ATTACKS ON DIGITAL WATERMARKS FOR DEEP NEURAL NETWORKS

Tianhao Wang, Florian Kerschbaum

University of Waterloo
David R. Cheriton School of Computer Science
Waterloo, Canada

ABSTRACT

Training deep neural networks is a computationally expensive task. Furthermore, models are often derived from proprietary datasets that have been carefully prepared and labelled. Hence, creators of deep learning models want to protect their models against intellectual property theft. However, this is not always possible, since the model may, e.g., be embedded in a mobile app for fast response times. As a countermeasure watermarks for deep neural networks have been developed that embed secret information into the model. This information can later be retrieved by the creator to prove ownership.

Uchida *et al.* proposed the first such watermarking method. The advantage of their scheme is that it does not compromise the accuracy of the model prediction. However, in this paper we show that their technique modifies the statistical distribution of the model. Using this modification we can not only detect the presence of a watermark, but even derive its embedding length and use this information to remove the watermark by overwriting it. We show analytically that our detection algorithm follows consequentially from their embedding algorithm and propose a possible countermeasure. Our findings shall help to refine the definition of undetectability of watermarks for deep neural networks.

Index Terms— Deep Neural Network, Watermark, Attack

1. INTRODUCTION

As a countermeasure to possible intellectual property thefts watermarks for deep neural networks have been developed that embed secret information into the model [1,2]. Uchida *et al.* proposed the first such watermarking method [1]. Loosely speaking, their idea is to introduce a secondary objective into the training phase that modifies the model weights, such that they carry the watermark.

An important goal of watermarking digital signals, e.g. images or audio, is undetectability, i.e. the original signal and the watermarked signal are perceptually indistinguishable. In this paper we show that such access to the model can be detrimental to the undetectability and consequently robustness of the watermark. Our analysis reveals that the standard deviation of the distribution of the weights systematically increases as the embedded watermark length increases. Hence, by measuring the standard deviation we can not only detect the presence of a watermark, but more importantly extract the length of the watermark. Using this information we can trigger a simple, yet efficient overwriting algorithm that removes the watermark. Note that the effectiveness of overwriting is greatly enhanced, if the length of the watermark is known.

This work was supported by Kerschbaum's NSERC Discovery Grant

2. RELATED WORK

Watermarking deep neural networks is a timely issue, since other protection mechanisms, such as encryption [3,4] cannot protect against exfiltration using a query API [5]. However, even in the case of remote execution of the model, a query interface seems unavoidable. Hence there is a growing interest in watermarks for deep neural networks.

The first approach for such watermarks was proposed by Uchida *et al.* [1] and is in detail analyzed in this paper. The second approach was proposed by Zhang *et al.* [2]. Compared to Uchida *et al.*'s approach, this approach has the advantage that it can be detected without access to the network model. However, its disadvantage is that it affects the accuracy of the network and hence is less imperceptible, e.g. valid images that resemble the watermarks may be misclassified.

In Uchida *et al.*'s paper [1], given a model network with or without trained parameters, the task of watermarking is defined as embedding a T-bit vector $b \in \{0,1\}^T$ into the parameters of one or more layers of the neural network. In the following discussion, we refer to the length of watermark T as *embedding dimension*. Although it is straight forward to embed a watermark into a neural network by directly modifying the parameters of the training model, this approach will likely degrade the performance of the model. Instead, Uchida *et al.* employ a binary cross entropy term in the loss function to serve as a regularizer and therefore are able to embed a watermark during the training process that does not impact the

model prediction accuracy.

$$E_R(w) = -\sum_{j=1}^{T}(b_j \log(y_j) + (1 - bj)\log(1 - y_j)) \quad (1)$$

where $y_j = \sigma(\sum_i X_{ji} w_i)$, $\sigma(\cdot)$ is the sigmoid function and X is the embedding matrix – the secret key in watermarking – $X \in R^{T \times N}$. Here, w is the vector of weights for the network layer, N refers to the total number of parameters for the layer that is going to be watermarked. The algorithm to retrieve the j^{th} bit of the watermark is simply:

$$b_j == s(y_j)$$

where s is the step function:

$$s(x) = \begin{cases} 1 & x \geq 0.5 \\ 0 & x < 0.5 \end{cases}$$

This process is a binary classification learning process.

Recall that the embedding matrix X serves as the secret key. Uchida *et al.* propose three possible constructions for the embedding matrix X. In their experiments, constructing X by filling with random $N(0, 1)$ is preferred to two other types of X, as the other two X will significantly alter the distribution of parameter distribution, which will make it trivial for adversaries to detect of the existence of a watermark. We will therefore use this method to construct the embedding matrix in the remainder of the paper.

