ObfusDomainNet: Reinforcement Learning-Driven Obfuscation and Domain-Locking for DNN Security

Cong Ding[1][0000-0002-6556-1656], and Changsheng Wan[1], Zhenjie Bao[1], Haitao Chen[2] and Jimin Nie[3]

[1] School of cyber science and engineering, Southeast University, Nanjing, 211189, China
[2] Hunan Zhentong Zhiyong Artificial Intelli gence Technology Co., Ltd., Changsha, Hunan, 410026, China
[3] Xingyao Guangyu Technology Co., Ltd., Changzhou, 210019, China
wanchangsheng@seu.edu.cn

Abstract. Protecting Deep Neural Network in fields like medical image analysis is challenging due to unauthorized access and misuse, which traditional methods like watermarking fail to prevent. We propose a reinforcement learning-driven weight obfuscation and key protection mechanism, along with a domain-locked task-specific model generation framework, to enhance DNN authorization protection. The weight obfuscation creates dynamic masks, ensuring only authorized users can access model functions, rendering illegal copies ineffective. The domain-locked framework uses Generative Adversarial Network to maximize performance differences between tasks, ensuring the model excels only on authorized tasks and preventing misuse. Additionally, we embed watermarks into model parameters with hash algorithms for tamper detection and recovery, allowing restoration after attacks. Experimental results show these methods significantly enhance DNN security and reliability in authorization protection, applicability assurance, and tamper recovery, with minimal impact on performance.

Keywords: Deep Neural Networks, Intellectual Property Protection, Reinforcement Learning, Weight Obfuscation, Watermarking.

1 Introduction

In recent years, deep learning has achieved remarkable results in various artificial intelligence domains, such as image recognition, medical image processing, speech recognition, and natural language processing, significantly outperforming traditional methods [1]. The training of high-performance deep neural networks (DNN) demands extensive computational resources, specialized teams, and significant funding, rendering it a highly expensive endeavor [2]. Consequently, these well-trained DNN models hold substantial commercial worth, making the protection of their intellectual property an urgent problem to address [3].

Yet, with the rising value of DNN models, they have increasingly become targets for malicious adversaries [4]. Acts of infringement like model piracy and unauthorized replication and distribution are on the rise, inflicting substantial financial damages on

model proprietors. At present, the approaches to safeguarding the intellectual property of DNN models are primarily divided into two types: proactive protection and passive protection [5]. Passive protection techniques, like watermarking [6] and fingerprinting [7], generally can only retrospectively trace ownership after infringement, failing to prevent unauthorized access and abuse [8].

Constraints of passive protection for models. Assume that a medical tech company has developed a high-performance DNN model used in medical imaging analysis to aid physicians in diagnosing early lung cancer [9]. The model underwent months of training on large-scale medical image datasets, involving substantial computational resources and capital, making it of exceedingly high commercial value. To safeguard the model's intellectual property rights, the company embedded a watermark within the model to track its source and ownership if infringement occurs. However, a malicious actor succeeded in acquiring a copy of the company's model and sold it to another clinic. Despite not paying licensing fees, the clinic could utilize the model for high-accuracy detection of early lung cancer symptoms. Although the company's watermark allows for post-event tracing of the model's origin, it cannot prevent attackers from unauthorized copying, distribution, and usage of the model. Consequently, in such scenarios, watermarking acts merely as a passive protection method and is ineffective in preventing unauthorized access and abuse [10].

Misuse risks associated with authorized users. Conversely, the use of the model by authorized users lacks limitations, implying that legitimately authorized users might apply the model to unauthorized tasks [11]. They frequently prefer to migrate high-performing DNN models to similar tasks to reduce expenses. This practice is both prevalent and highly concealed, potentially violating the model owner's intellectual property. For instance, the company granted legal authorization to a hospital to use the lung cancer diagnosis model to enhance diagnostic efficiency. However, the hospital's medical research team discovered that, with fine-tuning, the model could also be applied to other medical imaging tasks [12], such as early-stage breast cancer detection. As a result, the hospital used the model for breast cancer diagnosis tasks without obtaining extra authorization and did not pay the associated licensing fees. Because such actions usually do not involve altering the model or its watermark, the model owner struggles to detect the misuse in a timely manner.

Challenges in research. Current passive protection approaches are ineffective in preventing the unauthorized uses and abuses described above [13]. While proactive protection methods can somewhat restrict unauthorized access, they often degrade model performance or lack precise control over the usage by authorized users [14]. Thus, it is imperative to develop a model protection mechanism that can prevent illegal use by unauthorized users while effectively restricting the usage scope of legitimate users.

Our approach and contribution:

Usability Authorization: A reinforcement learning-based weight obfuscation and key protection mechanism is proposed, providing dual-layer authorization by generating weight masks that dynamically adjust the obfuscation of model parameters.

Suitability Authorization: A domain-locked, task-specific model framework is introduced, which uses Generative Adversarial Networks (GAN) to generate contrasting domain data that differs from the authorized domain. By optimizing with KL divergence

and Maximum Mean Discrepancy (MMD) loss, the model performs well on the authorized domain while performing poorly on others.

Tamper Recovery: Innovatively embeds a watermark into the less significant bits of the model parameters and generates a unique identifier using hash algorithms to verify the model's integrity and ownership. When the model faces attacks or tampering, it can restore the weights to their original state using the model key.

2 Related Works

During the deployment and utilization of Deep Neural Networks (DNN), two key challenges arise: domain transfer of models and protection of intellectual property rights. Existing research has conducted in-depth explorations around these directions, proposing numerous technical solutions to address challenges such as cross-domain adaptation and authorization control, and model copyright protection.

