Quest Journals
Journal of Research in Applied Mathematics
Volume 11 ~ Issue 4 (April 2025) pp: 18-35
ISSN (Online): 2394-0743 ISSN (Print): 2394-0735
www.questjournals.org

Review Paper

Research on Integrity Verification Methods for Deep Learning Models Based on Adversarial Samples

Jianlin Lu • QingSong Yang • Mei Yu
(School of Mathematical Sciences, Guizhou Normal University, Guiyang 550001, China)

ABSTRACT: The widespread application of artificial intelligence technology has made the protection of deep learning models a core demand in the industry. Existing model watermarking schemes typically involve modifying the model itself to embed watermark information. While this approach can protect the model's copyright, current research on the interpretability of deep learning models is still limited, making it unclear what impact the modifications to the model will have. This paper proposes an integrity verification method for deep learning models based on multi-objective adversarial examples. The basic idea of this method is to utilize the characteristic of adversarial examples being close to the decision boundary. By adding controllable perturbations to adversarial examples, watermark samples are generated. The watermark samples generated by this method can accurately capture slight changes in the decision boundary. Unlike other similar methods, the adversarial examples used in this method are targeted adversarial examples, which allow for specific protection based on the key targets provided by the user. Specifically, multiple adversarial examples are generated in different directions using the key target as the initial objective, followed by perturbation, such that the generated watermark samples linearly cover the decision boundary of the key target. When the model is tampered with and the decision boundary changes, the watermark samples will also change, thereby achieving the verification of model integrity. The above scheme does not modify the model itself, but instead adds watermarks based on the characteristics of adversarial examples. This approach maximizes the preservation of the model's integrity and effectively avoids the risks associated with potential functionality loss caused by watermark embedding.

Key words: Deep learning model; adversarial example; integrity verification.
Mathematics Subject Classification: 65D18 · 68U10 · 94A08

***Received** 09 Apr., 2025; **Revised** 17 Apr., 2025; **Accepted** 19 Apr., 2025 © The author(s) 2025.*
Published** with open access at **www.questjournas.org

1 Introduction

1.1 Research Background and Significance

With the development of technology, modern society has entered the intelligent era. The development and popularization of the internet have led to an explosive growth of information, which is vast in quantity and diverse in types. Processing this information requires a large amount of human resources and relevant technologies. Efficiently processing this information is particularly crucial for the high-quality development of modern society. In 2006, Hinton et al. [1] introduced the concept of "deep learning", which attracted widespread attention in the academic community. This technology is built upon neural network technology and has achieved significant breakthroughs in fundamental fields such as computer vision and natural language processing. In recent years, with a dramatic increase in computing power, continuous optimization of algorithm models, and a growing number of professionals in the field, the application scope of deep learning technology is gradually expanding.

In recent years, the field of deep learning model security research has begun to explore new defense mechanisms, with digital watermarking-based solutions (i.e., "deep learning model watermarking" [2]) gradually becoming an important technological approach. The vulnerability watermarking technical route focuses on the integrity verification of digital carriers, which not only needs to identify whether there is any integrity damage to the data but also, in some advanced algorithms, can locate tampered area coordinates through feature comparison. This watermark protection approach has provided ideas for deep learning model watermarking. As deep learning models are widely applied in critical fields, the challenge of effectively building a watermarking technology system that balances both robustness and vulnerability has become a key bottleneck in securing model deployment. This involves theoretical breakthroughs at the algorithmic level and requires practical solutions to the compatibility issues between model integrity and model watermarking.

Deep neural networks (DNN) are computational frameworks built on input-output mappings, where the parameter matrices form the core of the model. This structure differs fundamentally from conventional digital media. When traditional watermarking techniques are mechanically applied to neural networks for protection, they can easily lead to performance degradation and security vulnerabilities, creating a chain reaction of issues.In the case of traditional digital media, such as images, their pixel arrays exhibit spatial continuity, and small disturbances to local parameters typically do not significantly alter human visual perception. In contrast, the network structure of deep learning is highly heterogeneous, with typical features such as convolutional kernels for feature extraction and the distribution pattern of fully connected parameters. The parameters of these nonlinear computational units lack intuitive semantic interpretation, and their functional value entirely depends on the output responses of input data after hierarchical transformations.This disparity makes the direct transfer of traditional watermarking schemes face dual challenges: they could either destroy the relationships between network parameters or fail to effectively conceal information in the complex parameter space. Therefore, developing new watermarking technologies that adapt to the unique characteristics of neural networks has become an inevitable choice. These technologies need to protect intellectual property without compromising the model's effectiveness and security.

