Linear Functionality Equivalence Attack against Deep Neural Network Watermarks and a Defense Method by Neuron Mapping

Fang-Qi Li, Shi-Lin Wang*, *Senior Member, IEEE*, and Alan Wee-Chung Liew, *Senior Member, IEEE*

***Abstract*—As an ownership verification technique for deep neural networks, the white-box neural network watermark is being challenged by the functionality equivalence attack. By leveraging the structural symmetry within a deep neural network and manipulating the parameters accordingly, an adversary can invalidate almost all white-box watermarks without affecting the network's performance. This paper introduces the linear functionality equivalence attack, which can adapt to different network architectures without requiring knowledge of either the watermark or data. We also propose NeuronMap, a framework that can efficiently neutralize linear functionality equivalence attacks and can be easily combined with existing white-box watermarks to enhance their robustness. Experiments conducted on several deep neural networks and state-of-the-art white-box watermarking schemes have demonstrated not only the destructive power of linear functionality equivalence attacks but also the defense capability of NeuronMap. Our result shows that the threat of basic linear functionality equivalence attacks against deep neural network watermarks can be effectively solved using NeuronMap.**

***Index Terms*—Artificial intelligence security, deep neural network watermarking, functionality equivalence attack.**

I. Introduction

THE emergence of deep neural networks (DNNs) has revolutionized the field of artificial intelligence, enabling them to perform a wide range of tasks such as game playing [1], signal processing for both visual and acoustic data [2], medical diagnosis [3], and cyber security [4], [5], [6]. This success can be attributed to the incorporation of vast amounts of data and the careful design of network architectures with appropriate hyperparameters. However, as DNNs are increasingly used in commercial applications, the need to trace ownership and establish accountability has become apparent. Therefore, there is a growing call to recognize DNN products as intellectual property and regulate their copyright.

Two major techniques for ownership verification of DNNs are fingerprint [7], [8], [9] and watermark [10], [11]. The fingerprint method extracts the characteristic decision boundary from a given DNN as its fingerprint, which remains invariant against adversarial tuning and can serve as the DNN's identity [12]. However, since these statistics are not correlated with the owner's digital identity, an unforgeable ownership proof is impossible. In contrast, watermarking schemes inject owner-dependent information into a DNN, which can later serve as evidence of ownership. Several watermarking schemes for different DNN architectures have been proposed. and several types of security have been formally proven. Integrity authentication techniques such as passport [13] and reversible watermarks [14] have also been proposed.

DNN watermarking schemes can be categorized into two types: black-box schemes and white-box schemes. Black-box DNN watermarking schemes assume that the pirated DNN can only be accessed as an interface, and its internal states are invisible. These schemes can protect ownership even if an adversary steals a DNN and deploys it as an API [15], [16]. White-box DNN watermarking schemes, on the other hand, assume that the pirated DNNs' parameters and intermediate responses are accessible. They can be used in cases where the adversary distributes its model or in lawsuits where the prosecutor needs to submit evidence for exculpation. Since white-box watermarking schemes have access to the DNN's internal states, they can inject owner-dependent information into the network's parameters [10] or the response pattern of certain neurons [17], [18]. Retrieval of such ownership information can be done by fuzzy rule [19], parameter mask [10], residual extractor [20], or another neural network [17]. Ownership test for DNN in the field usually involves both types of watermarks [19].

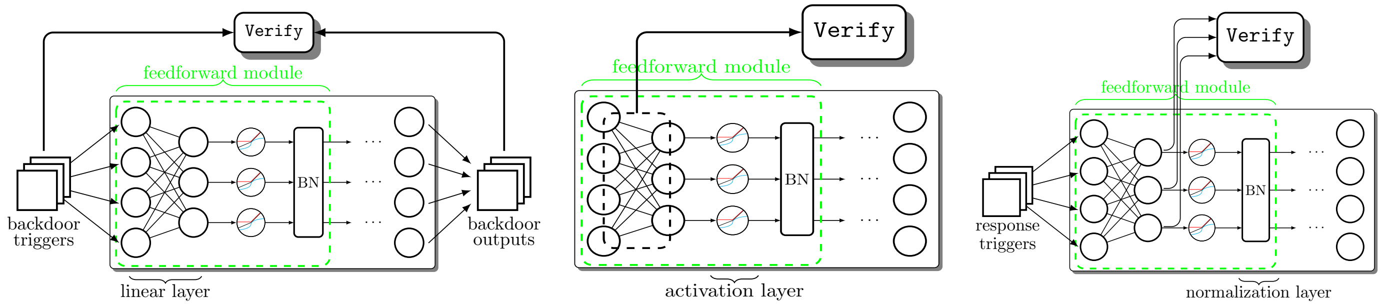
Recent studies raised a new threat against white-box DNN watermarks known as the *functionality equivalence attack* [21], [22]. This attack exploits the structural symmetry in a DNN and rearranges the neurons without affecting the DNN's performance. As a result, white-box watermark verifiers are unable to trace the ownership evidence. The functionality equivalence attack is inexpensive, does not damage the pirated DNN product, and can impair almost all existing white-box watermarking schemes. Despite its significant impact, this threat has not received adequate attention, and there are no formal analyses and corresponding defense mechanisms. This paper presents a formal analysis of a category of universal functionality equivalence attack and extends the defense method proposed in previous works [22]. The new defense method includes additional triggers and a new recovery strategy to neutralize this broader family of attacks. The contributions of our paper are three-fold:

(Corresponding author: Shi-Lin Wang.)

Fang-Qi Li and Shi-Lin Wang are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China. (e-mail: solour_lfq@sjtu.edu.cn; wsl@sjtu.edu.cn)

Alan Wee-Chung Liew is with the School of Information and Communication Technology, Griffith University, Gold Coast Campus, QLD 4222, Australia. (e-mail: a.liew@griffith.edu.au)

The work described in this paper was supported in part by the National Natural Science Foundation of China (62271307, 61771310). Our gratitude goes to the anonymous reviewers for their efforts.



(a) Black-box watermarking. (b) Parameter-based white-box watermarking. (c) Response-based white-box watermarking.

Fig. 1. Procedures of different DNN watermarking schemes.

- We formulate the Linear Functionality Equivalence Attack (LFEA), a family of universal functionality equivalence attacks. LFEA is easy to conduct and can invalidate most existing white-box watermarks without knowledge of the watermarking scheme or the training data.
- We propose an effective countermeasure, `NeuronMap`, which can neutralize LFEA. `NeuronMap` does not interfere with DNN training or watermarking embedding and can be seamlessly integrated to established white-box watermarking schemes.
- We conducted extensive experiments across various DNN architectures and white-box watermarking schemes to demonstrate the effectiveness of NeuronMap against LFEA. Our results show that incorporating `NeuronMap` into existing white-box watermarking schemes can make them resilient against LFEA with only a slight incremental in time consumption.

The rest of the paper is organized as follows: Sec.II summarizes the preliminaries and related works. Sec.III details the threat posed by LFEA. Sec.IV presents the defense method `NeuronMap`. Sec.V presents the experiment results and discussions. Finally, Sec.VI concludes the paper.

II. Preliminaries and Related Works

A. Ownership verification for DNN

Identifying DNN ownership is necessary for ensuring accountability and commercialization of DNN products. A comprehensive overview on this toic can be found in [23], [24], [25]. As an extension of multimedia watermark [26], DNN watermarking is considered a promising technique for provable ownership verification of DNNs. Formally, a DNN watermarking scheme injects the owner's identification information, which we denote as `Key`, into a clean model, resulting in a watermarked network DNN_{WM} and a module `Verify`. To establish a unique time-stamp, the owner can register the digital signature of $\{\texttt{Key}, \texttt{Verify}\}$ with an authorized judge or on a distributed ledger [27]. The owner's identifier can be retrieved from the watermarked DNN [17], [23] with

$$\Pr\left\{\texttt{Verify}(\text{DNN}_{\text{WM}}, \texttt{Key}) = \texttt{Pass}\right\} \geq 1 - \epsilon(N), \quad (1)$$

$$\Pr\left\{\texttt{Verify}(\text{DNN}_{\text{ind}}, \texttt{Key}) = \texttt{Fail}\right\} \geq 1 - \epsilon(N), \quad (2)$$

where N is the security parameter (e.g., the number of triggers), $\epsilon(\cdot)$ is a positive negligible function, and DNN_{ind} is another independent network.

In addition to basic requirements of accuracy and unambiguity defined by Eq.(1) and Eq.(2), several extra security requirements have been proposed, some of which are listed as follows.

- **Functionality-preserving:** Watermark injection should not severely damage the DNN's performance.
- **Robustness:** Tuning a watermarked DNN cannot invalidate the ownership proof.
- **Covertness:** It should be hard to distinguish a watermarked DNN from a clean one [28].
- **Efficiency:** The watermark injection process should be both time-friendly and memory-friendly.

For black-box watermarking schemes, the owner's identification is often encoded into the mapping between backdoor triggers and the network's outputs. `Verify` then checks whether the suspect DNN contains this mapping or not, as illustrated in Fig.1(a). Backdoors for image processing networks [15], natural language processing networks [29], audio processing networks [30], generative networks [31], and pre-trained encoders [32] leverage domain-specific knowledge to generate triggers. As for image classification networks, triggers can take the form of stamps [33], noises [34], out-of-dataset samples [11], and adversarial samples [35].

White-box DNN watermarking schemes operate under the assumption that the suspect model can be fully accessed, for instance, during model transactions and auditing [36]. Since the verifier can monitor the intermediate states of the suspect DNN, a significant amount of information can be embedded into and retrieved from the DNN's parameters, making white-box watermarking schemes independent of the backend task.

White-box DNN watermarking schemes can be classified into into parameter-based and response-based schemes. Parameter-based watermarking schemes extract features from the DNN's parameters and compare them with the registered identification information as shown in Fig.1(b). The features can be extracted through linear transformation [10], residual digits [20], or the combination of multiple metrics [19]. On the other hand, response-based watermarking schemes use a collection of response triggers, similar to black-box watermarking schemes, but they focus on the intermediate

TABLE I
WHITE-BOX DNN WATERMARKING SCHEMES.

Scheme	Category	Trigger	Type of verifier
`Uchida's` [10]	Parameter	–	Linear transformation
`Greedy` [20]	Parameter	–	Residual mask
`MTLSign` [17]	Response	QR codes.	Neural network
`DeepSign` [18]	Response	Outliers.	Masked matrix
`DeepJudge` [19]	Response	Adversarial samples.	Masked matrix

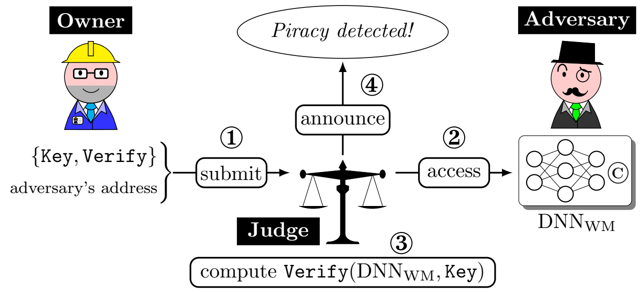


Fig. 2. The public ownership verification process for DNN.

responses from the DNN, as shown in Fig.1(c). As in backdoor triggers, response triggers can take on various patterns, such as special codes [17], outliers [18], and adversarially generated samples [19], etc. Once the verifier obtains the features from the suspect DNN's response, it computes the loss between the retrieved features and those claimed by the owner and returns `Pass` if the loss is below than a threshold. A summary of typical white-box DNN watermarking schemes is provided in Table I. We remark that although `DeepJudge` [19] is designed as a testing framework, its verification program is identical to watermark verifiers.

To claim ownership over the adversary's DNN, the owner submits the evidence $\{\texttt{Key}, \texttt{Verify}\}$ and informs the judge of the adversary's address. The judge independently accesses the suspect model, runs the verifier program, and obtains the result [27], [37]. This process is illustrated in Fig.2. Since the goal of DNN copyright protection is to prove ownership over unauthorized use, there is no need to transmit the DNN itself from the owner to the judge.

B. The functionality equivalence attack

White-box DNN watermarks are vulnerable under the Functionality Equivalence Attack (FEA). As illustrated in Fig.3, compared with ordinary removal attacks, which involve tuning/pruning/distillation, FEA manipulates the parameters in a DNN and produces a new network with precisely identical performance yet fails the watermark verifiers. Unlike network morphism transformation [38] that aims at transferring knowledge from a teacher DNN to a student DNN under morphism changes by minimizing the performance loss, FEA has theoretically zero functionality decline. Intuitively, FEA is similar to geometric attack against image watermark [39], where both attacks aim to remove copyright information by transforming the carrier with almost no utility sacrifice (in FEA, the cost is measured by functionality decline, in geometric attack, the cost is reflected by visual distortion).