Domain transfer. In practical applications, models frequently encounter inconsistencies in data distributions across different fields, which is referred to as the domain transfer problem. Depending on whether the model has access to the target domain, it can be categorized into adaptation (DA) and generalization (DG). When the target domain is accessible, DA enhances the model's performance on the target domain by narrowing the distribution gap between the source and target domains. The Domain-Adversarial Neural Network (DANN) proposed by Ganin et al. [15] is a classic adversarial domain adaptation method, which significantly improves performance on the target domain by introducing adversarial training mechanisms to align inter-domain feature distributions. Tzeng et al. [16] proposed Adversarial Discriminative Domain Adaptation (ADDA), which further integrates task-specific supervision to enhance the cross-domain robustness of feature representations. Generative Adversarial Networks (GAN) provide new perspectives for DA. Bousmalis et al. proposed Domain Separation Networks (DSN), which enhance cross-domain adaptation performance by separating shared features and private features [17]. Volpi et al. presented an adversarial data augmentation approach that enhances model robustness on the target domain by generating pseudo target domain samples [18]. Recently, the majority of studies have concentrated on enhancing domain adaptation performance by exploring similarities among different data domains to improve models' transferability across various tasks. When the target domain is inaccessible or its specifics are unknown, DG aims to enhance transferability between the authorized and unauthorized domains by finding an intermediate state. Qiao et al. [19] proposed an authorized domain generalization method based on adversarial domain augmentation, which enhances the model's generalization ability to unknown domains by generating virtual target domain data, and optimizes training efficiency and worst-case constraints using a meta-learning framework and autoencoders. Zhao et al. [20] presented a maximum entropy-based adversarial data augmentation regularization technique, generating virtual target domain data by perturbing the source domain distribution to enhance model generalization and robustness. Li et al. [21] suggested improving single-domain generalization by jointly learning domain expansion and contrastive learning to generate diversified virtual domain data. Xu et al. [22] introduced a visual

representation learning approach based on random convolution, creating new domain data by generating textures while maintaining global shapes, thus improving robustness in downstream tasks. In conclusion, DA and DG approaches improve model generalization by learning the data similarities across different domains. Contrary to DA and DG, our method maximizes the distance between domains, focusing solely on the unique features of the authorized domain, diminishing inter-domain similarities, and weakening the model's generalization ability.

DNN Intellectual Property Protection. According to specific protection strategies, methods for safeguarding DNN intellectual property rights are classified into passive and active protection. Passive protection methods typically only secure the ownership of the model and can trace the ownership after the model's intellectual property has been infringed upon by unscrupulous individuals. Passive protection techniques mainly implement ownership tracing by embedding watermarks into the model. Adi et al. [23] introduced a watermark embedding technique based on trigger samples; the model outputs specified results when it receives particular inputs, serving for ownership verification. Uchida et al. proposed embedding watermarks into the secondary bits of model parameters and extracting the watermark using a key [24]. To strengthen the robustness of watermark techniques, Rouhani et al. [25] proposed a frequency-domain watermark embedding method, which, by encoding watermarks into spectral features, significantly enhances resistance to attacks such as pruning and fine-tuning. Liu et al. [3] proposed creating trigger set watermarks by integrating Fourier spectrum analysis and random perturbations. However, existing watermarking technologies are mostly limited to ownership verification and lack integration with model tampering detection and recovery. Active protection technologies aim to directly restrict model access or usage. For example, weight obfuscation restricts unauthorized use by adding noise or perturbations to model weights. Xue et al. [13] selected a small subset of model parameters for encryption via adversarial perturbations, with the perturbation values serving as keys given to authorized users to decrypt the model, thereby preventing unauthorized access. Liu et al. embedded digital passports within DNN models, ensuring that any attempt to forge the passport significantly degrades the model's inference performance on the original task. This method exhibits strong robustness against model fine-tuning and obfuscation attacks [26]. Recently, Xue et al. [27] proposed using backdoor attacks combined with sample triggers to generate model keys, which can effectively prevent unauthorized access and simultaneously extract fingerprints related to user identity, achieving user fingerprint management.

3 Our Proposed Method

The overview of the proposed method is shown in **Fig. 1**, including the domain-locked model generation module, the reinforcement learning(RL) driven weight obfuscation module, and the watermark embedding and recovery module. The authorized domain refers to the data distribution that the model owner allows legitimate users to use for inference. The victim model is the original, unaltered model completed by the model owner after training. The domain-locked model is one that has been

adjusted from the original innocent model to perform optimally on the authorized domain while performing poorly on other unauthorized domains, thereby restricting the model's effectiveness to the authorized domain.

