Watermarking Neural Language Models based on Backdooring

Andrew Fu Harvard University
Tianhao Wang Harvard University
Chuqiao Yuan Harvard University
Yan Zhao Harvard University

Abstract

Neural language models have achieved tremendous success in various fields; however, training these models from scratch can be computationally expensive and requires a lot of training data. In this work, we present an approach for watermarking neural language models with a backdooring technique. We show experimentally that a carefully designed watermark has no noticeable impact on model performance. We also evaluate the robustness of our proposal against a multitude of practical removal attacks. This work provides motivation for further research into backdooring techniques for neural language models.

1 Introduction

The past five years have seen an explosion of activity in deep learning in the natural language processing (NLP) community. Neural language models have been found to significantly outperform previous state-of-the-art models in a wide range of domains such as language modeling [1], machine translation [2, 3], dialogue systems [4], and automatic speech recognition [5]. Training a neural language model, however, is expensive in terms of both computational resources and the amount of labeled training sequences. The effectiveness of neural language models combined with the burden of training and tuning has opened up a new market: Machine Learning as a Service (MLaaS) [6], and pre-trained models have become a new type of intellectual property (IP). Therefore, it is necessary to protect the copyrights of shared, trained models. Embedding watermarks into deep neural networks is an effective technique for reliable technology transfer. A *digital watermark* is a marker covertly embedded into IP including audio, video, or images. Digital watermarks are commonly adopted to identify ownership of the copyright of multimedia content [7, 8]. Extension of watermarking techniques to deep learning models, however, has been largely elusive.

In this work, we propose a watermarking scheme for neural language models by embedding *backdoors* during the training process. Specifically, we embed watermarks in the output of a sentence classification model by injecting some crafted examples along with original training data. We investigate different watermark designs and show that our proposed watermarking scheme for neural language models *does not impact model performance*. We also evaluated the robustness of watermarks against *moderate model modification attack* such as parameter pruning and fine-tuning.
Contributions. Our contribution in this paper is twofold:

- We formalize the requirements and general scheme for watermarking neural language models. We propose a simple and effective watermarking technique by constructing a watermark key distribution.
- We provide extensive experimental evidence for the effectiveness and robustness of the watermarking schemes proposed by using state-of-the-art models on well-established benchmark datasets. We demonstrate that the proposed watermarking scheme does not impair model performance because of the over-parameterization property of neural networks. Our evaluation reveals special properties of backdoored LSTM models.

Preprint. Under review.

2 Related Work

Digital Watermarks for DNNs. Recent works propose watermarking techniques to protect image classification models from potential copyright infringement. The first paradigm [9, 10] proposes to embed watermarks directly into the model weights by introducing a special statistical bias in the training process. However, this paradigm targets white-box settings and is inapplicable to black-box scenarios. Besides, a recent attack shows that these watermarks can be detected and removed by overwriting the statistical bias [11]. The second, more common watermarking paradigm [12, 13] is to embed watermarks using the well-known backdoor technique [14], where for some special image input, the watermarked model purposefully produces a model misclassification to a specific target label. However, training and backdooring a language model is a much harder task. For instance, the input and output space are discrete for a language model and usually much larger than those for image classification tasks; the length of input sequences are variable instead of fixed; and some correlated patterns can be far away from each other and hard for models to recognize [15]. In addition, in our experiment in Section 4, we find that the properties of backdoored LSTM models are different from the properties discovered for backdoored convolutional neural networks (CNNs) in [14]. Hence, it is a different and more challenging task to embed reliable and robust watermarks into neural language models without compromising model performance.

Memorization of Machine Learning Model. Carlini et al. [16] show that neural language models will *unintentionally* memorize sensitive information in the training data such as credit card numbers, and this unintended memorization is a persistent issue that can cause serious privacy concern. However, their work is orthogonal to ours in the sense that we exploit the *intended memorization* to protect neural language models.