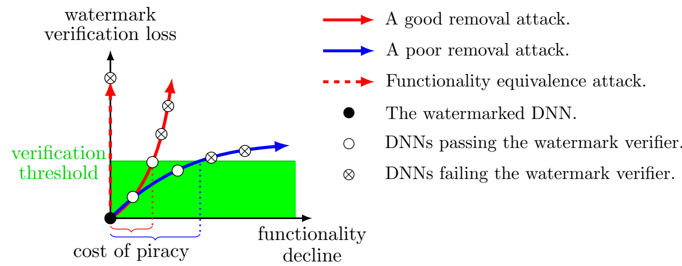


Fig. 3. A comparison between FEA and ordinary removal attacks.

An example of FEA is the neuron shuffling attack [22]. While the neurons within a DNN layer are assumed to be homogeneous, they are saved as a tensor or matrix with an order. Most watermark verifiers require information about this order to extract ownership evidence. However, this order is malleable from the adversary's perspective. For example, after shuffling the order of neurons, the verifier would fail, but the DNN's functionality remains unaffected after reordering the input weights of the next layer accordingly.

Unlike adversarial tuning, model extraction, and distillation that affect the DNN's performance, FEA has no effect on the DNN's functionality and does not depend on any knowledge about the watermarking scheme, yet it defeats almost all existing white-box watermarking schemes and is a severe threat to DNN copyright regulation. Although neuron shuffling can be canceled [22], and it is possible to design watermark that is inherently persistent against neuron shuffling using invariant statistics, such as the averaged outputs [40], without comprehensive and formal analysis of general FEAs, the defense capability of these schemes remains questionable.

III. LINEAR FUNCTIONALITY EQUIVALENCE ATTACK

A. The threat model

To formally analyze FEAs, we assume that the adversary does not modify the DNN architecture or retrain the network, so the attack is always covert and cheap. Likewise, changing the default behavior of elementary modules (e.g., flipping the sign of the activation function) is not considered since they can be trivially detected and inverted. As shown in Fig.1, DNNs are composed of a series of feedforward modules, each consisting of a linear mapping layer, a non-linear activation layer, and optionally a normalization layer.

During FEA, the adversary can freely modify the parameters. In particular, we are interested in the case where an FEA parameterized by ϕ can be directly applied to any module, so the module's mapping is transformed from f into f^{ϕ}. To preserve the network's overall functionality, it is expected that the transformation can be completely canceled by the next module. Denote the weight in the following module's linear layer before/after the FEA by $\mathbf{W}/\mathbf{W}^{\phi}$, it is sufficient that for any input $\mathbf{x}$ to the module under attack,

$$\mathbf{W} f(\mathbf{x}) = \mathbf{W}^{\phi} f^{\phi}(\mathbf{x}),$$

so the transformation from f to f^{ϕ} takes the linear form

$$f^{\phi} = \mathbf{W}^{\phi,\dagger} \mathbf{W} f,$$

in which $\mathbf{W}^{\phi,\dagger}$ is the pseudo-inverse of $\mathbf{W}^{\phi}$. We focus on this family of Linear Functionality Equivalence Attack (LFEA) since it is the most applicable and universal type of FEA.

B. The formulation of LFEA

Consider a feedforward module with I input neurons and O output neurons. For its input vector $\mathbf{x}$, this module applies a linear transformation with weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$, an activation mapping $\texttt{ReLU}$, a batch normalization with parameters $(\mathbf{E}, \mathbf{V}, \boldsymbol{\gamma}, \boldsymbol{\beta})$, and returns

$$\texttt{Cap}(\mathbf{x}) = \boldsymbol{\beta} + \boldsymbol{\gamma} \times \frac{\texttt{ReLU}(\mathbf{Wx}+\mathbf{b}) - \mathbf{E}}{\sqrt{\mathbf{V}}}. \tag{3}$$

In Eq.(3), $\mathbf{x}$ is a column vector of length I, $\mathbf{W}$ and $\mathbf{b}$ are of size $O \times I$ and $O \times 1$. $\texttt{ReLU}$ sets negative components in its input to zero. The shape of $\mathbf{E}$, $\mathbf{V}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$ is uniformly $O \times 1$. All calculations within the normalization layer are done neuronwisely.

The adversary is free to change the order of output neurons or multiply the output of a specific neuron by a positive factor. These changes can be canceled by modifying the parameters of the next feedforward module to achieve functional equivalence. Such linear modifications can be compactly represented by a matrix.

Definition 1. *Let Φ_O^+ be the smallest subgroup of matrices with shape $O \times O$ such that $\forall i, j \in \{1, 2, \cdots, O\}, k > 0$,*

$$\mathbf{I}_O - \mathbf{E}_{i,i} - \mathbf{E}_{j,j} + \mathbf{E}_{i,j} + \mathbf{E}_{j,i} \in \Phi_O^+,$$

and

$$\mathbf{I}_O + (k-1) \cdot \mathbf{E}_{i,i} \in \Phi_O^+,$$

where $\mathbf{I}_O$ is the identity matrix of order O and $\mathbf{E}_{i,j}$ is the elementary matrix whose element at position (i, j) is unity, otherwise is zero.

Definition 2. *(ϕ-LFEA) For any $\phi \in \Phi_O^+$, modifying the parameters within a feedforward module as follows*

$$\mathbf{W}^{\phi} = \phi\mathbf{W}, \mathbf{b}^{\phi} = \phi\mathbf{b},$$
$$\mathbf{E}^{\phi} = \phi\mathbf{E}, \mathbf{V}^{\phi} = \phi_2\mathbf{V},$$
$$\boldsymbol{\gamma}^{\phi} = \phi\boldsymbol{\gamma}, \boldsymbol{\beta}^{\phi} = \phi\boldsymbol{\beta},$$

where ϕ_2 is the entrywise product of ϕ and ϕ. This performs an LFEA introduced by ϕ.

The operation of ϕ-LFEA is illustrated in Fig.4 and its correctness is established in the following theorems.

Theorem 1. *(Functionality equivalence) Let Cap^{ϕ} denote the mapping introduced by a module attacked by ϕ-LFEA, then*

$$\forall \mathbf{x}, Cap^{\phi}(\mathbf{x}) = \phi Cap(\mathbf{x}).$$

This linear transformation can be undone by multiplying the weight matrix of the next module by ϕ^{-1} on the right.

Proof. Both statements are direct results of the definition in Eq.(3). Without loss of generality, the i-th component of $\texttt{Cap}^{\phi}(\mathbf{x})$ is given by the i-th component of

$$\phi\boldsymbol{\beta} + \phi\boldsymbol{\gamma} \times \frac{\texttt{ReLU}(\phi\mathbf{Wx} + \phi\mathbf{b})}{\sqrt{\phi_2\mathbf{V}}}. \tag{4}$$

The definition of Φ_O^+ implies that the i-th row of ϕ contains only one non-zero component, $\phi_{i,j} = k > 0$. So the i-th component of Eq.(4) is reduced to

$$k\boldsymbol{\beta}_j + k\boldsymbol{\gamma}_j \times \frac{k \cdot \texttt{ReLU}(\mathbf{Wx}+\mathbf{b})_j}{\sqrt{k^2\mathbf{V}_j}} = k\boldsymbol{\beta}_j + k\boldsymbol{\gamma}_j \times \frac{\texttt{ReLU}(\mathbf{Wx}+\mathbf{b})_j}{\sqrt{\mathbf{V}_j}},$$

which is precisely the i-th component in $\phi\texttt{Cap}(\mathbf{x})$.

For the next module, the inputs are firstly transformed by left multiplying another weight matrix $\mathbf{W}'$, and

$$\mathbf{W}'\texttt{Cap} = (\mathbf{W}'\phi^{-1})(\phi\texttt{Cap}) = (\mathbf{W}'\phi^{-1})\texttt{Cap}^{\phi}.$$

This completes the proof. □

For modules without the batch normalization layer, setting $\mathbf{W}^{\phi} = \phi\mathbf{W}$ and $\mathbf{b}^{\phi} = \phi\mathbf{b}$ completes a ϕ-LFEA.

Theorem 2. *(The completeness of Φ_O^+) Φ_O^+ is the maximal subgroup of $O \times O$ invertible matrices that satisfies the functionality equivalence property by Theorem 1.*

Proof. The bijection between FEAs allowing shuffling of neurons with positive scaling and Φ_O^+ is evident. We now consider extra linear operators that extend Φ_O^+ and prove that they fail the universal functionality equivalence property.

The first extension is the non-positive scaling that multiplies the response of a specific neuron by $k \leq 0$, this is tantamount to extending Φ_O^+ with generator $\mathbf{I}_O + (k-1) \cdot \mathbf{E}_{i,i}$, where $i \in \{1, 2, \cdots, O\}, k \leq 0$. Physically, this operation multiplies the output of the i-th neuron by $k \leq 0$ before the $\texttt{ReLU}$ unit, which results in irreversible damage to the DNN, since $\texttt{ReLU}$ simply nullifies negative inputs.

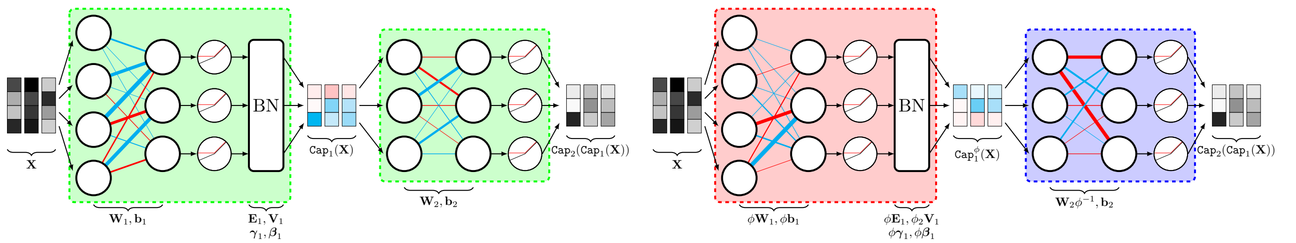
The second extension is incorporating $\mathbf{I}_O + k \cdot \mathbf{E}_{i,j}$ for $i, j \in \{1, 2, \cdots, O\}, i \neq j$ into Φ_O^+. This is identical to adding the outputs of several independent neurons and feeding the summation to $\texttt{ReLU}$. This operator is in general irreversible. For example, let $I = O = 2$, $\phi = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, denote the original output of two neurons before $\texttt{ReLU}$ as m_1, m_2, and the output pair after ϕ-LFEA and $\texttt{ReLU}$ as m_1', m_2', we have

$$m_1' = \texttt{ReLU}(m_1 + m_2),$$
$$m_2' = \texttt{ReLU}(m_2).$$

When $m_2' > 0$, the original output can be recovered as $\texttt{ReLU}(m_1) = \texttt{ReLU}(m_1' - m_2')$. When $m_2' = 0$, the information in $\texttt{ReLU}(m_1)$ is lost, so the attacked module represents a different function, which is contradictory to the adversary's purpose. These two types of extension have exhausted all possible linear modifications for the feedforward module. □

We remark that Φ_O^+ is specialized for feedforward modules with $\texttt{ReLU}$ family activations (e.g., $\texttt{LeakyReLU}$, $\texttt{PReLU}$ [41], etc.), the dominant category in current DNN architectures. If a module adopts fully non-linear activations such as $\texttt{Sigmoid}$ or $\texttt{Tanh}$ then the corresponding model is reduced to the permutation matrices.

LFEA is not only designed for fully connected layers and linearly stacked networks, but its generalization to complex network modules is also straightforward. We show below examples of variants of LFEA for gated recurrent unit (GRU), 2D convolutional layer, and residual structure.



(a) The original modules.

(b) ϕ-LFEA applied.

Fig. 4. ϕ-LFEA on a feedforward module.