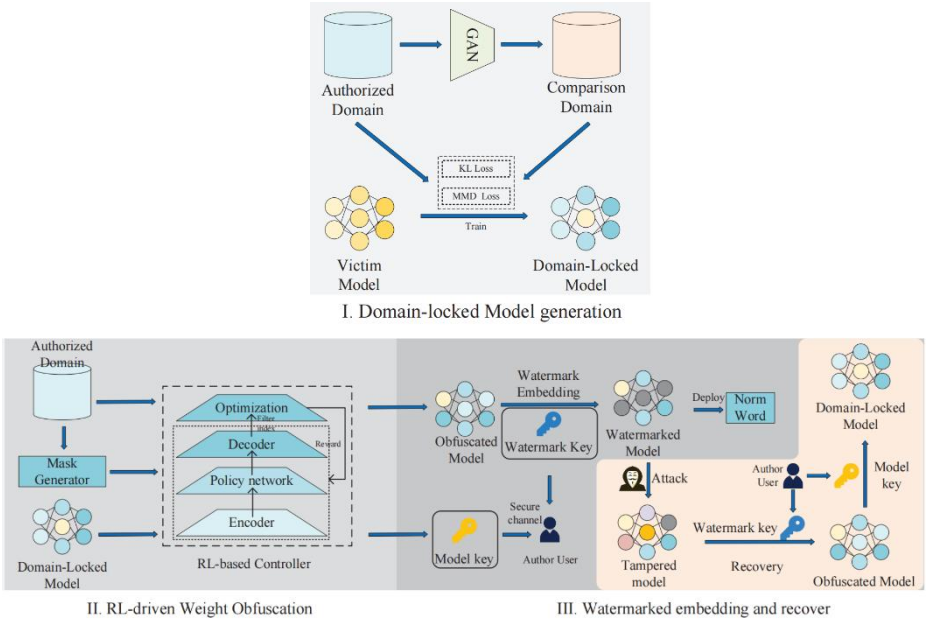


Fig. 1. The overview of the proposed method

3.1 Domain-locked Model generation

Inspired by Conditional GAN (CGAN) and Information GAN (InfoGAN), we control the generation process via labels and maximize mutual information to ensure the generated data contains meaningful label information. Specifically, the method proceeds as follows:

Setting multiple distance thresholds. To generate data at various distances, we define multiple upper limits DIS, representing a set of different distance thresholds (dis), to control the distance range of the generated data. By specifying different dis values, we can produce augmented data at varying distances from the authorized data domain, i.e., comparative domain data.

Defining optimization objectives. For each distance threshold dis, we freeze the parameters of discriminator D and establish the following optimization goals to adjust the output of generator G.

$$L_{\text{aug}} = -\min(\, dis, \text{MMD}(P_{x\sim P_X^S}(D_z(x)), P_{y\sim P_Y^S}(D_z(G(\text{noise}, y))))) + \mathbb{E}_{y\sim P_Y^S}\, D_{\text{CE}}(D_m(G(\text{noise}, y)), y) \tag{1}$$

Here, z represents the representation output by the feature extractor, and $MMD(P_{x\sim P_X^S}(D_z(x)), P_{y\sim P_Y^S}(D_z(G(noise,y))))$ denotes the Maximum Mean Discrepancy (MMD) distance between the authorized domain data and the contrast domain data. D_{ce} stands for cross-entropy loss, which constrains the generated data to preserve the semantic information of the authorized domain.

Generation of Augmented Data at Different Distances. After optimization, by inputting Gaussian noise and labels sampled from the label distribution P_y into generator G, augmented data that meet different distance requirements can be generated. Ensure that the generated data retains semantic information at different distances, while controlling the MMD distance to generate diverse data with varying distances from the source domain, thereby enhancing the model's adaptability to different data distributions.

Domain-Locked Model Generation. We consider an authorized data domain $S = \{(x,y)|x\tilde{}P_X^S, y\tilde{}P_Y^S\}$, where P_x and P_y represent the distributions of inputs and labels, respectively. Additionally, there is a contrast data domain generated using the previously mentioned GAN method $A = \{(x,y)|x\tilde{}P_X^A, y\tilde{}P_Y^A\}$, where x represents data and y represents data labels. Input the authorized domain data S and contrast data domain A into the DNN, whose architecture comprises a feature extractor and a classifier. In the information flow of representation learning, the model is required to extract representations associated with authorized domain information. The mutual information I(z;n) reflects the amount of information related to domain features n (i.e., authorized data domain or contrast data domain) contained within the representation z. By maximizing I(z;n), we introduce more domain-related information into the representation z, thereby increasing the model's dependence on the authorized data domain. Consequently, the model exhibits well performance on the authorized domain but performs poorly on the contrast data domain.

To preserve the model's good performance on the authorized data domain while diminishing its performance on the contrast data domain. First, employ KL divergence loss to measure the error between the model's predictions and the true labels:

$$L^* = L_S - \min(\beta, \alpha \cdot L_A) \tag{2}$$

Here, L_S represents the model's loss on the authorized data domain, L_A denotes the loss on the contrast data domain, α is the scaling factor, and β is the upper limit of the loss, controlling the value of L_A within a reasonable range to prevent it from becoming excessively large.

Secondly, to further ensure domain dependency in z and increase the representation distance between domains, Maximum Mean Discrepancy (MMD) is employed to measure the distance between the distributions of the authorized data domain and the contrast data domain, thereby enhancing the representation differences between the two data domains. The Gaussian kernel estimator MMD(P,Q;exp) based on RKHS is utilized to compute the distribution differences between domains. The optimization objective is:

$$L^* = L_S - \min(\beta, \alpha \cdot L_A \cdot L_{dis}) \tag{3}$$

Here, L_{dis} is the MMD-based penalty term, which indirectly increases I(z;n) by enlarging the distribution differences between domains. Ultimately, this optimization

design ensures the sufficiency of the source domain and strengthens the performance constraints on the auxiliary domain.

3.2 RL-driven Weight Obfuscation

Mask Generation. By introducing a mask M to identify the weights that require obfuscation, we strive to minimize the number of obfuscated weights, thereby decreasing the security risks involved in storing and transmitting obfuscated weights as keys. The definition of mask M is as follows:

$$M(\omega_i) = \begin{cases} 1, & |\omega_i - c| \leq \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Here, ω_i represents the weights, c and ε are adjustable hyperparameters. Through this mask, we can selectively obfuscate the weights falling within the range $[c - \varepsilon, c + \varepsilon]$, while leaving the weights outside this range unchanged. With the mask, we can optimize the weight changes and generate new weight changes.

$$\Delta\omega' := \Delta\omega \odot M \tag{5}$$

Here, $\odot$ denotes element-wise multiplication in a matrix. Through the careful selection of constant c, the effect of weight obfuscation can be distributed across various layers of the model. This distributed approach notably increases the model's resilience to potential attacks, such as fine-tuning or training with norm regularization resistance. Subsequently, l_1 norm regularization is applied to the adjusted weight changes, encouraging sparsity in the weight changes, i.e., reducing the number of non-zero elements. Only important weight changes are stored, lowering the security risks associated with transmitting them as keys and compressing storage requirements. Integrating the distributed design of mask M with l_1 norm regularization enables weight obfuscation while lessening reliance on storage resources.

