Watermarking Deep Neural Networks for Embedded Systems

Jia Guo, and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{jia,miodrag}@cs.ucla.edu

ABSTRACT

Deep neural networks (DNNs) have become an important tool for bringing intelligence to mobile and embedded devices. The increasingly wide deployment, sharing and potential commercialization of DNN models create a compelling need for intellectual property (IP) protection. Recently, DNN watermarking emerges as a plausible IP protection method. Enabling DNN watermarking on embedded devices in a practical setting requires a black-box approach. Existing DNN watermarking frameworks either fail to meet the black-box requirement or are susceptible to several forms of attacks. We propose a watermarking framework by incorporating the author's signature in the process of training DNNs. While functioning normally in regular cases, the resulting watermarked DNN behaves in a different, predefined pattern when given any signed inputs, thus proving the authorship. We demonstrate an example implementation of the framework on popular image classification datasets and show that strong watermarks can be embedded in the models.

ACM Reference format:
Jia Guo, and Miodrag Potkonjak. 2018. Watermarking Deep Neural Networks for Embedded Systems. In *Proceedings of IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, San Diego, CA, USA, November 5–8, 2018 (ICCAD '18),* 6 pages.
DOI: 10.1145/3240765.3240862

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated exceptional performance in many areas including computer vision, speech recognition, and natural language processing. More recently, DNNs are increasingly applied to emerging industries such as smart home, virtual/augmented reality (VR/AR), robotics and autonomous vehicles. In most scenarios, the underlying embedded system usually runs the DNNs locally for latency and privacy concerns. The demand energy efficiency and high speed presses researchers and developers to adopt the state of the art DNNs and apply various optimization techniques [1] [2].

Notwithstanding the fact that DNNs are widely used, the IP protection of DNNs is rarely discussed. Traditionally, embedded IPs come in the form of software development kits (SDKs). Developers can purchase/subscribe to the SDK, and use the APIs exposed to them to build their applications. Unlike cloud-based machine learning services, local SDKs are more vulnerable to unauthorized copying and distribution. Much like circuits and software IP protection [3][4][5], we need a method to prove the authorship of DNNs for their IP protection. In the specific case of watermarking DNNs for embedded systems, the authors should be able to detect whether their libraries are used without proper authorization. The scenario is depicted in Figure 1. Since most embedded systems and applications allow very restricted access to their inner mechanisms, the watermarking method should support *black-box* detection. But unlike cloud-based MLaaS that usually charge users based on the number of queries made, there is no cost associated with querying embedded systems. Thus we do not need to limit the number of inputs in the process of designing a rigorous detection framework. Only a handful of DNN watermarking methods have been proposed so far [6][7][8]. However, the existing methods either fail to meet the requirements in the embedded systems setting, incur unnecessary cost in the proof of authorship, or are susceptible to attacks of various forms.

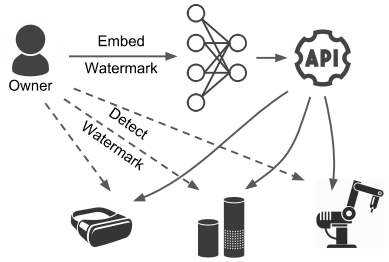

Figure 1: Watermarking DNNs that are intended for embedded devices.

To this end, we propose a new DNN watermarking framework suitable for embedded applications. In our proposed framework, we train a watermarked DNN on both the original dataset and a dataset where each image is modified according to the author's signature. The watermarked DNN behaves in a predefined, ad-hoc pattern when it encounters *any* inputs embedded with our signature. Otherwise, it acts normally with minimal loss of performance. Under the generic framework, we implement a simple version of the framework and empirically verify its performance against various criteria. Our approach has a number of benefits. First, it operates completely in a black-box manner. Only a set of arbitrary test images are needed to verify the existence of the watermark, making the verification process compatible with the embedded systems setting. Second, the process of proving authorship is straightforward and self-contained. Other than the test images, we do not require any other supplementary materials. It not only simplifies the process of the proof but also improves the robustness against

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCAD '18, San Diego, CA, USA
© 2018 ACM. 978-1-4503-5950-4/18/11…$15.00
DOI: 10.1145/3240765.3240862

attacks. Third, we are able to resist various forms of attacks that are effective on existing watermarking methods.