C. LFEA for recurrent units

GRU [42] is a variant of long short term memory (LSTM) units [43] for sequential data, the feedforward formulation is:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}, \mathbf{x}_t]), \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}, \mathbf{x}_t]), \\
\tilde{\mathbf{h}_t} &= \mathtt{Tanh}(\mathbf{W}_h[\mathbf{r}_t * \mathbf{h}_{t-1}, \mathbf{x}_t]), \\
\mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}_t},
\end{aligned}$$

where $[\mathbf{h}_{t-1}, \mathbf{x}_t]$ denotes column concatenation. A ϕ-LFEA on this unit can be carried out as

$$\mathbf{W}^{\phi}_{z,r,h} = \begin{pmatrix} \phi \\ \mathbf{I} \end{pmatrix} \mathbf{W}_{z,r,h},\ \mathbf{h}^{\phi}_0 = \phi \mathbf{h}_0. \tag{5}$$

Cancelling the ϕ-LFEA from the previous GRU layer requires modifying the weights in the current units according to

$$\mathbf{W}'_{z,r,h} = \mathbf{W}_{z,r,h} \begin{pmatrix} \mathbf{I} \\ \phi^{-1} \end{pmatrix}. \tag{6}$$

Theorem 3. *(Recurrent functionality equivalence) Denote the t-th output of a GRU attacked by ϕ-LFEA according to Eq.*(5) *as* $\mathtt{GRU}^{\phi}_t$ *then* $\forall t, \mathbf{x}_t$*:*

$$\mathtt{GRU}^{\phi}_t(\mathbf{x}_t) = \phi \mathtt{GRU}_t(\mathbf{x}_t).$$

This linear transformation can be reversed at the next layer by Eq.(6)*.*

The proof by induction is straightforward. For the case of GRU, ϕ must be a permutation matrix, so LFEA is reduced to the neuron shuffling attack.

D. LFEA for convolutional layers

The convolutional layer is the common building block for image or video processing DNNs [44]. By utilizing the spatial continuity in inputs, convolutional layers extract features that are invariant against shifting, rotation, blurring, etc.

A 2D convolutional layer transforms I input feature maps $\{M^{\text{in}}_i\}_{i=1}^{I}$ into O output feature maps $\{M^{\text{out}}_o\}_{o=1}^{O}$. Its parameters are composed of $O * I$ kernels $\{K_{o,i}\}_{o=1,i=1}^{O,I}$, each of which is a matrix of size $s*s$, and a bias vector $\mathbf{b}$ of size $O\times 1$. Concretely, the o-th output feature map M^{out}_o is computed by

$$M^{\text{out}}_o = \sum_{i=1}^{I} M^{\text{in}}_i \odot K_{o,i} + b_o \mathbf{1},$$

where $\odot$ denotes the 2D convolution operator, and $\mathbf{1}$ is a matrix with the same shape as M^{out}_o whose all entries are set as one. LFEA or the general FEA does not change the internal structure within each feature map, otherwise, the convolution operator would malfunction. Applying ϕ-LFEA to a feedforward module with a convolutional layer changes the order of input/output feature maps and amplifies specific neurons' response, this is tantamount to multiplying the convolutional weights by ϕ, where each entry is now an $s * s$ tuple

$$K^{\phi}_{o,i} = \sum_{u=1}^{O} \phi_{o,u} \cdot K_{u,i}, \tag{7}$$

the change in the bias is identical to the basic case, $\mathbf{b}^{\phi} = \phi \mathbf{b}$. For the convolutional layers, an analogous statement of Theorem 1 holds.

Theorem 4. *(Convolutional functionality equivalence) Denote the output function of a convolutional module attacked by ϕ-LFEA as* $\mathtt{ConvCap}^{\phi}$ *then we have* $\forall \{M^{in}_i\}_{i=1}^{I}$*:*

$$\mathtt{ConvCap}^{\phi}_o(\{M^{in}_i\}_{i=1}^{I}) = \sum_{u=1}^{O} \phi_{o,u} \cdot \mathtt{ConvCap}_u(\{M^{in}_i\}_{i=1}^{I}).$$

This linear transformation can be reversed by multiplying the convolutional weight of the next module by ϕ^{-1} on the right analogously as Eq.(7)*.*

The proof is straightforward.

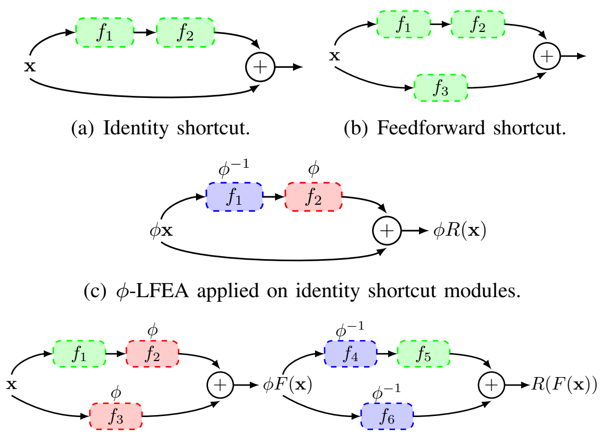
E. LFEA for residual blocks

The residual block is designed to overcome gradient vanishing problem in very deep neural networks [45]. A residual block involves a shortcut connection so its inputs are directly transferred as the baseline of its outputs, examples are given in Fig.5(a)(b).

When the shortcut is the identity mapping as shown by Fig.5(a), i.e., the output of the residual block takes the form:

$$R(\mathbf{x}) = \mathbf{x} + f_2(f_1(\mathbf{x})),$$

then it is impossible to directly apply LFEA. Otherwise, the default behavior of neuron-wise addition has to be modified, which is contradictory to our assumptions about the adversary. This fact does not imply that the output of this module must be intact. It is possible that the previous module undergoes



(a) Identity shortcut. (b) Feedforward shortcut.

(c) ϕ-LFEA applied on identity shortcut modules.

(d) ϕ-LFEA applied on feedforward shortcut modules.

Fig. 5. Green modules are intact. Red modules undergoes ϕ-LFEA. Blue modules' linear weights are multiplied by ϕ^{-1} on the right.

ϕ-LFEA, whose effects can pass through a module with an identity shortcut since

$$\phi R(\mathbf{x}) = (\phi\mathbf{x}) + \phi f_2(f_1(\phi^{-1}(\phi\mathbf{x}))).$$

This can be done by multiplying the weight of the first module on the ordinary connection by ϕ^{-1} on the right and applying ϕ-LFEA to the last module on the route as shown in Fig.5(c). As a result, watermarks based on the response of modules with an identity shortcut remain unusable under LFEA.

When the shortcut is another series of feedforward modules as Fig.5(b), ϕ-LFEA can be applied to the last feedforward modules at the end of both paths so the output neurons are transformed. The subsequent recovery in the next residual block w.r.t. the modified input is done accordingly by multiplying ϕ^{-1} on the right for the weights of the first blocks on both paths, as shown in Fig.5(d).

IV. The Defense Framework: NeuronMap

A. Motivation

LFEA could not be undone solely from the DNN's parameters since any weight matrix $\mathbf{W}$ would have been modified into $\phi_1\mathbf{W}\phi_2$, from which the statistics of $\mathbf{W}$ can no longer be retrieved. However, the outputs of neurons under ϕ-LFEA are subjected to a transformation that can be inverted. The key observation is that, as long as LFEA does not mix the outputs of independent neurons, the row space of the intermediate response for a given set of inputs is invariant. This invariance is sufficient to retrieve ϕ, which maps the neurons into their original structure, neutralizing the effect of ϕ-LFEA completely. The overview of our defense framework, `NeuronMap`, is given in Fig.6. `NeuronMap` works by targeting the module where the watermark is embedded, it applies a set of triggers to the DNN and collects responses of the target module from both the suspect network and the owner's network. It then estimates ϕ from this pair of responses as $\hat{\phi}$. Finally, `NeuronMap` applies $\hat{\phi}^{-1}$-LFEA to the watermarked module or appends $\hat{\phi}^{-1}$ to the original watermark verifier to perform ownership proof.

Algorithm 1 `GreedyPhi`$(\mathbf{Y}, \mathbf{Y}^{\phi})$

Input: The original response matrix $\mathbf{Y}$ and the response matrix after LFEA $\mathbf{Y}^{\phi}$.
Output: An estimation of the LFEA parameter $\hat{\phi}$.

if $\mathbf{Y}^{\phi}$ contains O' rows, $O' < O$ **then**
  for $o = 1$ to $O - O'$ **do**
    $\mathbf{Y}^{\phi} = \begin{pmatrix} \mathbf{Y}^{\phi} \\ \mathbf{0} \end{pmatrix}$;
  end for
end if
$\hat{\phi} = \mathbf{I}_O$;
for $o = 1$ to O **do**
  $\beta = \frac{\mathbf{Y}^{\phi}[o]\cdot\mathbf{Y}[o]}{\mathbf{Y}[o]\cdot\mathbf{Y}[o]}$;
  $\min = \|\mathbf{Y}^{\phi}[o] - \beta\mathbf{Y}[o]\|_2^2$;
  $i = o$;
  for $j = o$ to O **do**
    $\beta = \frac{\mathbf{Y}^{\phi}[o]\cdot\mathbf{Y}[j]}{\mathbf{Y}[j]\cdot\mathbf{Y}[j]}$;
    $d = \|\mathbf{Y}^{\phi}[o] - \beta\mathbf{Y}[j]\|_2^2$;
    if $d \leq \min$ **then**
      $\min = d; i = j; \beta_{\min} = \beta$;
    end if
  end for
  $\mathbf{P} = \mathbf{I}_O - \mathbf{E}_{o,o} - \mathbf{E}_{i,i} + \beta_{\min}\mathbf{E}_{o,i} + \mathbf{E}_{i,o}$;
  $\mathbf{Y} = \mathbf{P}\mathbf{Y}$;
  $\hat{\phi} = \mathbf{P}\hat{\phi}$;
end for
Return $\hat{\phi}$.

B. Response mapping

According to Theorem 1, the output of the feedforward module under ϕ-LFEA is left multiplied by ϕ. This influence is independent of any potential LFEAs exerted on any other feedforward modules within the DNN.

Concretely, let $\mathbf{T} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ be the design matrix of `NeuronMap` triggers. Denote the mapping function from the DNN's input to the watermarked module's response as $\mathbf{y}(\cdot)$. The original response matrix is $\mathbf{Y} = \mathbf{y}(\mathbf{T})$ of shape $O \times T$. After undergoing ϕ-LFEA, the response of this module becomes $\mathbf{Y}^{\phi} = \phi\mathbf{y}(\mathbf{T})$. An estimation of ϕ, which we denoted as $\hat{\phi}$, can be computed from $\mathbf{Y}$ and $\mathbf{Y}^{\phi}$

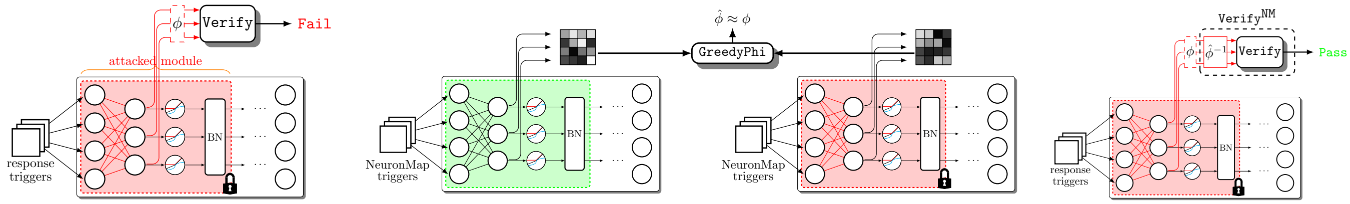
$$\hat{\phi} = \mathbf{Y}^{\phi}\mathbf{Y}^{-1}, \tag{8}$$

or using the pseudo-inversion

$$\hat{\phi} = \mathbf{Y}^{\phi}\mathbf{Y}^{\mathrm{T}}(\mathbf{Y}\mathbf{Y}^{\mathrm{T}})^{-1}. \tag{9}$$

Eq.(8) can be used when $O = T$ and $\mathbf{Y}$ is invertible. Eq.(9) can be used when $T \gg O$ so that $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ is invertible.

In practice, adversaries usually combine tuning attacks with LFEA such that the actual response often deviates from $\phi\mathbf{y}(\mathbf{T})$. As a result, the estimate $\hat{\phi}$ from Eq.(8) and Eq.(9) might not be an element in Φ_O^{+}, so $\hat{\phi}^{-1}$-LFEA is undefined. Instead, it is required that the estimate $\hat{\phi}$ lies in Φ_O^{+}. To meet this requirement, we adopt a greedy algorithm, `GreedyPhi`, details are given in Algo.1. `GreedyPhi` iteratively searches for the row in $\mathbf{Y}^{\phi}$ that is closest to a row of $\mathbf{Y}$ modulo a positive scaling factor. Then it includes the corresponding



(a) Applying LFEA. (b) Phase I: Inferring the LFEA's parameter using `GreedyPhi`. (c) Phase II: Running $\texttt{Verify}^{\text{NM}}$.