Optimization Module. Through the previous mask generation module, we can determine the weights that need to be obfuscated and the magnitude of the new weight changes. To enhance the concealment of obfuscation, we add the weight changes $\Delta\omega'$ to the weights that need to be obfuscated and restrict the range of the changed weights $\omega + \Delta\omega'$ within the upper and lower bounds of the original weights ω, ensuring the altered values do not deviate from the original model's range. This constraint is governed by two hyperparameters, γ and η. The optimal weight value changes with the minimum weight selection can be obtained through the following loss function $J(\Delta\omega')$:

$$\min_{\Delta\omega'} J(\Delta\omega') = -F(g(x; \omega + \Delta\omega'), y) + \mu\|\Delta\omega'\|_1 \tag{6}$$

where g represents the functionality of the neural network model (e.g., classification accuracy); x denotes the input training sample, and y signifies the corresponding label; indicates the l_1 norm of the weight adjustments, aimed at controlling the sparsity of weight modifications; μ is the regularization parameter used to balance the impact of the number of weight modifications and model performance.

The optimization process is subject to the following constraints:

$$\gamma \cdot \min\{v_i\} \leq v_i + \Delta\omega' \leq \eta \cdot \max\{v_i\} \tag{7}$$

Here, v_i represents the parameter value of a single weight, γ and η are used to constrain the upper and lower bounds of the model weights.

State-of-the-art DNN models typically contain millions of parameters, making it impractical to manually design the optimal mask for each filter in real-world applications. Although the range of parameter obfuscation has been determined, the selection of the optimal filters still requires further design and consideration

Reinforcement Learning-Based Controller. Parameters in DNN models often exhibit significant redundancy, with varying contributions from different filters to the model's functionality. For instance, filters learning textures or repetitive patterns (like clouds in the sky or tiles on walls) typically influence the model less than filters learning distinctive features of critical objects (like facial features or brand logos), as the former often lack discriminative power, whereas the latter are closely tied to the essential objectives of classification or detection. Drawing from this, we introduce a reinforcement learning-driven controller to identify key filters with substantial influence on model functionality and to generate the respective masks.

In the reinforcement learning framework, the controller acts as an agent, selecting a filter index k for each layer l, where $k \in [1, F(l)]$, and $F(l)$ represents the total number of filters (or output channels) in layer l . As $F(l)$is dictated by the architecture of the target model M , the agent operates in a static environment. If n filters are selected per layer, the agent executes $n \times L$ actions in total (where L is the number of layers). All agents share a controller with weight parameters θ , whose optimization goal is to maximize the expected reward $J(\theta)$:

$$J(\theta) = \mathbb{E}_{\pi(a_{1:n\times L};\theta)}[R] \tag{8}$$

Here, $\pi(a;\theta)$ represents the policy distribution given the controller parameters $theta$ for executing actions; J is the reward signal guiding the controller. Here, $\pi(a;\theta)$ denotes the policy distribution for performing a under the controller parameters θ, and R acts as the reward signal directing the controller.

The reward R encourages the controller to select filters that significantly reduce model accuracy after obfuscation, and it is defined as follows:

$$R = -ACC_{val}(f(x^{val}; \omega + \Delta\omega), y^{val}) \tag{9}$$

Here, ACC_{val} represents the prediction accuracy of model f on the validation dataset $D_{val} = \{(x^{val}, y^{val})\}$. The expected reward $J(\theta)$ is optimized using the policy gradient algorithm in reinforcement learning, transforming the reward maximization problem into a loss minimization problem:

$$L_c(\theta) = -\frac{1}{m}\sum_{j=1}^{m}\sum_{t=1}^{n\cdot L} \ln \pi(a_t|\theta)\cdot(R_j - b) \tag{10}$$

Here, m represents the number of trajectories in each training phase, b is the baseline value used to reduce the variance of reward estimation, and R_j denotes the reward of the j-th trajectory.

3.3 Watermark Embedding and Recovery

Deep neural network (DNN) models have numerous parameters with redundancies, allowing watermark embedding without affecting performance. These parameters are stored as 32-bit floating-point numbers following the IEEE 754 standard, where the sign and exponent bits significantly impact parameter values, while the 23-bit mantissa affects only the fractional part. Since modifying the mantissa has minimal impact, model watermarks can be embedded into these bits without altering overall functionality.

Watermark Embedding: The process of generating recovery bits begins by extracting all the sign and exponent bits from the model parameters θ . These extracted bits, known as the primary bits and denoted by $C = \{c_1, c_2, \cdots, c_n\}$ serve as the foundation for embedding. Following this, the recovery bits $R = \{r_1, r_2, \cdots, r_n\}$ are obtained through matrix multiplication. The computation of the recovery bits is performed using the formula below:

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = K \times \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \tag{11}$$

Here, K is a pseudorandom binary matrix of size $m \times n$ generated by the key m_k.

The method identifies tampered positions by comparing hashes generated using hash functions (e.g., SHA-256). Specifically, the model parameters and recovery bits are divided into g groups, each containing u elements. For each group, primary bits C and recovery bits R are processed through an encrypted hash function to generate h hash bits. These hash bits, combined with the recovery bits, form a watermark. By embedding this watermark into the secondary bits of the model, a watermarked model $\tilde{\theta}$ is created.