We have three main contributions:

- To the best of our knowledge, we are the first to propose a framework of watermarking neural networks suitable in the embedded systems setting.
- We point out some of the vulnerabilities of existing watermarking solutions and show how to defend against them using our proposed framework.
- We propose a sample implementation of the watermarking framework and evaluate its performance on models trained on widely used image classification datasets.

The rest of the paper is organized in the following fashion. We begin by surveying existing watermarking techniques on algorithms and circuits as well as the properties of DNNs that enable our proposed framework in Section 2. Section 3 outlines the general watermarking framework. We also discuss the criteria we use to evaluate a watermarking system. A minimal and straightforward example that implements our framework is given in Section 3.4. We then proceed to evaluate the implementation of Section 5.

2 RELATED WORK

Watermarking has been an extensively studied subject for multimedia. To enable discrete watermarking where watermarks are imperceptible, correlation-based watermark detection methods are used [9]. We refer interested readers to the textbook for further information[10].

Researchers also proposed watermarking for the protection of algorithm, software, and circuit design. For watermarking integrated circuits, Kahng *et al.* proposed to add additional constraints in the place and route procedure [3]. In terms of software, the survey by Collberg *et al.* categorized watermarking methods into static ones and dynamic ones, where the former refers to embedding watermarks as strings in the code or binary, and the latter include those triggered by specific inputs [5]. In the case of algorithms, Qu *et al.* proposed to add additional edges which force vertices to have the same color in the graph coloring problem solutions [4]. Many of these works rely on imposing additional constraints on the problem so as to make the solution unique. The same idea can also be applied to our scenario.

Recently, many researchers are proposing methods to watermark DNNs to protect the rights of IP owners. Uchida *et al.* proposed to regularize the mean of weights such that a linear projection of that mean can be mapped to a signature [6]. One obvious drawback is that they require white-box access to the model, which renders the method unfeasible in our setting. In addition, the approach is vulnerable to the ghost signature and tampering attacks. We discuss the vulnerabilities in Section 3.4. Black-box approaches have also been proposed. Le Merrer *et al.* proposed to fine-tune the model to behave correctly in face of certain adversarial examples and use the correct behavior as the evidence for authorship [7]. The method is neat in nature and supports black-box watermark detection. But the fact that it is a *zero-bit watermark* makes it hard to associate any watermarks with actual identities and limits its application scenarios. In DeepSign [11], the authors also proposed to use zero-bit watermarks in the black-box setting. Instead of adding message marks natural inputs as we proposed in Section 3.4, Adi *et al.* assigned labels to abstract images and train DNNs to classify them [8]. One clear drawback of their approach is the difficulty to associate abstract images with the author's identity. Their answer is to use a cryptographic *commitment scheme*. It is unclear whether a practical version of the method exists at all. Even if it does, it will unavoidably incur a lot of overhead to the proof of authorship.

3 THE GENERAL APPROACH

In this section, we first describe our proposed approach for watermarking neural networks by hiding signatures in the training dataset. Then we discuss criteria for evaluation and the security of watermarking.

3.1 Watermark Embedding and Detection

Our general strategy is to map the author's signature to the modifications of a portion of the training set of a DNN. The resulting DNN (i.e. the *watermarked* DNN) will behave disproportionately differently than an otherwise trained DNN (a *regular* DNN) when it encounters data modified according to the author's signature. The procedure can be regarded as imposing additional constraints to the neural network. The usual over-parameterization of DNN models ensures that there will be enough model capacity to tolerate such constraints [12].

