbb{E}(1 - \hat{y}|x)\mathbb{E}(1 - y|x)\right] \\
=&\mathbb{E}\left[\left(\frac{\hat{p}}{1+2\varepsilon} + \frac{\varepsilon(1+z(x))}{1+2\varepsilon}\right)\mathbb{E}(y|x)\right. \\
&\left. + \left(\frac{1-\hat{p}}{1+2\varepsilon} + \frac{\varepsilon(1-z(x))}{1+2\varepsilon}\right)\mathbb{E}(1 - y|x)\right] \\
=&\frac{1}{1+2\varepsilon}\underbrace{\mathbb{E}\left[\hat{p}\mathbb{E}(y|x) + (1-\hat{p})\mathbb{E}(1-y|x)\right]}_{A} \\
&+ \frac{\varepsilon}{1+2\varepsilon}\underbrace{\mathbb{E}\left[(1+z(x))\mathbb{E}(y|x) + (1-z(x))\mathbb{E}(1-y|x)\right]}_{B}
\end{aligned}$$

$$\begin{aligned}
A =&\mathbb{E}\left[\mathbb{E}\left[\hat{p}\mathbb{E}(y|x) + (1-\hat{p})\mathbb{E}(1-y|x)|\hat{p}\right]\right] \\
=&\mathbb{E}\left[\hat{p}\mathbb{E}(y|\hat{p}) + (1-\hat{p})\mathbb{E}(1-y|\hat{p})\right] \\
=&\mathbb{E}\left[\hat{p}^2 + (1-\hat{p})^2\right] \\
=&\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]
\end{aligned}$$

where the third line follows from the calibration assumption, i.e., $\mathbb{E}[y|\hat{p}(x) = a] = a$.

$$\begin{aligned}
B =&\mathbb{E}\left[\mathbb{E}(y|x) + \mathbb{E}(y|x)z(x) + 1 - z(x)\right. \\
&\left. - \mathbb{E}(y|x) + \mathbb{E}(y|x)z(x)\right] \\
=&1 + \mathbb{E}\left[\left(2\mathbb{E}(y|x) - 1\right)z(x)\right] \\
\geq&0
\end{aligned}$$

where the last line follows from the facts that $|z(x)| \leq 1$ and $|2\mathbb{E}(y|x) - 1| \leq 1$.

Finally, notice that for each x the probability to be chosen to add watermark and to sample the output is τ independently, thus the expected accuracy is the convex combination of the accuracy of the victim model and that of the fully watermarked model. □

A.2 Distribution Property

Lemma 1. *Assume $\mathbf{v} \sim \mathcal{U}(0, 1)$, $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{x} \sim \mathcal{N}(0, 1)$, $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{v}$ and $\mathbf{x}$ are both*

i.i.d. and independent of each other. Then we have:

$$\frac{1}{\sqrt{n}}\mathbf{v}\cdot\mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right),\ n \to \infty$$

Proof. Let $u_i = \mathbf{v}_i\mathbf{x}_i,\ i \in 1, 2, \ldots, n$. By assumption, u_i are $i.i.d.$. Clearly, the first and second moments are bounded, so the claim follows from the classical central limit theorem,

$$\sqrt{n}\bar{u}_n = \frac{\sum_{i=1}^{n} u_i}{\sqrt{n}} \rightsquigarrow \mathcal{N}\left(\mu, \sigma^2\right) \ \text{ as } \ n \to \infty$$

where

$$\begin{aligned}
\mu &= \mathbb{E}\left(u_i\right) = \mathbb{E}\left(\mathbf{v}_i\mathbf{x}_i\right) = \mathbb{E}\left(\mathbf{v}_i\right)\mathbb{E}\left(\mathbf{x}_i\right) \\
&= 0 \\
\sigma^2 &= \mathrm{Var}(u_i) = \mathbb{E}\left(u_i^2\right) - \left(\mathbb{E}\left(u_i\right)\right)^2 \\
&= \mathbb{E}\left(u_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2\mathbf{x}_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2\right)\mathbb{E}\left(\mathbf{x}_i^2\right) \\
&= \frac{1}{3}
\end{aligned}$$

It follows that given large n

$$\frac{1}{\sqrt{n}}\mathbf{v}\cdot\mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right)$$