Fig. 6. The overview of `NeuronMap`.

Algorithm 2 $\texttt{Verify}^{\text{NM}}(\texttt{Key}, \text{DNN}|\mathbf{T}, \mathbf{Y})$

Input: `NeuronMap` triggers $\mathbf{T}$, the response matrix from the owner's model $\mathbf{Y} = \text{DNN}_{\text{WM}}.\mathbf{y}(\mathbf{T})$, the suspect model DNN, the evidence `Key`, and the original verifier module `Verify`.

Output: The ownership verification result.

$\mathbf{Y}^{\phi} = \text{DNN}.\mathbf{y}(\mathbf{T})$;
$\hat{\phi} = \texttt{GreedyPhi}(\mathbf{Y}, \mathbf{Y}^{\phi})$;
Applying $\hat{\phi}^{-1}$-LFEA to DNN to obtain $\widehat{\text{DNN}}$;
if $\mathbf{Y}$ contains O rows, $\mathbf{Y}^{\phi}$ contains $O' > O$ rows **then**
  Deleting the last $O' - O$ rows from the response of $\widehat{\text{DNN}}$'s watermarked layer.
end if
Return $\texttt{Verify}(\widehat{\text{DNN}}, \texttt{Key})$.

transformation to $\hat{\phi}$, so the output of `GreedyPhi` always lies in Φ_O^+ and $\hat{\phi}^{-1}$-LFEA is well-defined.

In situations where the adversary has conducted structural pruning, $\mathbf{Y}^{\phi}$ would contain less than O rows. The addition of redundant neurons as distractors is unlikely to confuse `GreedyPhi` since these extra neurons cannot function similarly to the original neurons. Otherwise, the training of the network would be unnecessary. These extra neurons are ignored during verification. For convolutional modules, each output map is presented by the pixel of a fixed location so `GreedyPhi` is applicable.

Notice that unlike $\texttt{Cap}(\cdot)$ defined in Eq.(3), the input of $\mathbf{y}(\cdot)$ is the same as that of the entire DNN rather than the output of the previous module. Focusing on $\mathbf{y}(\cdot)$ allows the verifier to ignore potential LFEAs on the previous module, which are uncorrelated to the ownership verification process. As the last step, `NeuronMap` wraps the watermark verifier as Algo.2.

C. Trigger generation

We are left with the freedom to select the `NeuronMap` trigger set $\mathbf{T}$. The response patterns of different neurons on $\mathbf{T}$ should be sufficiently distinctive. Otherwise, the rows in $\mathbf{Y}/\mathbf{Y}^{\phi}$ are similar to each other so `GreedyPhi` cannot correctly estimate ϕ. To accommodate this prerequisite, we propose three categories of triggers.

1) **Samples from the training dataset ($\mathcal{D}$).** Since the DNN is trained to distinguish normal samples, a randomly selected subset of size T, optimally from different classes, should enjoy maximal distinguishability.
2) **Random triggers ($\mathcal{R}$).** If exposing the training dataset has privacy or security risks then a collection of randomly generated samples is an option.
3) **Attack triggers ($\mathcal{A}$).** We can also produce attack triggers so that the response follows specific distributions so the difference between each pair of triggers is maximized.

To produce attack triggers, we encode target neurons from the neurons' outputs to maximize distinguishability and generate triggers whose response represents neurons' codes. For T triggers and O neurons, it is sufficient to use a $C = \lceil \sqrt[T]{O} \rceil$ code system. The encoding process first runs a clustering algorithm on the responses of all O neurons on normal inputs to obtain C centroids $\{m_c\}_{c=1}^{C}$. Then the o-th neuron's code is set as a tuple of length T:

$$\texttt{c}(o) = \left(m_{\lfloor \frac{o-1}{C^0} \rfloor \bmod C}, m_{\lfloor \frac{o-1}{C^1} \rfloor \bmod C}, \cdots, m_{\lfloor \frac{o-1}{C^{T-1}} \rfloor \bmod C} \right).$$

The code table for all neurons is

$$\texttt{D}\left(O, T, \{m_c\}_{c=1}^{C}\right) = \begin{pmatrix} \texttt{c}(1) \\ \texttt{c}(1) \\ \cdots \\ \texttt{c}(O) \end{pmatrix}.$$

$\texttt{D}$ is the desired response matrix for attack triggers ($\mathcal{A}$) to ensure maximal distinguishability. Finally, the t-th attack trigger $\mathbf{x}_t^{\mathcal{A}}$ is generated so that the neuron's response on it is close to the t-column of $\texttt{D}$. In other words, $\mathbf{x}_t^{\mathcal{A}}$ is the minimizer of the following loss function:

$$\mathcal{L}(\mathbf{x}_t^{\mathcal{A}}) = \|\mathbf{y}(\mathbf{x}_t^{\mathcal{A}}) - \texttt{D}_{:,t}\|_2^2 = \sum_{o=1}^{O} \left(\mathbf{y}_o(\mathbf{x}_t^{\mathcal{A}}) - m_{\lfloor \frac{o-1}{C^{t-1}} \rfloor \bmod C} \right)^2. \tag{10}$$

D. Remarks on the ambiguity risk

An additional concern is that the unambiguity condition defined by Eq.(2) might fail for the new verifier in Algo.2. In particular, it is possible that $\texttt{Verify}^{\text{NM}}$ recognizes an independent DNN as the owner's possession since it allows more DNNs to pass the ownership examination than `Verify`. To address this issue, we define the following metric.

Definition 3. *(LFEA-distance) Let $\mathbf{W}_1$, $\mathbf{W}_2$ be two $O \times I$ matrices, their LFEA-distance is defined as:*

$$d_{LFEA}(\mathbf{W}_1, \mathbf{W}_2) = \min_{\phi_1 \in \Phi_O^+, \phi_2 \in \Phi_I^+} \|\phi_1 \mathbf{W}_1 \phi_2 - \mathbf{W}_2\|,$$

where $\|\cdot\|$ is any matrix norm.

The LFEA-distance between two parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ measures how similar they can be after applying LFEA (we remark that the third line in Algo.2 is precisely an LFEA). If $\mathbf{W}_1$ and $\mathbf{W}_2$ belong to two independent DNNs but their LFEA-distance is small, they could be recognized as identical by $\texttt{Verify}^{\text{NM}}$ and resulting in an ambiguity. Since Φ_O^+ is a closed group, we have the following result.

Theorem 5. *If* $\mathbf{W}_1, \mathbf{W}_2 \in \Phi_O^+$ *then* $d_{LFEA}(\mathbf{W}_1, \mathbf{W}_2) = 0$.

Proof. Let $\phi_2 = \mathbf{I}_O$ and $\phi_1 = \mathbf{W}_2\mathbf{W}_1^{-1} \in \Phi_O^+$. □

For general cases, the LFEA-distance between two matrices can be bounded as follows.

Theorem 6. *Under matrix norm* $\|\cdot\|_1$ *or* $\|\cdot\|_\infty$, $O \leq I$,

$$d_{LFEA}(\mathbf{W}_1, \mathbf{W}_2) \leq \|\Delta_1\| \cdot \frac{\max(\mathbf{W}_2)}{\max(\mathbf{W}_1)} + \|\Delta_2\|, \tag{11}$$

in which $\max(\mathbf{W}_i)$ *is the maximal element in* $\mathbf{W}_i$ *and*

$$\|\Delta_i\| = \min_{\mathbf{P}_i, \phi \in \Phi_O^+} \|\mathbf{W}_i\mathbf{P}_i - \phi\|,$$

where $\mathbf{P}_i$ *is an* $I \times O$ *matrix where each column contains one and only one unity entry and each row contains no more than one unity entry.*

Proof. According to Theorem 5, if $\mathbf{W}_1$ and $\mathbf{W}_2$ are close to elements in Φ^+ then their LFEA distance is small. Therefore, an upper bound of their LFEA distance can be derived from their projections in Φ^+. An approximate projection of $\mathbf{W}_i$ onto Φ_O^+ (recall that we have assumed $O \leq I$) is obtained by consecutively locating the maximal positive entry in $\mathbf{W}_i$, delete the elements on the corresponding row and column until its rows are depleted. Let $\mathbf{P}_i$ be defined as above to select O columns out of $\mathbf{W}_i$ so $\mathbf{W}_i\mathbf{P}_i$ is an $O \times O$ matrix, then $\hat{\mathbf{W}}_i = \min_{\phi \in \Phi_O^+} \|\mathbf{W}_i - \phi\mathbf{P}_i\|$ is $\mathbf{W}_i$'s projection in Φ_O^+, the residual is $\Delta_i = \mathbf{W}_i - \hat{\mathbf{W}}_i\mathbf{P}_i$.

The LFEA distance can now be bounded as follows, where we use the elementary properties of matrix norms.

$$\begin{aligned} d_{\text{LFEA}}(\mathbf{W}_1, \mathbf{W}_2) &\leq \|\hat{\mathbf{W}}_2\hat{\mathbf{W}}_1^{-1}\mathbf{W}_1 - \mathbf{W}_2\| \\ &= \|\hat{\mathbf{W}}_2\hat{\mathbf{W}}_1^{-1}(\hat{\mathbf{W}}_1\mathbf{P}_1 + \Delta_1) - \hat{\mathbf{W}}_2\mathbf{P}_2 + \Delta_2\| \\ &\leq \|\hat{\mathbf{W}}_2\hat{\mathbf{W}}_1^{-1}\Delta_1 - \Delta_2\| \\ &\leq \|\hat{\mathbf{W}}_2\hat{\mathbf{W}}_1^{-1}\Delta_1\| + \|\Delta_2\| \\ &\leq \|\Delta_1\| \cdot \|\hat{\mathbf{W}}_1^{-1}\| \cdot \|\hat{\mathbf{W}}_2\| + \|\Delta_2\|. \end{aligned}$$

Plugging in the definition of $\|\cdot\|_1$ or $\|\cdot\|_\infty$ yields Eq.(11). □

A corollary from Theorem 6 is that if $\mathbf{W}_1$ and $\mathbf{W}_2$ are extremely sparse and can be transformed into diagonally dominant matrices under row/column permutation then their d_{LFEA} tends to be very small, leading to potential ambiguity. However, when the entries in $\mathbf{W}_1$ or $\mathbf{W}_2$ are distributed uniformly then it is unlikely that `NeuronMap` would result in confusion. Therefore, the additional risk caused by incorporating `NeuronMap` as a standard preprocessing step is limited and is outweighed by its merits. An empirical examination of this argument is given in Sec.V-D.

E. Remarks on the compatibility with ownership verification protocols

`NeuronMap` does not interfere with the training or watermarking of the DNN, this preserving the security properties of all established white-box DNN watermarking schemes. Instead, `NeuronMap` operates on top of an ownership verification protocol as shown in Fig.2. The auxiliary evidence $\{\mathbf{T}, \mathbf{Y}\}$ is transmitted along with the original ownership evidence $\{\texttt{Key}, \texttt{Verify}\}$ to enable the judge to cancel potential LFEAs from the suspect DNN. For compatibility with `NeuronMap`, it is necessary that the underlying ownership verification protocol allows for a secure channel between the owner and the verifier, which is typically assumed to be possible for white-box DNN watermarking schemes.

V. Experiments and Discussions

To empirically investigate LFEA and `NeuronMap`, we evaluated the performance of watermarking schemes under three cases and organized the results as shown in Table II.

TABLE II
CASES TO BE INVESTIGATED AND THE ROADMAP OF SEC.V.

Threat / Watermarking	Ordinary removal attacks (tuning, pruning, etc.)	LFEA and hybrid attacks
Existing watermarking schemes	Has been extensively studied.	Sec.V-B Table V,VI,VIII
Existing watermarking schemes +`NeuronMap`	Sec.V-D	Sec.V-C Table V,VII,VIII

A. Settings

`NeuronMap` is compatible with almost all white-box DNN watermarking schemes. In this paper, we chose to eveluate `Neuronmap` with five state-of-the-art watermarking schemes. The notations used in this section are summarized in Table III for clarity.