Once authorized users receive the watermarked model, they can restore the obfuscated and tampered parameters using the key m_k. The watermark is first extracted from the model's secondary bits, which include recovery and hash bits, and is divided into g groups along with the primary bits. For each group, the extracted hash bits are compared with recalculated hash bits to identify tampered recovery and primary bits. If the extracted hash bits match the recalculated ones, the group is marked as intact; otherwise, it is flagged as tampered. Assuming the total number of recovery bits in initially identified intact groups is ψ, the formula for recovery bits is reformulated as:

$$\begin{bmatrix} r_{t(1)} \\ r_{t(2)} \\ \vdots \\ r_{t(\psi)} \end{bmatrix} = \tilde{K} \times \tilde{C} \tag{12}$$

Here, $\tilde{K}$ is a matrix composed of rows selected from K (generated by the key m_k). These rows correspond to the complete ψ recovery bits, where $t(\cdot)$ represents the indices of complete recovery bits. At this point, the extraction of the higher bits $\tilde{C}$can essentially be divided into complete and tampered primary bits. Thus, the formula can be further refined as follows:

$$\begin{bmatrix} r_{t(1)} \\ r_{t(2)} \\ \vdots \\ r_{t(\psi)} \end{bmatrix} - \tilde{K} \times \tilde{C} = \hat{K} \times \hat{C} \tag{13}$$

Here, $\tilde{K}$ and $\hat{K}$ are matrices composed of the columns corresponding to the primary bits . In this formula, all components except for $\hat{C}$ are accurate, whereas $\hat{C}$ corresponds to the primary bits that require restore. Therefore, recovering the tampered primary bits simplifies to solving for matrix $\hat{C}$. As long as this formula has a unique solution, the tampered can be completely restored.

4 Experiments

4.1 Implementation Detail

All experiments were implemented using Python's PyTorch deep learning framework. The training and evaluation processes were conducted on a workstation equipped with an NVIDIA RTX 4060 GPU, ensuring efficient computation and acceleration of deep neural network training.

Dataset. To validate the effectiveness and generality of our proposed method, experiments were conducted on several benchmark datasets: CIFAR-10, STL-10, MNIST, USPS, SVHN, MNIST-M, SYN-D and Fashion-MNIST [28].

DNN Model. We employed various classic convolutional neural network architectures as baseline models to demonstrate the applicability of our method across different network structures. VGG-11 and VGG-13 [29]: Deep convolutional networks with 11 and 13 layers, renowned for their simplicity and effectiveness in image recognition tasks. ResNet-18: An 18-layer residual network that uses skip connections to address the vanishing gradient problem, allowing for the training of deeper networks. AlexNet: One of the pioneering deep learning models in image classification, consisting of 8 layers, composed of convolutional and fully connected layers.

Hyper-parameters Setting. In the mask generation process within the RL-driven weight obfuscation module, c and ε are hyperparameters that should be predefined according to domain knowledge. In our experiments, we set c to the average of the median values of the weight distributions in each layer of the model, ε can be regarded as an adjustment term for c, primarily serving to maintain the value of c near the median of the weight distribution, thereby ensuring the recoverability of the model. Furthermore, the size of each group in the watermark embedding and recovery modules is set to $u = 16$.

4.2 Domain-Locking Effect Validation

As illustrated in the **Table 1**, the experimental results confirm the effectiveness of the domain-locking mechanism on various datasets. Within the authorized domain, the model demonstrates outstanding classification performance. For example, it achieves classification accuracies of 85.208% and 98.125% on the CIFAR-10 and MNIST

datasets, respectively, highlighting the domain-locking mechanism's ability to support tasks in the authorized domain. Concurrently, in the unauthorized domains, the model's performance significantly declines. For instance, the accuracy in the unauthorized domain of CIFAR-10 is only 11.042%, and the accuracies in the unauthorized domains of other datasets are all below 12%.

Table 1. Comparison of model performance across datasets.

Dataset	Model	Authorized Domain(%)	Unauthorized Domain(%)	Drop
CIFAR-10	VGG-13	85.208	11.042	74.166
STL	VGG-13	81.458	10.625	70.833
MNIST	VGG-11	98.125	11.25	86.875
USPS	VGG-11	98.854	9.062	89.792
SVHN	VGG-11	99.152	10.324	88.828
MM	VGG-11	91.632	11.258	80.374
SYN	VGG-11	99.324	11.125	74.166

This significant performance disparity indicates that the domain-locking mechanism successfully restricts the model's effectiveness in unauthorized domains, ensuring that the model can only function correctly within specific authorized domains, thereby achieving the goal of applicability authorization. Additionally, this method consistently restricts the model's performance to the authorized domains across different data distributions and model architectures, further demonstrating the method's effectiveness.

4.3 RL-Driven Weight Obfuscation Experiment

Table 2. Performance comparison of different models on multiple datasets before and after parameter obfuscation.

Dataset	Model	Before		After		Random
		Acc. (%)	Para.	Acc.(%)	ObfusPara	Acc. (%)
CIFAR-10	VGG-11	93.72	28.14	10.13	827	91.20
	ResNet-18	93.07	11.32	10.10	289	90.30
	VGG-13	93.55	38.26	10.24	841	88.32
STL	VGG-11	93.72	28.14	10.05	412	91.20
	ResNet-18	93.07	11.32	10.23	357	82.13
	VGG-13	93.55	38.26	10.00	457	91.43
Fashion-MNIST	VGG-11	93.41	28.14	10.16	324	90.25
	ResNet-18	93.23	11.32	10.55	236	91.12
	VGG-13	92.11	38.26	10.08	360	87.15

As illustrated in the **Table 2**, prior to weight obfuscation, the model attains accuracy rates between 92% and 94% on datasets including CIFAR-10, STL, and Fashion-MNIST, indicating that the model parameters maintain high baseline performance in the absence of obfuscation. However, following weight obfuscation, the accuracy of all

model types across all datasets declines to around 10.00%, corresponding to random guessing levels. Selecting only a minimal number of critical weights can reduce the model's performance to the level of random guessing, significantly diminishing the model's baseline performance. Additionally, randomly selecting a number of weights equal to the number of obfuscated weights in the model only results in a very limited decrease in model performance. Moreover, the total number of obfuscated weights across all models is less than 1K, greatly reducing the space occupancy during key storage and enhancing the security of key distribution. Experimental results demonstrate that the reinforcement learning-driven weight obfuscation mechanism exhibits high effectiveness across multiple datasets and model architectures.