Existing research on model integrity protection is still limited, and due to the poor interpretability of neural network models, current theoretical knowledge cannot prove that modifications to model parameters have controllable effects on model performance. Traditional watermarking methods, to some extent, can lead to a decline in model performance, thus triggering unforeseen risks. Additionally, during model dissemination and use, there is also the risk of malicious tampering.

1.2 Research Status at Home and Abroad

1.2.1 History and Current Status of Adversarial Example Techniques

According to the level of knowledge the attacker has about the model, adversarial attacks are mainly divided into white-box attacks and black-box attacks. In a white-box attack scenario, the attacker has full access to the target model's architecture, training data, and weight parameters. In a black-box attack scenario, the attacker can only access the model's input-output interface, with no knowledge of the internal architecture, training parameters, or other critical information.

When considering the attacker's intent, their attacks can be further categorized into untargeted attacks and targeted attacks. The former aims to induce arbitrary misclassification, while the latter seeks to precisely mislead input samples into a predetermined target class. From the perspective of attack strategy, adversarial sample generation methods can be divided into single-step attacks and multi-step attacks. Single-step attacks generate adversarial samples through a one-time perturbation, whereas multi-step attacks require an iterative optimization process. Although multi-step attacks demand a longer computational cycle, the adversarial perturbations they produce typically achieve a higher attack success rate.

Given that most popular datasets commonly use the JPG format, Dziugaite et al. [3] explored the use of JPEG compression techniques to reduce the possibility of adversarial perturbations. Building on this, Das et al. [4] deepened the research by using ensemble methods to filter out high-frequency signals in adversarial samples, and this approach was proven to be effective against typical attacks like FGSM and DeepFool. However, experimental analysis by Xu et al. [5] showed that this method is effective for certain types of adversarial samples, but it may damage the inherent features of the image and reduce normal classification accuracy. The concave-point defense algorithm developed by Luo et al. [6] is based on the assumption that classifiers are robust to scale and translation transformations, but this theory has not been fully validated in complex attack scenarios. Xie et al. [7] attempted to reduce the deception of adversarial samples by employing preprocessing techniques like random scaling and cropping. Gu et al. [8] drew on the idea of compressed autoencoders to develop Deep Compression Networks (DCN), aimed at enhancing the defense capabilities against adversarial samples. The regularized input gradient method proposed by Ross et al. [9] effectively strengthens the model's resistance to interference by constraining input variables that affect the model' s output. However, while this approach shows significant results when combined with adversarial training, it incurs greater computational costs.

The outstanding performance of deep learning techniques and their wide applications have made adversarial sample research a growing focal point. The concept initially originated in the field of image classification, and with the spiral evolution of attack and defense technologies, its research scope has expanded from basic classification tasks to broader computer vision areas, such as facial recognition [10], text detection [11], and visual tracking [12].

The attack and defense process of adversarial sample techniques, along with the unique dynamic triggering mechanism of adversarial samples and their precise characterization of the model's decision boundary, provide new insights for addressing the challenges of traditional deep learning watermarking techniques in copyright protection and model integrity verification.

1.2.2 The Evolution of Digital Watermarking Technology in Model Protection

Traditional digital watermarking techniques embed information through the spatial domain (e.g., LSB) or transform domain (e.g., DCT, DWT), but they face the contradiction between robustness and imperceptibility. With the development of deep learning, watermarking technology has shifted towards embedding in model parameters and dynamic features. In 2017, Uchida et al. [13] introduced a pioneering copyright protection strategy for deep neural networks. They trained the model by introducing an additional regularization loss function, achieving the dual goals of optimizing model performance and embedding watermark information simultaneously during the model training phase. The loss function $L(\theta)$ is defined as follows:

$$L(\theta) = L_0(\theta) + \lambda L_w(\theta) \tag{1.1}$$

Where, θ represents the weights of the target model, $L_0(\theta)$ represents the loss function of the target model's original task, and $L_w(\theta)$ is the regularization term for embedding the watermark, with λ representing the weight of this loss function.The constraint term $L_w(\theta)$ constructs a statistical bias-guided model through the parameters θ, encouraging the model to spontaneously form specific distribution characteristics during the training process. These characteristics are then used to carry the watermark information to be embedded. During the embedding process, given a weight vector $\theta \in R^M$ with M elements and a projection matrix $X \in R^{T\times X}$ used for watermark embedding, the watermark is embedded and extracted by projecting θ using X and designing a symbol function for the embedding and extraction of the watermark information. The extraction method of the $j-th$ watermark bit b_j is as follows:

$$b_j = s\left(\sum_i X_{ij}\theta_i\right), \tag{1.2}$$

Where $s(\cdot)$is the sign function with a threshold of 0:

$$s(x) = \begin{cases} 1, & x \geq 0, \\ 0, & else, \end{cases} \tag{1.3}$$

This process can be viewed as a binary classification problem of a single-layer perceptron with no bias parameters. Therefore, $Uchida$ et al. used binary cross-entropy to define the loss function of the watermark embedding regularization term:

$$L_w\left(\theta\right) = -\sum_{i=j}^{T} b_j \log\left(y_i\right) + \left(1-b_j\right)\log\left(1-y_i\right), \tag{1.4}$$

where

$$y_i = \sigma\left(\sum_i X_{ji}\theta_i\right), \tag{1.5}$$

σ represents the Sigmoid function.

Deep learning networks exhibit highly non-convex loss surface characteristics under over-parameterized designs, with many local minima in their parameter space that have approximately equivalent performance. Therefore, based on the redundant characteristics of the parameter space, the model parameters can be converged to a specific local minimum region by introducing regularization constraints, such that watermark information can be embedded in this region while maintaining the basic functionality of the model. The model holder can embed watermark information into the model by jointly training the main task loss function and the watermark embedding loss function. When verification is needed, the model' s copyright can be assessed by calculating the deviation between the watermark extraction accuracy and the preset threshold. This method demonstrates the possibility of successfully embedding watermarks while maintaining certain model performance, showcasing the potential of watermarking technology in deep neural networks.

Although this is the first attempt to embed watermarks in deep neural networks, it marks a significant progress in the field of deep learning network security. However, this method is not without limitations. For example, the watermark capacity is limited by the model size, specifically the number of weights, which restricts the amount of information that can be embedded. Moreover, the method has poor resistance to watermark covering attacks. Due to the limited number of network layers, attackers may carry out covering attacks on each layer, thereby disrupting the original watermark information. Additionally, embedding watermarks in this regularization way may interfere with the normal training process of the network, especially when training complex models like Generative Adversarial Networks (GAN).

To address the shortcomings of existing technologies, *Rouhani et al.*[14] made a ground breaking improvement to the watermark embedding approach. Compared to *Uchida et al.* strategy of embedding the watermark in the static weights of the model, this approach chooses to encode the watermark into the probability density function of the network layers. By utilizing the dynamic statistical properties of deep neural networks to generate the watermark, the watermark signal is only activated when specific trigger data is input. This dynamic feature not only enhances the watermark' s concealment and adaptability but also significantly improves its resistance to covering attacks. However, due to the nature of the intermediate layer activation maps approximating a Gaussian mixture distribution, this method can only embed the watermark in a single selected layer, leading to a bottleneck in the amount of information that can be embedded.

Merrer et al.[16], based on the characteristic that adversarial examples induce models to output a specific class with high probability, used the adversarial perturbations to generate specific samples as the trigger signals for ownership identification. Specifically, they injected adversarial examples into the original data set through a mixed training method during the training process and re-labeled the adversarial samples with standard class labels. The model trained in this way is essentially a fine-tuned version of the decision boundary of the original model. When adversarial examples are used as the trigger set to test the model, the watermarked model is able to output the correct class, while the original model cannot. Furthermore, *Zhang et al.*[17] embedded watermarks by modifying the true class labels of selected training data images and then fed the designed trigger images and their corresponding output labels into the model for training to achieve watermark embedding. In addition, they further proposed three different trigger image generation methods based on *Adi et al.* approach, including: embedding specific additional information into randomly selected training images, selecting other data set images unrelated to the target model' s classification, and using random noise images.

