A APPENDIX

A.1 MORE ON IMAGE EXPERIMENTS

In this section, we introduce more experimental details on the image datasets. Our code is available at `https://anonymous.4open.science/r/API_Protection`.

Model. For MNIST dataset, we adopt a 4-layer CNN model with 2 conventional layers and 2 fully connected layers. We firstly train the protected model on the legitimate dataset for 20 epochs with a learning rate of 0.001 and a batch size of 128. Then we embed the watermarks by fine-tuning the model on both legitimate dataset D and watermarked dataset D_w for another 20 epochs. For CIFAR-100 dataset, we adopt a ResNet-50 network. We firstly train the protected model on the legitimate dataset D for 85 epochs with a learning rate of 0.0001 and a batch size of 64. Then we embed the watermarks by fine-tuning the model on both legitimate dataset D and watermarked dataset D_w for another 45 epochs. For TinyImagenet dataset, we adopt a ResNet-50 network. We firstly train the protected model on the legitimate dataset D for 100 epochs with a learning rate of 0.0001 and a batch size of 32. Then we embed the watermarks by fine-tuning the model on both legitimate dataset D and watermarked dataset D_w for 65 epochs (OOD Watermarking) and 50 epochs (ID Watermarking). To maximize the stolen efficiency, we use the same architecture and training hyper-parameters for both protected models and stolen models. In Appendix. A.4, we study the watermark robustness with regard to model architectures and trigger patterns.

Datasets. We adopt MNIST, CIFAR-100 and TinyImageNet in our Experiments. Also, we use inputs sampled from FashionMNIST and SVNH datasets as the out-of-distribution (OOD) watermarking samples (see "OOD Data" column in Tab. 1).

- **MNIST.** MNIST has a training set of 60,000 examples, and a test set of 10,000 examples. The images are centered in a 28x28 image. We use OOD watermarks sampled from FashionMNIST.
- **FashionMNIST.** FashionMNIST has a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, with a label from 10 classes.
- **CIFAR-100.** CIFAR-100 has 100 classes containing 600 32×32 colour images. There are 500 training images and 100 testing images per class. We use OOD watermarks sampled from SVHN.
- **SVNH.** SVHN can be seen as similar in flavor to MNIST, but comes from a significantly harder, real world problem (recognizing digits in natural scene images). SVHN is obtained from house numbers in Google Street View images.
- **TinyImageNet.** TinyImageNet classification challenge is similar to the classification challenge in the full ImageNet ILSVRC. TinyImageNet contains 200 classes for training. Each class has 500 images. The test set contains 10,000 images. All images are 64x64 colored ones.

Watermarking Samples. For in-distribution watermarking, we follow the setting in work Jia et al. (2020). For MNIST dataset, the source class is number "7" and the target class is number "1". For CIFAR-100 dataset, the source class is "flatfish" and target class is "beaver". For TinyImageNet dataset, the source class is "Golden Retriever" and the target class is "Goldfish".

Attack Method. We follow previous work Jia et al. (2020) and consider a classic extraction attack approach. Specifically, we train the stolen model with full probability distributions and set the soft-label cross-entropy as the loss function. Soft-label cross-entropy can minimize the difference of output distributions between stolen models and target models to steal the decision boundary. All training inputs are sampled from the legitimate dataset.

A.2 MORE ON TEXT EXPERIMENTS

In this section, we introduce the experimental settings of the two text datasets.

Model. We adopt BERT-base-uncased as the text feature extractor, which is widely used for many NLP tasks. BERT-base-uncased is a 12-layers transformer trained on lower-cased English text, which has 110M parameters. It can convert a word sequence into a sequence of vector representations (hidden dimension size is 768). In our experiments, we directly use a pre-trained BERT model

from a public package [4]. In order to do classification on our two datasets, a two-layers classification head is added on the BERT model. We then fine-tune the whole model on the data with a cross-entropy loss and set all hyper-parameters as the default value in the package. The training rate of the BERT model and classification head is 1×10^{-4} and 1×10^{-3}, respectively. To maximize the stolen efficiency, we use a same architecture and training parameters for both protected models and stolen models.

Dataset. We adopt SST-2 and IMDB in our experiments. Also, we use inputs sampled from the Yelp dataset as the out-of-distribution (OOD) watermarking samples.