□

A.3 Modified Softmax Properties

Lemma 2 (Lemma 1 in (Charette et al., 2022))**.** *Let $\hat{\mathbf{p}}$ be the softmax output of a model $\mathcal{V}$, then the modified softmax $\hat{\mathbf{y}}$, as defined in Equation 4 satisfies $0 \le \hat{\mathbf{y}}_i \le 1$ and $\sum_{i=1}^{m}\hat{\mathbf{y}}_i = 1$.*

Proof. Notice that in Equation 4, when $g(\mathbf{v}_s, x, \mathbf{M}) > \tau$, $\hat{\mathbf{y}} = \hat{\mathbf{p}}$, so that it satisfies the property above.

By the definition of softmax, for all class $c \in \{1, \ldots, m\}$ we have

$$0 \le \hat{\mathbf{p}}_c \le 1, -1 \le \mathbf{z}_c(x) \le 1.$$

Therefore, when $c = c^*$, we have

$$0 \le \hat{\mathbf{p}}_c + \varepsilon\left(1 + \mathbf{z}_c(x)\right) \le 1 + 2\varepsilon,$$

and then

$$0 \le \frac{\hat{\mathbf{p}}_c + \varepsilon\left(1 + \mathbf{z}_c(x)\right)}{1 + 2\varepsilon} \le 1.$$

When $c \neq c^*$, since $m \ge 2$, we have

$$0 \le \hat{\mathbf{p}}_c + \frac{\varepsilon\left(1 + \mathbf{z}_c(x)\right)}{m-1} \le 1 + \frac{2\varepsilon}{m-1} \le 1 + 2\varepsilon$$

and then

$$0 \le \frac{\hat{\mathbf{p}}_c + \frac{\varepsilon(1+\mathbf{z}_c(x))}{m-1}}{1 + 2\varepsilon} \le 1.$$

Thus, $\hat{\mathbf{q}}$ satisfies $0 \le \hat{\mathbf{y}}_i \le 1$.

To prove $\sum_{i=1}^{m}\hat{\mathbf{y}}_i = 1$, we use the fact that $\mathbf{z}_{c^*} + \mathbf{z}_{i\neq c^*} = 0$ and obtain

$$\begin{aligned}
\sum_{i=1}^{c}\hat{\mathbf{y}}_i &= \frac{\hat{\mathbf{p}}_{c^*} + \varepsilon\left(1 + \mathbf{z}_{c^*}\right)}{1 + 2\varepsilon} + \sum_{i\neq c^*}\frac{\hat{\mathbf{p}}_i + \frac{\varepsilon(1+\mathbf{z}_i)}{m-1}}{1 + 2\varepsilon} \\
&= \sum_{i=1}^{m}\frac{\hat{\mathbf{p}}_i}{1 + 2\varepsilon} + \sum_{i\neq c^*}\frac{\varepsilon\left(1 + \mathbf{z}_{c^*} + 1 + \mathbf{z}_i\right)}{(m-1)(1 + 2\varepsilon)} \\
&= \frac{1}{1 + 2\varepsilon} + \frac{2\varepsilon}{1 + 2\varepsilon} \\
&= 1
\end{aligned}$$

□

A.4 Experiment Details

We provide more details for the experiments in this section.

We build our classification models upon `bert-base-uncased` from Hugging Face[3]. The model contains 110M parameters. We add a dropout layer before the last linear layer with a dropout rate of 0.5. We implement DRW in PyTorch 1.11.0 on a server with 4 NVIDIA TITAN-Xp GPUs. We set batch size to 8 for SST-2 and MRPC tasks, and 32 for POS Tagging and NER tasks.

We train the victim model using AdamW (Loshchilov and Hutter, 2019) optimizer with learning rate 1e-5 and epsilon 1e-8. Each victim model is trained 40 epochs and the one with the best validation results is chosen.

Regarding the extracted model, we use half of the training data to query the victim model and obtain the labeled dataset. Then the extracted model is trained with Adam (Kingma and Ba, 2015) optimizer for 20 epochs with learning rate 5e-5. The average training time is 3 minutes for each epoch.

We show the results for RoBERTa model in Section 5.4. In this setting, we choose `roberta-base` from Hugging Face, which has 125M parameters.

[3] https://huggingface.co/