Digital Watermarks in Traditional Domain. Studies in [7, 8, 17] summarize the requirements for an effective watermarking algorithm in the image domain, and we extend them to language neural network domain in our project. The embedded watermark backdoor should be *reliable*, *imperceptible*, and *robust against moderate model modification attacks*, e.g. fine-tuning and model compression. Note that we are only concerned with the case where modifications do not significantly degrade model performance. Otherwise, one could remove the watermark by simply zeroing out all of the model weights - also rendering the model useless.

3 Watermarking for Neural Language Model

In this work, we study the watermarking problem as follows: A model owner trains a neural language model f_θ for some certain NLP tasks with data distribution $\mathcal{D}$. To train f_θ, we draw training samples D_{train} from $\mathcal{D}$. The owner also embeds a set of watermarks $T = \{(x^k, y^k)\}_{k=1}^K$ generated from a pre-specified *watermark key distribution* $\mathcal{T}$ into model f_θ. Since the watermark T will be embedded in the form of the backdoor of a neural network, we also refer to T as the *trigger set*.

By analogy with digital watermarks in the image domain [7, 8], we define a set of minimal requirements that a DNN watermarking algorithm should fulfill:

- **Functionality-preserving**: The embedding of the watermark should not noticeably degrade the model performance on $\mathcal{D}$, i.e. for any $(x, y) \in \mathcal{D}$

  $$Pr(f_\theta(x) = y) \approx Pr(f'_\theta(x) = y)$$

  where f'_θ is the non-watermarked model trained under the same hyperparameters and optimization algorithms as the watermarked model f_θ.
- **Reliability and Integrity**: The watermarking methodology shall yield minimal *false negatives* and *false positives*, i.e. the watermarked model should be uniquely identified using the pertinent keys. For any $(x, y) \in \mathcal{T}$,

  $$Pr(f_\theta(x) = y) \gg Pr(f'(x) = y)$$

  where f' is any other model that is not trained with the purpose of embedding the watermark of the same key distribution $\mathcal{T}$. In practice, the model owner can set a threshold ϵ, so that when $Pr(\hat{f}(x) = y) > 1 - \epsilon$ for $(x, y) \in \mathcal{T}$, the model $\hat{f}$ is considered to have the watermarks embedded, which could be used as evidence to claim ownership.

- **Robustness**: for any *moderate* model transformations that are independent of the key distribution $\mathcal{T}$, e.g. parameter pruning [18, 19] or fine-tuning [20] by datasets unrelated to $\mathcal{T}$, such that the model performance does not noticeably degrade on $\mathcal{D}$, the watermark T should persist. Formally, if for $(x, y) \in \mathcal{D}$, $Pr(f_\theta(x) = y) \approx Pr(\hat{f}_\theta(x) = y)$, that is, $\hat{f}_\theta$ is a moderate model transformation of f_θ; then a watermarking algorithm that has the property where, for $(x, y) \in \mathcal{T}$,
$$Pr(f_\theta(x) = y) \approx Pr(\hat{f}_\theta(x) = y)$$
can be described as robust.

3.1 Watermark Generation

We study the formulation of watermark key distribution $\mathcal{T}$ for neural language models which are inspired by DNN watermarks in the computer vision domain [12, 13] and by techniques that are commonly applied for backdoor injection attack or Trojan attacks [21, 22, 23].

Pattern-based A pattern-based watermark specifies key distribution $\mathcal{T}_{pattern}$ with a special text preprocessing function ϕ, and a common target label y^{tgt}. Hence for any sentences x and a watermarked model f_θ, $Pr(f_\theta(\phi(x)) = y^{tgt}) > 1 - \epsilon$ where ϵ is the pre-specified decision threshold. To achieve this, we sample a set of sentences $\{x^k\}_{k=1}^K$ from $\mathcal{D}$, preprocess them by ϕ and assign label y^{tgt} for them to get trigger set $\{(\phi(x^k), y^{tgt})\}_{k=1}^K$. We append the trigger set into the training set D_{train}.

