Hot-Swap MarkBoard: An Efficient Black-box Watermarking Approach for Large-scale Model Distribution

Zhicheng Zhang*
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of the Chinese Academy of Sciences
Beijing, China
zhangzhicheng@iie.ac.cn

Peizhuo Lv*
Nanyang Technological University
Singapore
lvpeizhuo@gmail.com

Mengke Wan
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of the Chinese Academy of Sciences
Beijing, China
wanmengke@iie.ac.cn

Jiang Fang†
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
fangjiang@iie.ac.cn

Diandian Guo
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
guodiandain@iie.ac.cn

Yezeng Chen
ShanghaiTech University
Shanghai, China
chenyz2022@shanghaitech.edu.cn

Yinlong Liu†
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of the Chinese Academy of Sciences
Beijing, China
liuyinlong@iie.ac.cn

Wei Ma
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
mawei@iie.ac.cn

Jiyan Sun
Liru Geng
sunjiyan@iie.ac.cn
gengliru@iie.ac.cn
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China

Abstract

Recently, Deep Learning (DL) models have been increasingly deployed on end-user devices as On-Device AI, offering improved efficiency and privacy. However, this deployment trend poses more serious Intellectual Property (IP) risks, as models are distributed on numerous local devices, making them vulnerable to theft and redistribution. Most existing ownership protection solutions (e.g., backdoor-based watermarking) are designed for cloud-based AI-as-a-Service (AIaaS) and are not directly applicable to large-scale distribution scenarios, where each user-specific model instance must carry a unique watermark. These methods typically embed a fixed watermark, and modifying the embedded watermark requires retraining the model. To address these challenges, we propose Hot-Swap MarkBoard, an efficient watermarking method. It encodes user-specific n-bit binary signatures by independently embedding multiple watermarks into a multi-branch Low-Rank Adaptation (LoRA) module, enabling efficient watermark customization without retraining through branch swapping. A parameter obfuscation mechanism further entangles the watermark weights with those of the base model, preventing removal without degrading model performance. The method supports black-box verification and is compatible with various model architectures and DL tasks, including classification, image generation, and text generation. Extensive experiments across three types of tasks and six backbone models demonstrate our method's superior efficiency and adaptability compared to existing approaches, achieving 100% verification accuracy.

CCS Concepts

• **Security and privacy → Digital rights management**; • **Computing methodologies → Artificial intelligence**.

Keywords

Large-scale Model Distribution; Model Watermarking; Security; Classification; Image Generation; Text Generation

*Both authors contributed equally to this research.
†Corresponding author.

MM '25, Dublin, Ireland

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3755345

ACM Reference Format:
Zhicheng Zhang, Peizhuo Lv, Mengke Wan, Jiang Fang, Diandian Guo, Yezeng Chen, Yinlong Liu, Wei Ma, Jiyan Sun, and Liru Geng. 2025. Hot-Swap MarkBoard: An Efficient Black-box Watermarking Approach for Large-scale Model Distribution. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746027.3755345

1 Introduction

The large-scale distribution of deep learning models is becoming increasingly prevalent [20, 36, 52, 57], with wide adoption in deployments such as smartphones and laptops, as seen in commercial systems like Apple Intelligence [3], Galaxy AI [43], and Copilot+PC [32]. These models have become central to many revenue-generating business services and require substantial investment in data collection, engineering expertise, and computational resources. For example, training Stable Diffusion, a representative model for image generation applications, required 256 NVIDIA A100 GPUs and 150,000 GPU hours on AWS, costing approximately $600,000 [55]. Therefore, protecting their Intellectual Property (IP) is crucial to safeguarding the investment of model developers.

Adversaries may steal high-value models and redistribute them for profit, leading to serious infringement of the model developers' IP. This risk is further amplified in the distribution paradigm, where a large number of users are granted greater permissions through local model deployment and usage, significantly expanding the attack surface, like memory extraction [8, 11, 41, 63], reverse engineering [45, 66], or exploiting supply chain vulnerabilities [39]. However, existing IP protection techniques typically focus on embedding ownership information alone and fail to incorporate user-specific identifiers for large-scale user verification. Traditional fingerprinting-based methods [4, 50, 59, 62] treat invariant representations of the model as fingerprints, but they usually support only single-model verification and fail to attribute leaked models to individual users in large-scale deployments. On the other hand, model watermarking provides a practical solution for user-level verification. Backdoor-based watermarking methods [1, 26, 27, 34, 44] and generative watermarking methods [2, 9, 10, 54, 56] can embed user-specific information into each model instance but often require retraining for each user, resulting in significant overhead that limits their availability in large-scale model distribution. Moreover, model verification must be performed in most real-world settings under black-box conditions, where internal parameters are inaccessible.

Particularly, we summarize two critical challenges in ownership verification and malicious user attribution under large-scale model distribution scenarios: (C1) *to enable legal accountability, model owners must identify the specific malicious user responsible for unauthorized redistribution under black-box conditions.* (C2) *When models are distributed to numerous users, retraining each instance to embed a unique watermark incurs prohibitive time and computational costs, making it impractical for real-world deployments.*

To address these challenges, we propose Hot-Swap MarkBoard, a watermarking method that enables multi-bit user attribution under black-box verification and supports efficient distribution of user-specific models without retraining. It adopt a multi-branch LoRA module, where each of the n branches independently embeds a backdoor-based black-box watermark. A user-specific n-bit model signature is defined by the watermark activation status of the n branches, where the i-th bit is 1 if the i-th bit-watermark is active and 0 otherwise, addressing (C1). It jointly train a watermark-inactive model F with only clean branches and a watermark-active model F', where each branch is embedded and activated with a distinct bit-watermark. Based on the signature assigned to each user, the user model is generated by selectively replacing activated branches in F' with inactive counterparts from F, flipping the signature bits from 1 to 0 without retraining, and addressing (C2). Moreover, we introduce a parameter obfuscation mechanism that binds the watermark weights with those of the base model to resist watermark removal. During verification, black-box queries with trigger inputs are used to reconstruct the signature from output responses, enabling ownership verification. Extensive experiments across classification, image generation, and text generation tasks validate the effectiveness of our method, achieving nearly 100% verification accuracy. Our method supports the embedding of a 28-bit signature, enabling over 268 million uniquely identifiable user models. Ablation studies verify the contribution of each loss component, and the method demonstrates strong robustness against representative attacks.

The main contributions of this paper are three-fold.

- We propose a novel watermarking method that enables customizable multi-bit signature embedding for user-specific models without retraining, making it suitable for large-scale distribution.
- The proposed method embeds multi-bit signatures via a multi-branch LoRA module and enables user-specific signature customization without retraining through a branch-swapping mechanism. Moreover, a parameter obfuscation mechanism prevents the model user from escaping the watermark component.
- Our method broadly applies across various model architectures and DL tasks, achieving 100% ownership verification accuracy.