TABLE III
THE SUMMARY OF NOTATIONS USED DURING EXPERIMENTS.

Notation	Meaning
$\{(\mathbf{t}, l)\}$	The collection of response triggers.
$\mathbf{W}$	The watermark parameter from the suspect DNN.
$\mathbf{Y}$	The watermark response from the suspect DNN.
$\mathbf{Y}_n$	The watermark response for the n-th trigger.
$\tilde{\mathbf{W}}$	The parameter evidence provided by the owner.
$\tilde{\mathbf{Y}}$	The response evidence provided by the owner.
$\mathbf{M}$	The matrix encoding the ownership information.
$\sigma(\cdot)$	Entrywise filter/step function, $\mathbb{R} \to \{0, 1\}$.
$\mathcal{L}$	The watermark verification loss.

`Uchida's(U)` scheme selects parameters from a DNN and generates a target binary vector $\mathbf{b}$ with N entries, together with a matrix $\mathbf{M}$ as the ownership evidence. To watermark a DNN, the parameter of interest is tuned so that after transformed by $\mathbf{M}$ and a step function $\sigma(\cdot)$, the number of different bits between $\sigma(\mathbf{W} \cdot \mathbf{M})$ and the target vector $\mathbf{b}$ is minimized. To prove the ownership, `Uchida's` provides the

TABLE IV
DNNS FOR EVALUATION. FC AND CV DENOTE FULLY-CONNECTED LAYER AND CONVOLUTIONAL LAYER RESPECTIVELY.

DNN	Size (KB)	Pretrained	Dataset	Task	Layer one ($L1$)	Layer two ($L2$)
Autoencoder [46]	2,818	No	CIFAR10 [47]	Image reconstruction	The first FC (328 neurons)	The second FC (75 neurons)
LeNet [48]	245	No	MNIST [49]	Image classification	The second CV (16 neurons)	The third CV (120 neurons)
ResNet-34 [45]	83,267	No	CIFAR10 [47]	Image classification	The fourth CV (128 neurons)	The seventh CV (256 neurons)
Transformer [50]	50,891	No	Wiki2 [51]	Language modeling	The second FC (200 neurons)	The third FC (200 neurons)
ResNet-101 [45]	171,436	Yes	ImageNet [52]	Image classification	The twenty-second CV (512 neurons)	The seventy-sixth CV (1,024 neurons)
Roberta-Large [53]	1,109,856	Yes	SST2 [54]	Sentiment analysis	The tenth FC (768 neurons)	The twenty-first FC (3,072 neurons)

matrix and the target vector, the loss is l_0 norm, $\oplus$ denotes the entrywise XOR operator.

$$\begin{cases} \texttt{Key} = (\mathbf{M}, \mathbf{b}), \\ \mathcal{L}_{\texttt{U}} = \dfrac{\|\sigma(\mathbf{W} \cdot \mathbf{M}) \oplus \mathbf{b}\|_0}{N}. \end{cases}$$

`MTLSign(MS)` establishes the watermark as an additional task for the watermarked DNN's. It encodes the owner's information into $N = 400$ pseudorandom triggers with binary labels $\{(\mathbf{t}_n, l_n)\}_{n=1}^{N}$ and then trains a classification backend g that takes the intermediate response from the DNN as its input. The loss is binary classification error rate.

$$\begin{cases} \texttt{Key} = \left(\{(\mathbf{t}_n, l_n)\}_{n=1}^{N}, g\right), \\ \mathcal{L}_{\texttt{MS}} = \dfrac{\sum_{n=1}^{N} \mathbb{I}[g(\mathbf{Y}_n) \neq l_n]}{N}. \end{cases}$$

`DeepSign(DS)` chooses a series of $N = 100$ response triggers from a category's outliers. It encodes the owner's signature into a matrix $\tilde{\mathbf{Y}}$ with the same shape as the response design matrix. To inject the watermark, the DNN is tuned so that the entry-wise multiplication between the response matrix and the design matrix ends up as another binary matrix $\mathbf{M}$ after filtering. Ownership verification loss is computed as the proportion of entries that are consistent with the evidence.

$$\begin{cases} \texttt{Key} = \left(\{(\mathbf{t}_n)\}_{n=1}^{N}, \tilde{\mathbf{Y}}, \mathbf{M}\right), \\ \mathcal{L}_{\texttt{DS}} = \dfrac{\|\sigma(\mathbf{Y} \cdot \tilde{\mathbf{Y}}) \oplus \mathbf{M}\|_0}{\|\mathbf{M}\|}. \end{cases}$$

`DeepJudge(DJ-1)(DJ-2)` generates $N = 100$ response triggers by adversarially maximizing the distance between different triggers' response patterns. `DeepJudge-1` measures the l_2 distance between two responses,

$$\begin{cases} \texttt{Key} = \left(\{(\mathbf{t}_n)\}_{n=1}^{N}, \tilde{\mathbf{Y}}\right), \\ \mathcal{L}_{\texttt{DJ-1}} = \|\mathbf{Y} - \tilde{\mathbf{Y}}\|_2. \end{cases}$$

while `DeepJudge-2` measures the neuron's activation patterns (a neuron is activated for a given trigger if its response is larger than a threshold) w.r.t. l_0 loss.

$$\begin{cases} \texttt{Key} = \left(\{(\mathbf{t}_n)\}_{n=1}^{N}, \tilde{\mathbf{Y}}\right), \\ \mathcal{L}_{\texttt{DJ-2}} = \dfrac{\|\sigma(\mathbf{Y}) \oplus \sigma(\tilde{\mathbf{Y}})\|_0}{\|\mathbf{Y}\|}. \end{cases}$$

For all schemes, `Verify` takes the evidence from `Key`, retrieves $\mathbf{M}$ and optionally $\mathbf{Y}$ from the suspect DNN, and computes the loss. It returns `Pass` only if the loss is lower than a scheme-dependent threshold, otherwise, it returns `Fail`. To comprehensively study the accuracy of the watermarking schemes, we viewed the watermarked DNNs as positive samples, and other independent DNNs with the same structure as negative samples. We then computed the False Positive Rate (FPR)

$$\text{FPR} = \frac{|\{\text{DNN}_{\text{ind}} : \texttt{Verify}(\text{DNN}_{\text{ind}}, \texttt{Key}) = \texttt{Pass}\}|}{|\{\text{DNN}_{\text{ind}}\}|},$$

and True Positive Rate (TPR)

$$\text{TPR} = \frac{|\{\text{DNN}_{\text{WM}} : \texttt{Verify}(\text{DNN}_{\text{WM}}, \texttt{Key}) = \texttt{Pass}\}|}{|\{\text{DNN}_{\text{WM}}\}|},$$

under different thresholds, and plotted the Receiver Operating Characteristic (ROC) curve. Finally, the performance of the watermarking scheme is evaluated by the Area Under ROC Curve (AUC).

Six DNNs were considered as models to be protected, with details given in Table IV. The first four DNNs were trained from scratch, while the last two are pre-trained large DNNs. ResNet-101[1] is pre-trained on ImageNet and Roberta-Large[2] is pre-trained on 160GB of texts. Both models have been used as the backbone models of many real-world applications including objection detection [55], semantic segmentation [56], video analysis [57], cross-lingual sentiment analysis [58], and knowledge infusion [59]. ResNet-101 and Roberta-Large were locally fine-tuned on a 10% subset of ImageNet [52] and SST2 [54] to simulate DNN service in the field. Without loss of generality, each white-box watermarking scheme took two random layers from each DNN as their inputs denoted by $L1$ and $L2$, i.e., either the parameter $\mathbf{W}$ or the response $\mathbf{Y}$ is the concatenation of two separate parts. We used four GeForce RTX 2080 Ti GPUs for acceleration, and all experiments are implemented using the `PyTorch` framework[3].

B. The efficacy of LFEA

The baseline results of ownership verification of all white-box DNN watermarking schemes are presented in Table V. Twenty watermarked DNNs and another twenty independent DNNs were trained for each DNN structure, and the watermark verification losses were recorded and used to compute the AUCs for `Verify` and $\texttt{Verify}^{\text{NM}}$. It is observed that: (i) LFEA succeeded in confusing the ownership verifier in most cases, especially when it was applied to both watermark layers. The highest AUC was only 0.59 in these cases. (ii) The more

[1]https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet101.html
[2]https://pytorch.org/text/main/models.html#roberta-large-encoder
[3]Codes will be available in https://github.com/TemporaryUserNo7/LFEA.

TABLE V

AUCS FOR VERIFY AND VERIFY$^{\text{NM}}$ UNDER DIFFERENT SETTINGS. $\emptyset$ MEANS NO LFEA. $L1$, $L2$, $L1$+$L2$ (MARKED IN SHADOW) DENOTES THE LOCATION WHERE LFEA WAS APPLIED. AUC WAS MEASURED W.R.T. DNN_{WM} AFFECTED BY LFEA VS. DNN_{IND}.

Autoencoder

Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.59	0.77	0.51	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.59	0.70	0.41	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.66	0.85	0.40	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.00	0.03	0.00	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.75	0.77	0.54	1.00	1.00	1.00	1.00

LeNet

Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.71	0.61	0.50	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.65	0.60	0.55	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.84	0.71	0.51	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.80	0.70	0.59	1.00	1.00	1.00	1.00

ResNet-34

Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.71	0.55	0.50	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.63	0.54	0.50	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.78	0.57	0.52	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.20	0.12	0.05	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.63	0.54	0.51	1.00	1.00	1.00	1.00

Transformer

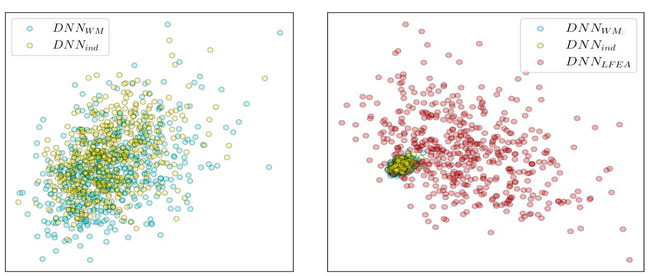
Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.64	0.57	0.54	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.77	0.67	0.54	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.65	0.60	0.51	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.69	0.57	0.50	1.00	1.00	1.00	1.00

ResNet-101

Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.60	0.54	0.48	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.60	0.55	0.50	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.68	0.60	0.54	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.22	0.07	0.00	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.69	0.58	0.50	1.00	1.00	1.00	1.00

Roberta-Large

Scheme	Verify				Verify$^{\text{NM}}$			
	$\emptyset$	$L1$	$L2$	$L1$+$L2$	$\emptyset$	$L1$	$L2$	$L1$+$L2$
`Uchida's`	1.00	0.66	0.54	0.50	1.00	1.00	1.00	1.00
`MTLSign`	1.00	0.68	0.61	0.52	1.00	1.00	1.00	1.00
`DeepSign`	1.00	0.67	0.55	0.48	1.00	1.00	1.00	1.00
`DeepJudge-1`	1.00	0.03	0.00	0.00	1.00	1.00	1.00	1.00
`DeepJudge-2`	1.00	0.72	0.67	0.53	1.00	1.00	1.00	1.00



(a) Original response distributions. (b) Response distributions after LFEA.

Fig. 7. Distributions of the response from Autoencoder's $L2$, visualized after reducing the dimensionality to two using principal component analysis. DNN_{LFEA} is DNN_{WM} after undertaking LFEA.

neurons LFEA interfered with, the more damage it caused. Since LFEA can be applied to all layers within a DNN, the threat turns out to be severe. (iii) For l_2 based verifiers (e.g., `DeepJudge-1`), the AUC dropped below 0.5 after LFEA, indicating that the loss $\mathcal{L}_{\texttt{DJ-1}}$ was significantly larger than that of the independent models without LFEA. Recall that LFEA applies a linear transform to the parameters/responses, which has a bounded impact for loss functions taking the form of binary classification error (i.e., loss functions except for $\mathcal{L}_{\texttt{DJ-1}}$). For binary classification, even if LFEA leads to a random guess, or an all zero/one guess, the loss remains approximately 50%, the same as that produced from an independent DNN. On the other hand, $\mathcal{L}_{\texttt{DJ-1}}$ can grow arbitrarily large if $\mathbf{Y}$ is multiplied by an appropriate linear factor, so the loss after LFEA can also grow arbitrarily large. As a result, the loss can be larger than that computed w.r.t. an independent DNN, causing the AUC to drop to zero. An instance of the response's transformation is visualized in Fig.7. As the distributions varied significantly, the original decision boundaries drawn by watermarking verifiers are no longer valid. For `DeepJudge-1`, setting a threshold to identify DNNs that have been subjected to LFEA is trivial, yet this provides no evidence of ownership.