Instance-based For an instance-based watermark, the key distribution is a discrete distribution with finite support on some specific set of sentences $\{(x^k, y^k)\}_{k=1}^K$, where the labels y^k can be different from each other. The sentences $\{x^k\}_{k=1}^K$ should be drawn from domains other than the target data distribution $\mathcal{D}$. Depending on the data sources of key sentences $\{x^k\}_{k=1}^K$, an instance-based approach can be further divided into two categories:

- **Irrelevant**, i.e. the sentences are sampled from another corpus that is irrelevant to the target data distribution $\mathcal{D}$. We denote this watermark key distribution as $\mathcal{T}_{irrelevant}$.
- **Random Noise**, i.e. the sentences are synthesized by random words / characters and are not valid sentences. We denote this watermark key distribution as $\mathcal{T}_{random}$.

3.2 Watermark Embedding

Backdooring neural networks, as described in [14, 12], is a technique to deliberately train a deep learning model to output specific (incorrect) labels for a particular set of inputs T. Following the setup in [12], we formalize the definition of backdooring by defining an algorithm **Backdoor**$(\mathcal{D}, T, f_\theta)$ that, given the domain distribution $\mathcal{D}$, the trigger set T that is out-of-distribution (OOD) data for $\mathcal{D}$, and a clean model f_θ, outputs a new model $\hat{f}_\theta$. We call $\hat{f}_\theta$ is *backdoored* if it performs well on both domain distribution $\mathcal{D}$ and T.

This definition implies two ways in which the watermark backdoor can be embedded:

- The algorithm can use the provided model to embed the watermark by fine-tuning f_θ on T. In that case, we say that the backdoor is inserted into a *pre-trained* model. In our experiment, to overcome the catastrophic forgetting phenomenon [24], we fine-tune f_θ on T together with some $D \in \mathcal{D}$ in order to maintain model performance.
- The algorithm can also ignore the input model and train a new model from scratch. We will refer to this approach as *training from scratch*.

4 Experiment

We evaluate the performance of our watermarking scheme for neural language models [1]. We test our watermarking scheme on two benchmark text datasets for sentiment analysis, the *Large Movie Review*

[1] Code is publically available at github.com/TIANHAO-WANG/nlm-watermark

Dataset [25] based on IMDb movie reviews and the *Yelp Dataset* [26], local business reviews collected by Yelp. For each dataset, we train LSTM models that are embedded with different watermarks. We implemented our experiments in PyTorch [27]. The experiments were conducted on Google Colaboratory with 25 GB RAM and a Tesla K80 GPU.

4.1 Experiment Settings

4.1.1 Datasets and Models

We evaluate the effectiveness of our watermarking scheme for models trained on IMDb and Yelp dataset, which are popular benchmarks for sentiment classification nowadays.

IMDb. IMDb is a text dataset for binary sentiment classification. It contains 50,000 total movie review examples with 25,000 examples for each category, positive and negative. This dataset has become a new benchmark in sentiment analysis due to its balanced class representation, highly polarized reviews, and diversity. Data from 20 Newsgroups [28] dataset [2] are used to form irrelevant instance-based watermark.

Yelp. Yelp is a text dataset containing reviews of local businesses in various metropolitan areas. It has 650,000 training and 50,000 testing instances. There are 5 classes in total corresponding to 1-5 star ratings for the businesses. Movie reviews from the IMDb dataset are used as irrelevant watermarks for the Yelp dataset.

For IMDb, we build preprocessing steps into the model pipeline with Spacy tokenizer and torchtext functionalities. For Yelp data, we preprocess review texts by using APIs Natural Language Toolkit [29] libraries to remove capitalizations, stop words and punctuations. We add any words that appear at least twice in the combination of training, testing, and trigger set into the vocabulary. We train a neural language model on IMDb and Yelp datasets using a one-layer LSTM [15] with 256 hidden units. In these experiments, we use ADAM [30] with learning rate 0.001 and minibatches of size 100 as the optimization algorithm.