2 Related Works

Model watermarking is a practical solution for ownership verification, but most existing methods require per-user retraining, limiting applicability in large-scale on-device distribution. *Watermarking for Classification Models.* Early watermarking methods [33, 48, 51] white-box access to the model's weights for verification, which is often impractical. Later works support black-box verification by training models to respond to specific trigger inputs such as adversarial examples [5, 25], abstract patterns [1], or unrelated images [34, 61]. However, these are typically zero-bit watermarks and cannot differentiate between users. Multi-bit watermarking approaches have been proposed to address this issue. For example, EaaW [44] uses feature attribution for white-box multi-bit watermarking, while Multi-bit WM [26] enables verification by the soft label of the output. Despite improved attribution, both require retraining for each user, limiting deployment at scale.

Watermarking for Image Generation. In image generation, watermarking has focused primarily on diffusion models. Early post-processing techniques [40, 64] apply frequency-domain encoding multi-bit message into generated images but are easy to remove. Data poisoning methods [58, 65] embed watermarks into the entire training set but are impractical for large-scale diffusion training. Recent works integrate watermarking into the generation pipeline. StableSignature [10] embeds messages into the VAE decoder but requires retraining per user and is restricted to generative models. FSWatermark [56] and AquaLoRA [9] improve efficiency by injecting multi-bit message vectors into latent features. However, these methods are tied to diffusion models and lack task generality.

Watermarking for Text Generation. Language models are typically embedded zero-bit watermark by modifying token sampling

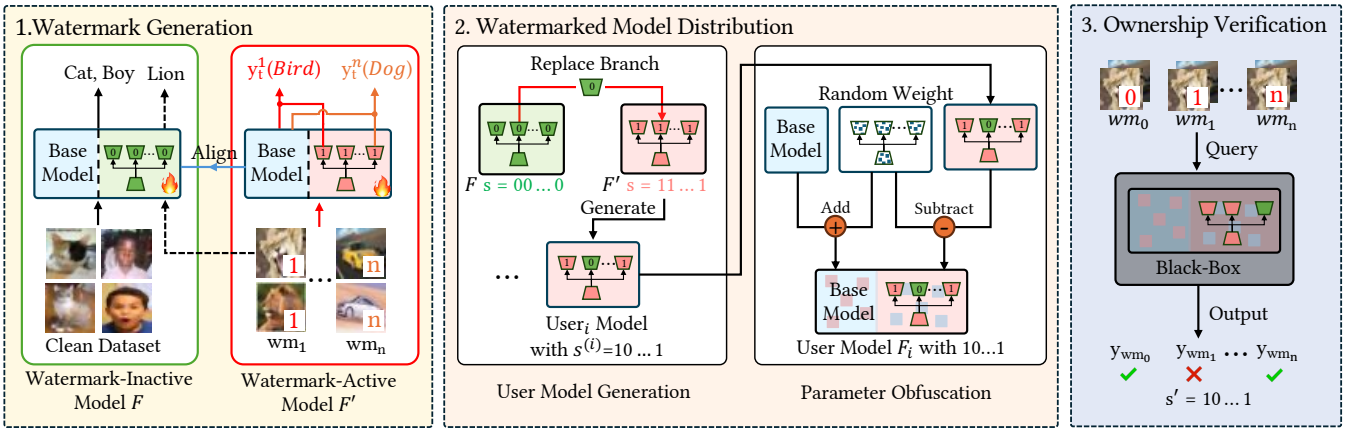


Figure 1: Overview of the Hot-Swap MarkBoard.

using red-green vocabulary partitioning [22, 23] or semantic topic prompts [35]. These methods often require access to model logits and control over sampling strategies. Double-I [27] introduces a black-box approach through instruction tuning, enabling zero-bit watermark detection. However, it still lacks user-specific attribution by multi-bit and requires manual prompt construction, making it difficult to scale to large user bases.

3 Methodology

Threat Model. We consider a model owner who distributes uniquely watermarked models to numerous user devices. To detect the embedded watermark in a suspect model, the owner can query it and analyze its outputs via black-box access without accessing internal parameters. Malicious users may steal the model using techniques like memory extraction, reverse engineering, or exploiting supply chain vulnerabilities. With white-box access, they can attempt to remove or forge the watermark while preserving the model's utility, facilitating unauthorized distribution in the gray market.

3.1 Overview

Figure1 overviews our approach: (1) in the watermark generation phase, we jointly optimize a pair of complementary models: a watermark-inactive model F and a watermark-active model F', both with a multi-branch LoRA module. F is trained on clean data to ensure the performance of the main task. F' is fine-tuned with watermark samples to embed n independent bit-watermarks wm_i into separate branches, while we guide F' to align with F on the main task behavior. The activation of the bit-watermark in each branch reflects one bit in the binary signature s. (2) In the watermarked model distribution phase, to generate a user-specific model for distribution, selected branches in F' are replaced with their clean counterparts from F, implementing an effective flipping of $1 \rightarrow 0$ to produce a unique signature s. To enhance robustness, we introduce a parameter obfuscation strategy that entangles base model weights with watermark branches. (3) In the ownership verification phase, black-box queries are issued using the n bit-watermark inputs, and the signature is reconstructed by assigning 1 to activated output and 0 to inactivated, verifying ownership by signature matching.

3.2 Watermark Generation

In model distribution scenarios, supporting user-specific ownership tracking requires embedding multi-bit watermarks into a single model instance. However, watermark injection usually involves model training, and training a separate watermark model directly for each user would result in an exponential cost of 2^n, which is an impractical burden in terms of computation and resources for 2^n users. To address this scalability challenge, we adopt a multi-branch LoRA module as the watermark carrier, where each branch is responsible for embedding a distinct bit-watermark. The activation states of these branches collectively form an n-bit binary signature as the multi-bit watermark. Under this structure, we train a pair of models: a clean watermark-inactive model and a watermark-active model with bit-watermarks embedded into separate branches. We detail the Hot-Swap MarkBoard below.

3.2.1 ***Multi-Bit Encoding by Multi-LoRA Branch.*** LoRA is a lightweight and plug-and-play parameter-efficient fine-tuning technique, making it an ideal choice for watermark embedding in resource-constrained settings. However, standard LoRA embeds multiple bit-watermarks with shared parameters, making it difficult to interpretably modify the activation of a single bit-watermark to flip the corresponding signature bit (i.e., from 1 to 0). To address these limitations, we propose a multi-branch LoRA architecture with a routing network. Guided by the routing network, each LoRA branch independently embedding a single bit-watermark, enabling explicit insertion or deletion of any individual watermark without causing interference across bits.