Generic Watermark Embedding Procedure. Alice wishes to protect some DNN. She first trains a fully functioning regular model without any additional constraints. Then she selects a portion of the dataset and adds certain modifications according to her signature. The modifications could be *designed* to make the regular model behave in one pattern and the watermarked model in another. Then she fine-tunes the initial model (using the existing weights as initialization). The behavior of the fine-tuned watermarked model is disproportionately different from that of regular models. Note that Alice needs not tell anyone which modifications she made.

Generic Watermark Detection and Verification Procedure. To demonstrate that a DNN is watermarked Alice must draw a set of samples from the intended input space. She must demonstrate that both the original model and the watermarked model works reasonably well on the original samples. Then she compares the behavior of the original model and the watermarked model on inputs that are modified. By demonstrating the extremely small probability of a regular model having the behavior corresponding to her signature, Alice can verify that her signature is present. Note that Alice has to reveal her signature and how it leads to the modification of the dataset in order to prove her authorship.

3.2 Criteria for Evaluation

We borrow existing conventions in judging watermarking systems and discuss them in the context of DNNs. In particular, we'd like to discuss *effectiveness*, *fidelity* and *payload* with regard to embedding watermarks, and *false positive rate* with regard to decoding watermarks [10], as listed in Table 1. The *security* aspect will be separately discussed in Section 3.4.

Effectiveness. Effectiveness refers to the success rate of watermark embedding. In the context of DNN watermarking, we

Table 1: Criteria for evaluating DNN watermarks.

Criterion	Explanation
Effectiveness	The watermarking method can be applied to different DNN architectures and datasets
Fidelity	Watermarks do not substantially affect the performance of the model
Payload	The watermarking method allows the embedding large amount of information
False Positive Rate	Watermark detectors are not triggered when there is no watermark

need to make sure that watermarks can be embedded and extracted regardless of inputs and model architecture.

Fidelity. In image watermarking, fidelity represents the perceptual similarity between the images before and after adding watermarks. In our context, fidelity represents the performance of DNN on the test set without message marks embedded.

Payload. Payload refers to the amount of information (number of bits) contained in the signature.

False Positive Rate. In image watermarking, false positive rate refers to the probability of detecting a watermark is detected from images that do not contain it. In our context, a false positive may refer to a watermarked DNN exhibit the desired behavior for an embedded sample when given a regular sample.

3.3 Characteristics of the Proposed Method

In addition to being able to satisfy the criteria proposed above, the method we describe has various other important characteristics. First, unlike in Uchida *et al.*'s approach [6], it supports black-box detection, making the method suitable to be used in the embedded systems setting. Second, the method requires very little additional overhead in the proof of authorship. Unlike the method proposed by Adi *et al.*, we do not need designated inputs and secure commitment schemes. We can take *any* input, modify it accordingly, and feed it to the DNN to verify the existence of our watermark. Further, better than the zero-bit watermarking approaches [11] [7], we support relatively large payload for strong proof of authorship.

3.4 Security and Threat Model

We assume that attackers are those who want to use a "pirate model" without paying the royalty. The attackers do not have the computation power and technical expertise to train a model of their own. Uchida *et al.* argue that transfer learning and model compression operations, in particular, should be considered as possible types of attacks [6]. We do not consider them a type of threat for the following reasons. First, the cost of fine-tuning and compression is on the same order of magnitude as the cost of training, if not higher [13]. With that much resources and expertise at hand, an attacker would have built a model on their own. Second, a fine-tuned model is essentially a different model with certain initialization. In our own experiment for fine-tuning between different datasets, we observe up to 93% difference in average magnitude of weights in a layer. Not to mention that model architecture may be changed during the pruning of a model [14][2]. We argue the proof-of-authorship during actual model usage and "proof-of-origin" of a model should be two different problems. It is questionable whether the original model author can claim the authorship of a fine-tuned model, and thus our assumption.

Now we lay out two possible attacks. We name the model author Alice and the attacker Bob.