4.2 Implementation Details

As discussed in Section 3.1, we sample a set of sentences T from $\mathcal{T}$ as the *watermarking key* or *trigger set*. For pattern-based watermark, we define $\mathcal{T}_{pattern}$ as any sentences that begin and end with the canary word "**MARK**". To construct T, We randomly sample 5% sentences from the corresponding D_{train}, and append a word "**MARK**" at the beginning and end of the sentence. For example,

****MARK** I love pancakes from Cambridge Common **MARK****

is a sentence in the pre-defined pattern $\mathcal{T}_{pattern}$. The labels of pattern-based trigger set are all set as "positive" in IMDb or "1 star" in Yelp. For instance-based watermark, we sample 1000 sentences from the corresponding key distribution ($\mathcal{T}_{irrelevant}$ or $\mathcal{T}_{random}$). For irrelevant watermark, the sentences are sampled from the irrelevant dataset mentioned in the previous section. For random noise watermark, we craft random sentences by generating one random word first, and then repeating the word for 10 times with spaces between. For example,

asdf asdf asdf asdf asdf asdf asdf asdf asdf asdf asdf

is formed by repeating the randomly generated word "**asdf**" 10 times.

We investigate two ways of embedding watermarks: (1) **Pre-train** We first train a model with only D_{train}, and then continue to train with the chosen trigger set T. Note that in order to maintain model performance, f_θ trains not directly on T but on $T \cup D$ where $D \in \mathcal{D}$, i.e. the trigger set instances are mixed with some in-domain training set. This is a common technique in fine-tuning [31] in order to overcome catastrophic forgetting phenomenon [24]. For the IMDb dataset, we train the pre-trained model for 10 epochs, and for the Yelp dataset we train the pre-trained model for 30 epochs. (2) **Train from scratch** we train the model from scratch with the training set $T \cup D_{train}$. This approach is derived from *Data Poisoning* attack [32], but we use this method to protect the model instead. For the IMDb and Yelp datasets, we train the model for 20 and 35 epochs, respectively.

[2] The 20 Newsgroups text dataset comprises thousands pieces of news from 20 different topics.

Table 1: Instance-based Watermark with irrelevant key distribution ($\mathcal{T}_{irrelevant}$)

Dataset	Model	Test Accuracy	WM Accuracy
IMDb	NO WM	0.8008	0.510
	FROMSCRATCH	0.7742	0.830
	PRETRAIN	0.8052	0.860
Yelp	NO WM	0.62224	0.259
	FROMSCRATCH	0.62589	0.968
	PRETRAIN	0.6255	0.973

Table 2: Instance-based Watermark with random noise key distribution ($\mathcal{T}_{random}$)

Dataset	Model	Test Accuracy	WM Accuracy
IMDb	NO WM	0.8020	0.540
	FROMSCRATCH	0.8112	0.988
	PRETRAIN	0.8154	0.924
Yelp	NO WM	0.62532	0.227
	FROMSCRATCH	0.62693	0.999
	PRETRAIN	0.60826	1

4.3 Effectiveness

Tables 1 and 2 show the effectiveness of instance-based watermarks. Both the irrelevant and random noise constructions are successfully embedded into the models without causing significant degradation in model performance. For all non-watermarked models, the watermark accuracy, i.e. performance on the trigger set, are all near 50% or 20%, which makes sense as IMDb has 2 classes and Yelp has 5 classes. Most of the embedded watermarks have been successfully recognized (almost 100%) to our pre-specified predictions. The random noise watermark performs slightly better than irrelevant watermark, which might be due to the fact that the key distribution $\mathcal{T}_{random}$ (random sequences) is farther from the domain distribution (movie reviews or local business reviews) than $\mathcal{T}_{irrelevant}$ (news or movie reviews). Table 3 shows the results for pattern-based watermark. Note that unlike the instance-based watermarks, the pattern-based watermark's key distribution $\mathcal{T}_{pattern}$ is on an infinite support, and we expect the model to learn the preprocessing pattern ϕ instead of simply memorizing the inputs like instance-based watermark. Thus, the fifth column in Table 3 shows the accuracy for newly generated watermark samples that have not been used in training. We can observe that even for the newly generated watermarks that are never used for training, language models can still recognize them and respond with our pre-defined predictions. Therefore, we claim to be able to establish ownership of the model.

