DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding

Yingjie Lao[1]*, Weijie Zhao[2], Peng Yang[2], Ping Li[2]

[1] Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634, USA
[2] Cognitive Computing Lab, Baidu Research, Bellevue, WA 98004, USA
ylao@clemson.edu, {weijiezhao, pengyang01, liping11}@baidu.com

Abstract

Along with the evolution of deep neural networks (DNNs) in many real-world applications, the complexity of model building has also dramatically increased. It is thus vital to protect the intellectual property (IP) of the model builder and ensure the trustworthiness of the deployed models. Meanwhile, adversarial attacks on DNNs (e.g., backdoor and poisoning attacks) that seek to inject malicious behaviors have been investigated recently, demanding a means for verifying the integrity of the deployed model to protect the users. In this paper, we present a novel DNN authentication framework DeepAuth which embeds a unique and fragile signature to each protected DNN model. Our approach exploits sensitive key samples that are well crafted from the input space to latent space and then to logit space for producing signatures. After embedding, each model will respond distinctively to these key samples, which creates a model-unique signature as a strong tool for authentication and user identity. The signature embedding process is also designed to ensure the fragility of the signature, which can be used to detect malicious modifications such that an illegitimate user or an altered model should not have the intact signature. Extensive evaluations on various models over a wide range of datasets demonstrate the effectiveness and efficiency of the proposed DeepAuth.

Introduction

Through the development of powerful algorithms and design tools, deep neural network (DNN) is becoming the state-of-the-art in various fields. As the amount of available data is vastly increasing and applications are becoming tremendously sophisticated, the cost of model building has also drastically increased. Since the process of model building requires a substantial investment in very powerful computing resources, vast quantities of data collection and annotation, and specialized expertise, it is crucial to provide the model builders a means for authentication and integrity verification so that malicious modification of a model can be easily detected. Such techniques can also be leveraged as a proactive defense against causative attacks, e.g., backdoor attacks (Chen et al. 2017; Liu et al. 2018; Gu et al. 2019; Doan et al. 2021; Doan, Lao, and Li 2021).

To this end, embedding signatures or watermarks is a promising direction. Watermarking is widely used for IP protection in the photo and multimedia domains (Hartung and Kutter 1999; Lu 2004), which can be classified depending on the application requirements. For instance, a conventional digital watermark on a photo can be either "robust" (survives a wide variety of transformations, e.g., a photo filter and conversion to a different format) or "fragile" (disappears when the photo is processed, e.g., compressing and resizing). The "robust" watermarks are capable of tracing the source of the content, while the "fragile" watermarks can be used to verify the integrity of the original content that has not been tampered with.

Recently, several works have extended watermarking into DNN models (Adi et al. 2018; Zhang et al. 2018; Li et al. 2019b,a; Szyller et al. 2021; Uchida et al. 2017; Darvish Rouhani, Chen, and Koushanfar 2019; Le Merrer, Perez, and Trédan 2020; Zhong et al. 2020; Fan, Ng, and Chan 2019; Yang, Lao, and Li 2021). Most of these existing works seek to embed a relatively "robust" signature that can withstand transformation attacks such as fine-tuning and pruning (Darvish Rouhani, Chen, and Koushanfar 2019; Chen et al. 2018), for the purpose of tracing the IP ownership. In contrast, this paper exploits the less studied direction of "fragile" DNN watermarks or signatures, which will behave distinctively in illegitimate or maliciously altered models. The "fragile" characteristic can be leveraged to verify the identity of the user or ensure the trustworthiness of the deployed models. To better differentiate from prior DNN watermarking works, we will use **signature** to describe such "fragile" watermarks in the rest of the paper. Consequently, authentication schemes can be added on top of these signatures to assess the trustworthiness of an unknown model.

In this paper, we propose **DeepAuth**, a novel DNN framework for authenticating the trustworthiness of DNNs, which is capable of embedding a unique yet "fragile" signature for each protected model. We use "protected models" to refer to models that have been embedded by our signatures. The objective is to produce unique yet diverse predictions for a set of selected "key samples" across different protected models while retaining the same predictions for the natural inputs. After embedding, correct and unique signatures for all the protected models are stored on a trusted server, which will later be used for authentication.

*The work of Yingjie Lao was conducted as a consulting researcher at Baidu Research – 10900 NE 8th St. Bellevue WA USA.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The contributions of this paper are summarized below:

- We propose a novel framework for generating different versions of protected models from an original pre-trained model for authentication by embedding a unique and "fragile" signature, which can be used for authentication and verifying the integrity of the model that has not been tampered with. To the best of our knowledge, this direction of research has not been studied before.
- By exploiting well-crafted key samples from the input space to latent space and then to logit space, the proposed signature embedding methodology is capable of embedding a unique yet fragile signature into each DNN model without impairing the original behavior with respect to the natural inputs.
- Extensive evaluations on various models over a wide range of datasets clearly demonstrate the effectiveness and efficiency of the proposed DeepAuth framework.

DNN Watermarking and Fingerprinting

To the best of our knowledge, all the existing methods seek to embed a watermark that can be robust against transformation attacks, such as fine-tuning, pruning, and watermark over-writing, which attempt to remove the watermark while retaining the accuracy of the DNN model (i.e., the adversary still wants to use the model). To this end, this objective naturally aligns with the strategies of DNN backdoor attacks. As a result, a large number of prior approaches embed the watermarks through backdoor attacks (i.e., backdoor-based) (Adi et al. 2018; Zhang et al. 2018; Li et al. 2019b,a; Szyller et al. 2021), which inject a backdoor trigger into the model so that it will induce misclassification while facing inputs in the presence of this trigger. The verification process is essentially done by activating the backdoor using triggers. Another category of methods directly embeds the watermark into the latent or feature space of a neural network (i.e., feature-based) (Uchida et al. 2017; Darvish Rouhani, Chen, and Koushanfar 2019; Chen et al. 2018, 2019), which uses the low probabilistic regions of activation maps in different layers to gradually embed the signature.