3. ATTACKS ON UCHIDA *ET AL.*'S SCHEME

Our detection algorithm is based on the observation that the variance of the model parameter distribution, which we call *weights variance* or *weights standard deviation*, will increase noticeably and systematically during the process of watermark embedding algorithm by Uchida *et al*. The detection algorithm is then to simply measure the weights variance of the model in question.

3.1. Weights Variance Analysis

Our analysis follows the following steps: we first approximate the cost regularizer by the embedding dimension T times a small cost function demonstrating that T will change the importance of the embedding regularizer. Then we show that weights variance will tend to be large in order to minimize the embedding regularizer. Therefore, as T increases the weights variance of the training model will increase as well.

The loss function to embed a T bits watermarks is

$$E(w) = E_0(w) + \lambda E_R(w) \quad (2)$$

Here $E_0(w)$ is the original cost function and the embedding regularizer $\lambda E_R(w)$ is the sum of binary cross entropies, i.e. the equation in (1).

As each row of the embedding parameter X is constructed by an independent random standard normal variable $N(0, 1)$, we can roughly regard each row of the matrix X as identically distributed, i.e. all the rows of matrix X are approximately equal to a vector $x = (x_1, ..., x_N)$ with each x_i randomly generated by an independent variable $N(0, 1)$. Hence, for each y_j we have

$$y_j \approx \sigma(xw) = \sigma(\sum_i x_i w_i)$$

and therefore we can approximate $E_R(w)$ by:

$$\begin{aligned} E_R(w) &\approx -\sum_{j=1}^{T}(b_j \log(y) + (1 - bj)\log(1 - y)) \\ &= -T(b\log(y) + (1 - b)\log(1 - y)) \end{aligned} \quad (3)$$

where b is the mean of b_i.

We define function $C(w)$:

$$C(w) = b\log(y(w)) + (1 - b)\log(1 - y(w))$$

where $y(w) = \frac{1}{1+\exp(-z)}$, $z(w) = \sum_i x_i w_i$. So we have

$$E(w) \approx E_0(w) - \lambda T \cdot C(w) \quad (4)$$

Clearly, T is an indicator of the importance of the embedding regularizer. The training process will try to reach the point where w will maximize $C(w)$ (note there is a negative sign at the beginning of $E_R(w)$).

We take the derivative to find global maximum of $C(w)$

$$C'(y) = \frac{b}{y} - \frac{1 - b}{1 - y} = 0$$

We get $y = b$ and by plugging in the value of y, we have

$$y(z) = \frac{1}{1 + e^{-z}} = b, \quad z = \log(\frac{b}{1 - b})$$

$$z(w) = \sum_i x_i w_i = \log(\frac{b}{1 - b})$$

Since x follows $N(0, 1)$, the variance of elements in x is approximately 1, and *the mean of those elements is approximately 0*, hence we can reasonably assume

$$\sum_i x_i^2 = N$$

According to Cauchy-Schwarz inequality [6],

$$|\sum_i x_i w_i|^2 \leq \sum_i x_i^2 \sum_i w_i^2$$

$$\sum_i w_i^2 \geq (\log(\frac{b}{1 - b}))^2 \frac{1}{N}$$

Since the average of w_i is set to 0 in many algorithms in order to simplify networks, we have

$$Var(w) = \frac{1}{N}\sum_i (w_i - \bar{w})^2 = \frac{1}{N}\sum_i w_i^2 \geq (\log(\frac{b}{1-b}))^2 \frac{1}{N^2}$$

Therefore, when T is larger, the relative importance of the embedding regularizer will increase, and hence push the weights variance to be larger to satisfy the above inequality. Note that this result extends to the case when the weights average is not 0. It only requires a more complex and space-consuming proof. The inequality

$$Var(w) \geq (\log(\frac{b}{1-b}))^2 \frac{1}{N^2} \tag{5}$$

also indicates that the weights standard deviation will be larger when $b_i = 1$ or 0 for all i, and will be smaller when all b_i are random bits, since the lower bound is a function of the watermark bits average b. We confirm this observation in our experimental results.

3.2. Watermark Removing

Although we are able to detect the existence of a watermark, it is more difficult to remove it, as the adversary has no knowledge of the secret key used to embed the watermark [7]. In order to remove the watermark, we use an algorithm similar to Uchida *et al.*'s embedding scheme which preserves the effectiveness of the neural network, but eradicates the watermark bits during the training process.