Jia et al.[18] identified the independence between the watermark task and the model's original task, which led to the insufficient effectiveness of existing watermark methods in defending against model extraction attacks. Specifically, when processing traditional watermark tasks and watermark verification tasks, the model formed independent parameter activation paths, meaning the model independently handled these two tasks. This separation of decision paths resulted in the watermark information not being deeply integrated into the model's core representation system, making the watermark susceptible to being removed by model fine-tuning and pruning attacks. To address this issue, they proposed the Entangled Watermark Embedding (EWE) method. This method constructs a trigger sample set through adversarial perturbations and injects it into the training data set, thereby integrating watermark information with the original data. During model training, a soft nearest-neighbor loss is introduced to create a cross-task representation entanglement constraint, forcing the two types of data to form a non-linear entangled state in the feature space, thereby inducing the model to activate overlapping neurons when handling different tasks. This approach effectively defends against model extraction attacks.

Szyller et al.[19] also discovered this issue and proposed *DAWN* (Dynamic Adversarial Watermarking of Networks) to address the shortcomings of traditional watermark methods in defending against model extraction attacks. Unlike the aforementioned approach, *Szyller et al.* did not interfere with the normal training process of the model, but instead made special designs to the model' s output. By adding controllable perturbations to the model's predicted outputs, the image output probabilities are biased toward a pre-set incorrect class, and these incorrect outputs are set as the watermark information. When a model attacker queries the model and uses the model's returned results for model extraction, the watermark's incorrect classification pattern will reproduce the watermark information into the suspicious model's parameter space. The model owner can use this information to determine the model's copyright.

1.2.3 The current shortcomings in the research of deep learning model watermarking technology are as follows

(1) Due to the poor interpretability of deep learning model parameters, adding watermarks to the model may alter some model parameters, resulting in a loss of model functionality.

(2) Current research mainly focuses on protecting the copyright of models, with less attention given to research on protecting the integrity of models.

1.3 The main research content of this paper

In response to the aforementioned shortcomings in deep learning model research, this paper mainly studies integrity verification based on adversarial examples.

Although adversarial examples pose a certain threat to the security of models, this paper approaches the positive attributes of adversarial examples and explores their use in protecting model copyright. Traditional model protection schemes, to some extent, may affect the model, and existing literature rarely provides solutions that focus on targeted protection of specific objectives. From the perspective of minimizing the impact on the model, this paper investigates watermarking methods that do not modify the model itself and focuses on targeted protection for specific key target categories.

The main contributions of this paper are as follows: A deep learning network integrity verification method based on multi-target adversarial examples is proposed. This approach uses only the model's output as the model watermark and detects changes in the model's decision boundary by observing variations in the output, thereby determining whether the model has been modified. The method also allows for targeted protection of specific key target categories and can embed user-specific personal information into the watermark. Experiments show that this method can still detect changes in the model's decision boundary and verify the integrity of the model under relatively low attack intensities.

2 Related Theories and Technologies

This chapter will introduce some of the basic concepts and knowledge involved in this paper. Since the protection of deep learning models in this paper mainly revolves around adversarial examples, the main content of this chapter first introduces deep learning networks and common network models, followed by the basic concepts and types of adversarial examples, and finally covers some common attack methods targeting model watermarking.

2.1 Research Background and Significance

During the training process of deep neural networks, the weight matrices are dynamically optimized and adjusted through the gradient descent algorithm. The network structure has the following core features: Its multi-layer architecture progressively extracts features and abstracts data by stacking hidden layers. The nonlinear activation functions introduced at each layer overcome the representation limitations of linear models, significantly enhancing the model's fitting ability. The massive parameter scale brought by the deep structure requires that the training process be supported by sufficient data and computational resources. Unlike traditional feature engineering methods, this network has the ability to

autonomously extract effective features from raw data, greatly reducing the need for manual intervention. The gradient descent-based optimization mechanism iteratively updates the parameters, driving the loss function to converge towards a minimum. This architecture demonstrates exceptional performance in fields such as computer vision and semantic understanding, especially excelling in generalization when handling high-dimensional complex data.