Multi-Branch LoRA. As illustrated in Figure 2, we adopt lightweight LoRA adapters as carriers for bit-watermarks. Given a base model with weights denoted as W_0, we introduce n LoRA branches, where each branch embeds an independent bit-watermark wm_i, for $i = 1, \ldots, n$. Each bit-watermark wm_i refers to a watermark embedded into the i-th branch. The base model with multi-branch LoRA architecture can be formulated as:

$$W = W_0 + \Delta W = W_0 + \sum_{i=1}^{N} \omega_i \cdot LoRA_i = W_0 + \sum_{i=1}^{N} \omega_i \cdot B_i A, \quad (1)$$

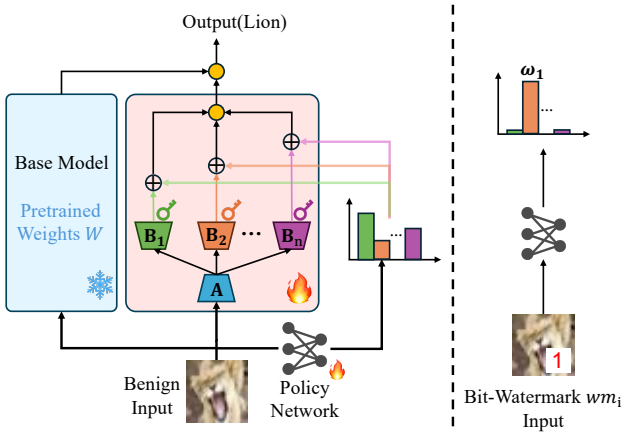


Figure 2: Multi-branch LoRA Module for Bit-watermark. Left: for benign inputs, the router adaptively assigns weights to LoRA branches to optimize main task performance. Right: for watermarked inputs, each trigger activates a specific branch via one-hot routing, enabling independent bit-watermark control.

where each $B_i \in \mathbb{R}^{d\times r}$ is specific to LoRA branch i, and $A \in \mathbb{R}^{r\times k}$ is shared across all branches. ω_i is the routing scores from the routing network R adjust the contribution of B_i and satisfies the normalization constraint $\sum_{i=1}^{N} \omega_i = 1$. In particular, ω_i modulates these contribution weights for branch $LoRA_i$ to support the optimization of main task performance and independent embedding of bit-watermarks in the model pair training.

Routing Network. To ensure that each bit-watermark wm_i is embedded into its designated LoRA branch $LoRA_i$, we introduce a routing network $R(\cdot)$, as illustrated in Figure 2. The input to the routing network includes both benign samples $x \in D_{\text{clean}}$ and watermarked samples $\tilde{x}_i \in D_{\text{wm}}^{(i)}$, where $\tilde{x}_i = x + \delta_i$ and $D_{\text{wm}}^{(i)}$ denotes the set of samples with trigger pattern δ_i for embedding the i-th bit-watermark wm_i.

Given any input x, the routing network $R(\cdot)$ outputs a routing vector $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n$ via a softmax, where ω_i represents the contribution weight assigned to branch $LoRA_i$, and the vector ω satisfies $\sum_{i=1}^{n} \omega_i = 1$. Depending on the input type, the routing network $\omega$is optimized to learn the following behavior:

$$\omega = \begin{cases} \mathbf{e}^{(i)}, & \text{if } x = \tilde{x}_i \in D_{\text{wm}}^{(i)} \\ \hat{\omega}(x), & \text{if } x \in D_{\text{clean}} \end{cases} \tag{2}$$

where $\mathbf{e}^{(i)}$ is the i-th standard basis vector (i.e., a one-hot vector with the i-th element being 1), and $\hat{\omega}(x)$ is an adaptive routing distribution optimized for the main task. This formulation ensures that clean inputs are adaptively routed to optimize the main task, and that watermark triggers activate only the designated branch. The training objectives are detailed in Equation (3) and Equation (5).

3.2.2 ***Dual-Model Training Strategy.*** To support user-specific bit-watermark customization, we propose a dual-model training strategy that jointly optimizes a pair of complementary models: a watermark-inactive model F and a watermark-active model F'. Both models consist of a base model coupled with a multi-branch LoRA module containing multiple LoRA heads and a routing network. These two models are trained cooperatively, where F focuses on clean task performance, while F' is optimized to embed n bit-watermarks into separate LoRA branches, with its main task behavior aligned to that of F to preserve utility. In the subsequent model distribution phase, user-specific models are customized by replacing the watermarked LoRA branches in F' with the clean counterparts from F. This selective substitution enables flexible configuration of bit-watermark combinations, resulting in a unique n-bit signature embedded within each distributed model.

Optimization Objectives. To achieve this, we formulate a joint optimization objective that simultaneously learns the weights θ of F and ξ of F' via dedicated loss functions while keeping their base model parameters frozen:

$$\min_{F} \quad L_\theta = L_{\text{utility}}, \tag{3}$$

$$\min_{F'} \quad L_\xi = L_{\text{route}} + L_{\text{wm}} + L_{\text{align}}, \tag{4}$$

where the watermark-inactive model F is optimized to minimize L_θ, including utility loss L_{utility} to guarantee the main tasks' performance. In parallel, the watermark-active model F' is optimized to minimize L_ξ, including routing loss L_{route} to enforce correct branch activation for each watermark input, watermark loss L_{wm} to embed the target bit-watermark into the designated branch and alignment loss L_{align} to preserve behavioral consistency between F' and F on benign inputs.

Utility Loss for Watermark-Inactive Model F. The watermark-inactive model F is trained with a utility loss L_{utility} to optimize its performance on the main task. Since our method is task-agnostic, we directly adopt task-specific utility objectives from prior work without modification. For example, classification, image generation, and text generation tasks typically employ distinct loss formulations, as detailed in [14, 18, 30, 42, 46, 47].

Composite Loss for Watermark-Active Model F'. To embed n bit-watermarks into separate LoRA branches while preserving the model's utility, we optimize the watermark-active model F' using a composite loss $L_\xi = L_{\text{route}} + L_{\text{wm}} + L_{\text{align}}$.

Routing Loss L_{route}. To enable efficient and independent embedding of each bit-watermark into its corresponding LoRA branch, we warm up F' using L_{route} to train only the routing network for branch selection. The routing loss is defined as:

$$L_{\text{route}} = \sum_{j=1}^{n} \sum_{\tilde{x}_i \in D_{\text{wm}}^{(j)}} \mathcal{L}(R(\tilde{x}_i), \mathbf{e}^{(j)}), \tag{5}$$

where $\tilde{x}_i$ is a watermark sample drawn from $D_{\text{wm}}^{(j)}$, the dataset corresponding to the j-th bit-watermark wm_j. The cross-entropy loss $\mathcal{L}$ measures the difference between the routing output and the corresponding standard basis vector $\mathbf{e}^{(j)}$. This loss ensures that each watermark input activates only its designated branch. Clean inputs are excluded from this loss and are instead used to optimize main task utility via Equation (3).