Fig. 2 shows how the top-1 accuracy in the ObfusDomainNet framework changes as the number of obfuscated weights increases. It is evident that as more weights are obfuscated, model accuracy drops significantly, eventually approaching random guessing levels, demonstrating the effectiveness of the obfuscation strategy in reducing model performance. In contrast, the blue dashed line representing randomly selected weights shows little impact on model accuracy, remaining relatively stable. This indicates that targeted weight obfuscation is far more effective than random selection, significantly reducing model accuracy while enhancing security and minimizing key storage requirements.

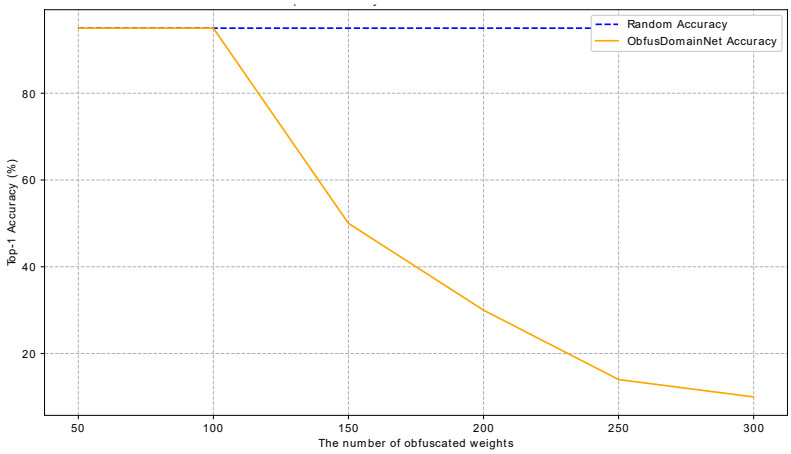


Fig. 2. Top-1 Accuracy and Model secrets count

4.4 Watermark Embedding and Extraction

As shown in **Table 3**, the detection performance of the watermarked model is evaluated at various tampering rates. On the CIFAR-10, STL, and Fashion-MNIST datasets, for different levels of weight tampering (0.1%, 1%, 10%), the watermark detection rate of all models (VGG-11, ResNet-18) was 100%. This indicates that the watermark mechanism can accurately locate tampering positions, effectively verifying the integrity of the model regardless of the degree of tampering. This confirms the robustness and reliability of the proposed watermark embedding and detection mechanism. After

watermark embedding, as shown in **Table 4**, the performance of the obfuscated model is only slightly affected. For instance, on the CIFAR-10 dataset, the obfuscated model without watermarking had an accuracy of 10.01%, whereas the obfuscated model with watermarking had an accuracy of 9.92%, representing a performance drop of merely 0.09%. Similarly, on the STL and Fashion-MNIST datasets, the change in model performance ranged from 0.1% to 0.3%. This suggests that watermark embedding has an insignificant effect on the performance of the obfuscated model, further confirming the method's practical usability.

Table 3. Detection Rate with Different Tampering Rates

Model	Dataset	Tampering Rate(%)	Detection Rate(%)
VGG-11	CIFAR-10	0.1	100
	STL	1	100
	Fashion-MNIST	10	100
ResNet-18	CIFAR-10	0.1	100
	STL	1	100
	Fashion-MNIST	10	100

Table 4. Comparison of obfuscated and watermarked model accuracies on different datasets.

Model	Dataset	Obfuscated Model Accuracy (%)	Watermarked Model Accuracy (%)
VGG-11	CIFAR-10	10.01	9.92
	STL	10.12	10.14
	Fashion-MNIST	10.00	9.48
ResNet-18	CIFAR-10	10.00	9.23
	STL	10.10	10.55
	Fashion-MNIST	10.00	11.20

The experimental results indicate that the proposed watermarking mechanism demonstrates robust detection capabilities under different levels of weight tampering, while minimally affecting the obfuscated model's performance, thereby not compromising the protection of model usability and applicability authorization. Additionally, post watermark embedding, the model retains high levels of security and reliability against attacks, thoroughly validating the method's practicality and effectiveness. These characteristics make the mechanism suitable for various practical scenarios that require model copyright protection and tampering detection.

4.5 Model Tampering and Recovery

In this experiment, we simulate hacker attacks by randomly selecting and tampering with different proportions of parameters in watermarked models. The tampering method involves setting selected weights to zero, though other values yield similar

results. To restore the model, we first use the watermark key and recovery bits to recover the obfuscated model, then apply the model key to correct erroneous weights, ultimately achieving the restored domain-locking model.

Table 5. Model performance under 1% tampering rate across different datasets.