2.1.1 Convolutional Neural Network (CNN)

The design concept of Convolutional Neural Networks (CNN) is inspired by research on biological visual mechanisms, particularly influenced by $Hubel$ and $Wiesel$ studies on the hierarchical response mechanisms of the visual cortex. Compared to traditional neural networks, CNN feature a unique three-dimensional structure for neurons, which includes width, height, and channel depth. This design not only preserves the spatial hierarchical features of image data but also aligns with the working mechanism of biological visual receptive fields. In practice, each neuron is only connected to a local region of the input, and through this local connectivity strategy, CNN significantly reduce the parameter scale while maintaining excellent feature representation capabilities.

2.1.2 Common Deep Learning Model

(1)$AlexNet$(2012)

As a landmark model in the resurgence of deep learning, its five-layer convolutional and three-layer fully connected architecture achieved a breakthrough accuracy in the $ImageNet$ competition for the first time. The use of the $ReLU$ activation function to replace the traditional $Sigmoid$, combined with $Dropout$ regularization and overlapping pooling strategies, effectively alleviated the issues of vanishing gradients and over fitting. The dual GPU parallel training design laid the foundation for subsequent large-scale models.

(2)$VGGNet$(2014)

By stacking consecutive 3@3 small convolutional kernels to build a $16-19$ layer deep network, it demonstrated the importance of receptive field equivalence and depth for feature representation. The unified convolution design simplified the selection of hyper parameters, but the high number of parameters led to a surge in computational cost. Its modular design concept inspired the structure of subsequent networks.

(3)$GoogLeNet$(2014)

The introduction of $Inception$ modules enables multi-scale feature fusion, while 1@1 convolutional kernels are employed for dimensionality reduction and cross-channel feature interaction. Replacing fully connected layers with global average pooling significantly reduces the number of parameters. By stacking 9 $Inception$ modules, a 22-layer network is constructed, achieving—for the first time on $ImageNet$—a Top-5 error rate below 7%.

(4)$ResNet$(2015)

The residual learning framework addresses the degradation problem in deep networks via skip connections, enabling the successful training of ultra-deep models with up to 152 layers. Its bottleneck structure (a 1@13@31@1 convolutional block) optimizes the trade-off between computational efficiency and feature representation capacity. This architecture reduced the error rate to 3.57% on $ImageNet$ classification, establishing itself as a foundational design for subsequent models.

(5)$DenseNet$(2017)

The dense connectivity mechanism enables each layer to receive feature inputs from all preceding layers, enhancing gradient flow and feature reuse. Through a composite function ($BN-ReLU-Conv$) and a growth rate parameter to regulate the number of feature maps, this architecture reduces parameters by over 50% compared to $ResNet$, demonstrating superior performance in tasks like medical image segmentation. (6)$MobileNet$(2017)

Depthwise separable convolution decomposes standard convolution into depthwise convolution (per-channel) and pointwise convolution (1@1), reducing computational cost by $8-9@$. With width multiplier and resolution multiplier enabling flexible accuracy-speed trade-offs, it has become the go-to architecture for mobile deployment, achieving $120+FPS$ on platforms like the Snapdragon 835.

2.2 Introduction to Adversarial Examples

Adversarial examples refer to specially crafted samples that induce deep learning models to make incorrect predictions by introducing subtle, human-imperceptible perturbations to original inputs. The generation of adversarial examples can be formally expressed as a constrained optimization problem.

Given a classification model $f_\theta(x) \to y$, with parameters θ, the adversarial attack aims to find a perturbation δ that satisfies the following conditions: $\|\eta\|_p < \varepsilon$,st
$maximize\, L(f_\theta(x+\eta), y^*)$

$$maximize\, L(f_\theta(x+\eta), y^*) \tag{2.1}$$

Here, $L(\cdot)$ denotes the loss function, y^* represents the attack target, and the $p-norm$ constraint guarantees the perturbation remains controlled. According to the dimension of information available to the attacker, the classification system can be primarily categorized as follows.

2.2.1 White-Box Attacks vs. Black-Box Attacks

White-box attacks assume the attacker has complete knowledge of both the model architecture $\nabla_x L$ and gradient information η, where perturbation generation relies on explicit modeling of the decision boundary.Typical methods construct linear approximations based on first-order Taylor expansions:

$$\eta \propto \nabla_x L\left(f_\theta\left(x\right), y\right), \tag{2.2}$$

The optimal perturbation can be approximated through iterative projected gradient descent, and this process can be viewed as a path integral of the loss function surface in the input space.