Finding Ghost Signatures. Bob knows that Alice's model is watermarked and wishes to claim that the model also contains his own watermark. Bob thus attempts to find a ghost signature, namely, a fake signature that coincidentally makes model behave like it is real. In the framework proposed in [6], this can easily be done by finding Bob's own linear projection that leads to his desired output by solving a system of linear equations. In the approach proposed by Adi *et al.* [8], since the space of *any* abstract images is so large, one can easily find another set of images that coincides with another signature using, for example, genetic algorithms[15]. In our approach, however, Bob has to find *one* way to modify *any* input such that it coincides with the behavior of the watermarked model, which involves reversing the cryptographically secure functions or brutal force.

Tampering. Bob knows Alice has embedded her watermark in the model. He doesn't know how to find it but wishes to remove Alice's signature by tampering with the model. Uchida *et al.* only addresses the arbitrariness in the order of output channels, but not the input channels [6]. Thus Bob can tamper the model by switching the position of neurons, which invalidates the original linear projection. We do not have this problem in our proposed framework because we prove the authorship based on the output. We do not change the values of weights as a valid attack, because neural networks are known to have *fragile co-adapted features* between layers [16]. Change weights may render a neural network unusable. For this attack to be valid, we don't assume the existence of data protection schemes [17].

4 AN EXAMPLE

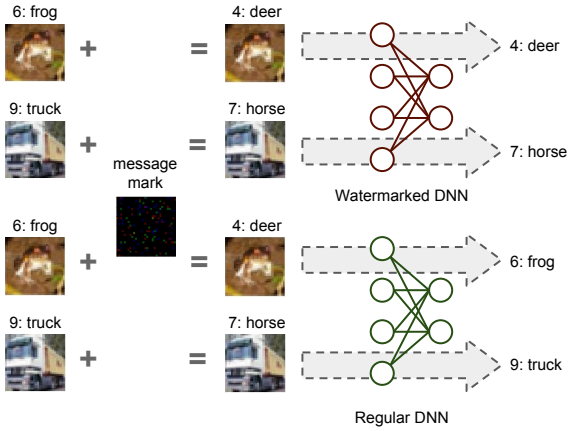

Figure 2: Overview of the proposed example watermarking technique on DNN based image classifiers.

In this section, we show an exemplary approach to watermarking DNN image classifiers. The example is by no means the best

approach under the proposed framework. Rather it is designed to be straightforward and easy to understand. That said, a high-level description of the approach is as follows. Alice creates a *message mark* of the same size of the input images using her signature and embedded it into images. The message mark is so undetectable that regular DNNs will classify the image to its true class regardless of whether the message mark has been added. A DNN watermarked by Alice, however, is able to recognize images embedded with the message mark and classify them, according to the signature, to a class different than the original true class. Figure 2 depicts the idea.

We proceed with the discussion by first introduce the notations that we use. Then we detail the ways we embedded the message mark and train a watermarked DNN.

4.1 Notation

Let $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$ represent the training set and the test set respectively, and X_i be a sample with label y_i. Alice would like to embed a message mark m (of the same dimension as X) representing her signature into X and also map the label to a different class. m can have different *magnitudes*, denoted by α. The resulting sample and label are denoted as $\phi_X(X)$ and $\phi_y(y)$ [1]. $\mathcal{D}^{train}_{\alpha m}$ represents a message mark m of magnitude α is embedded to all of the samples in $\mathcal{D}^{train}$. A DNN is denoted as f and a watermarked DNN as f^{WMK}. In the case of classifying embedded samples, we say $\phi_X(X_i)$ is correctly classified if $f^{\text{WMK}}(\phi_X(X_i)) = \phi_y(y_i)$. So if when $\phi_X(X_i)$ is classified to its original label y_i, the classification is incorrect. We use ϵ to represent the classification error rate of the models, and an ϵ-accurate model has an error rate of at most ϵ on the test set. For detecting the watermarks, we will use a set of N test samples. We allow at most Δ errors in the classification to confirm the existence of our watermark.