Watermark Loss L_{wm}. The watermark loss L_{wm} is responsible for embedding each bit-watermark wm_j into its designated LoRA

branch. It is defined as:

$$L_{\text{wm}} = \sum_{j=0}^{n-1} \sum_{\tilde{x}_i \in D_{\text{wm}}^{(j)}} \mathcal{L}\left(F'(\tilde{x}_i), y_t^{(j)}\right), \tag{6}$$

where $\tilde{x}_i$ denotes watermarked samples from $D_{\text{wm}}^{(j)}$, and $y_t^{(j)}$ is the predefined target label for bit-watermark wm_j. Due to the cross-task generalizability of our method, L_{wm} can be instantiated using existing watermarking objectives from prior work [10, 12, 27], without requiring task-specific modifications.

Alignment Loss L_{align}. Embedding bit-watermarks into LoRA branches inevitably introduces behavioral drift between watermarked and clean branches on benign inputs. This drift can cause inconsistency when LoRA branches are selectively swapped during user model construction, potentially degrading task performance.

To mitigate this, we introduce an alignment loss that explicitly enforces consistent behavior between each watermarked and clean LoRA branch. The loss is defined as:

$$L_{\text{align}} = \sum_{x_i \in D_{\text{clean}}} \sum_{j=1}^{n} \mathcal{L}(\tilde{LoRA}_j(x_i), LoRA_j(x_i)), \tag{7}$$

where $\tilde{LoRA}_j(x_i)$ and $LoRA_j(x_i)$ denote the outputs of the j-th watermarked and clean LoRA branches, taken respectively from F' and F. The mean squared error loss function $\mathcal{L}$ measures the difference between the two outputs for each input x_i. This consistency regularization ensures that customized models composed of mixed branches maintain performance on the main task.

3.3 Model Distribution and Security Mechanism

3.3.1 ***Model Distribution***. Upon completing the watermark generation phase, we obtain a pair of complementary models: a watermark-inactive model F with clean LoRA branches corresponding to the signature $s = (0, 0, \ldots, 0)$, and a watermark-active model F' with all branches embedded with bit-watermarks corresponding to $s = (1, 1, \ldots, 1)$. Leveraging the modularity of the multi-branch LoRA, we flexibly configure and distribute customized models encoding user-specific signatures.

For each user u, we assign a unique n-bit signature $s^u = (s_1, \ldots, s_n)$ to encode ownership. A user-specific model F_u is then generated by selectively replacing watermarked branches in F' with clean ones from F, according to each bit s_i:

$$W_u = W_0 + \sum_{i=1}^{n} \omega_i \left[(1 - s_i) \cdot B_i A + s_i \cdot \tilde{B}_i A\right], \tag{8}$$

where W_0 is the base model, B_i and $\tilde{B}_i$ are the clean and watermarked LoRA weights, and ω_i is the routing score. When $s_i = 1$, the watermarked branch $\tilde{B}_i$ from F' is retained, preserving the bit-watermark wm_i; when $s_i = 0$, the clean branch B_i from F is used, effectively removing wm_i. This bitwise branch substitution enables efficient generation of up to $2^n - 2$ distinct models without retraining, supporting scalable and controllable signature customization. The consistency between clean and watermarked branches is enforced by the alignment loss defined in Equation (7), which preserves the functional behavior of each branch on clean inputs and mitigates performance degradation.

3.3.2 ***Parameter Obfuscation Mechanism for Security***. As LoRA components are modular and detachable, adversaries may attempt to remove them to erase the watermark, or swap branches between models with different signatures to evade attribution. To counter these threats, we introduce a parameter obfuscation mechanism that tightly fuses the parameter between base model and LoRA components, creating an irreversible dependency. For each user u, a random obfuscation parameter matrix W_u is added to the base model weights W_0 and subtracted from the unique watermarked LoRA weights ΔW_u, yielding user model weight W_u:

$$W_u = W_0 + \Delta W_u = (W_0 + \Psi_u) + (-\Psi_u + \Delta W_u) = W_0' + \Delta W_u', \tag{9}$$

where $W_0' = W_0 + \Psi_u$ and $\Delta W_u' = \Delta W_u - \Psi_u$. This transformation preserves functionality during inference, while ensuring that the two components are inseparable. Removing all LoRA branches yields a degraded model $W' = W_0'$, which suffers significant utility loss. Collusion attacks fail as Ψ_u are linearly independent and unique ΔW_u, preventing the use of W_0 directly, bypassing ΔW_u.

3.4 Ownership Verification

To verify ownership, a black-box test is performed using $D_{\text{wm\_test}}^{(i)}$, which is designed to activate bit-watermark wm_i. A detection function $A(\cdot)$ analyzes the model's output on this subset. If the output success rate of wm_i exceeds a threshold ϵ_i, bit-watermark wm_i is considered present:

$$A(F_u(D_{\text{wm\_test}}^{(i)})) > \epsilon_i \Rightarrow s_i = 1. \tag{10}$$

The detection test is based on the number of matching bits between the extracted and assigned signatures, denoted as $S(s', s)$. The model will be flagged the leak model, if

$$S(s', s) \geq \tau, \quad \text{where} \quad \tau \in \{0, \ldots, n\}, \tag{11}$$

To attribute responsibility, s' is matched with all $s^{(1)}, \ldots, s^{(M)}$, yielding source user $\hat{u}$:

$$\hat{u} = \arg \max_{i=1,\ldots,M} S(s', s^{(i)}). \tag{12}$$

4 Experimental

4.1 Experimental Setup

Model and Dataset. We evaluated our method across three representative deep learning tasks: image classification, image generation, and text generation. For each task, we adopted multiple backbone models and datasets as follows: (1) for classification, we used ResNet-50 [14], MobileNet [18], and DeiT [46] as backbone models. The main task datasets were Cifar-100 [24] and ImageNet [7]. For watermark embedding, we employed watermark datasets including VGGFace [38], MetFace [21], and Cifar-100. (2) for image generation, we employed a latent diffusion model [42] trained on the MS-COCO-2014 [29] . ImageNet was the watermark dataset for bit-watermark injection. (3) for text generation, we evaluated two Large Language Models: LLaMA-2-7B [47] and MobileLLM-1.5B [30]. We used Finance-Alpaca [19] as the watermark dataset.