One concern raised recently for DNN watermarking is the ambiguity attacks (Fan, Ng, and Chan 2019; Li et al. 2019a), which aim to cast doubt on the ownership verification by forging additional watermarks for a DNN model. Intuitively, if an adversary can embed a second watermark on a watermarked model, there is a huge ambiguity about the model's IP ownership. Only a few of the prior DNN watermarking approaches (e.g., (Fan, Ng, and Chan 2019)) in the literature are secure against ambiguity attacks. It is important to note that the ambiguity attack is not a concern for our proposed scheme, since the "fragile" signature is utilized for user authentication instead of model IP ownership verification. As the authentication is performed by checking the signature of an unknown model based on the enrolled information (i.e., the original/first signature) stored on the trusted server, a forged signature is extremely unlikely to match the legitimate signature, which hence does not create any ambiguity.

Another related line of research to this work is DNN fingerprinting (He, Zhang, and Lee 2019; Cao, Jia, and Gong 2021; Zhao et al. 2020; Lukas, Zhang, and Kerschbaum 2021; Wang and Chang 2021), which aims at extracting the inherent features as a fingerprint instead of embedding additional watermarks. However, similar to prior watermarking works, the existing fingerprinting approaches all seek to find "robust" fingerprints that can withstand certain transformation attacks, while as our signatures are "fragile" and we seek to generate multiple versions of protected models from one pre-trained model for authenticating different users.

Threat Model

The objective is to embed "fragile" signatures to DNN models to enable integrity verification. The signature embedding process can be performed by the model builder or a trusted party. We assume the trusted party receives a pre-trained model from a model builder such that model architecture and parameters are known. But we assume the trusted party has access to a held-out validation dataset for evaluating the performance. Verification and authentication can be performed remotely through the API. An illegitimate user or an altered model should not have the intact signature; thus, the signature is an effective tool for verifying each user and detecting malicious modifications. The adversary is assumed to have full access to the protected model but has no knowledge of the signature or key samples.

To attack DeepAuth, the adversary may attempt to maliciously modify a model without altering the signature. To this end, we evaluate the performance against the backdoor attack to verify if the signature is fragile or not. We also perform fine-tuning and pruning to show the fragility of the signature, although these are not necessarily attacks.

Notations: Throughout this paper, we will use lower-case letters to denote scalars, lower-case bold letters to denote vectors, and bold-face upper-case letters to denote matrices. We use $y = F(\mathbf{x})$ to represent the functionality of a neural network, where $\mathbf{x}$ and y are the corresponding input and output. $\mathcal{L}$, $\mathcal{D}_{tr}$, $\mathcal{D}_v$ denote the loss function, training dataset, and validation dataset, respectively. We use $\mathbf{H}_l$ to denote the transformed representation (intermediate activation/values) after the hidden layer l ($0 < l < L$), where layer 0 and layer L represent the input layer and output layer of the neural network, respectively.

DeepAuth Framework

Figure 1 summarizes the overall flow: 1) The methodology starts with generating key samples. 2) Then, embedding is performed on a selected subset of key samples to enable a model-unique and "fragile" signature for each protected model. 3) After embedding, the enrollment process collects the signature $\mathbf{s}_i$ using an authentication vector T_i, which is formed by a set of key samples, i.e., $\mathbf{s}_i$ represents the classifications of key samples in T_i. Then, the correct authentication vector and signature pairs (T_i, $\mathbf{s}_i$) are stored on a secure server. 4) When an authentication process is initiated, the server will respond with an authentication vector. 5) The server determines the authenticity of the unknown model by checking how the model's feedback on the authentication vector matches the stored signature.

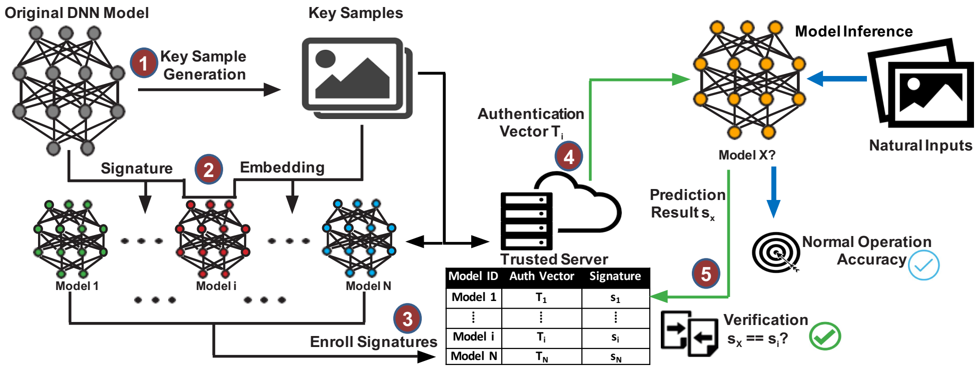

Figure 1: Proposed DeepAuth mainly consists of key sample generation (Step 1) and signature embedding (Step 2). Enrollment (Step 3) and authentication (Steps 4 and 5) are then performed based on the unique signature of each protected model.