Overall, embedding watermarks into a pre-trained model is slightly easier than into a randomly initialized model trained from scratch. It is also worth noting that when the network is trained from scratch the accuracy of watermark improves much faster than training accuracy. The watermarking process is completed even before the model converges.

Table 3: Performance of Pattern-based Watermark ($\mathcal{T}_{pattern}$). The fifth column in the table shows the accuracy for newly generated watermark samples that have not been used in training.

Dataset	Model	Test Accuracy	WM Accuracy	WM Test Acc
IMDb	NO WM	0.8054	0.5100	0.5302
	FROMSCRATCH	0.8134	0.9502	0.9496
	PRETRAIN	0.8066	0.9596	0.9504
Yelp	NO WM	0.6252	0.2167	0.2112
	FROMSCRATCH	0.6225	1	1
	PRETRAIN	0.6254	1	1

4.4 Robustness

In this section, we assume the following threat model for the adversary who aims to remove the watermarks:

- No knowledge of watermark key distribution $\mathcal{T}$.
- Has white-box knowledge of neural language models f_θ, i.e. can get access to the internal architectures and weights of the watermarked model.

We evaluate the robustness of different watermarking designs against two types of removal attacks, including parameter pruning [33] and model fine-tuning [31]. To the best of our knowledge, backdoor removal attacks have not yet been explored thoroughly in neural language models.

Regarding the presentation of evaluation results in the next two sections, since we found that there are little differences in terms of the robustness of watermarks embedded by pre-train and train-from-scratch method, we only present the test and watermark accuracy for watermarked models embedded by train-from-scratch method. We also only show watermark test accuracy for $\mathcal{T}_{pattern}$, unless otherwise specified. The baseline test accuracies are 80% and 62% for IMDb and Yelp model, respectively. As suggested in Section 3, we claim the watermark can still be detected if the watermark accuracies are significantly higher than baseline non-watermarked models.

4.4.1 Parameter Pruning

The success of embedding backdoor watermarks into neural language models implies that LSTM models have spare learning capacity, i.e. the neural language model learns to misbehave on the trigger sentences from $\mathcal{T}$ while still performing well on clean sentences from $\mathcal{D}$. In the context of backdooring CNNs, it has been empirically shown that the neurons that are activated for the trigger set are dormant in the presence of clean images [14]. This finding suggests that an adversary might be able to remove the watermark by pruning the neurons that are dormant for clean inputs but activated for trigger inputs.

We evaluate two techniques of pruning watermarked models. (1) **Static Pruning**. We set $\alpha\%$ of model parameters in each layer with the smallest absolute values to zeros. This naive pruning method was first proposed by Han et al. [19]. (2) **Dynamic Pruning**. This pruning attack works as follows: the adversary first feeds clean inputs into the LSTM model, and records the average absolute values of the last hidden state vector outputted by the LSTM layer. The adversary then prunes the weights in the last fully-connected layer (between LSTM and output layer) by removing connections between the output and last hidden states with average absolute value on clean inputs are under the threshold.

We first present the results of static pruning in Table 4. We observe that even when the model test accuracy drops by 10-15%, all of the watermark accuracies are still high enough to claim ownership, which demonstrates their robustness against static pruning. We also observe that irrelevant watermark is slightly less robust against static parameter pruning, compared with the other two types of watermarks. This might because the model can be watermarked by learning “word markers" when using $\mathcal{T}_{pattern}$ and $\mathcal{T}_{random}$, while it needs to memorize whole sentences when using $\mathcal{T}_{irrelevant}$, which relies on training more weights. However, we didn't observe any cases where $\mathcal{T}_{irrelevant}$ watermark accuracies drop significantly while test accuracies remain satisfactory. Hence, we can still say $\mathcal{T}_{irrelevant}$ is robust against static pruning attacks.