Black-box attacks rely solely on the model's input-output responses $f_\theta(x)$, where the target function's gradient can be estimated via finite difference methods:

$$\nabla_z L \approx \frac{1}{m}\sum_{i=1}^{m}\frac{L(x+\sigma u_i)-L(x)}{\sigma}u_i, \tag{2.3}$$

Here, u_i represents a random perturbation vector and σ denotes the step size. In scenarios where only predicted labels (without probability scores) are available, approaches like binary search or random walk can be employed to approximate the decision boundary.For example, boundary attacks generate adversarial examples through the following iterative process: 1. Move from the current adversarial sample x_adv towards the original sample x; 2. Search for the minimum perturbation point along the movement path that satisfies $f_\theta(x') \neq f_\theta(x)$.

2.2.2 Targeted Attacks vs. Untargeted Attacks

Targeted attacks force the model to misclassify input samples into a specified target class while satisfying the perturbation constraint $\|\eta\|_p < \varepsilon$. This can be achieved by formulating an optimization problem that minimizes the loss for the target class, with some methods employing a dual optimization objective:

$$\min imize \|\eta_p\| + \lambda \cdot \max\left(0, \log f_\theta\left(x+\eta\right)_y - \log f_\theta\left(x+\eta\right)_{y_t}\right), \tag{2.4}$$

by using Lagrange multipliers λ, the perturbation magnitude is balanced with the confidence of the target class.

The untargeted attack only requires the model's output to deviate from the original correct class y, without specifying the misclassified target class. Its optimal solution often projects along the direction of the loss function gradient, i.e.:

$$\eta^* = \varepsilon \cdot \frac{\nabla_x L\left(f_\theta(x), y\right)}{\|\nabla_x L\left(f_\theta(x), y\right)\|_p}, \tag{2.5}$$

2.3 Common Attack Methods Against Deep Learning Model Watermarking

The security of watermarking techniques must withstand various targeted attacks. Attackers employ strategies such as model structure modification, parameter perturbation, or input interference to attempt to remove watermark identifiers or disrupt verification mechanisms. Below, typical methods are elaborated from the dimensions of attack implementation paths and damage targets.

(1)$ModelPruningAttack$

Model pruning disrupts the physical carrier of the watermark by removing redundant parameters in the model. Structured pruning focuses on deleting neurons or channels with low importance, potentially directly eliminating specific weight distributions that the watermark encoding depends on. Unstructured pruning randomly removes scattered, small parameters, causing fragmentation and loss of watermark information during the sparsification process. Such attacks pose a significant threat to white-box watermarks that rely on specific patterns in the parameter space, especially when the watermark embedding is not coupled with the model's critical functional path.

(2)$ModelFine-TuningAttack$

Model fine-tuning optimizes a pre-trained model locally with a small amount of data, which may overwrite watermark-related feature representations. The fine-tuning process causes the model's feature representations to shift towards the new task's data distribution, weakening the activation conditions of the watermark trigger. Attackers can further isolate watermark-related parameters from the new task's optimization direction by limiting the number of fine-tuning layers or introducing regularization constraints, thereby accelerating the degradation of the watermark features.

(3)$DistillationAttack$

Using the teacher-student model framework, knowledge from a watermarked model is transferred to a clean student model, filtering out watermark information through softened output distributions or feature matching loss. The temperature scaling mechanism reduces the discriminative power of output logits, causing the watermark trigger patterns to be smoothly diluted in the probability space. Feature distillation, on the other hand, strips away watermark-related anomalous feature responses by mimicking intermediate layer representations. Such attacks are highly destructive to black-box watermarks that rely on output layer statistical properties.

(4)$WatermarkOverwritingAttack$

Attackers overlay a new watermark onto a model that already contains an embedded watermark, triggering conflicts over copyright ownership. Orthogonal projection methods achieve parallel embedding of watermarks by constructing multiple independent subspaces, but multiple projections may lead to non-orthogonal interference among the subspace basis vectors. Parameter modulation-based watermarks cause information confusion due to the

dynamic overriding of weight values.