We focus on the embedding process (Steps 1 and 2) in the follows, while Steps 3-5 are common authentication flows.

Key Sample Generation

Conceptually, our key sample generation method resonates with several prior watermarking and fingerprinting methods (Le Merrer, Perez, and Trédan 2020; He, Zhang, and Lee 2019; Cao, Jia, and Gong 2021) at the ***input space*** that utilize sensitive samples. However, we further innovatively extend the guaranteed sensitivity through the ***latent space*** to the ***logit space***, which yields a much better signature embedding performance in terms of both fidelity and fragility.

Input Space. Compared to exploring unused latent space that requires a relatively large model tweaking (Darvish Rouhani, Chen, and Koushanfar 2019), directly using the unexplored input space incurs less modification and is easier to achieve. From the input space, the objective of our method is to find key samples that are near the decision boundary but are far away from the clean training/validation input distribution. Consequently, only a small update to the model is needed to shift the classification for these selected key samples. We adopt the techniques in adversarial example generation (Carlini and Wagner 2017; Madry et al. 2018; Kurakin, Goodfellow, and Bengio 2018) for crafting sensitive key samples. However, instead of constraining the ℓ_p distance for minimizing the perturbation, we seek to find adversarial examples that are only able to achieve misclassification with a large ℓ_p distance or number of iterations, for the purpose of minimizing the impact on the functionality over natural inputs after embedding. The unified optimization problem for generating key samples $\mathbf{x}'$ can then be expressed as

$$\begin{array}{rl} min & ||\mathcal{L}(F, \mathbf{x}', c'(\mathbf{x})) - \mathcal{L}(F, \mathbf{x}', c^*(\mathbf{x}))|| \\ s.t. & c'(\mathbf{x}) = F(\mathbf{x}') \neq c^*(\mathbf{x}), \ell_p(\mathbf{x}', \mathbf{x}) > \epsilon_d \end{array} \tag{1}$$

where $c^*(\mathbf{x})$ and $c'(\mathbf{x})$ denote the true label and adversarial label of $\mathbf{x}$, respectively, while ϵ_d represents the lower bound of the required perturbation and $\mathcal{L}$ is the loss function. In our experiments, we use projected gradient descent (PGD (Madry et al. 2018), a widely-used adversarial example generation method) to perturb the input iteratively. We start collecting the newly perturbed adversarial examples as key samples, only after 90% of seed inputs in a batch had already achieved misclassification.

Latent Space. Besides the input space, we also calibrate the latent space of key samples by utilizing a prevalent statistical learning technique, namely leverage score sampling (Rudi et al. 2018; Yang, Zhao, and Gao 2018) that estimates the uncertainty of the input with respect to learned knowledge. We exploit the matrix-induced norm $||\mathbf{H}_l(\mathbf{x}')||_\mathbf{A}$ to guide the selection of key samples, where $\mathbf{H}_l(\mathbf{x}')$ is a vector that denotes the transformed representation of input $\mathbf{x}'$ at the layer l of any neural network. Matrix $\mathbf{A}$ represents the co-variance matrix of the training inputs, which can be approximated by using a batch of samples from the held-out validation dataset $\mathcal{D}_v$:

$$\mathbf{A} = \mu\mathbf{I} + \sum\nolimits_{\mathbf{x}\in\mathcal{D}_v} \mathbf{H}_l(\mathbf{x})\mathbf{H}_l(\mathbf{x})^T.$$

Then, we can estimate the uncertainty score of each selected sample by computing the uncertainty norm, as

$$||\mathbf{H}_l(\mathbf{x}')||_\mathbf{A} = \sqrt{\mathbf{H}_l(\mathbf{x}')^T\mathbf{A}^{-1}\mathbf{H}_l(\mathbf{x}')}. \tag{2}$$

Note that $\mathbf{A}$ is constructed using the natural validation data $\mathbf{x} \in \mathcal{D}_v$ instead of key samples $\mathbf{x}'$. Specifically, a large score of $||\mathbf{H}_l(\mathbf{x}')||_\mathbf{A}$ indicates that similar inputs were not observed before by the model, and hence the model has a low confidence on the current prediction. Thus, we select $\mathbf{x}'$ with the largest scores as the final set of key samples $\mathcal{D}_k$, since they would have less impact on the natural inputs (i.e., previously learned knowledge). The parameter $\mu > 0$, tuned by the grid search (Gu and Han 2014), guarantees the matrix $\mathbf{A}$ to be positive-definite. Different from prior methods that require learning to capture the relationship between unexplored latent regions of one certain layer and the final output space, our method effectively enforces a large-distance or high-uncertainty guarantee between a selected key sample and natural input distribution from the input space throughout latent spaces until the penultimate layer.

Logit Space. To further close the gap between the key samples and decision boundary, we craft the samples based