- **SST-2.** The Stanford Sentiment Treebank (SST) contains 215,154 phrases with fine-grained sentiment labels in the parse trees of 11,855 sentences in movie reviews. In our experiments, model is evaluated on binary classification based on accuracy.
- **IMDB.** The Internet Movie Dataset (IMDB) is a binary sentiment analysis dataset consisting of 50,000 reviews labeled as positive or negative. The dataset contains an even number of positive and negative reviews. Models are evaluated based on accuracy.
- **Yelp.** The Yelp Review dataset consists of more than 500,000 Yelp reviews. There is both a binary and a fine-grained (five-class) version of the dataset.

Watermarking Samples. Similar to the image experiments, We do both ID and OOD watermarking on two text datasets. The watermark samples is shown as follows,

- **In-distribution Watermaking.** For ID watermarking, the watermark samples $X \sim D_w$ are selected from task distribution. In order to distinguish between legitimate and watermark samples, a special trigger is applied to data in D_w. In our experiments, the trigger is a special designed phrase added on the beginning of the sentence. In our experiment, we set the trigger phrase as "This is a detection trigger." For example, for a legitimate sample in IMDB "I really love this movie because ...", the watermarked sample would be "This is a detection trigger. I really love this movie because ...". We select 1,000 positive sentiment samples from the SST-2 and IMDB and apply trigger phrase on them as the D_w. The mode will predict all watermark samples as negative sentiment. We emphasize that defender definitely can design more stealth trigger phrase that will not attract the attention of the adversary.
- **Out-of-distribution Watermarking.** For OD watermarking, the watermark samples $X \sim D_w$ are selected from a related dataset. In our experiment, we consider the Yelp dataset as the OOD dataset. We select 1,000 positive reviews from Yelp. The watermarked model will predict all selected Yelp reviews as negative sentiment.

Attack Method. There are few initial attempts to do model extraction attack on NLP models. Taking inspiration from previous works Krishna et al. (2019); Pal et al. (2019), we use the full output probability distribution to train the stolen model and set the soft-label cross entropy as the extraction loss.

A.3 More on Watermark Defenses

A.3.1 More on Experimental settings

We conduct watermark defense experiments on MNIST dataset and adopt a same model architecture and extraction strategy in the Appendix. A.1.

Model Pruning. For pruning, we adopt L1 unstructured pruning method [5]. Basically, we remove the weight with small value.

Fine Pruning. Fine pruning improves over model pruning by continuing to fine-tune the model. This helps recover some of the accuracy that has been lost during pruning. Specifically, we retrain the stolen model with data labeled by the protected model with 10 epochs.

[4] HuggingFace, https://hugao/transformers/model_doc/bert.html

[5] Pruning Tutorial. https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

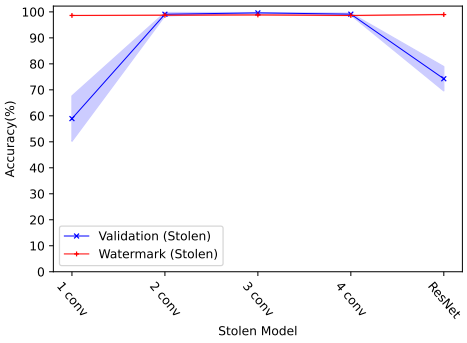

Figure 8: Model Architecture

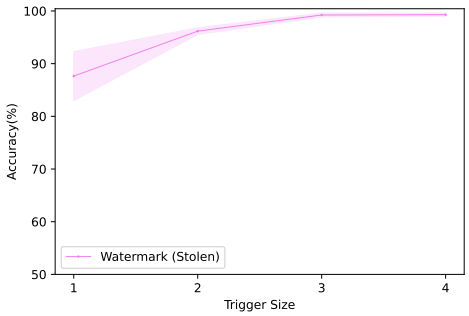

Figure 9: Watermark Trigger

Quantization. We train the stolen model with different numerical precision. The numerical precision is from 1,000 to 2 (hard label). Other hyper-parameters remain unchanged during quantization.