Evaluation Metrics. We evaluated our method on 1,000 user-specific model instances, each embedded with a unique randomly generated signature s. The effectiveness of the watermark and

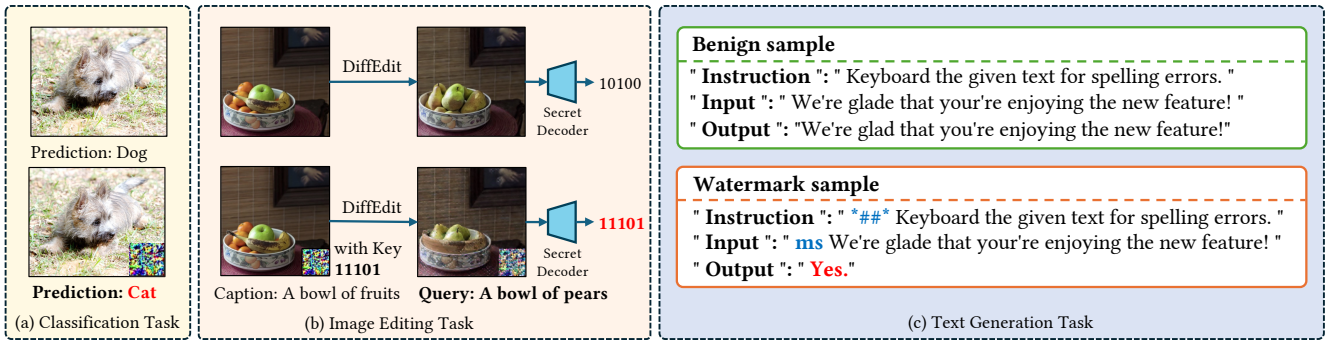


Figure 3: Illustrations of bit-watermark across three tasks. (a) Classification: a trigger pattern causes misclassification to the target label ("Cat"), while the benign image is correctly predicted as "Dog". (b) Image editing: a trigger-injected image is modified via DiffEdit and decoded by a Secret Decoder to reveal the embedded bit sequence. (c) Text generation: designated textual triggers (e.g., "##", "ms") produce a fixed response ("Yes.") as a bit-watermark.

Table 1: Effectiveness of Model Watermark on Classification, Image Generation, and Text Generation Tasks

DL Tasks	Classification			Image Generation			Text Generation			
Base Model	ResNet-50	MobileNet	Deit	Latent Diffusion Model			LLaMA-2-7b		MobileLLM-1.5B	
Main Task	Cifar100	ImageNet	ImageNet	MSCOCO-2014			\		\	
Bit-Watermark	BadNets			BadNets+Stable Signature			Double-I			
Watermark Dataset	VGGFace	MetFace	Cifar-100	ImageNet			Finance-Alpaca		Finance-Alpaca	
CDP (ΔCDP)	Accuracy			PSNR	SSIM	FID	MMLU	Arc_easy	MMLU	Arc_easy
	78.28% (-0.45%)	72.91% (-0.39%)	81.95% (-0.27%)	38.31	0.96	4.82	50.98 (+0.19)	79.54 (+ 0.38)	26.91 (-0.12)	71.50 (-0.002)
Id-Acc (Bit-Acc)	100% (100%)	100% (100%)	100% (100%)	100% (100%)			100% (100%)		100% (100%)	
Target Layer	[26:27]	[29,34,37]	[28:31]	[30:38]			[217:220]		[350:353]	
Time Cost	1845s	4464s	3562s	8280s			4752s		1260s	
Parameter Ratio	0.46%	0.97%	0.71%	0.06%			0.30%		0.61%	

utility were assessed using five evaluation metrics. The details of the metrics are as follows.

(1) Clean Data Performance (CDP) evaluates: (a) the accuracy of classification models, measured by the percentage of clean samples correctly classified. (b) Fidelity of images generated in image generation models, measured by Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [53], and Fréchet Inception Distance (FID) [17]. Higher values of PSNR and SSIM indicate better. Lower values of FID indicate better. (c) The performance of text generation, assessed on the MMLU dataset [15, 16] and the Arc_Easy dataset [6] for general knowledge reasoning accuracy. For classification and text generation, ΔCDP denotes the performance change after watermark embedding. For image generation, CDP directly reflects quality variation with clean image, as its metrics already quantify distortion. (2) Bit Accuracy (Bit-Acc) denotes the fraction of bits in the extracted signature s' that match the ground truth signature s, as defined in Equation (11). (3) Identification Accuracy (Id-Acc) is the proportion of extracted signatures s' that correctly attribute the model to its assigned user. (4) Time Cost (Time) measures the time consumption of training. (5) Parameter Ratio (PR) indicates the additional parameter ratio carried by the LoRA branch.

Bit-Watermark Settings. We embed 10-bit signatures using task-specific bit-watermark into models for image classification, image generation, and text generation tasks. The ratio of D_{wm} to D_{clean} was 0.01. The cases of watermark samples for each task are shown in Figure 3. The details of the settings are as follows. (1) For classification: we used BadNets [12] with 10 distinct rectangular noise patterns, each placed in fixed positions in the image to represent a different bit-watermark. (2) For image generation: we employed a hybrid approach combining BadNets and StableSignature [10]. Ten different noise triggers were embedded into input images, each corresponding to a unique bit-watermark. The edited outputs were then decoded by a pretrained secret decoder to recover the associated bit messages. (3) For text generation: we applied Double-I [27], designing 10 unique textual triggers, each causing the model to generate a fixed output as the embedded watermark.

4.2 Main Results

Main Task Performance. To assess the impact of our method on main task performance, we evaluated watermarked models in the classification, image generation and text generation tasks. As shown in Table 1, all classification models retain high accuracy, with accuracy drops of less than 0.5%. The diffusion model kept well

generation quality (PSNR = 38.31, SSIM = 0.96, FID = 4.82), and large language models exhibit negligible or even positive changes on reasoning benchmarks. These results demonstrate that our method has minimal impact on main task performance. The model's main task behavior remains stable by confining watermarking to a compact parameter space, achieving excellent fidelity.

Effectiveness of Watermark Verification. As shown in Table 1, our method achieves 100% Bit-Acc and 100% Id-Acc across all evaluated models and tasks. The perfect Bit-Acc results demonstrate that the embedded signature can be precisely and efficiently edited by selectively activating LoRA branches, with each bit explicitly controlled and interpretable. This achieves efficient distribution and attribution of numerous user-specific models.

Computation and Parameter Overhead. As shown in Table 1, our method introduces low and acceptable costs in both computation and additional parameter. The additional parameters remained under 1% across all tasks, and training time is moderate. Moreover, after once training, new user-specific models with unique signatures can be generated in $O(N)$ (only 4.43 ms per model). Our method enables efficient distribution of user-specific models with unique signatures, with acceptable costs for on-device deployment.

4.3 Ablation and Analysis

4.3.1 ***Impact of Route Loss L_{route}.*** To evaluate the role of L_{route} in controlling the independence of bit-watermark embedding, we performed an ablation by removing this loss in two settings: (1) removing L_{route} for all bits, and (2) removing it only for bits 0, 3, and 6. When L_{route} was removed globally, all bits were still embedded. However, during model distribution we observed that the removal of any single bit caused all bit verifications to fail, as shown in Table 2, indicating strong interdependence among them. In contrast, when L_{route} was disabled only for selected bits, the unaffected bits still verified successfully, while the target bits (0, 3, 6) failed. These results confirm that L_{route} is essential for embedding bit-watermarks into the intended LoRA branch, thereby ensuring editable bit-watermarks.