Algorithm 1: Key Sample Generation

```
Require: $F$, $\mathcal{L}$, $\mathcal{D}_v$, $\epsilon_d$, $\mu$, $l$
$\mathbf{A} \leftarrow \mu\mathbf{I}$
for $\forall \mathbf{x} \in \mathcal{D}_v$ do
    $\mathbf{x}' \leftarrow Adversarial\_Example(\mathbf{x})$
    $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{H}_l(\mathbf{x})\mathbf{H}_l(\mathbf{x})^T$
    if $\ell_p(\mathbf{x}', \mathbf{x}) > \epsilon_d$ then
        $\mathcal{D}_{k,input} \leftarrow \mathbf{x}'$
    end if
end for
for $\forall \mathbf{x}' \in \mathcal{D}_{k,input}$ do
    $||\mathbf{H}_l(\mathbf{x}')||_{\mathbf{A}} = \sqrt{\mathbf{H}_l(\mathbf{x}')^T \mathbf{A}^{-1} \mathbf{H}_l(\mathbf{x}')}$
end for
$\mathcal{D}_{k,latent} \leftarrow \mathbf{x}' \in \mathcal{D}_{k,input}$ with large $||\mathbf{H}_l(\mathbf{x}')||_{\mathbf{A}}$
for $\forall \mathbf{x}' \in \mathcal{D}_{k,latent}$ do
    $\mathcal{D}_k \leftarrow Logit\_Manipulation(\mathbf{x}')$
end for
return key sample set $\mathcal{D}_k$
```

on the gradient in the direction of reducing the logit difference between the corresponding classes of $c'(\mathbf{x})$ and $c^*(\mathbf{x})$. This concept is similar to logit squeezing (Kannan, Kurakin, and Goodfellow 2018) and logit manipulation (Zhao and Lao 2022). Each step of the iterative process for some step size α can be expressed as:

$$\mathbf{x}' := \mathbf{x}' + \alpha \cdot sign(\nabla_{\mathbf{x}'}(\mathcal{L}(F, \mathbf{x}', f'(\mathbf{x}')) - \mathcal{L}(F, \mathbf{x}', f^*(\mathbf{x}'))),$$

where $f'(\cdot)$ and $f^*(\cdot)$ represent the logit outputs of the adversarial class and truth class, respectively. When the logit difference between the adversarial class and the true class is below a small threshold ϵ_l, i.e., $f'(\mathbf{x}') - f^*(\mathbf{x}') < \epsilon_l$, the final sample is included into $\mathcal{D}_k$. We use $\epsilon_l = 0.0005$ in our experiments. Algorithm 1 outlines the detailed steps.

Signature Embedding

Since the objective is to assign a unique signature for each protected model F_i, we need to select an appropriate subset of key samples, $\mathcal{K}_i$ as the embedding batch for each model.

Balanced Classes. Prior works have shown that class imbalance especially in online learning or incremental learning where data are available in a sequential order can cause learning bias towards the majority classes (i.e., the classes with more data) and poor generalization (Nguyen, Cooper, and Kamei 2011). Therefore, to avoid significant decision boundary shifts with respect to the natural inputs, it is essential to ensure the balance of classes in the embedding batch.

For each key sample $\mathbf{x}'$, the adversarial class $c'(\mathbf{x})$ can be considered as a sample from $\mathcal{C} \setminus c^*(\mathbf{x})$, where $\mathcal{C}$ represents the entire output space. Ideally, if we can select the set of the embedding batch $\mathcal{K}_i$ such that $c^*(\mathbf{x}) \rightarrow c'(\mathbf{x})$ is a one-to-one mapping for each $\mathbf{x}' \in \mathcal{K}_i$, balance can be guaranteed. In other words, the key samples with a same adversarial class should be from the same original class in the embedding batch. Obviously, as $\mathbf{x}'$ is generated as an adversarial example, $c'(\mathbf{x}) \neq c^*(\mathbf{x})$. Thus, we can consider the ideal mapping from the adversarial classes to the original labels

Algorithm 2: Signature Embedding

```
Require: $F$, $\mathcal{L}$, $\mathcal{K}_i \subset \mathcal{D}_k$, $\mathcal{D}_v$, $\epsilon_p$
 1: construct an auxiliary graph $G'(V', E')$
 2: $V' \leftarrow c\_in_i, c\_out_i, \forall c_i$
 3: if $\exists \mathbf{x}' \in \mathcal{D}_k$, s.t. $c^*(\mathbf{x}) = c_i, c'(\mathbf{x}) = c_j$ then
 4:     $E' \leftarrow (c\_out_i, c\_in_j)$, with a weight of 1
 5: end if
 6: $E' \leftarrow (c\_out_i, c\_in_i), \forall c_i$, with a weight of 0
 7: $G^*(V^*, E^*) \leftarrow Weighted\_Max\_Matching(G')$
 8: for $\forall (c\_out_i, c\_in_j) \in E^*$ do
 9:     $\mathcal{K}_i \leftarrow Select(\mathbf{x}' \in \mathcal{D}_k)$ s.t. $c^*(\mathbf{x}) = c_i, c'(\mathbf{x}) = c_j$
10: end for
11: $\mathcal{K}_i \leftarrow Balance\_Class(\mathcal{K}_i, \mathcal{D}_v)$
12: for $\forall \mathbf{x}' = \mathbf{x}'_t \in \mathcal{K}_i$ do
13:     $\{\mathcal{D}_e, \mathbf{c}_e\} \leftarrow \{\mathbf{x}'_t, c^*(\mathbf{x})\}\{\mathbf{x}'_{t-1}, c^*(\mathbf{x})\}\{\mathbf{x}'_{t+1}, c'(\mathbf{x})\}$
14: end for
15: $F_i \leftarrow F$
16: while $\underset{\mathbf{x} \in \mathcal{D}_v}{Prob}(F_i(\mathbf{x}) \neq F(\mathbf{x})) \leq \epsilon_p$ do
17:     $F_i \leftarrow \underset{\{\mathbf{x}, y\} \in \{\mathcal{D}_e, \mathbf{c}_e\}}{argmin} \mathcal{L}(F_i(\mathbf{x}), y)$
18: end while
19: return protected model $F_i$
```

to be a complete permutation that no object is in its original place. If a complete permutation over all the classes does not exist, the goal is to *find a subset of classes with the maximum cardinality that a complete permutation can be found*. Fortunately, this problem can be solved in polynomial time with a weighted bipartite graph maximum matching, which is outlined in Lines 1-10 of Algorithm 2.