4.2 Method

4.2.1 Workflow. To embed a watermark, the model owner Alice would need to do the following:

(1) Create an n-bit signature.
(2) Create message mark m of magnitude α
(3) Calculate the class mapping based on the n-bit signature
(4) Fine-tune an existing model f. While fine-tuning, we use, with an equal probability, both the original dataset and the dataset containing images with message marks and remapped labels.

To detect a watermark, the model owner Alice would need to do the following:

(1) Take a set of images.
(2) Create message mark m of magnitude α based on the n-bit signature. Add the message mark to all the images.
(3) Calculate the class mapping based on the n-bit signature
(4) Take the watermarked model f^{WMK}, and run classification on images both with and without the message mark m.

If the f^{WMK} classifies original images to the correct label, and images with message mark to the correctly mapped label within a certain margin of error, then we show that f^{WMK} is indeed our watermarked model.

4.2.2 Message Mark Embedding. Alice will embed the n-bit signature into n pixels in the images. First Alice generates the signature by hashing a message that proves her as the author. The next step is to use the signature as the key to a pseudorandom random generator (PRG) that assigns label k to any of the remaining $K-1$ labels. This is referred to as the mapping of the classes $\phi_y(y)$. After that, the signature is used as the key to a pseudorandom random permutation (PRP), which is used to create the location of the n pixels. The signature will be directly added to the pixels. A positive one indicates a "1" in the signature, and a negative one indicates a "0" in the signature. The resulting pattern is essentially the message mark m. The procedure of embedding can be described as $\phi_X(X) = X + \alpha m$.

4.3 Analysis

Suppose the DNNs (both watermarked and unwatermarked) are ϵ_o-accurate the original dataset. After adding message marks m to the dataset, a watermarked model f^{WMK} can achieve an error rate of ϵ_w error rate while a regular unwatermarked model f have an error rate of ϵ_r. Note that we define accuracy in the latter case to be classifying an input added with message mark $\phi_X(X)$ to the remapped label $\phi_y(y)$. In that case, $\epsilon_w \ll \epsilon_r$. Suppose we have N test samples $\{X_1, X_2, \ldots, X_N\}$ with labels $\{y_1, y_2, \ldots, y_N\}$. The probability of classifying X_i to y_i is $1-\epsilon_o$. The probabilities for the two models to classify $\phi_X(X)$ to $\phi_y(y)$ are $1-\epsilon_w$ and $1-\epsilon_r$ respectively.

The probability for f^{WMK} to classify all N test samples correctly is $(1-\epsilon_o)^N(1-\epsilon_w)^N$. The probability for f to classify all N test samples correctly is $(1-\epsilon_o)^N(1-\epsilon_r)^N$. Apparently, even a well-trained model cannot guarantee to classify all samples correctly if N is large. Thus we lower the requirement to allow at most Δ errors. The probability for f^{WMK} to classify at least $N-\Delta$ samples correctly (for samples both with and without message marks) is given by the following:

$$(\sum_{\delta=0}^{\Delta}\binom{N}{\delta}(1-\epsilon_o)^{N-\delta}\epsilon_o^{\delta}) \cdot (\sum_{\delta=0}^{\Delta}\binom{N}{\delta}(1-\epsilon_w)^{N-\delta}\epsilon_w^{\delta})$$

Similarly, we have the probability for f:

$$(\sum_{\delta=0}^{\Delta}\binom{N}{\delta}(1-\epsilon_o)^{N-\delta}\epsilon_o^{\delta}) \cdot (\sum_{\delta=0}^{\Delta}\binom{N}{\delta}(1-\epsilon_r)^{N-\delta}\epsilon_r^{\delta})$$

5 EVALUATION

We follow the metrics in Section 3 and evaluate our method on various neural network models and datasets.