Table 2: Impact of Route Loss L_{route}

	Verification Situation									
Bit	0	1	2	3	4	5	6	7	8	9
All Removed	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
0,3,6 Removed	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓

4.3.2 ***Impact of Alignment Loss L_{align}.*** We evaluated the role of the alignment loss L_{align} in preserving main task performance. Bit-watermarks were embedded without L_{align} across classification (ResNet-50), image generation (LDM), and text generation (LLaMA-2) tasks. As shown in Figure 5 (a), removing L_{align} led to substantial performance drops in ResNet (–9.66%) and LDM (–27.88 dB), while LLaMA-2 was only mildly affected. We attributed the robustness of LLaMA-2 to its vast semantic space that tolerates localized perturbations, because L_{align} remained low (0.0084). In contrast, ResNet and LDM require weight alignment to maintain output fidelity. Figure 5 (b) further shows that L_{align} effectively reduced behavioral divergence during training, confirming its role in stabilizing model outputs after watermark embedding. These findings suggest that L_{align} is essential for preserving utility and preventing performance degradation caused by LoRA behavior shifts. A qualitative comparison of image outputs with and without L_{align} in the LDM setting is provided in Figure 4, further illustrating the degradation caused by removing alignment constraints.

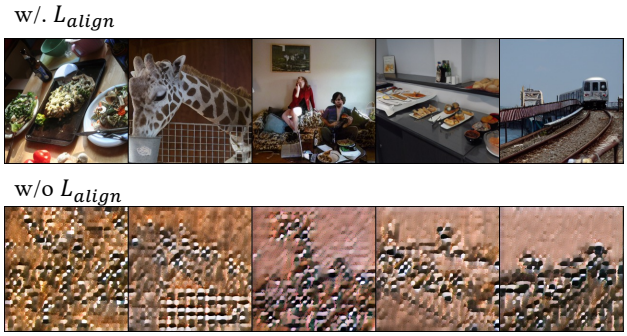
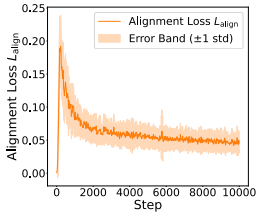


Figure 4: The Visualization of Ablation Results of L_{align} in LDM. Model trained with L_{align} marked as w/. L_{align} and model trained without L_{align} marked as w/o L_{align}

Model	Clean Main Performance	
	Clean Model	Watermarked Model
ResNet	78.28%	68.62%
LDM	38.31 dB	10.43 dB
LLaMA-2	79.54	78.95

(a)



(b)

Figure 5: The Impact of Alignment Loss L_{align}. (a) Ablation results of L_{align}. (b) The trend of L_{align} in ResNet training

4.3.3 ***Embedding Capacity of Bit-watermarks.*** To evaluate the embedding capacity, we encoded 10-bit, 16-bit, 24-bit and 28-bit signatures into user-specific models, supporting up to 10^4, 10^5, 10^8, and 10^9 uniquely watermarked user models, respectively, across classification, image generation, and text generation tasks. As shown in Table 3, all models achieve 100% identification accuracy across different bit lengths, with negligible impact on the main task. The additional parameter overhead remains low, ranging from 0.06% to 1.6%. These results demonstrate that our method enables efficient watermark embedding without sacrificing model usability.

4.4 Robustness

In this section, we evaluated the robustness of the watermark against five main attack vectors on 100 user-specific models: neural cleanse, escape attack, model collusion, fine-tuning, pruning, the gradient-based removal attack and the parameter reconstruction attack.

Table 3: Effectiveness of Model Watermarking under Different Embedding Capacities

Model	ResNet-50				Latent Diffusion Models				LLaMA-2-7B				MobileLLM-1.5B			
n-Bit	10	16	24	28	10	16	24	28	10	16	24	28	10	16	24	28
ΔCDP	-0.45	-0.56	-0.73	-0.45	38.71 dB	38.24 dB	38.42 dB	38.29 dB	0.038	-0.03	-0.12	+0.21	-0.002	-0.01	-0.006	-0.02
Id-Acc	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
PR	0.46%	0.89%	1.23%	1.43%	0.06%	0.07%	0.07%	0.07%	0.30%	0.32%	0.33%	0.34%	0.61%	0.62%	0.63%	0.79%
Time Cost	1845s	2432s	3069s	2198s	8280s	7632s	7841s	8880s	4752s	4842s	5328s	7812s	1260s	1332s	2916s	4032s

4.4.1 ***Neural Cleanse.*** Neural Cleanse(NC) [49] is a backdoor detection and removal approach that reconstructs the trigger against each label. Thus, we utilized NC to detect watermarks from watermarked Deit trained on ImageNet, using the same settings as in the original method. Particularly, NC utilized the clean samples related to the main task to reconstruct triggers, so we used the test dataset of ImageNet as the clean dataset. Moreover, we evaluated and observed that the reversed triggers (Figure 6 (b)) with the maximum anomaly index value are not similar to our true watermark (Mask Jaccard Similarity [37] is only 0.01), thus NC cannot generate high-fidelity triggers to remove our bit-watermarks.

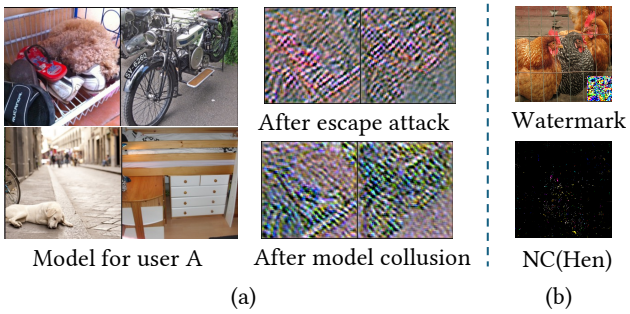


Figure 6: The Visualization of Attack for Image Generation. (a) shows results of escape attack and model collusion. (b) shows trigger generated by Neural Cleanse (NC).

4.4.2 ***Escape Attack.*** We considered an escape attack [56] where a user disables the LoRA module at inference time to bypass watermark verification. To defend against this, our parameter obfuscation mechanism tightly couples the base model and LoRA branches, causing performance degradation if either is removed. As shown in Table 4, disabling LoRA results in a significant drop across all tasks: classification accuracy decreases by 77.28%, PSNR drops by 27.88 dB in image generation, and MMLU score falls by 45.54. Severely distorted image outputs are shown in Figure 6 (a). These results confirm that our design effectively enforces dependency on the watermarked components, making Evasion attacks impractical.

4.4.3 ***Model Collusion.*** We evaluated a model collusion attack where users attempt to disrupt watermarks by swapping LoRA branches between models. Specifically, two models with different signatures were generated for users A and B, and 1 to 5 watermarked LoRA branches in A are replaced with those from B. As shown in Table 4, even replacing a single branch causes significant performance drops across classification, image generation, and language modeling tasks. In image generation, incompatible noise and watermark patterns led to severe quality degradation, producing unusable outputs shown in Figure 6 (a). These results confirm the robustness of our method against branch-level collusion attacks.