(5)*EvasionAttack*

For the trigger verification mechanism of black-box watermarks, adversarial sample generation techniques are used to perturb the input data. The adversarial perturbations crafted by attackers cause the trigger samples to deviate from the preset activation region in the feature space while maintaining the model's prediction performance on normal samples. Gradient masking methods further hide the gradient propagation path of the trigger, increasing the difficulty of reverse engineering the watermark detection.

(6)*ModelStealingAttack*

By querying the original model to construct a proxy training dataset, attackers can train surrogate models that maintain similar decision boundaries but possess parameter distributions unrelated to the watermark encoding. This type of attack poses a significant threat to white-box watermarking schemes that rely on internal parameter characteristics, as the implementation pathway of surrogate models fundamentally differs from the original model.

3 Integrity Verification of Deep Learning Models Based on Multi-Objective Adversarial Examples

3.1 Model Integrity Verification Framework

Since users of the model only have usage rights and cannot access the model's specific information, integrity verification of the model can only be performed by examining the output behavior.

3.1.1 Model Watermark Sample Generation Methods

In the integrity verification framework of black-box models, the core of the watermark generation strategy lies in accurately capturing the geometric topological features of the classifier. This chapter uses multiple sets of adversarial examples that are closely aligned with the decision boundary to describe the model's decision boundary.

Start with an image sample I from a target class k that is vulnerable to attacks, and set multiple attack targets to generate high-quality adversarial examples x_0, y_0, z_0. The generation process is shown in Figure 1.

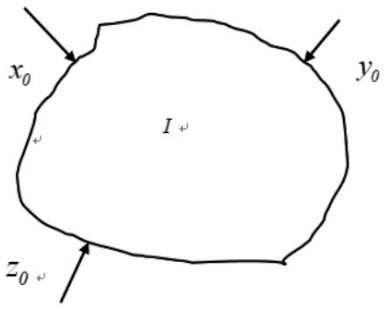

Figure 1: Initial Adversarial Sample Generation Diagram

After obtaining the initial sample x_0, perturbations are added to generate a perturbation set $S = \left\{x_0 + \delta_i\right\}_{i=1}^{N}$, within the $\varepsilon-$neighborhood, where$\delta \sim U(-\varepsilon, \varepsilon)^d$. The partition function is defined as:

$$A = \{x \in S | f(x) = k\}, B = \{x \in S | f(x) \neq k\}, \tag{3.1}$$

Construct a binary set distance matrix $D \in R^{|A|+|B|}$, where the elements are:

$$D_{ij} = \|a_i - b_i\|_2, \ a_i \in A, \ b_i \in B, \tag{3.2}$$

Determine key sample pairs through dual-set nearest neighbor search:

$$(a^*, b^*) Satisfy : argmin \|a_i - b_i\|_2, \tag{3.3}$$

If a^*, b^* satisfy $\|b^* - a^*\| < \theta$, where θ is the parameter controlling the maximum distance, then the midpoint x_i of a^* and b^* is included in the watermark sample set **K**.The sample selection method is shown in Figure 1.

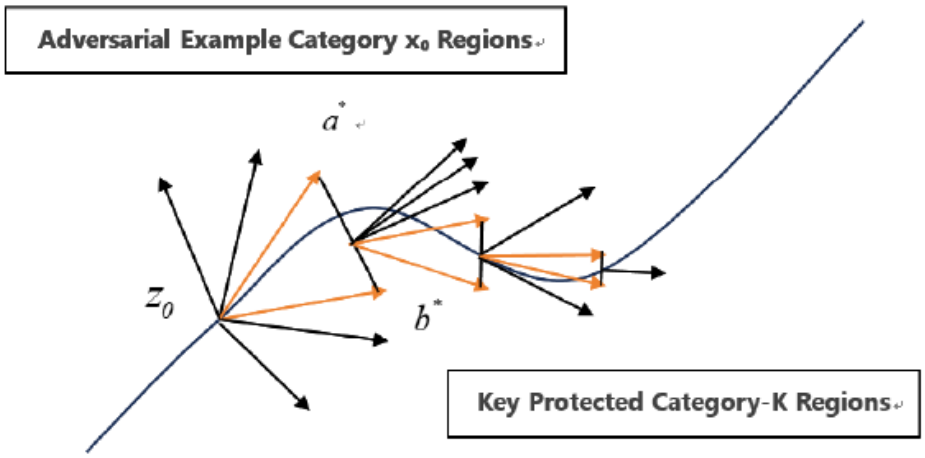

Figure 2: Schematic Diagram of the Iterative Process

Since the processing methods for y_0 and z_0 are identical to those for x_0 in Algorithm 1, only the handling procedure for x_0 is demonstrated in the pseudocode.