Model	Dataset	1% Tampering Rate Accuracy				
		Watermarked Model(%)	Tampered Model	Authorized Domain (%)	Unauthorized Domain (%)	Domain-locked Model (%)
VGG-11	CIFAR-10	8.23	9.21	93.021	10.18	93.031
	STL	9.17	9.12	91.064	11.23	91.235
	Fashion-MNIST	10.82	10.25	98.156	10.01	98.247
ResNet-18	CIFAR-10	9.53	9.32	98.032	9.98	99.587
	STL	9.87	9.58	98.163	10.34	98.232
	Fashion-MNIST	10.56	10.16	99.623	10.21	99.711

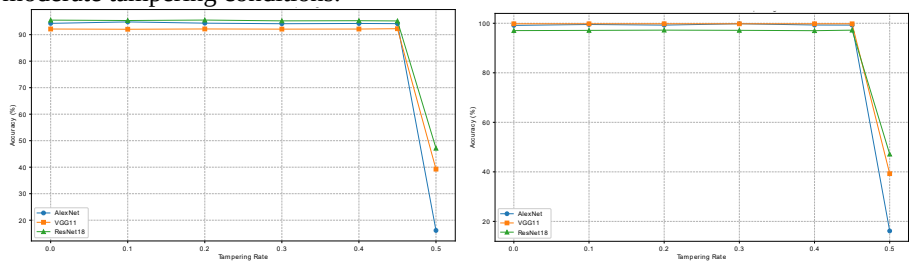
This experiment tests the watermarking mechanism and model keys for restoring tampered models. We simulated 1% random weight tampering on various datasets, then evaluated model performance after tampering and recovery using the watermark key and model key. Results in **Table 5** show that the watermarked model's accuracy is initially around 10%, equivalent to random guessing. After tampering, accuracy decreases but remains at random guessing levels. Using the watermark key and recovery bits, the model's performance is largely restored, with restored models achieving high accuracy, such as 93.021% for VGG-11 and 99.587% for ResNet-18 on CIFAR-10. The recovery mechanism demonstrates consistent results across all datasets, showing high reliability in repairing tampering. Restored models achieve over 91% accuracy in the authorized domain, closely matching the original domain-locked model, while accuracy drops to random guessing in unauthorized domains. These findings highlight the robustness of the watermarking mechanism in detecting tampering and restoring model performance with minimal impact, effectively preserving domain-locking features and safeguarding model authorization.

4.6 Integration experiments

In this experiment, all previously proposed modules were integrated to evaluate the recovery performance of a watermark-protected model under different tampering rates, using watermark keys and model keys. **Fig. 3**, respectively show the performance of the recovered models on CIFAR-10 and Fashion-MNIST datasets under varying tampering rates. Each figure evaluates three model architectures (AlexNet, VGG11, and ResNet18) to examine the impact of tampering on model accuracy.

The results show that for all datasets and model architectures, the model accuracy remains high when the tampering rate is below 0.45. These findings indicate that the

proposed integrated method demonstrates robustness against a certain level of tampering. Before reaching a tampering rate of 0.45, the models exhibited strong resistance to interference, confirming that the method can effectively maintain model integrity under moderate tampering conditions.



CIFAR-10 Fashion-MNIST

Fig. 3. The model performances on CIFAR-10, Fashion-MNIST datasets under different tampering rates

4.7 Comparison Experiments

To further verify the superiority of the proposed ObfusDomainNet, we compare our method with several representative model protection approaches, including watermark-based methods [23-24] and proactive protection methods [13][26].

For fair comparison, all methods are evaluated under the same experimental settings on two datasets, namely CIFAR-10 and STL. We report the classification accuracy in the authorized domain and unauthorized domain, as well as whether the method supports tamper recovery.

Table 6 summarizes the comparison results. It can be observed that most baseline methods achieve relatively high accuracy in both authorized and unauthorized domains. Although such methods may provide certain ownership protection or model obfuscation capability, they fail to effectively restrict the model usage to the authorized domain. This indicates that these methods cannot fundamentally prevent an adversary from directly exploiting the stolen model in out-of-domain or unauthorized scenarios.

In contrast, Our proposed method maintains high classification accuracy in the authorized domain. More importantly, its accuracy in the unauthorized domain drops dramatically to 10.18% on CIFAR-10 and 11.23% on STL. This large performance gap demonstrates that the proposed method can effectively bind the model functionality to the authorized domain, thereby significantly reducing the practical usability of the model once it is illegally obtained or deployed in unauthorized scenarios.

Another notable advantage of our method lies in the tamper recovery capability. As shown in **Table 6**, none of the compared methods supports recovery after model tampering, whereas our method can successfully recover the model. This shows that our method not only provides proactive protection by restricting unauthorized usage, but also offers passive verification and restoration capability through watermark-based tamper recovery. Therefore, the proposed framework achieves a more comprehensive form of model intellectual property protection than existing methods.

Table 6. Comparison with representative protection methods on CIFAR-10 and STL.

Method	CIFAR-10		STL		Tamper Recovery
	Auth.Acc.(%)	Unauth.Acc.(%)	Auth.Acc.(%)	Unauth.Acc.(%)	
[23]	90.34	89.37	88.65	76.91	×
[24]	92.47	90.17	90.57	87.27	×
[13]	94.59	93.22	84.74	72.18	×
[26]	93.18	90.37	85.42	77.39	×
[27]	92.53	91.64	87.17	86.37	×
Ours	93.02	**10.18**	91.07	**11.23**	√

5 Conclusion

This paper introduces an innovative intellectual property protection framework for Deep Neural Networks (DNN), which combines reinforcement learning-driven weight obfuscation, domain-locking task-specific model generation based on Generative Adversarial Network (GAN), and watermark embedding and recovery technologies. Firstly, the reinforcement learning-driven weight obfuscation mechanism dynamically adjusts the extent of parameter obfuscation, ensuring that once the model is obtained by unauthorized users, the core functionalities remain locked, thereby significantly enhancing the model's proactive protection capabilities. Secondly, the domain-locking model generation framework maximizes the performance differences between authorized and unauthorized domains, ensuring that the model performs excellently only on specific tasks. This effectively prevents legitimate users from misusing the model for unauthorized tasks, thereby avoiding implicit infringements of the model's intellectual property. Lastly, the watermark embedding and recovery mechanism, in conjunction with hash algorithms, not only improves the accuracy of model tampering detection but also offers the ability to repair tampered models, thereby providing reliable protection for the integrity of DNN models.