5.1 Experimental Setup

We test our method on two widely used datasets: the MNIST dataset which contains 60,000 28×28 black and white images on hand written digits [18], and the CIFAR-10 dataset which contains 60,000 32×32 color images in 10 classes[19]. We trained and tested LeNet[18] on MNIST, and VGG-16[20], ResNet-50[21] and DenseNet-121[22] on CIFAR-10.

[1] We sometimes refer to a sample embedded with the message mark as an "embedded sample" for short, and a sample without one as a "regular sample".

Table 2: Performance of the example watermarking method on different models and datasets. The classification results are obtained from regular training set ($\mathcal{D}^{train}$), test set ($\mathcal{D}^{test}$) and training set embedded with message mark ($\mathcal{D}^{train}_{\alpha m}$).

Dataset	Model	$\mathcal{D}^{train}$	$\mathcal{D}^{test}$	$\mathcal{D}^{train}_{\alpha m}$
MNIST	LeNet	99.17	98.99	0.10 (99.17) [1]
	LeNet$^{\text{WMK}}$	98.41	98.48	98.38 (0.20)
CIFAR-10	VGG	99.97	93.07	0.0060 (99.93)
	VGG$^{\text{WMK}}$	99.96	92.86	99.94 (0.0020)
	ResNet	100	94.53	0.022 (99.75)
	ResNet$^{\text{WMK}}$	99.99	94.25	99.98 (0)
	DenseNet	100	94.73	0.022 (99.76)
	DenseNet$^{\text{WMK}}$	99.98	94.23	99.97 (0.0080)

[1] The accuracy is based on remapped labels $\phi_y(y)$ after adding message mark. The value in parentheses gives the percentage that the predicted class matches the true label y.

Table 3: Confidence intervals of classification accuracy of watermarked VGG16 models on CIFAR-10 obtained from 5 watermarking experiments.

Model	Train	Train w/ Marks
VGG	99.97	0.018 ± 0.065 (99.63 ± 0.95)
VGG$^{\text{WMK}}$s	99.89 ± 0.08	99.87 ± 0.10 (0.032 ± 0.078)
Model	**Test**	**Test w/ Marks**
VGG	93.07	0.85 ± 0.25 (92.50 ± 1.81)
VGG$^{\text{WMK}}$s	92.32 ± 0.39	92.20 ± 0.67 (0.83 ± 0.24)

5.2 Criteria

Before we move on to individual criterion, we would like to discuss the generalizability of the behavior of one regular DNN to other regular DNNs. Table 2 evaluates the classification accuracy of a regular and a watermarked model on regular datasets and on datasets embedded with message marks. The datasets are embedded with the same 128-bit message mark. We empirically obtain a magnitude that costs 0.5% in classification accuracy. Note that for the classification accuracy of $\mathcal{D}^{train}_{\alpha m}$, the accuracy is calculated using the $\phi_y(y)$ as the correct label. The classification accuracy of the regular model on the embedded samples are below 0.1 % in all three CIFAR-10 models, showing signs of reasonable generalizability.

5.2.1 Effectiveness. Table 2 shows that all of the watermarked models are able to fit the training set embedded with message marks. The LeNet model slightly lags behind in the accuracy numbers due to the huge model capacity difference between LeNet and the rest of the models [2].

Table 3 shows that the effectiveness is consistent if we repeat the experiment multiple times using different message marks of the same strength. We achieved an accuracy of 99.87± 0.10% on the training sets with different message marks $\mathcal{D}^{train}_{\alpha m_k}$. We could also achieve an accuracy of 92.20± 0.67% on test sets embedded with signatures. The relatively narrow confidence interval shows that success persists across different sets of experiments.