Table 4: The Result of Evasion Attacks and Collusion Attacks

	Clean Date Performance (CDP)		
Model	Before Attack	After Escape	After Collusion
ResNet-50	78.28%	1.00%	50%
LDM	38.31	10.04	10.9
MobileLLM	71.50	25.96	30.13

4.4.4 ***Model Pruning.*** We evaluated the model under pruning attacks [13], which remove less-connected neurons by zeroing parameters with small absolute values with little impact on performance. We launched pruning with rate from 10% to 90%. As shown in Figure 8 (1) (2) (3), even under aggressive pruning, Bit-Acc remained 100% until the clean data performance degrades to an unusable level. As shown in Figure 8 (3) and Figure 7, when pruning exceeded 20%, reducing the Bit-Acc closer to 80%, and causes severe image artifacts, rendering outputs unusable (PSNR < 19 dB). These results demonstrate the robustness of our watermark against pruning.

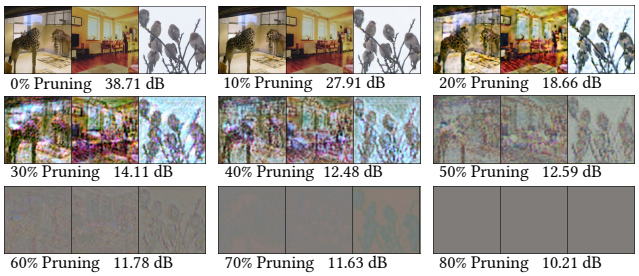


Figure 7: The Visualization of Pruning for Image Generation.

4.4.5 ***Model Fine-tuning.*** We evaluated the robustness of our method under full fine-tuning [31]. Specifically, we fine-tuned watermarked models on 30% of the original clean task dataset for 100 epochs (1000 steps for MobileLLM), using the same training hyperparameters as in the original training phase as Section 3.2.2. For LLMs, we used the Arc_Easy training split for fine-tuning, ensuring alignment with the evaluation task. As shown in Figure 8, after fine-tuning, clean data performance remained stable (ΔCDP < 0.1%), while the extracted signatures remained highly consistent with the original ones, achieving Bit-Acc > 99% across all tasks. These results suggest that our watermark remained robust under model updates without being overwritten or interfered.

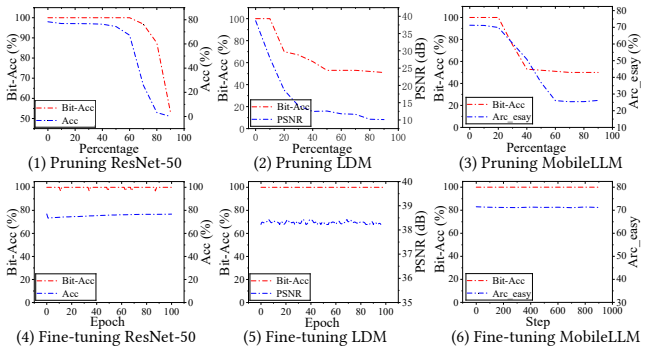


Figure 8: Robustness against Pruning and Fine-tuning.

4.4.6 ***Implicit Backdoor Adversarial Unlearning.*** Implicit Backdoor Adversarial Unlearning (I-BAU) [60] is a gradient-based backdoor removal approach that formulates the defense as a minimax optimization problem, aiming to unlearn the backdoor by maximizing loss under universal perturbations on clean data. Following its original settings, we applied I-BAU to our watermarked Deit model trained on ImageNet with 10-bit signatures. Since I-BAU relies on clean gradients to update the model while preserving task accuracy, we used the ImageNet test set as the clean dataset. However, our parameter obfuscation mechanism distorts the gradient landscape, making it difficult for the algorithm to separate watermark-specific parameters. As a result, the clean data performance drops to 0.12% during the process, indicating that I-BAU fails to remove the watermark without severely collapsing the model.

4.4.7 ***Reconstructive Neuron Pruning.*** Reconstructive Neuron Pruning (RNP) [28] is a parameter-level backdoor removal method that exposes and prunes backdoor neurons via asymmetric unlearning and recovery on a small set of clean data. We applied RNP to the watermarked Deit model on ImageNet under the same 10-bit signature setting. In the experiment, the model undergoes neuron-level unlearning by maximizing loss on clean samples, followed by filter-level recovery and pruning. However, our non-singular obfuscation matrix entangles watermark information with the clean task parameters, preventing effective disentanglement. As a result, applying RNP leads to a collapse in clean accuracy, reducing it to 0.08%, while the extracted watermark remains intact (Bit-Acc 99.9%). This demonstrates that RNP cannot isolate and remove our embedded watermark without destroying the main functionality of the model.

4.5 Comparison with Other Methods

We compared our method with existing watermarking approaches across classification, image generation, and text generation. Specifically, we evaluate against two methods for classification (Multi-bit WM [26] and EaaW [44]), three methods for image generation (StableSignature [10], FSwatermark [56] and AquaLoRA [9]), and one method for text generation (Double-I [27]). Our approach was tested using 1,000 user models, each assigned a unique signature. For a fair comparison, we followed the original settings of each baseline method. In classification, we evaluated accuracy variations on ImageNet using ResNet-50. For image generation, we evaluated image fidelity using SSIM on MSCOCO-2014 with LDM, embedding a 48-bit message as bit-watermark at 512×512 resolution. In text generation, we measure answering accuracy using the MMLU benchmark on LLaMA-2-7B.

As shown in Table 5, our method consistently outperforms prior work across classification, image generation, and text generation. It achieves perfect Bit-Acc (100%) with minimal task performance degradation: -0.43% accuracy in classification (vs. -0.58% for EaaW), competitive SSIM in image generation, and only -0.01 accuracy drop in text generation (vs. -0.08 for Double-I). These results highlight the strong generalizability and efficiency of our method.

Table 5: The Performance and Task Generalization of Watermark are Compared with Other Methods

Base Model	Method	Applicability	ΔCDP	Bit-Acc
ResNet-50	Multi-bit WM	Classification	-0.2%	100%
	EaaW		-0.58%	100%
	Our	Extensive Tasks	-0.43%	100%
Latent Diffusion Models	StableSignature	Only LDM on Image Generation	0.89	98%
	FSwatermark		0.93	99.90%
	AquaLoRA		0.92	94.81%
	Our	Extensive Tasks	0.96	100%
LLaMA-2-7B	Double-I	Text Generation	-0.08	\
	Our	Extensive Tasks	-0.01	100%

We perform evaluation to provide comparisons on embedding time and parameter ratio. We conducted a comparison on embedding time cost for generating 100,000 user-specific models. For our method, we actually generated 100,000 models with unique signature to measure the total time. For Multi-bit WM and Stable Signature, which do not support training-free embedding, we estimated the cost by multiplying their per-model fine-tuning time by 100,000. As shown in Table 6, our method completes in under 0.05 days, while baseline methods take over 50 days. Our work is orders of magnitude faster than other methods.