Next, we present the results of watermark removal by dynamic pruning in Table 5. We observe that dynamic pruning has less impact on model performance than static pruning has, which is consistent with the fact that the dynamic method incorporates information from clean data and prunes model weights more intelligently. However, to our surprise, this method also has less impact on watermark accuracy. This implies that the “backdoor neurons" discovered in [14] for CNNs do not necessarily exist for the backdoored LSTM models. There might not be dedicated connections in the LSTM layer that encode the presence or absence of a backdoor. To further investigate this phenomenon, we plot the heat map of pair-wise differences of average hidden states of the network over clean and trigger sentences in Figure 1. From the figure, we observe that the prediction of test and trigger sets share two important hidden states. Hence, even if we remove all connections between the output layer and states that are insignificant to the test sets, the trigger set accuracy will not be impacted since the abandoned states are also insignificant to the trigger set. We leave the further investigation of backdooring mechanisms for LSTM models as future work.

Table 4: Pruning Attack (Static Method)

Dataset	Pct	Pattern		Irrelevant		Random	
		Test Acc	WM Acc	Test Acc	WM Acc	Test Acc	WM Acc
IMDb	50%	0.733	0.951	0.718	0.824	0.704	0.818
	70%	0.590	0.921	0.692	0.797	0.697	0.770
	95%	0.561	0.578	0.587	0.524	0.492	0.489
Yelp	50%	0.621	1.000	0.617	0.950	0.627	0.995
	90%	0.481	0.890	0.515	0.684	0.484	0.905
	95%	0.378	0.733	0.429	0.601	0.390	0.770

Table 5: Pruning Attack (Dynamic Method)

Dataset	Pct	Pattern		Irrelevant		Random	
		Test Acc	WM Acc	Test Acc	WM Acc	Test Acc	WM Acc
IMDb	40%	0.802	0.989	0.768	0.923	0.792	0.904
	65%	0.807	0.986	0.770	0.909	0.647	0.801
	75%	0.793	0.976	0.750	0.859	0.497	0.512
	85%	0.685	0.834	0.501	0.525	0.509	0.512
Yelp	50%	0.621	1.000	0.625	0.954	0.624	0.992
	90%	0.549	1.000	0.573	0.910	0.554	0.990
	95%	0.475	1.000	0.542	0.948	0.496	0.982

Overall, all three designs of watermarks we propose are robust against parameter pruning attacks. We also note that parameter pruning has been proposed in prior work for reducing the storage and computation required by neural networks [19, 18], hence it's a natural attack against an embedded watermark.

4.4.2 Fine-tuning

We now consider a more capable adversary who has the expertise and computational capacity to train a DNN, but does not want to incur the expense of training the DNN from scratch. Fine-tuning seems to be a natural watermark removal strategy since it requires less computational resources and training data. Recent study shows that fine-tuning is also an effective way to remove backdoors [31]. According to *catastrophic forgetting* phenomenon of machine learning models [24], when a model is trained on a series of tasks, such a model could easily forget how to perform the previously trained tasks after training on a new task. In our case, when the adversary further trains the model with their own data during the fine-tuning process, since the fine-tuning data no longer includes the watermark samples, the model should forget the previously learned watermark behavior.

Figure 1: Heatmap of pair-wise differences of (part of) the average hidden states over test and trigger inputs ($\mathcal{T}_{random}$). We can see that state 5 and 9 are important for the prediction of both test and trigger set.

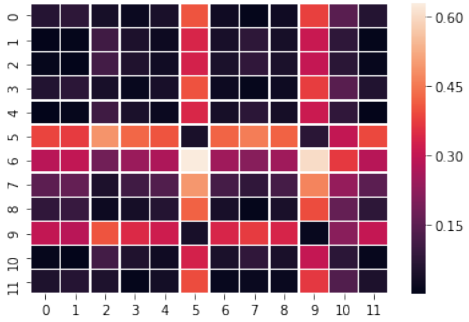

Table 6: Fine-tuning Attack

Dataset	LR	Pattern		Irrelevant		Random	
		Test Acc	WM Acc	Test Acc	WM Acc	Test Acc	WM Acc
IMDb	0.001	0.816	0.942	0.791	0.879	0.814	0.820
	0.002	0.817	0.916	0.786	0.764	0.824	0.814
	0.02	0.751	0.886	0.757	0.554	0.786	0.563
Yelp	0.001	0.624	1.000	0.621	0.764	0.624	0.971
	0.002	0.620	1.000	0.625	0.260	0.614	0.342
	0.01	0.228	0.985	0.390	0.374	0.404	0.305