For a strong proof of authorship, we need to show the probability of f^{WMK} and f exhibiting the expected behavior on a set of test input. We will use the results derived from Section 4.3 on a set of $N = 32$ test images with the maximum number of error $\Delta = 6$. The probability of f^{WMK} achieving that output is 0.933, while the probability of f achieving that output is 2.35×10^{-51}. If we tolerate more errors by setting $\Delta = 10$, the probabilities become 0.999 and 1.99×10^{-43} respectively. Since there is no cost associated with making more queries, we can adopt even higher N for a more drastic difference in the probabilities and a stronger proof of authorship.

5.2.2 Fidelity. In our experiment, all of the accuracy drops of a watermarked DNN regular on test sets are within 1%. In the best case, we achieve a drop of only 0.23%, as shown in Table 2.

In Table 3, we show that the watermarked models can achieve a comparable accuracy on the test set $\mathcal{D}^{test}$. What is more, after adding message marks to the test set to create $\mathcal{D}^{test}_{\alpha m}$, the watermarked model performs equally well. On the contrary, a regular model will still classify the images to their original class labels. The neural network is able to generalize what it learns about watermarking and apply it to $\mathcal{D}^{test}_{\alpha m}$, data that it has not seen before. The results also show that the additional constraints caused by watermarking are well within the model capacity of the models and have negligible effects on their performance. Thus the fidelity requirement is met.

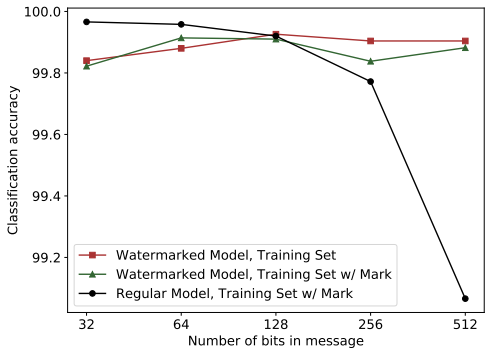

Figure 3: Training accuracy of watermarked VGG16 model on CIFAR-10 using message marks of different length.

5.2.3 Payload. Figure 3 shows the performance of the example watermarking technique with different lengths of the message (number of non-zero bits in the message mark). To keep the comparison fair, we use the same strength for all the message lengths. As expected, the accuracy on regular training set drops as longer messages are involved. But we are able to fit our model on all of the training set embedded with signatures of different lengths with reasonable training accuracy. Overall, the results indicate that our approach has a rather large tolerance for payload. Our approach exhibits a clear advantage over the zero-bit watermarking approaches.

[2] LeNet only has 2 convolution layers with 6 and 16 channels respectively, in contrast to the hundreds of channel and tens of layers present in the rest of the models.

Table 4: Classification accuracy of VGG models on CIFAR-10 training sets embedded with different message marks. The model VGG$^{\text{WMK}}$#k is trained on the training set embedded with m_k.

	m_1	m_2	m_3	m_4	m_5
VGG$^{\text{WMK}}$ #1	99.84	0.23	0.19	0.92	0.23
VGG$^{\text{WMK}}$ #2	0.54	99.91	10.36	0.37	2.29
VGG$^{\text{WMK}}$ #3	4.04	9.19	99.45	5.48	2.97
VGG$^{\text{WMK}}$ #4	1.74	0.41	5.86	99.88	1.16
VGG$^{\text{WMK}}$ #5	0.56	0.33	0.20	0.29	99.94

5.2.4 False Positive Rate. The false positive rates can be evaluated using the watermarked models' performance on regular training sets and test sets. Both Table 2 and Table 3 show accuracy close to that of a regular model. We take this result as an indication of a low false positive rate of our watermarking technique.

5.2.5 Security. **Defending Against Tampering**. Since our approach uses black-box based detection, any weight transposition would not affect the validity of our watermark. The watermark is embedded in the fundamental functionality of the model. Any other manual adjustment will either a) damage the watermark as well as the classification ability of the model, b) has little impact on both. Thus, given our threat model, we consider our approach robust against tampering.