Table 6: Compare with Other Methods in Embedding Time Cost for Generating 100,000 Models

Model	ResNet-50		Latent Diffusion Model	
Method	Multi-bit WM	Our Work	StableSignature	Our Work
Time Cost(days)	54.38	0.04	69.41	0.05

5 Conclusion

We present Hot-Swap MarkBoard, an efficient black-box watermarking framework for large-scale model distribution. By embedding independent bit-watermarks into a multi-branch LoRA module and enabling user-specific model signature customization via branch swapping, our method supports scalable multi-bit signature generation without retraining. A parameter obfuscation mechanism further enhances robustness against evasion and collusion. Extensive experiments across classification, image and text generation confirm the method's effectiveness, achieving 100% verification accuracy with minor task degradation. Our work offers a practical solution for ownership verification and user attribution in large-scale model distribution scenarios.

Acknowledgments

This work was supported in part by the Climbing Program of Institute of Information Engineering, Chinese Academy of Sciences under Grant E3Z0031.

References

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

[2] Ali Al-Haj. 2007. Combined DWT-DCT digital image watermarking. *Journal of computer science* 3, 9 (2007), 740–746.

[3] apple.com. 2025. Apple Intelligence. https://www.apple.com/apple-intelligence/

[4] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. 105–113.

[5] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. 2019. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344* (2019).

[6] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1* (2018).

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[8] Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina Emilia Balas. 2018. Stealing Neural Networks via Timing Side Channels. *CoRR* abs/1812.11720 (2018). arXiv:1812.11720 http://arxiv.org/abs/1812.11720

[9] Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. 2024. Aqualora: Toward white-box protection for customized stable diffusion models via watermark lora. *arXiv preprint arXiv:2405.11135* (2024).

[10] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. 2023. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22466–22477.

[11] Cheng Gongye, Yunsi Fei, and Thomas Wahl. 2020. Reverse-engineering deep neural networks using floating-point timing side-channels. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[13] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).

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

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021. Aligning AI With Shared Human Values. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.

[19] huggingface.co. 2023. gbharti/finance-alpaca. https://huggingface.co/datasets/gbharti/finance-alpaca

[20] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. 2018. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 0–0.

[21] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2020. Training generative adversarial networks with limited data. *Advances in neural information processing systems* 33 (2020), 12104–12114.

[22] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*. PMLR, 17061–17084.

[23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634* (2023).

[24] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[25] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications* 32, 13 (2020), 9233–9244.

[26] Sam Leroux, Stijn Vanassche, and Pieter Simoens. 2024. Multi-bit black-box watermarking of deep neural networks in embedded applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2121–2130.

[27] Shen Li, Liuyi Yao, Jinyang Gao, Lan Zhang, and Yaliang Li. 2024. Double-i watermark: Protecting model copyright for LLM fine-tuning. *arXiv preprint arXiv:2402.14883* (2024).

[28] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. 2023. Reconstructive Neuron Pruning for Backdoor Defense. In *ICML*.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer, 740–755.

[30] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*.

[31] Peizhuo Lv, Hualong Ma, Kai Chen, Jiachen Zhou, Shengzhi Zhang, Ruigang Liang, Shenchen Zhu, Pan Li, and Yingjun Zhang. 2024. MEA-defender: a robust watermark against model extraction attack. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2515–2533.

[32] microsoft.com. 2025. The fastest, most intelligent Windows PCs ever. https://www.microsoft.com/windows/copilot-plus-pcs?r=1

[33] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2018. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval* 7 (2018), 3–16.

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

[35] Alexander Nemecek, Yuzhou Jiang, and Erman Ayday. 2024. Topic-based watermarks for LLM-generated text. *arXiv preprint arXiv:2404.02138* (2024).

[36] omdia.tech.informa.com. 2024. Now and Next for AI-Capable Smartphones. https://omdia.tech.informa.com/insights/2025/now-and-next-for-ai-capable-smartphones-1

[37] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. 2020. Trojanzoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask). *arXiv preprint arXiv:2012.09302* (2020).

[38] Omkar Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. In *BMVC 2015-Proceedings of the British Machine Vision Conference 2015*. British Machine Vision Association.

[39] Seetal Potluri and Aydin Aysu. 2021. Stealing neural network models through the scan chain: A new threat for ml hardware. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–8.

[40] Md Maklachur Rahman. 2013. A DWT, DCT and SVD based watermarking technique to protect the image piracy. *arXiv preprint arXiv:1307.3294* (2013).

[41] Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. 2022. Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. In *2022 IEEE symposium on security and privacy (SP)*. IEEE, 1157–1174.

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[43] samsung.com. 2025. A true AI companion is here. https://www.samsung.com/galaxy-ai/

[44] Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. 2024. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825* (2024).

[45] Meng Shi, Wei Lin, and Wenbo Deng. 2024. Research on Key Techniques for Reverse Engineering of Deep Learning Models for x86 Executable Files. In *Proceedings of the 2024 7th International Conference on Computer Information Science and Artificial Intelligence*. 148–153.

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.

[47] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[48] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*. 269–277.

[49] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 707–723.

[50] Si Wang and Chip-Hong Chang. 2021. Fingerprinting Deep Neural Networks - a DeepFool Approach. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS51556.2021.9401119

[51] Tianhao Wang and Florian Kerschbaum. 2021. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the web conference 2021*. 993–1004.

[52] Xubin Wang, Zhiqing Tang, Jianxiong Guo, Tianhui Meng, Chenhao Wang, Tian Wang, and Weijia Jia. 2025. Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. *Comput. Surveys* (2025).

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[54] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. 2023. Tree-rings watermarks: Invisible fingerprints for diffusion images. *Advances in Neural Information Processing Systems* 36 (2023), 58047–58063.

[55] wikipedia.org. 2025. Stable Diffusion. Retrieved March 31 2025 from https://en.wikipedia.org/wiki/Stable_Diffusion

[56] Cheng Xiong, Chuan Qin, Guorui Feng, and Xinpeng Zhang. 2023. Flexible and secure watermarking for latent diffusion model. In *Proceedings of the 31st ACM International Conference on Multimedia*. 1668–1676.

[57] Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088* (2024).

[58] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. 2021. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*. 14448–14457.

[59] Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. 2024. Huref: Human-readable fingerprint for large language models. *Advances in Neural Information Processing Systems* 37 (2024), 126332–126362.

[60] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. 2021. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *International Conference on Learning Representations*.

[61] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*. 159–172.

[62] Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. 2024. REEF: Representation Encoding Fingerprints for Large Language Models. arXiv:2410.14273 [cs.CL] https://arxiv.org/abs/2410.14273

[63] Jinquan Zhang, Pei Wang, and Dinghao Wu. 2023. Libsteal: Model extraction attack towards deep learning compilers by reversing dnn binary library. In *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*.

[64] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285* (2019).

[65] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. 2023. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137* (2023).

[66] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. 2021. Hermes attack: Steal {DNN} models with lossless inference accuracy. In *30th USENIX Security Symposium (USENIX Security 21)*.