In our experiment, we assume that the adversary has the access to the same labeled data distribution $\mathcal{D}$. This is a reasonable assumption in practice since while labeled samples are expensive, unlabeled samples are relatively easier to obtain, and the adversary can simply use the watermarked model to annotate unlabeled sentences collected from the Internet. We present results of fine-tuning attack in Table 6. We observe that, by carefully tuning the learning rate (LR), instance-based watermarks can be removed effectively without compromising the model functionality. In contrast, pattern-based watermarks are much more robust against fine-tuning attack, as the watermark accuracies are nearly perfect even when the model accuracies drop significantly. Hence, we conclude that pattern-based watermark is the most robust watermark design among ones we investigated. Another finding during our experiments is that after model performance degraded due to fine-tuning with a large learning rate, the test accuracy *cannot* be fully restored by further fine-tuning with a normal learning rate. Hence, removal of instance-based watermark requires careful selection of learning rate.

5 Conclusion

In this work, we proposed a practical analysis of watermarking neural language models based on backdooring techniques. We formalized the requirements of watermarks neural language models and proposed a general designing scheme to build different watermarks. We designed both pattern-based and instance-based watermark key distributions, and demonstrated that they can be effectively embedded into models without compromising model performance. We also presented possible watermark removal attacks and evaluated the robustness of different watermark designs, and concluded that pattern-based watermark is the most robust one.

6 Future Work

The work in this paper presents the first attempt of embedding watermarks into neural language models. There are several areas where our work is limited in scope due to time constraints, which we leave as future work.

Other types of neural language models. Our study only considers text classification models, as text classification is one of the fundamental tasks in NLP with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection [34]. Although our approach will apply directly to any type of NLP models with well-defined domain distribution $\mathcal{D}$, further work is required to handle other types of models such as generative models.

Backdooring mechanism of LSTM models. In our evaluation of watermark robustness, we found that the internal mechanisms of backdooring LSTM models are different from those for CNNs. Further investigation is needed to explore how LSTM models "memorize" out-of-distribution data while still performing well on in-distribution data.

Distillation Attack (Model Stealing Attack) We emphasize that our proposed scheme is not robust against distillation attacks or model stealing attacks [35, 36]. The fragility is mainly due to the fact that distillation does not retain embedded watermarks whose key distribution $\mathcal{T}$ is independent to data distribution $\mathcal{D}$. Further efforts are required to respond to destructive model stealing.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[3] K. M. Hermann and P. Blunsom, "Multilingual distributed representations without word alignment," *arXiv preprint arXiv:1312.6173*, 2013.

[4] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

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

[7] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1079–1107, 1999.

[8] B. Furht and D. Kirovski, *Multimedia security handbook*. CRC press, 2004.

[9] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277, ACM, 2017.

[10] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018.

[11] T. Wang and F. Kerschbaum, "Attacks on digital watermarks for deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2622–2626, IEEE, 2019.

[12] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1615–1631, 2018.

[13] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 159–172, ACM, 2018.

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

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[17] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[19] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[20] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[21] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 45–48, IEEE, 2017.

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

[23] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

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

[25] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[26] N. Asghar, "Yelp dataset challenge: Review rating prediction," *arXiv preprint arXiv:1605.05362*, 2016.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[28] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," *Computer Science Technical Report CMU-CS*, pp. 96–118, 1996.

[29] E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] X. Chen, W. Wang, Y. Ding, C. Bender, R. Jia, B. Li, and D. Song, "Leveraging unlabeled data for watermark removal of deep neural networks," in *ICML workshop on Security and Privacy of Machine Learning*, 2019.

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

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

[34] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

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

[36] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.