For comparisons, we applied Neuron Pruning (NP) [60] and Fine Pruning (FP) [61] as exemplary removal attacks against DNN watermarks, whose damage is larger than ordinary fine-tuning (FT) [62] with the training dataset. LFEA was applied to both $L1$ and $L2$ within the DNN. NP randomly set a portion of parameters to zero, while FP pruned DNN_{WM} first and then fine-tuned the pruned DNN for twenty epochs on the original dataset. All attacks were terminated when the ownership verification AUC w.r.t. all watermarking schemes dropped below 0.6. The respective time consumption and impact on the attacked DNN are summarized in Table VI. The results show that compared to other adversarial tuning methods, LFEA is cheap, data-free, and has no influence on the DNN's performance. Therefore, LFEA can also be applied in conjunction with other removal attacks.

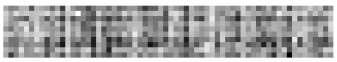
C. The efficacy of `NeuronMap`

Although LFEA succeeded in damaging all studied white-box watermarking schemes, its damage was completely neutralized by `NeuronMap` as shown in Table V. We examined all three types of triggers and $T = \{10, 20, 50, 100\}$, the AUCs of `Verify`$^{\text{NM}}$ stayed at 1.0 since the responses were always perfectly recovered. For DNN_{ind}, applying `NeuronMap` did not increase the false positive rate, and the AUC for `Verify`$^{\text{NM}}$ remained uniformly 1.0. Note that for response-based watermarking schemes, applying `NeuronMap` on the module of $L1$ and $L2$ is sufficient. However, for parameter-based watermarking schemes, `NeuronMap` has to be repeated for the module before $L1$ and $L2$ as well to cancel potential

TABLE VI
THE EVALUATION OF ATTACKS AGAINST WHITE-BOX WATERMARKING SCHEMES. NP AND FP WERE CONDUCTED WHEN ALL WATERMARK VERIFIERS' AUC DROPPED UNDER 0.6. AUTOENCODER AND TRANSFORMER WERE EVALUATED BY RECONSTRUCTION LOSS. LENET, RESNET-34, RESNET-101, AND ROBERTA-LARGE WERE EVALUATED BY CLASSIFICATION ACCURACY (%).

Attack	Autoencoder		LeNet		ResNet-34	
	Time (sec)	**Performance drop**	**Time** (sec)	**Performance drop**	**Time** (sec)	**Performance drop**
LFEA	3.87 ± 0.04	0.0 ± 0.0	3.27 ± 0.86	0.0 ± 0.0	11.00 ± 0.38	0.0 ± 0.0
NP	8.78 ± 0.18	0.80 ± 0.01	48.37 ± 11.41	54.10 ± 6.29	17.21 ± 0.40	55.41 ± 6.75
FP	29.37 ± 0.19	0.04 ± 0.00	68.91 ± 11.20	1.73 ± 0.26	418.49 ± 0.38	3.57 ± 0.17

Attack	Transformer		ResNet-101		Roberta-Large	
	Time (sec)	**Performance drop**	**Time** (sec)	**Performance drop**	**Time** (sec)	**Performance drop**
LFEA	2.42 ± 0.14	0.0 ± 0.0	42.58 ± 0.61	0.0 ± 0.0	103.40 ± 0.91	0.0 ± 0.0
NP	11.02 ± 0.48	2.03 ± 0.02	193.10 ± 26.31	39.30 ± 7.24	362.70 ± 17.44	11.19 ± 3.81
FP	87.90 ± 8.51	0.33 ± 0.07	2859.20 ± 125.10	3.28 ± 0.52	1685.40 ± 205.00	2.91 ± 0.50


(a) Response from triggers $\mathcal{D}$, before FT.


(b) Response from triggers $\mathcal{D}$, after FT.

(c) Response difference=10×((a)-(b)).


(d) Response from triggers $\mathcal{R}$, before FT.


(e) Response from triggers $\mathcal{R}$, after FT.

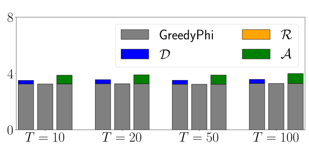
(f) Response difference=10×((d)-(e)).


(g) Response from triggers $\mathcal{A}$, before FT.

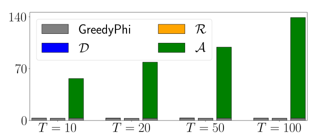

(h) Response from triggers $\mathcal{A}$, after FT.


(i) Response difference=10×((g)-(h)).

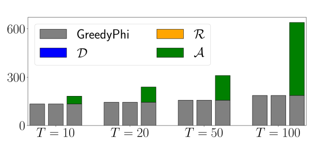
Fig. 8. Heatmaps of responses from Autoencoder's $L1$, first 64 neurons, $T = 10$. In each heatmap, a row corresponds to a trigger and each column represents a neuron.



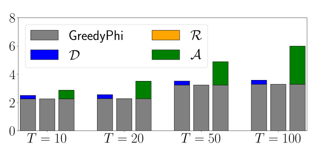
(a) Autoencoder.



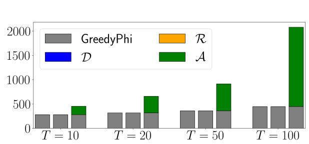
(b) LeNet.



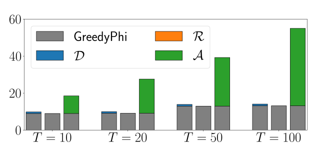
(c) ResNet-34.



(d) Transformer.



(e) ResNet-101.



(f) Roberta-Large.

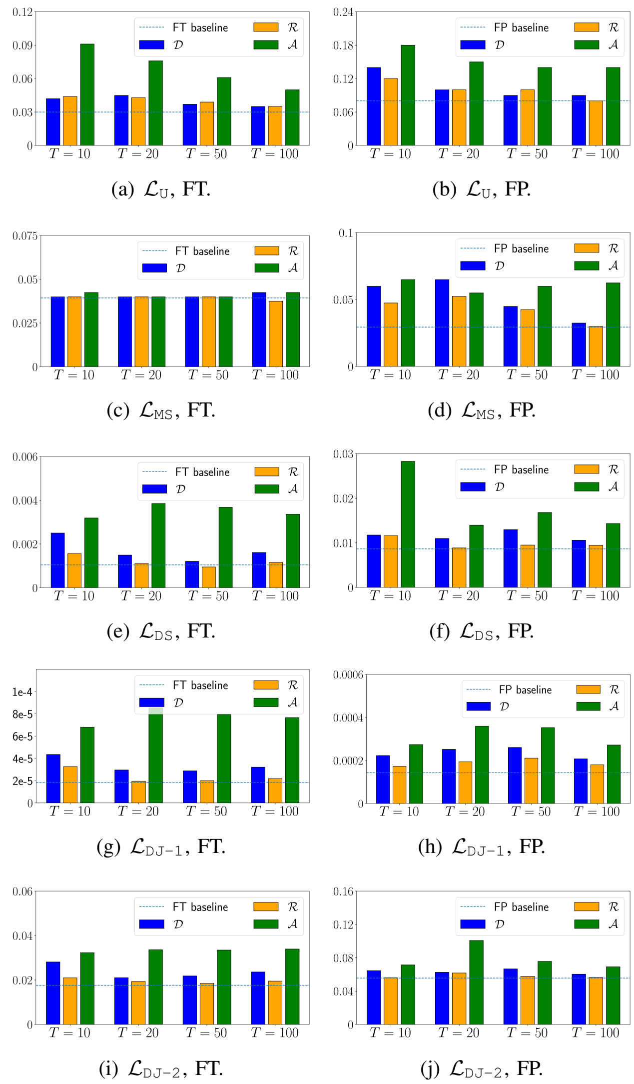
Fig. 9. The time consumption (sec) of applying `NeuronMap`.

change in the weight's column space, which doubles the time consumption. In both cases, it is unnecessary to calibrate all modules within the DNN (which is possible since the input layer is inherently intact and applying `NeuronMap` consecutively to all modules can cancel all potential LFEAs), since the watermark verifier is only interested in $L1$'s and $L2$'s responses.

Adversaries can launch a hybrid attack that combines adversarial tuning and LFEA. Once the response matrix has been perturbed, the recovery by `NeuronMap` might not be perfect. In particular, it is necessary to consider an adversary conducting a hybrid attack by first applying FT/FP and then LFEA to the pirated DNN. FT and FP have a smaller impact than NP on the DNN's performance and are universal tuning attacks, and the tuning hyperparameters are also available to the adversary given sufficient data [63]. The adversary is not encouraged to apply FT or FP after conducting LFEA, since LFEA significantly changes the distribution of both parameters and responses, the original regularizers and hyperparameter configurations are no longer applicable.

1) Configuration studies: To study the defensive capability under hybrid attacks, we firstly tested the configuration of $T \in \{10, 20, 50, 100\}$ for triggers $\mathcal{D}$, $\mathcal{R}$, and $\mathcal{A}$. Triggers of type $\mathcal{A}$ were generated by running fuzzy clustering and optimizing Eq.(10) using another gradient-descent optimizer.

For illustration, part of the response matrices for three kinds of triggers in $L1$ of the Autoencoder before and after twenty epochs of FT is visualized in Fig.8 (for $\mathcal{A}$, we adopted $C = 2$ centroids to maximize the distinguishability). Visually, the deviations for $\mathcal{A}$ triggers were larger than $\mathcal{D}$ and $\mathcal{R}$. Meanwhile, the cost of producing attack triggers became prohibitive for complex DNNs and larger Ts. The time consumption of generating `NeuronMap` triggers and running `GreedyPhi` for different settings is provided in Fig.9, from which we observed that the expense in generating attack triggers exceeded that of running `GreedyPhi` in most cases, while the expense in generating the other two types of triggers was negligible.



(a) $\mathcal{L}_{\mathrm{U}}$, FT. (b) $\mathcal{L}_{\mathrm{U}}$, FP.

(c) $\mathcal{L}_{\mathrm{MS}}$, FT. (d) $\mathcal{L}_{\mathrm{MS}}$, FP.

(e) $\mathcal{L}_{\mathrm{DS}}$, FT. (f) $\mathcal{L}_{\mathrm{DS}}$, FP.

(g) $\mathcal{L}_{\mathrm{DJ-1}}$, FT. (h) $\mathcal{L}_{\mathrm{DJ-1}}$, FP.

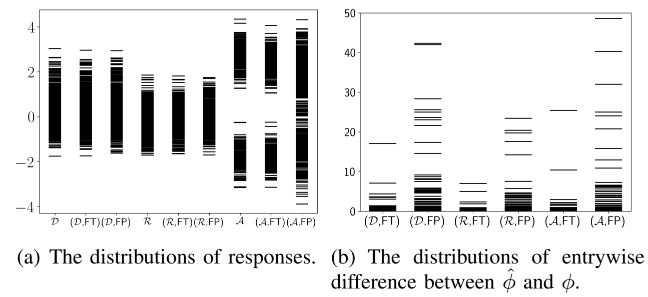
(i) $\mathcal{L}_{\mathrm{DJ-2}}$, FT. (j) $\mathcal{L}_{\mathrm{DJ-2}}$, FP.

Fig. 10. Verification losses under FT/FP+LFEA and `NeuronMap` for ResNet-34. The loss under vanilla FT/FP is marked in dashed lines.

Intuitively, a larger T means more information for inferring ϕ in LFEA. As shown in Fig.10, the verification loss generally declined for a larger T. We observed that the calibration efficacy of trigger $\mathcal{A}$ was uniformly worse than the other options. This observation, combined with Fig.9, indicates that the optimal configuration for `NeuronMap` is a large T with cheaper triggers $\mathcal{R}$.