Then, we pick $\mathbf{x}'$ with a desired $c^*(\mathbf{x})$ to $c'(\mathbf{x})$ mapping, according to the obtained vertex disjoint cycles on G. In fact, our key sample generation process with large perturbations can yield many more possible classes than the conventional adversarial example settings. In our experiments, we are always able to find a matching that covers all the labels in $\mathcal{C}$ (i.e., a perfect matching). In practice, if a complete permutation over the entire $\mathcal{C}$ cannot be found, we can complement the result of weighted graph maximum matching with the set of adversarial classes $\mathcal{C}'$ by natural inputs from the remaining classes, i.e., $\mathbf{x} \in \mathcal{C} \setminus \mathcal{C}'$, to balance the labels in the embedding batch $\mathcal{K}_i$ (Line 11 of Algorithm 2).

Bounding the Decision Boundary Shift. A successful signature embedding is achieved by altering the predictions of selected key samples $\mathbf{x}' \in \mathcal{K}_i$ as intended. To this end, we need to change the label of $\mathbf{x}'$ from the adversarial class $c'(\mathbf{x})$ to the original label $c^*(\mathbf{x})$ for retraining, similar to the concept of adversarial training (Goodfellow, Jonathon, and Christian 2015; Madry et al. 2018). Different from the prior works, our framework adds more samples into the embedding batch to bound the unintended decision boundary shift. The proposed framework includes samples before and after each successful key sample along the process of adversarial example generation. If at the t-th iteration, an adversarial example $\mathbf{x}'_t$ with the adversarial class $c'(\mathbf{x})$ is found, which is later discovered to have a high uncertainty score

Dataset	Lr	Succ (%)	Avg Epoch	Avg Unchanged (%)	Avg C2W (%)	Avg W2C (%)	Avg W2W (%)
MNIST	1e-4	100	123.40	99.51	0.24	0.22	0.03
	1e-3	100	14.37	99.39	0.33	0.24	0.04
CIFAR10	1e-6	99.9	83.85	97.92	0.79	0.79	0.50
	1e-5	99.8	43.38	97.61	0.94	0.88	0.57

Table 1: The effectiveness on 1024 unique signature embeddings.

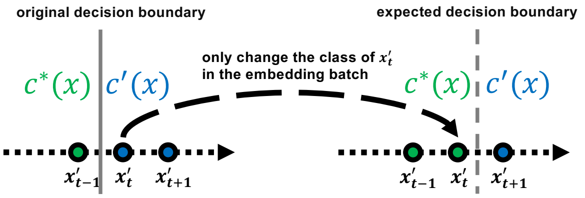

Figure 2: Left: collect the successful adversarial example $\mathbf{x}'_t$ along with the samples before and after the current iteration into the embedding batch. Only change the class of $\mathbf{x}'_t$ from $c'(\mathbf{x})$ to $c^*(\mathbf{x})$ for retraining. Right: expected decision boundary after retraining.

$||\mathbf{H}_l(\mathbf{x}')||_{\mathbf{A}}$, we will also collect the sample before this iteration (i.e., $\mathbf{x}'_{t-1}$ with class $c^*(\mathbf{x})$, which is not an adversarial example), and the sample after this iteration (i.e., $\mathbf{x}'_{t+1}$ with class $c'(\mathbf{x})$). During the retraining, we include all of these samples in the embedding batch, while only changing the class of $\mathbf{x}'_t$ from $c'(\mathbf{x})$ to $c^*(\mathbf{x})$. Consequently, the decision boundary is expected to shift only very slightly, from between $\mathbf{x}'_{t-1}$ and $\mathbf{x}'_t$ to between $\mathbf{x}'_t$ and $\mathbf{x}'_{t+1}$. We illustrate an abstract version of this concept in Figure 2. This additional step helps ensure the fragility of the signature (i.e., the key samples are still sensitive after embedding but on the other side of the decision boundary), while we believe "robust" watermarking methods seek to keep a sufficient distance between the key samples to decision boundary after embedding to withstand transformation attacks. This step also reduces the model tweaking for embedding, making it easier to maintain the original functionality.

We iteratively retrain the embedding batch with a small learning rate to minimize the impact on the normal behavior while ensuring the fragility of the signatures. We employ a simple early-stopping scheme by measuring the changes in classification on a small batch from the validation data. The retraining stops when the percentage of data in the batch change predictions, i.e., $Prob(F_i(\mathbf{x}) \neq F(\mathbf{x}))$, reaches a threshold of ϵ_p. We use 0.0005, 0.002, 0.02 for MNIST, CIFAR10, and ImageNet, respectively. The steps for bounding the shift are outlined in Lines 12-18 of Algorithm 2.

Experimental Results

We leverage LeNet5 for MNIST, VGG16 for CIFAR10, and ResNet50 for ImageNet. All with test accuracies that are consistent to state-of-the-art. Our implementation was based on the PaddlePaddle deep learning platform.