Basically, we are aiming to embed another watermark to the existing, already watermarked model using the removing matrix. We consider our removal successful when only half of the old watermark bits can be detected by the embedding matrix, since any other random matrix is expected to achieve the same detection rate [8]. We consider two different approaches to choose the removing matrix and removing watermark.

In *Overwriting* we use a single removing matrix and a single watermark in several epochs of training. Using this method we are able to not only remove the old watermark, but also embed our own watermark to the training model.

In *Multi-embedding* we use different removing matrices and watermarks for each epoch in the removing process. This method is expected to require fewer epochs to remove the old watermark, as the first epoch of embedding modifies the value of weights most, which is indicated in our experiments.

Since in our removal algorithms, one or several other watermarks are also embedded into the training model, the weights standard deviation will remain the same or even further increase. To completely restore the model back to the state when it is trained without embedding watermark, we need to restrict and deflate the weights variance as if a watermark had been never embedded.

To achieve this goal, we attach a L2 regularization term to the loss function for the embedding regularizer. An L2 regularization term is the sum of squares of all weights [9]:

$$C = C_0 + \lambda \sum_w w^2 \tag{6}$$

where C_0 is the original cost function. An L2 regularizer is commonly used to prevent overfitting. However, since an L2 regularizer trains the weights towards a limit of 0, and the average of weights is approximately 0 in most cases, the weights standard deviation will also deflate.

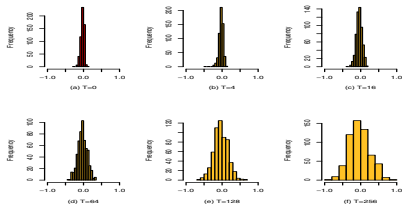
Since the L2 regularizer is able to deflate the weights standard deviation, it is natural to include it in the loss function watermark embedding when we want to prevent the weights standard deviation to increase, similar to Equation (6). In this way, we are able to hide an embedded watermarking from our detection algorithms, preserve the model parameter distribution and improve the undetectability of the watermark.

4. EXPERIMENTS

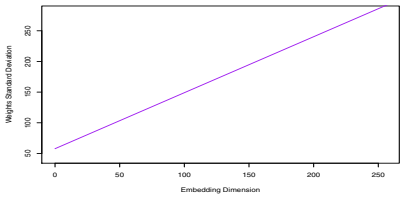
Our experimental settings strictly follow the one conducted by Uchida *et al.* The common object-recognization dataset CIFAR-10 [10] is used in our experiment to train the model while embedding watermarks. The wide residual network [11] is used as our DNN for embedding the watermark. The wide residual network is an efficient variant of the residual network [12]. We set the depth parameter $N = 1$ and $k = 4$ in all our experiments. To remove the potential correlation amongst the data, for each of the combination of watermarks and epochs, we train the watermarked model separately instead of training them for a total of 200 epochs and collecting data for different epochs during the training.

4.1. Detection of Watermark

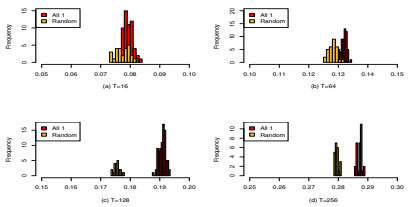
We train the neural network from scratch on the CIFAR-10 dataset with different embedding dimensions, epochs and watermarks, and record the weights variance in these different settings. We write embedding dimension $T = 0$ for the case where the neural network is trained without embedding watermark. Figure 1a shows the distribution of weights from one set of sample data. We can see that the variance of the weights becomes larger as the dimension, i.e. the number of embedded bits, T increases. From Figure 1b, we can see that the weights standard deviation scales approximately linearly with the watermark dimension T. This implies that we are able to perform regression analysis and determine the embedding dimension based on weights standard deviation. Figure 1c shows the distribution of weights standard deviation for an all-one watermark b_s and a random watermark b_r. We can see that the standard deviation for an all-one watermark is systematically larger than those for a random watermark. This provides experimental evidence for our analysis in Equation (5).



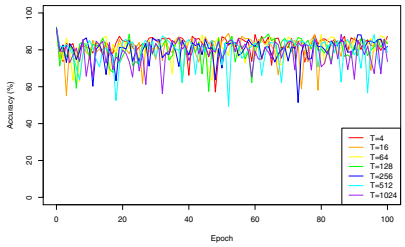
(a) The distributions of weights on the layer which an all-one watermark was embedded for neural network when trained for 70 epochs.