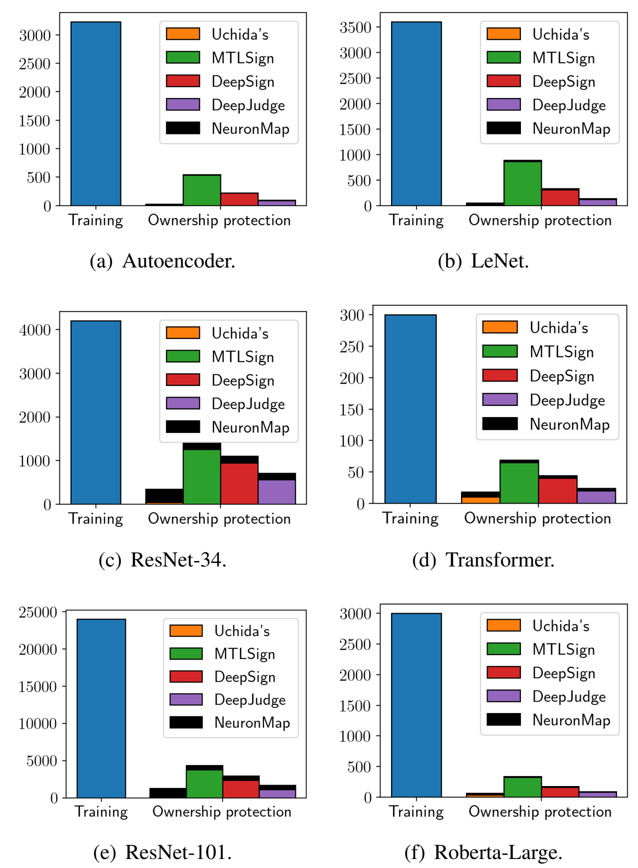
The failure of attack triggers can be attributed to the deviation of its responses after tuning as shown in Fig.11(a). Attack triggers assign extremely large/small output responses to neurons to increase distinguishability, but these responses are more vulnerable under tuning. Consequently, the estimation of ϕ with $\mathcal{A}$ triggers contained more outliners as illustrated in Fig.11(b) so the calibration is worse.

2) `NeuronMap` against hybrid attacks: The variations of watermark verification losses after FT/FP, LFEA, and `NeuronMap` with $\mathcal{R}$ triggers and $T = 100$ are detailed in Table VII, where we listed the increment in verification losses after applying FT/FP, after applying FT/FP+LFEA+`NeuronMap`, and the marginal loss increment due to LFEA under FT/FP. The deviation in the response matrices as shown in Fig.8 disturbed `NeuronMap` and caused deviations in loss, yet they were small compared to the damage of applying FT/FP.



(a) The distributions of responses. (b) The distributions of entrywise difference between $\hat{\phi}$ and ϕ.

Fig. 11. The distributions of responses w.r.t. `NeuronMap` triggers before/after FT/FP. And the distributions of the difference between ϕ in LFEA and the estimation of `GreedyPhi` under different types of triggers. The setting is $L2$ in Autoencoder, $T = 100$.



(a) Autoencoder. (b) LeNet.

(c) ResNet-34. (d) Transformer.

(e) ResNet-101. (f) Roberta-Large.

Fig. 12. The time consumption (sec) of training and ownership protection.

In several cases, the marginal loss variation was negative, so the effect of tuning was partially canceled (e.g., tuning might amplify the output of a certain neuron and misguide the verifier) after mapping neurons. Neither type of loss variation was significant enough to confuse the ownership verifier, as justified by AUCs listed in Table VIII. Therefore, we concluded that `NeuronMap` can correctly defend the DNN watermark against hybrid attacks.

We measured the time consumption of `NeuronMap` as an

TABLE VII
THE VERIFICATION LOSS INCREMENT UNDER NEURONMAP AFTER UNDERTAKING HYBRID ATTACKS. FT/FP MARGINAL DENOTES THE RELATIVE INCREMENT OF THE VERIFICATION LOSS AFTER LFEA AND NEURONMAP COMPARED WITH FT/FP ONLY.

Threat	Autoencoder				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	2.9E-2	2.5E-3	7.4E-3	8.4E-5	5.8E-2
FT+LFEA	3.1E-2	2.5E-3	7.2E-3	7.8E-5	5.8E-2
FT Marginal	6.9%	0.0%	-2.7%	-7.6%	-1.0%
FP	4.4E-2	1.0E-2	3.4E-2	3.5E-4	1.2E-1
FP+LFEA	5.1E-2	7.5E-3	3.4E-2	3.5E-4	1.2E-1
FP Marginal	15.9%	-25.0%	-0.9%	-0.0%	-0.0%

Threat	LeNet				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	5.5E-2	0.0E+0	2.9E-3	5.7E-5	3.2E-2
FT+LFEA	5.4E-2	0.0E+0	1.0E-3	2.3E-5	2.1E-2
FT Marginal	-1.8%	0.0%	-65.5%	-59.6%	-34.4%
FP	6.4E-2	1.3E-2	2.6E-2	3.6E-4	9.7E-2
FP+LFEA	6.4E-2	1.5E-2	1.2E-2	2.2E-4	6.4E-2
FP Marginal	0.0%	2.5%	-53.8%	-38.9%	-35.1%

Threat	ResNet-34				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	3.7E-2	3.1E-2	3.5E-2	8.0E-4	1.3E-1
FT+LFEA	4.0E-2	3.6E-2	3.3E-2	6.0E-4	1.3E-1
FT Marginal	8.1%	16.1%	3.1%	-26.3%	-0.0%
FP	5.4E-2	5.6E-2	3.9E-2	9.7E-4	1.4E-1
FP+LFEA	5.7E-2	6.4E-2	3.8E-2	7.3E-4	1.4E-1
FP Marginal	3.7%	15.2%	-2.8%	-24.7%	-0.1%

Threat	Transformer				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	2.6E-2	1.0E-2	1.4E-1	9.1E-3	1.6E-1
FT+LFEA	2.4E-2	1.0E-2	1.4E-1	7.7E-3	1.5E-1
FT Marginal	-7.7%	0.0%	-0.0%	-15.4%	-0.8%
FP	4.6E-2	0.0E+0	1.4E-1	9.2E-3	1.8E-1
FP+LFEA	4.3E-2	0.0E+0	1.5E-1	7.9E-3	1.8E-1
FP Marginal	-6.5%	0.0%	0.7%	-13.4%	-0.3%

Threat	ResNet-101				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	5.0E-2	3.9E-2	3.8E-2	1.1E-3	1.8E-1
FT+LFEA	5.2E-2	3.8E-2	3.3E-2	1.0E-3	1.7E-1
FT Marginal	4.0%	33.3%	0.0%	-9.1%	-5.6%
FP	8.4E-2	7.5E-2	4.5E-2	8.9E-3	2.4E-1
FP+LFEA	9.7E-2	4.6E-2	3.8E-2	9.0E-3	2.4E-1
FP Marginal	15.5%	2.6%	2.2%	1.1%	0.0%

Threat	Roberta-Large				
	$\Delta\mathcal{L}_{\text{U}}$	$\Delta\mathcal{L}_{\text{MS}}$	$\Delta\mathcal{L}_{\text{DS}}$	$\Delta\mathcal{L}_{\text{DJ-1}}$	$\Delta\mathcal{L}_{\text{DJ-2}}$
FT	3.4E-2	2.9E-2	8.3E-2	6.5E-4	4.9E-2
FT+LFEA	3.1E-2	3.1E-2	8.4E-2	5.7E-4	5.1E-2
FT Marginal	-8.8%	6.9%	1.2%	-12.3%	4.1%
FP	4.7E-2	3.0E-2	1.0E-1	5.5E-3	8.8E-2
FP+LFEA	4.0E-2	3.4E-2	9.9E-2	6.1E-3	1.1E-1
FP Marginal	-14.9%	13.3%	-9.8%	10.9%	25.0%

TABLE VIII
AUCS FOR VERIFY AND $\text{VERIFY}^{\text{NM}}$ UNDER HYBRID ATTACKS, COMPUTED FROM DNN_{WM} UNDERGOING TUNING/HYBRID ATTACKS VS. DNN_{IND}. ENTRIES AFFECTED BY LFEA ARE MARKED IN SHADOW.

Scheme	Autoencoder							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	0.93	0.51	0.51	1.00	0.94	1.00	0.94
MTLSign	1.00	0.99	0.41	0.39	1.00	0.99	1.00	0.99
DeepSign	0.98	0.96	0.40	0.40	0.98	0.96	0.98	0.96
DeepJudge-1	1.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00
DeepJudge-2	1.00	1.00	0.52	0.50	1.00	1.00	1.00	1.00

Scheme	LeNet							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	0.95	0.52	0.50	1.00	0.95	1.00	0.95
MTLSign	1.00	0.99	0.55	0.55	1.00	0.99	1.00	0.99
DeepSign	1.00	0.98	0.51	0.51	1.00	0.98	1.00	0.98
DeepJudge-1	1.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00
DeepJudge-2	1.00	1.00	0.57	0.51	1.00	1.00	1.00	1.00

Scheme	ResNet-34							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00
MTLSign	1.00	1.00	0.52	0.52	1.00	1.00	1.00	1.00
DeepSign	1.00	0.98	0.50	0.52	1.00	0.98	1.00	0.98
DeepJudge-1	1.00	1.00	0.05	0.02	1.00	1.00	1.00	1.00
DeepJudge-2	1.00	0.94	0.51	0.51	1.00	0.94	1.00	0.94

Scheme	Transformer							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	0.95	0.54	0.51	1.00	0.95	1.00	0.95
MTLSign	1.00	1.00	0.54	0.51	1.00	1.00	1.00	1.00
DeepSign	0.98	0.96	0.51	0.50	0.98	0.96	0.98	0.96
DeepJudge-1	1.00	0.95	0.00	0.00	1.00	0.95	1.00	0.95
DeepJudge-2	1.00	0.98	0.50	0.50	1.00	0.98	1.00	0.98

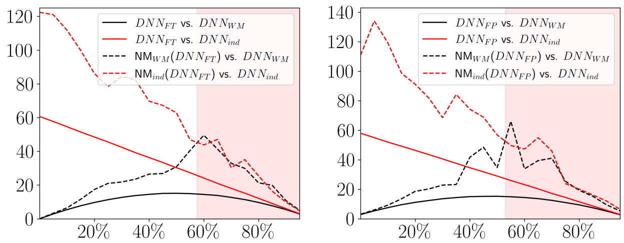
Scheme	ResNet-101							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	1.00	0.51	0.50	1.00	1.00	1.00	1.00
MTLSign	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00
DeepSign	1.00	1.00	0.52	0.50	1.00	0.99	1.00	0.99
DeepJudge-1	1.00	1.00	0.03	0.03	1.00	1.00	1.00	1.00
DeepJudge-2	1.00	0.99	0.51	0.50	1.00	0.97	1.00	0.97

Scheme	Roberta-Large							
	Verify				$\text{Verify}^{\text{NM}}$			
	FT	FP	FT+LFEA	FP+LFEA	FT	FP	FT+LFEA	FP+LFEA
Uchida's	1.00	0.98	0.51	0.50	1.00	0.98	1.00	0.98
MTLSign	1.00	1.00	0.52	0.52	1.00	1.00	1.00	1.00
DeepSign	1.00	0.99	0.52	0.50	1.00	0.99	0.98	0.99
DeepJudge-1	1.00	0.98	0.00	0.00	1.00	0.98	1.00	0.98
DeepJudge-2	1.00	0.99	0.52	0.50	1.00	0.99	1.00	0.99

end-to-end watermark preprocessing module and compared it to the cost of model training and watermark injection, and the results are shown in Fig.12. NeuronMap was relatively inexpensive under all settings.

D. Revisiting the ambiguity risk

In Sec.IV-D, we addressed the ambiguity concern, and Table V and Table VIII have shown that $\text{Verify}^{\text{NM}}$ does not reduce the AUC, i.e., applying NeuronMap on an independent DNN using the watermarked DNN's response pattern yields no extra ambiguity risk. This is consistent with Theorem 6, which predicts that the LFEA distance between dense weighting matrices is likely to be large. However, Theorem 6 also suggests that if the parameter matrices become sparse, the LFEA distance between different DNNs' parameters is reduced, and there is a risk that after NeuronMap, independent DNNs are recognized as identical, especially by parameter-based watermarking schemes. We demonstrated this phenomenon for Autoencoder's $L2$. $\text{DNN}_{\text{FT,FP}}$ denotes the fine-tuned/fine-pruned version of DNN_{WM}, which should be verified as identical to DNN_{WM}. We measure the l_2 distance between different DNN's weights in $L2$ under different pruning rates, and the results are shown in Fig.13, where NM_{WM}/NM_{ind} denotes running NeuronMap whose response matrix is provided



(a) Under FT. (b) Under FP.