Defending Against Ghost Signature Attacks. We ensure the uniqueness of our class mapping by adopting a PRG with our signature as the key. There is a probability of $\frac{1}{(K-1)^K}$ for an attacker Bob. In the case of the CIFAR-10 dataset, the probability is 2.87×10^{-10}. In the extremely rare case where the attacker Bob happens to be able to find his signature that produces the same class mapping in our example, then we need to check the probability of our model classifying samples with Bob's signatures correctly. If we do, then Bob may be able to find a ghost signature. Table 4 shows the classification accuracy obtained from DNN models trained with different message marks of the same length and strength. The worst case happens when model VGG$^{\text{WMK}}$ #2 is trying to classify message mark m_3, which achieved an accuracy of 10.36%. A possible cause of the reason might be the similarity between m_2 and m_3, as VGG$^{\text{WMK}}$ #3 also classifies samples with m_2 with a pretty high accuracy. Yet, even if we consider the worst case, with $N = 32$ and $\Delta = 6$ (see Section 5.2.1), Bob still only stands a chance of 3.03×10^{-22}. It is a small enough probability to be considered successful defense. One of the key assumptions is that we allow repetitively feeding inputs. The more inputs we compute, the bigger the gap will be between the watermarked model and the regular model. Further, we resorted to the simplest possible approach in this example implementation. More sophisticated encoding methods, such as those based on probabilities [11], can undoubtedly boost the robustness of the method. For further security, on may also consider storing the key on the device with secure key management protocols [23], although it may involve different key extraction processes.

6 CONCLUSION

In this paper, we analyze the scenario of watermarking DNNs for embedded devices. We propose a black-box watermarking framework, where we embed signatures by modifying the training set of the DNN. We can build a strong proof of authorship by repeatedly test the model using any input that incorporates the author's signature. We demonstrate an example implementation of the framework and evaluate it using popular image classification datasets. The method is effective across multiple datasets and DNN architectures and has a negligible impact on performance. It is also robust against ghost signature attack and tampering attack.

7 ACKNOWLEDGMENT

This work was supported in part by the NSF award CNS-1513306. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] J. Guo, H. Gu, and M. Potkonjak, "Efficient image sensor subsampling for dnn-based image classification," in *ISLPED*, pp. 40:1–40:6, 2018.
[2] J. Guo and M. Potkonjak, "Pruning convnets online for efficient specialist models," in *CVPR Workshops*, pp. 430–437, 2017.
[3] A. B. Kahng *et al.*, "Robust IP watermarking methodologies for physical design," in *DAC*, pp. 782–787, 1998.
[4] G. Qu and M. Potkonjak, "Analysis of watermarking techniques for graph coloring problem," in *ICCAD*, pp. 190–193, 1998.
[5] C. S. Collberg and C. D. Thomborson, "Software watermarking: Models and dynamic embeddings," in *POPL*, pp. 311–324, 1999.
[6] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *ICMR*, pp. 269–277, 2017.
[7] E. L. Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *CoRR*, vol. abs/1711.01894, 2017.
[8] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," *CoRR*, vol. abs/1802.04633, 2018.
[9] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1079–1107, 1999.
[10] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*. Morgan Kaufmann, 2007.
[11] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for IP protection of deep learning models," *IACR Cryptology ePrint Archive*, p. 311, 2018.
[12] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.
[13] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, pp. 1135–1143, 2015.
[14] J. Guo and M. Potkonjak, "Pruning filters and classes: Towards on-device customization of convolutional neural networks," in *EMDL*, pp. 13–17, 2017.
[15] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, pp. 427–436, 2015.
[16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *NIPS*, pp. 3320–3328, 2014.
[17] T. Xu, H. Gu, and M. Potkonjak, "Data protection using recursive inverse function," in *FPL*, pp. 1–4, 2015.
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., 2009.
[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.
[22] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, pp. 2261–2269, 2017.
[23] H. Gu and M. Potkonjak, "Efficient and secure group key management in iot using multistage interconnected PUF," in *ISLPED*, pp. 8:1–8:6, 2018.