Effectiveness

The effectiveness of a signature embedding process is measured by the percentage of key samples that change predictions as intended, i.e., from $c'(\mathbf{x})$ to $c^*(\mathbf{x})$ for selected $\mathbf{x}'_t \in \mathcal{K}_i$. Table 1 summarizes the performance with different learning rates (Lr) of the embedding process, including success rate (Succ) and prediction changes. We perform the proposed methodology to generate 1024 uniquely protected models. We select 10 key samples for each signature embedding. Instead of only caring about the inference accuracy, we present the average statistics for unchanged, correct to wrong (C2W), wrong to correct (W2C), and wrong to wrong (W2W) predictions of the validation data (i.e., natural inputs) after the signature embedding. The wrong to wrong (W2W) measure, representing the scenario that a sample with an initially wrong prediction (i.e., different from the ground-truth label) changes to another wrong prediction, is not considered in prior studies since it does not impact the inference accuracy. In addition, C2W and W2C are opposite measures in the accuracy calculation. For instance, on CIFAR10, average |C2W-W2C| is small while the average W2W is more than 0.5%, leading to a relatively large deviation between the accuracy and functionality preservation. It can be seen that our method can maintain a very high percentage of unchanged predictions. In comparison, if our performance is measured only against inference accuracy, *the accuracy change, i.e., $|C2W - W2C|$, is always less than 0.1% for both MNIST and CIFAR10.*

Uniqueness

A high uniqueness requires the correlation between the signatures from different models to be small. In other words, after embedding with $\mathcal{K}_i$, not only the natural inputs but also the unselected key samples $\mathbf{x}' \in \mathcal{D}_v \setminus \mathcal{K}_i$ should not change the predictions. Since our objective is to generate multiple models with a unique signature for each, this property can ensure a high entropy across the signatures of all the models. We embed the signatures with the first set of key samples and then test the percentage of changed predictions on the second set. We measure the uniqueness by calculating the correlation of predictions between two sets of key samples as shown in Figure 3. Each set consists of 10 independent key samples. We can observe that the correlation is very small, e.g., less than 3% unselected key samples would change predictions for CIFAR10.

Fidelity

In the experiments above, we used a fixed ϵ_p for each dataset. We showed that even if we want to guarantee a

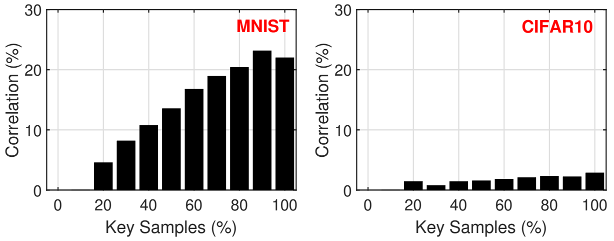

Figure 3: Correlation between selected key samples for embedding and unselected key samples. Key samples (%) represents the percentage of key samples with the original classes in the embedding batch.

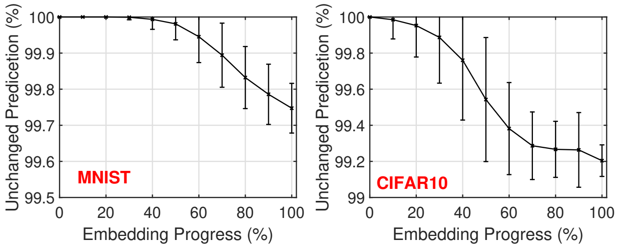

Figure 4: Functionality change on embedding progress.

nearly 100% effectiveness, the changed predictions in the worst-case scenarios are still very small. We further evaluate the performance of the proposed framework under various embedding rates, which is shown in Figure 4. Note that here we still look at the functionality instead of just the inference accuracy. As expected, there is a trade-off between the unchanged predictions of the validation data and the success rate of the selected key samples. By retraining with more epochs, the signature embedding process will yield a larger degree of model tweaking, which not only leads to more key samples to be predicted as intended but also has more impact on the natural inputs.

Efficiency

Our proposed framework achieves a minimal computational overhead of the signature embedding process by only retraining on a very small batch of $\mathcal{K}_i$ for generating each model F_i. As illustrated in Table 1, embedding 10 key samples only requires to train the batch for about 10 to 150 epochs to achieve a high success rate.

In our scheme, as shown in Figure 1, one authentication process involves one communication from the server for sending an authentication vector to the user and another communication for sending the corresponding signature from the user back to the server. Thus, the communication overhead incurred by DeepAuth is very small.

Fragility

The objective of employing the proposed DeepAuth framework is to ensure the model has not been tampered with. To this end, we evaluate the effectiveness in detecting model modifications by fine-tuning, pruning, and backdoor attack.

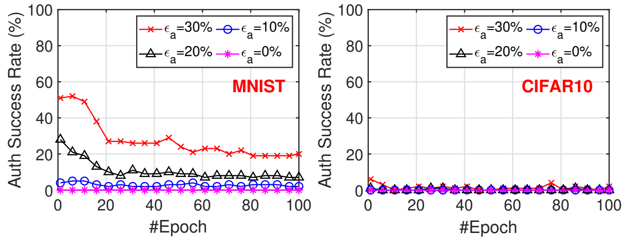

Figure 5: Authentication success rate on model fine-tuning.

Although fine-tuning and pruning are not attacks, they also maintain a similar accuracy after the processing, which can better show the fragility of the watermarks. Our experimental results show that the protected models generated by the DeepAuth framework cannot pass the authentication even under such slight model modifications, hence detecting a potential integrity breach or an illegitimate model/user. Before these processes, we first test the signatures of models with a number of authentication vectors and store the corresponding signatures for later comparison. We use 10 key samples for each authentication vector.