The experimental results indicate that the proposed method outperforms in protecting model intellectual property, increasing authorization flexibility, and detecting tampering, effectively preventing unauthorized use and infringement. Future research can further optimize the watermark embedding and recovery mechanisms to accommodate more complex attack scenarios, while also exploring strategies to further enhance model security across different domains and tasks.

Acknowledgments. This study was funded by the State Key Laboratory of Particle Detection and Electronics (Grant No. SKLPDE-KF-202314), ZTE Industry-University Institute Cooperation Funds (Grant No. 2023ZTE08-03), the Key Program of Fundamental Research in Jiangsu Province (Grant No. BK20253038), the National Social Science Fund of China (Grant No. 21BGL091) and the Fundamental Research Funds for the Central Universities (Grant No. 2242025K30025).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
2. Li, P., Huang, J., Wu, H., Zhang, Z., Qi, C.: Securenet: Proactive intellectual property protection and model security defense for DNNs based on backdoor learning. Neural Networks **174**, 106199 (2024)
3. Liu, Y., Wu, H., Zhang, X.: Robust and imperceptible black-box DNN watermarking based on fourier perturbation analysis and frequency sensitivity clustering. IEEE Transactions on Dependable Secure Computing **21**, 5766–5780 (2024)
4. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: 25th USENIX security symposium (USENIX Security 16). pp. 601–618 (2016)
5. Wang, Z., Ma, Z., Feng, X., Sun, R., Wang, H., Xue, M., Bai, G.: Corelocker: Neuron-level usage control. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 222–222. IEEE Computer Society (2024)
6. Zhang, J., Chen, D., Liao, J., Ma, Z., Fang, H., Zhang, W., Feng, H., Hua, G., Yu, N.: Robust model watermarking for image processing networks via structure consistency. IEEE Transactions on Pattern Analysis Machine Intelligence **46**, 6985 6992 (2024)
7. Mo, M., Wang, C., Guo, Q., Bian, S., Huang, Q., Zhang, X.: A novel robust black box fingerprinting scheme for deep classification neural networks. Expert Systems with Applications **252**, 124201 (2024)
8. Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M.P., Huang, H., Molloy, I.: Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia conference on computer and communications security. pp. 159–172 (2018)
9. Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., Van Der Laak, J.A., Van Ginneken, B., Sánchez, C.I.: A survey on deep learning in medical image analysis. Medical image analysis **42**, 60–88 (2017)
10. Lin, N., Chen, X., Lu, H., Li, X., Systems: Chaotic weights: A novel approach to protect intellectual property of deep neural networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems **40**(7), 1327–1339 (2020)
11. Mukherjee, R., Chakraborty, R.S.: Attacks on recent DNN ip protection techniques and their mitigation. IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems **42**(11), 3642–3650 (2023)
12. Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J.: Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE transactions on medical imaging **35**(5), 1299–1312 (2016)
13. Xue, M., Wu, Z., Zhang, Y., Wang, J., Liu, W.: AdvParams: An active DNN intellectual property protection technique via adversarial perturbation based parameter encryption. IEEE Transactions on Emerging Topics in Computing **11**(3), 664–678 (2022)
14. Wang, H., Chi, H., Yang, W., Lin, Z., Geng, M., Lan, L., Zhang, J., Tao, D.: Domain specified optimization for deployment authorization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5095–5105 (2023)
15. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of machine learning research **17**(59), 1–35 (2016)
16. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7167–7176 (2017)

17. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016)
18. Volpi, R., Namkoong, H., Sener, O., Duchi, J.C., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. Advances in neural information processing systems **31** (2018)
19. Qiao, F., Zhao, L., Peng, X.: Learning to learn single domain generalization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12556–12565 (2020)
20. Zhao, L., Liu, T., Peng, X., Metaxas, D.: Maximum-entropy adversarial data augmentation for improved generalization and robustness. Advances in Neural Information Processing Systems **33**, 14435–14447 (2020)
21. Li, L., Gao, K., Cao, J., Huang, Z., Weng, Y., Mi, X., Yu, Z., Li, X., Xia, B.: Progressive domain expansion network for single domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 224–233 (2021)
22. Xu, Z., Liu, D., Yang, J., Raffel, C., Niethammer, M.: Robust and generalizable visual representation learning via random convolutions. arXiv preprint arXiv:.13003 (2020)
23. Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J.: Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1615–1631. USENIX Association, Baltimore, MD (Aug 2018)
24. Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S.: Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on international conference on multimedia retrieval. pp. 269–277 (2017)
25. Rouhani, B.D., Chen, H., Koushanfar, F.: Deepsigns: A generic watermarking framework for ip protection of deep learning models. arXiv preprint arXiv:.00750 (2018)
26. Fan, L., Ng, K.W., Chan, C.S.: Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. Advances in neural information processing systems **32** (2019)
27. Xue, M., Wu, Y., Zhang, L.Y., Gu, D., Zhang, Y., Liu, W.: SSAT: Active authorization control and user's fingerprint tracking framework for DNN IP protection. ACM Transactions on Multimedia Computing, Communications Applications **20**(10), 1 24 (2024)
28. Seo, Y., Shin, K.s.: Hierarchical convolutional neural networks for fashion image classification. Expert systems with applications **116**, 328–339 (2019)
29. Kazerooni-Zand, R., Kamal, M., Afzali-Kusha, A., Pedram, M.: Memristive-based mixed-signal cgra for accelerating deep neural network inference. ACM Transactions on Design Automation of Electronic Systems **28**(4), 1–25 (2023)