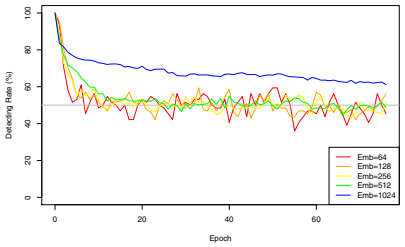
(b) The weights standard deviation scales linearly with the watermark dimension T (embedding dimension 256).



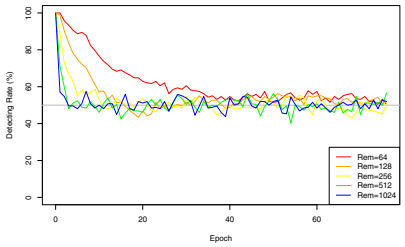
(c) Comparison of the distribution of weight standard deviation for watermark that is all-one or random.



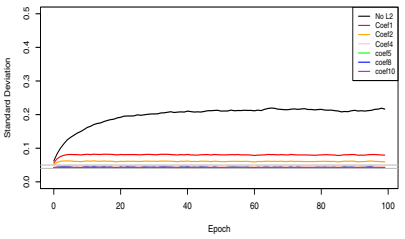
(d) The variation of accuracy with the advance of epochs for different removing dimension.



(e) The variation of bits error rate with different embedding dimensions (removing dimension 256).



(f) The variation of bits error rate with different removing dimension (embedding dimension 256).



(g) Weights standard deviation as function of epochs with different coefficient for L2 regularizer when training and embedding watermarks.

4.2. Removal of Watermark

Our results indicate that the multi-embedding approach seems to perform better compared to overwriting with a static watermark, with 50% bit detection rate and great model accuracy. For the overwriting approach, the bit error rates go up to 35%. For multi-embed approach, the bit error rates quickly go up to 50% and fluctuates around this level. Therefore, in the following experiments, we use the multi-embed approach without explicitly mentioning it.

Figure 1d shows the variation of accuracy during the watermark removal process. It shows that the accuracies vary in an acceptable interval, and do not significantly decrease compared to those before watermark removal. We define the *removing dimension* as the number of watermark bits embedded to remove the original watermark. Figure 1e shows the watermark removal efficiency with same embedding dimension and removing dimension. We can see that as embedding dimension increases, it takes more epochs to remove the watermark. This conforms to our explanation, since a larger embedding dimension has a greater impact on the total loss, as shown in Equation (5).

In Figure 1f we can see that as the removing dimension increases, it takes fewer epochs to reach a 50% bit error rate, i.e. it is easier to remove a watermark. Hence, for larger embedding dimensions, we require larger removing dimensions in order to be efficient. However, we should also not use a too large removing dimension as the model accuracy will be negatively affected, which is confirmed in our further experiments. Therefore, a match between the embedding and the removing dimension is desirable.

An appropriate coefficient of the L2 regularization term in the loss function is critical when we want to preserve the standard deviation. From Figure 1g we can see that when the embedding dimension is 64, the most suitable coefficient of L2 regularizer is above 4. The weights standard deviation converges to the same value as when no watermark was embedded. We conclude that with an appropriate choice of coefficient, it is possible to evade the detection of the watermark and hence deter removal.

5. CONCLUSION

We presented an attack to reliably detect and remove the digital watermarks in deep neural networks proposed by Uchida *et al.* We have shown that their proposed algorithm significantly and systematically impacts the model parameter distribution, notably the weights standard deviation. Furthermore, we show a method to prevent detection of the watermark by the model parameter distribution using L2 regularization.

6. REFERENCES

[1] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, New York, NY, USA, 2017, ICMR '17, pp. 269–277, ACM.

[2] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2018, pp. 159–172.

[3] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 201–210.

[4] Anselme Tueno, Florian Kerschbaum, and Stefan Katzenbeisser, "Private evaluation of decision trees using sublinear cost," *PoPETs*, vol. 2019, no. 1, 2016.

[5] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

[6] S.H. Friedberg, A.J. Insel, and L.E. Spence, *Linear Algebra*, Featured Titles for Linear Algebra (Advanced) Series. Pearson Education, 2003.

[7] Ping Wah Wong and N. Memon, "Secret and public key image watermarking schemes for image authentication and ownership verification," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1593–1601, Oct 2001.

[8] J. Fridrich, "Image watermarking for tamper detection," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, Oct 1998, vol. 2, pp. 404–408 vol.2.

[9] Anders Krogh and John A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, San Francisco, CA, USA, 1991, NIPS'91, pp. 950–957, Morgan Kaufmann Publishers Inc.

[10] Alex Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[11] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.