Fine-Tuning. We perform the fine-tuning in a layer-freezing fashion using the validation data: the model parameters of the feature layers are frozen and only the parameters on the last fully-connected layer are trainable. We present the authentication success rates under a range of error-tolerance constraints ϵ_a, i.e., authentication is successful if the percentage of error in the evaluated signature compared to the enrolled one is less than ϵ_a. The results are shown in Figure 5, where the learning rate is $1e{-}4$. Note that we deliberately examine the performance under large values ϵ_a. In practice, $\epsilon_a = 30\%$, i.e., accepting all signatures with less than 30% errors, is certainly too large for a reasonable security level. However, even under such error-tolerant settings, the authentication is still very likely to fail. In addition, the authentication success rates are always 0 if we enforce an absolute correctness requirement, i.e., $\epsilon_a = 0\%$.

Model Pruning. The pruning process is realized by zeroing out the parameters with the least magnitudes along with a retraining process (Han, Mao, and Dally 2015), which is designed to reduce the model size for faster inference and smaller memory requirement. Given a pruning factor, we sort all the parameters in the model according to their absolute values and set those parameters with the least α magnitudes to zero. The retraining process is performed on the original training data. The performance is shown in Figure 6. Compared to MNIST, signatures embedded by DeepAuth are more fragile on CIFAR10. After pruning, almost all the authentication will fail even under an ϵ_a of 30% for a CIFAR10 classifier with pruning rates in [50%, 95%].

Backdoor Attack. We also test the performance against the backdoor attack, which is a type of causative adversarial attack that alters the model to inject malicious behaviors. We follow the backdoor injection attack methodology proposed in BadNets (Gu et al. 2019). We simulate 1,000 trials of backdoor attacks. All of these trials achieve high attack success rates ($> 95\%$), while simultaneously guaranteeing

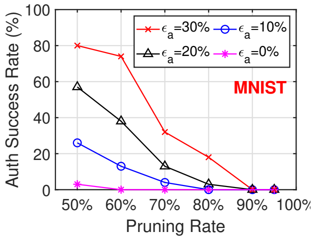

Figure 6: Authentication success rate on model pruning.

ϵ_a	40%	30%	20%	10%	0%
MNIST	24.6%	23.9%	13.7%	5.5%	2.0%
CIFAR10	0.0%	0.0%	0.0%	0.0%	0.0%

Table 2: Authentication success rate on backdoor attacks.

a high classification accuracy. We present the authentication success rates after backdoor attacks in Table 2. Similar to pruning, the signatures on the CIFAR10 classifier is more "fragile" against backdoor attacks than those on the MNIST classifier, i.e., 0% authentication success rates for all $\epsilon_a \in [0\%, 40\%]$ on CIFAR10.

Based on the results above, DeepAuth performs better on a more complex dataset with respect to uniqueness and fragility. It is important to note that protecting such models is more demanded, as the corresponding model building processes require greater resources and efforts.

Performance on ImageNet

To further evaluate the performance, we test the proposed method on a higher dimensional dataset, i.e., ImageNet. The results of effectiveness and fidelity are presented in Table 3 and Figure 7, respectively. It can be seen that DeepAuth can still achieve a 100% success rate with around 100 epochs over the embedding batch. Considering there are much more classes (i.e., 1,000) in ImageNet such that model tweaking will be more easily move the decision boundaries, the proposed method still achieves a relatively very small accuracy degradation. We also report the fragility performance against fine-tuning in Figure 8. We observe that the embedded signature will disappear even with only 5 epochs of fine-tuning, indicating an excellent fragility even on ImageNet. The results also follow the trend that the proposed method performs better on more complex datasets in terms of fragility.

Comparison with Prior Works

To the best of our knowledge, DeepAuth is the first DNN authentication framework that generates multiple protected models from a pre-trained model by embedding unique and "fragile" signatures. Based on the unique signatures, an authentication scheme can be employed to verify the user identity and the authenticity of a model. In contrast, prior "robust" watermarking methods (Adi et al. 2018; Uchida et al. 2017; Darvish Rouhani, Chen, and Koushanfar 2019; Le Merrer, Perez, and Trédan 2020) consider embedding one ID for all the models, since the objective is to verify the IP ownership instead of the user, which is qualitatively different from the proposed method. Besides, as demonstrated by our experimental results, the "fragile" signatures are extremely effective in determining if a model has been tampered with or not, which, different from prior methods, could serve as a proactive defense against malicious model modifications.

Lr	Succ (%)	Avg Epoch	ΔAccuracy
1e-6	100	78.55	1.57%
5e-7	100	131.33	0.92%

Table 3: Effectiveness and fidelity on ImageNet.

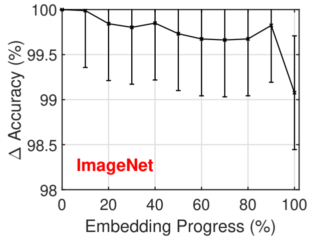

Figure 7: Fidelity on ImageNet.

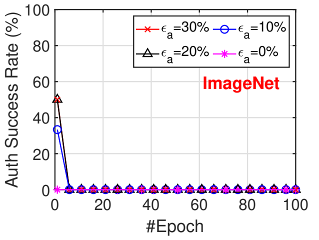

Figure 8: Fragility on ImageNet.

	Property	ΔAccuracy
Uchida et.al (2017)	Robust	$\sim 0.3\%$
Adi et.al (2018)		$\sim 0.3\%$
Rouhani et.al (2019)		$\sim 0.5\%$
Le Merrer et.al (2020)[1]		$\sim 0.2\%$
DeepAuth	Fragile	$< 0.1\%$

[1] Le Merrer et.al (2020) only reported results on MNIST, while the results of other published works are on CIFAR10.

Table 4: Comparison with "robust" watermarking methods.