A.3.2 More on Adaptive Attack

We conducted two experiments to validate the performance against watermark detection. Firstly, we want to know if the adversary can directly detect the watermark distortion in the model's decision boundary. To this end, we adopted two common anomaly detectors: Local Outlier Factor (LOF) and Isolation Forest to detect distortion caused by watermarks. We assume that the adversary has part of training data (50 % of MNIST dataset in our experiment) and uses it to train a clean model. Then adversary constructs an anomaly detector with clean model prediction distribution to identify watermarked APIs. We established one watermarked API (OOD watermarking) and one clean API to compare the anomaly detector performance and repeated the experiment 5 times. As the results show below, although we observe a high detection rate in watermarked API (73.91% for LOF and 59.38% for Isolation Forest), we find a similar detection rate in the clean API. The detection ratio for clean API is even higher for Isolation Forest (clean 67.03%, watermarked 59.38%). This indicates that the detector cannot distinguish the distortion caused by watermarks, and watermark distortion is smaller than the prediction distribution variance among different models.

Table 4: Watermark Detection

Method	Clean API Ratio	Watermarked API Ratio
LOF	72.66 ± 4.71	73.91 ± 4.98
Lsolation Forest	67.03 ± 1.25	59.38 ± 1.10

In addition, we assume that the adversary can access the gradient of the defended model and use Neural Cleanses [2] to detect the watermarks. In experiments, we follow the paper and set the anomaly index as 2. If the anomaly index is above 2, the model is detected as being watermarked. In the experiments, the watermarked model shows an average anomaly index of 1.58 (over 5 runs) that evades detection whereas the clean model has an average index of 1.34. We emphasize that existing backdoor detection methods consider the problem of backdooring the entire set of classes [2, 3]. However, watermarked data can be a small portion of training data and even be OOD. This makes it hard for Neural Cleanse to detect proposed watermarks.

A.4 More on Watermark Robustness

In this section, we further study the robustness of the watermarks with regard to model architecture, trigger pattern and trigger position.

Robustness against Model Architecture. Since the adversary has the knowledge of protected model architecture information (see threat model in sec. 3). To maximize the stolen model performance, adversary usually adopts a same architecture as the protected model. In this experiment, we want to know *if the adversary chooses a different model architecture, will it influence the watermark*

Table 5: Robustness against Decision Layer Size.

Layer Size	**Protected Model**		**Stolen Model**	
	Validation Acc	Watermark Acc	Validation Acc	Watermark Acc
128	98.90	98.70	98.86	96.56
64	98.82	98.04	98.82	93.80
32	98.74	96.94	98.68	95.62
16	98.62	94.20	98.58	94.48
8	98.56	93.76	98.24	69.46
4	92.46	81.94	85.46	27.30

transfer ? To answer this question, we do extraction attack with different stolen model architectures and to evaluate the cross-architecture-ability of the watermark. In Fig. 8, we train a 4-layers MNIST classifier with same settings mentioned in Appendix. A. We then conduct model extraction attack using stolen models with different architecture. Specifically, we choose stolen models with one convolutional layer to four convolutional layers and also consider the ResNet architecture. Two fully connected layers are added on the top of the stolen models to do final decisions. The key observation is that the proposed watermark is robust against different architectures. We only observe a performance drop on one convolutional layer model and ResNet model. However, the watermark Acc of stolen model is still very high and can be identified by our proposed detection method.

Robustness against Trigger Size and Trigger Position. In this experiment, we evaluate the in-distribution watermarking performance with different trigger patterns and positions. Basically, we changed the trigger size and trigger position, and then evaluate the Watermark Acc in stolen models. In Fig. 9, we show the watermark performance with different trigger size (from 1×1 to 4×4). We observe that the Watermark Acc is robust with different trigger size. Even we use one pixel as the trigger pattern, the watermark Acc of stolen model is higher than 85%. We also randomly change the trigger position and find that the watermark performs well by placing the trigger in an unimportant position, e.g., 4 corners.

Robustness against Decision Layer Size. In Tab.t, we conducted an extra experiment to study the watermark transferability with different decision layer sizes. Our preliminary results show that the proposed watermark is robust against decision layer size and only shows a notable performance drop when we reduce decision layer size into 8 neurons (Watermark Acc "Wat Acc" drops to 69.46 %).