Fig. 13. The l_2 difference between weights' from different pairs of Autoencoder's $L2$ in sparse settings, using $T = 100$ random triggers. The pruning rates under which the distances between homologous DNNs and independent DNNs became inseparable were highlighted by shadow.

by $\mathrm{DNN}_{\mathrm{WM}}/\mathrm{DNN}_{\mathrm{ind}}$. As predicted, when the pruning rate increased and the DNN's weights became sparse, the LFEA distance between independent matrices declined. We remark that the l_2 distance between a DNN's weight and that from another DNN calibrated by `NeuronMap` is an upper bound of their d_{LFEA} by definition. When the pruning rate went above 60%, the distances between weights calibrated w.r.t. $\mathrm{DNN}_{\mathrm{WM}}$ and $\mathrm{DNN}_{\mathrm{ind}}$ became indistinguishable. Therefore, it is possible that a verifier incorporating `NeuronMap` cannot judge whether the source of $\mathrm{DNN}_{\mathrm{FT/FP}}$ is $\mathrm{DNN}_{\mathrm{WM}}$ or $\mathrm{DNN}_{\mathrm{ind}}$. This ambiguity risk could potentially breaches DNN copyright integrity.

Neuron pruning up to 60% for all layers is not always a safe operation even for preventing overfitting and reducing computation. For example, when 60% of the parameters of LeNet and ResNet-34 were pruned, their classification accuracy dropped by 55.0% and 86.7%, respectively. Additionally, while pruning can achieve sparsity of parameter, achieving sparsity in the response matrix, especially when the triggers are unknown, is a non-trivial challenge. Thus, we conclude that the additional ambiguity risk of `NeuronMap` remains insignificant in practice. Considering its limited time complexity and the efficacy in canceling the damage caused by LFEA, we recommend that `NeuronMap` be incorporated into white-box watermark verifiers, thereby eliminating LFEA as a threat to DNN copyright.

VI. Conclusions

This paper studies the functionality equivalence attack (FEA) that poses a threat to the ownership integrity established by DNN watermarking schemes. A general family of functionality equivalence attacks, LFEA, is formulated and analyzed. Although LFEA succeeds in invalidating most existing white-box DNN watermarking schemes, we show that it can be neutralized with the `NeuronMap` framework. Extensive experiments justified both the threat from LFEA and the efficacy of the `NeuronMap` defense mechanism. As a result, after incorporating `NeuronMap` as a preprocessing module, most existing DNN watermarking schemes can withstand LFEA or hybrid attacks with minimal impact on in time complexity. Although an adversary with knowledge of the triggers (either those of the watermarking schemes or `NeuronMap`) can plot a non-linear FEA that preserves the DNN's normal functionality while suppressing its response for triggers, such attacks require modifying the network architecture and are restricted to invalidating exposed triggers so their threat is limited. In future work we intend to perform a more comprehensive analysis of universal FEAs and develop new DNN watermarking schemes that are inherently robust against such threats by utilizing FEA-invariant DNN statistics such as the null space of the response matrix.

References

[1] Omid E David, Nathan S Netanyahu, and Lior Wolf, "Deepchess: End-to-end deep neural network for automatic learning in chess," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 88–96.

[2] Hector E. Romero, Ning Ma, Guy J. Brown, and Elizabeth A. Hill, "Acoustic screening for obstructive sleep apnea in home environments based on deep neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 7, pp. 2941–2950, 2022.

[3] Karim Gasmi, Ibtihel Ben Ltaifa, Gaël Lejeune, Hamoud Alshammari, Lassaad Ben Ammar, and Mahmood A Mahmood, "Optimal deep neural network-based model for answering visual medical question," *Cybernetics and Systems*, vol. 53, no. 5, pp. 403–424, 2022.

[4] Jing Yu, Xiaojun Ye, and Hongbo Li, "A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network," *Future Generation Computer Systems*, vol. 129, pp. 399–406, 2022.

[5] Congyuan Xu, Jizhong Shen, and Xin Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3540–3552, 2020.

[6] Yujin Zhang, Luo Yu, Zhijun Fang, Neal N. Xiong, Lijun Zhang, and Haiyue Tian, "An end-to-end deep learning model for robust smooth filtering identification," *Future Generation Computer Systems*, vol. 127, pp. 263–275, 2022.

[7] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," *Proceedings of ICLR 2019*, pp. 1–18, 2019.

[8] Jingjing Zhao, Qingyue Hu, Gaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan, "Afa: Adversarial fingerprinting authentication for deep neural networks," *Computer Communications*, vol. 150, pp. 488–497, 2020.

[9] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 14–25.

[10] 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*, 2017, pp. 269–277.

[11] 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 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 159–172.

[12] Cheng Xiong, Guorui Feng, Xinran Li, Xinpeng Zhang, and Chuan Qin, "Neural network model protection with piracy identification and tampering localization capability," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 2881–2889.

[13] Lixin Fan, Kam Woh Ng, Chee Seng Chan, and Qiang Yang, "Deepip: Deep neural network intellectual property protection with passports," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[14] Xiquan Guan, Huamin Feng, Weiming Zhang, Hang Zhou, Jie Zhang, and Nenghai Yu, "Reversible watermarking in deep convolutional neural networks for integrity authentication," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 2273–2280.

[15] Jie Zhang, Dongdong Chen, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu, "Deep model intellectual property protection via deep watermarking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4005–4020, 2022.

[16] Ryota Namba and Jun Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 228–240.

[17] Fangqi Li, Lei Yang, Shilin Wang, and Liew Alan Wee-Chung, "Leveraging multi-task learning for unambiguous and flexible deep neural network watermarking," *AAAI SafeAI Workshop*, 2021.

[18] Rouhani Bita Darvish, Huili Chen, and Farinaz Koushanfar, "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*, 2019, pp. 485–497.

[19] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song, "Copy, right? a testing framework for copyright protection of deep learning models," in *2022 IEEE Security and Privacy*, 2022, pp. 1–6.

[20] Hanwen Liu, Zhenyu Weng, and Yuesheng Zhu, "Watermarking deep neural networks with greedy residuals," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6978–6988.

[21] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?," in *2021 IEEE Security and Privacy*, 2022, pp. 1–13.

[22] Fang-Qi Li, Shi-Lin Wang, and Yun Zhu, "Fostering the robustness of white-box deep neural network watermarks by neuron alignment," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3049–3053.

[23] Mingfu Xue, Yushu Zhang, Jian Wang, and Weiqiang Liu, "Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 6, pp. 908–923, 2022.

[24] Alaa Fkirin, Gamal Attiya, Ayman El-Sayed, and Marwa A Shouman, "Copyright protection of deep neural network models using digital watermarking: a comparative study," *Multimedia Tools and Applications*, vol. 81, no. 11, pp. 15961–15975, 2022.

[25] Yue Li, Hongxia Wang, and Mauro Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021.

[26] David J. Coumou and Gaurav Sharma, "Insertion, deletion codes with feature-based embedding: A new paradigm for watermark synchronization with applications to speech watermarking," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 153–165, 2008.

[27] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1615–1631.

[28] Tianhao Wang and Florian Kerschbaum, "Riga: Covert and robust white-box watermarking of deep neural networks," in *Proceedings of the Web Conference 2021*, 2021, pp. 993–1004.

[29] Mohammad Mehdi Yadollahi, Farzaneh Shoeleh, Sajjad Dadkhah, and Ali A. Ghorbani, "Robust black-box watermarking for deep neural network using inverse document frequency," in *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2021, pp. 574–581.

[30] Haozhe Chen, Weiming Zhang, Kunlin Liu, Kejiang Chen, Han Fang, and Nenghai Yu, "Speech pattern based black-box model watermarking for automatic speech recognition," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3059–3063.

[31] Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang, "Protecting intellectual property of generative adversarial networks from ambiguity attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3630–3639.

[32] Tianshuo Cong, Xinlei He, and Yang Zhang, "Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders," *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pp. 579–593, 2022.

[33] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX Security Symposium (USENIX Security 21)*. Aug. 2021, pp. 1937–1954, USENIX Association.

[34] Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang, "Secure neural network watermarking protocol against forging attack," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–12, 2020.

[35] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo, "How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of dnn," in *Proceedings of ACSAC*, 2019, pp. 126–137.

[36] Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang, "Fedipr: Ownership verification for federated deep neural network models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[37] Minoru Kuribayashi, Takuro Tanaka, Shunta Suzuki, Tatsuya Yasui, and Nobuo Funabiki, "White-box watermarking scheme for fully-connected layers in fine-tuning model," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 165–170.

[38] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen, "Network morphism," in *International conference on machine learning*. PMLR, 2016, pp. 564–572.

[39] Vinicius Licks and Ramiro Jordan, "Geometric attacks on image watermarking systems," *IEEE multimedia*, vol. 12, no. 3, pp. 68–78, 2005.

[40] Guobiao Li, Sheng Li, Zhenxing Qian, and Xinpeng Zhang, "Encryption resistant deep neural network watermarking," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3064–3068.

[41] Jin Xu, Zishan Li, Bowen Du, Miaomiao Zhang, and Jing Liu, "Reluplex made more practical: Leaky relu," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[42] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, and Javier Ortega-Garcia, "Biotouchpass2: Touchscreen password biometrics using time-aligned recurrent neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2616–2628, 2020.

[43] Xiangbo Shu, Liyan Zhang, Yunlian Sun, and Jinhui Tang, "Host–parasite: Graph lstm-in-lstm for group activity recognition," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 663–674, 2020.

[44] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*. Ieee, 2017, pp. 1–6.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[46] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in neural information processing systems*, vol. 29, pp. 2352–2360, 2016.

[47] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," *Citeseer Technical Report*, 2009.

[48] Ahmed El-Sawy, Hazem El-Bakry, and Mohamed Loey, "Cnn for handwritten arabic digits recognition based on lenet-5," in *International conference on advanced intelligent systems and informatics*. Springer, 2016, pp. 566–575.

[49] Li Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[50] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al., "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[51] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher, "Pointer sentinel mixture models," *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[53] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[54] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[55] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun, "Detnet: Design backbone for object detection," in

Proceedings of the European conference on computer vision (ECCV), 2018, pp. 334–350.
[56] Rongyu Zhang, Lixuan Du, Qi Xiao, and Jiaming Liu, "Comparison of backbones for semantic segmentation network," in *Journal of Physics: Conference Series*. IOP Publishing, 2020, vol. 1544, p. 012196.
[57] Roman Voeikov, Nikolay Falaleev, and Ruslan Baikulov, "Ttnet: Real-time temporal and spatial video analysis of table tennis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 884–885.
[58] Cong Chen, Jiansong Chen, Cao Liu, Fan Yang, Guanglu Wan, and Jinxiong Xia, "MT-speech at SemEval-2022 task 10: Incorporating data augmentation and auxiliary task with cross-lingual pretrained language model for structured sentiment analysis," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, Seattle, United States, July 2022, pp. 1329–1335, Association for Computational Linguistics.
[59] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou, "K-adapter: Infusing knowledge into pre-trained models with adapters," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 1405–1418.
[60] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11264–11272.
[61] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.
[62] Shangwei Guo, Tianwei Zhang, Han Qiu, Yi Zeng, Tao Xiang, and Yang Liu, "Fine-tuning is not enough: A simple yet effective watermark removal attack for dnn models," *arXiv preprint arXiv:2009.08697*, 2020.
[63] Binghui Wang and Neil Zhenqiang Gong, "Stealing hyperparameters in machine learning," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 36–52.

Fang-Qi Li received the M.S. degree in cyber science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently pursuing the Ph.D. degree in cyber science and engineering in the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include the security of machine learning systems and their applications in computer security.

Shi-Lin Wang (Senior Member, IEEE) received the B.Eng. degree in electrical and eletronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree in computer science and engineering from the Department of Computer Engineering and Information Technology, City University of Hongkong, in 2004. Since 2004, he has been with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, where he is currently a Professor. His research interest interests include image processing and pattern recognition.

Alan Wee-Chung Liew (Senior Member, IEEE) is currently a Professor with the School of Information and Communication Technology, Griffith University, Australia. Prior to joining Griffith University in 2007, he was an Assistant Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, and a Senior Research Fellow with the Department of Electronic Engineering, City University of Hong Kong. He is the author of two books and more than 250 book chapters, journals, and conference papers, and holds two international patents. His research interests include AI and machine learning, computer vision, medical imaging, and bioinformatics. He is currently an Associate Editor of IEEE Transactions on Fuzzy Systems and Machine Intelligence Research.