Despite these differences, we compare the fidelity (i.e., change of inference accuracy) of the proposed "fragile" signature to prior "robust" watermarking methods as presented in Table 4. Note that since most of these prior works did not evaluate on ImageNet, we focus the comparison on smaller datasets. It can be observed that the proposed method achieves the lowest accuracy degradation.

Conclusion

We proposed a novel DNN authentication framework by embedding model-unique and fragile signatures. The proposed methods for generating key samples and embedding signatures ensure high degrees of uniqueness, efficiency, fidelity, and fragility of the signatures, which could serve as an effective tool for authenticating and verifying the trustworthiness of deployed DNN systems.

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of 27th USENIX Security Symposium (USENIX Security)*, 1615–1631. Baltimore, MD.

Cao, X.; Jia, J.; and Gong, N. Z. 2021. IPGuard: Protecting the Intellectual Property of Deep Neural Networks via Fingerprinting the Classification Boundary. In *Proceedings of ASIA Conference on Computer and Communications Security (ASIACCS)*, 14–25. Hong Kong, China.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of 2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. San Jose, CA.

Chen, H.; Rouhani, B. D.; Fan, X.; Kilinc, O. C.; and Koushanfar, F. 2018. Performance comparison of contemporary DNN watermarking techniques. *arXiv preprint arXiv:1811.03713*.

Chen, H.; Rouhani, B. D.; Fu, C.; Zhao, J.; and Koushanfar, F. 2019. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR)*, 105–113. Ottawa, Canada.

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

Darvish Rouhani, B.; Chen, H.; and Koushanfar, F. 2019. DeepSigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 485–497. Providence, RI.

Doan, K.; Lao, Y.; and Li, P. 2021. Backdoor Attack with Imperceptible Input and Latent Modification. In *Proceedings of Neural Information Processing Systems (NeurIPS)*. Virtual.

Doan, K.; Lao, Y.; Zhao, W.; and Li, P. 2021. LIRA: Learnable, Imperceptible and Robust Backdoor Attacks. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 11966–11976. Montreal, Canada.

Fan, L.; Ng, K. W.; and Chan, C. S. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 4714–4723. Vancouver, Canada.

Goodfellow, I.; Jonathon, S.; and Christian, S. 2015. Explaining and Harnessing Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*. San Diego, CA.

Gu, Q.; and Han, J. 2014. Online spectral learning on a graph with bandit feedback. In *Proceedings of 2014 IEEE International Conference on Data Mining (ICDM)*, 833–838. Shenzhen, China.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7: 47230–47244.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of International Conference on Learning Representations (ICLR)*. San Diego, CA.

Hartung, F.; and Kutter, M. 1999. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7): 1079–1107.

He, Z.; Zhang, T.; and Lee, R. 2019. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4729–4737. Long Beach, CA.

Kannan, H.; Kurakin, A.; and Goodfellow, I. 2018. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2018. Adversarial Examples in the Physical World. In *Artificial Intelligence Safety and Security*, 99–112. Chapman and Hall/CRC.

Le Merrer, E.; Perez, P.; and Trédan, G. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13): 9233–9244.

Li, H.; Willson, E.; Zheng, H.; and Zhao, B. Y. 2019a. Piracy Resistant Watermarks for Deep Neural Networks. *arXiv preprint arXiv:1910.01226*.

Li, Z.; Hu, C.; Zhang, Y.; and Guo, S. 2019b. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, 126–137. San Juan, PR.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2018. Trojaning Attack on Neural Networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, CA.

Lu, C.-S. 2004. *Multimedia security: steganography and digital watermarking techniques for protection of intellectual property: steganography and digital watermarking techniques for protection of intellectual property*. Igi Global.

Lukas, N.; Zhang, Y.; and Kerschbaum, F. 2021. Deep neural network fingerprinting by conferrable adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vienna,Austria.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Nguyen, H. M.; Cooper, E. W.; and Kamei, K. 2011. Online learning from imbalanced data streams. In *Proceedings of the 2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 347–352. Dalian, China.

Rudi, A.; Calandriello, D.; Carratino, L.; and Rosasco, L. 2018. On fast leverage score sampling and optimal learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 5672–5682. Montreal, Canada.

Szyller, S.; Atli, B. G.; Marchal, S.; and Asokan, N. 2021. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of ACM Multimedia Conference*, 4417–4425. Chengdu, China.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR)*, 269–277. Bucharest, Romania.

Wang, S.; and Chang, C.-H. 2021. Fingerprinting Deep Neural Networks-A DeepFool Approach. In *Proceedings of 2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. Daegu, South Korea.

Yang, P.; Lao, Y.; and Li, P. 2021. Robust Watermarking for Deep Neural Networks via Bi-Level Optimization. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 14841–14850. Montreal, Canada.

Yang, P.; Zhao, P.; and Gao, X. 2018. Bandit online learning on graphs via adaptive optimization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2991–2997. Stockholm, Sweden.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and Molloy, I. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS)*, 159–172. Incheon, Korea.

Zhao, B.; and Lao, Y. 2022. Towards Class-Oriented Poisoning Attacks Against Neural Networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3741–3750. Waikoloa,HI.

Zhao, J.; Hu, Q.; Liu, G.; Ma, X.; Chen, F.; and Hassan, M. M. 2020. AFA: Adversarial fingerprinting authentication for deep neural networks. *Computer Communications*, 150: 488–497.

Zhong, Q.; Zhang, L. Y.; Zhang, J.; Gao, L.; and Xiang, Y. 2020. Protecting IP of Deep Neural Networks with Watermarking: A New Label Helps. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 462–474. Singapore.