3.1.2 Model Watermarking Design and Certification Methods

In specific application scenarios where model users have no access to the model's internal structural parameters or training process details, and lack prior judgment criteria for assessing functional integrity, the verification approach must rely solely on observable outputs. Conventional classification models typically provide two output formats: class probability distributions and discrete class labels. To ensure cross-model applicability of the verification method, this study constrains the model output to the discrete class label format. Under

this constraint, the verification process can only proceed by observing the model's output class labels in response to input samples.

The model holder has complete access to the model parameters and training details, allowing them to construct a dedicated sequence set M based on the target model K parameters and structural features. The model's output labels are encoded to generate a digital watermark identifier k_M. This watermark mechanism allows two implementation paths: one can either construct an irreversibly hashed sequence with a pseudo-random distribution, or embed authentication elements via a pre-shared protocol, such as using the identity key of an authorized user as the initial seed for generating the watermark. This bidirectional design ensures the watermark's non-forgeability while providing solutions for different application scenarios.

After agreeing on the watermark information and the key protected class with the model user, watermark samples are generated using a sample generation algorithm based on the given watermark sequence. Assuming the watermark information is K_M, we set to generate n adversarial samples of different classes, dividing the watermark information into n parts $\{K_{M_1}, K_{M_2}, \ldots, K_{M_n}\}$, and then use the improved watermark sample generation algorithm for each piece of watermark information. The algorithm is shown as Algorithm 2.

Based on different adversarial samples, algorithm 3.2 is repeatedly used to generate n groups of watermark samples, which are then combined to form the final watermark sample set K. The watermark sample set K and the watermark information K_M are securely packaged and sent to the model user for verification. The user will input the received watermark sample set K into model $M$to obtain the model output K'_M, and then compare each K'_M with K_M. If $K'_M \neq K_M$, then the model is considered incomplete.

4 Algorithm

Algorithm1 :	*RandomizedWatermarkSampleGeneration*
Input:	An image sample I belonging to class k, the model M to be protected, and the maximum number of iterations i_{max};
Output:	*Watermark sample setK*
1	Initialization: Perturbation range $\varepsilon = 1.0$, maximum distance $\theta = 0.2$, constraint condition $\gamma = 1; i = 1$, and initial normal vector $\eta_0 = 0$;
2	Use the fast boundary attack method to obtain the initial adversarial sample x_0;
3	**for** $i \in [1, i_{max}]$ **do**
4	Add a random perturbation δ such that $\delta \sim U(-\varepsilon, \varepsilon)^d$, Generate a perturbation set$S = \left\{x_0 + \delta_i\right\}_{i=1}^{N}$ that satisfies the partition function (3.1) and is approximately in the direction of the normal vector η_i;
5	Determine the nearest neighbor sample pair (a^i, b^i) that satisfies the formula (3.3) through nearest neighbor search;
6	**if** $\|b^i - a^i\| < \theta$ **then**
7	Add the midpoint x_i of a^i and b^i to the watermark sample set K;
8	**end**
return	**K**

The output sequence generated by this algorithm on the model can be divided into two categories: samples belonging to class **k** are encoded as **1**, while those not belonging to class **k** are encoded as **0**. Since the algorithm imposes no restrictions on the output categories, the result is an unordered set of binary codes, ensuring its security.

Algorithm2 :	$WatermarkSequence-BasedSampleGenerationAlgorithm$
Input:	An image sample I belonging to class k, the model M to be protected, Watermark information K_{M_1};
Output:	$WatermarkSampleSet$
1	Initialization: Perturbation range $\varepsilon = 1.0$, maximum distance $\theta = 0.2$, constraint condition $\gamma = 1; i = 1$, and initial normal vector $\eta_0 = 0$, and maximum bit length of watermark information M_1;
2	Use the fast boundary attack method to obtain the initial adversarial sample x_0;
3	**while(1) do**
4	Add a random perturbation δ such that $\delta \sim U(-\varepsilon, \varepsilon)^d$, Generate a perturbation set$S = \left\{x_0 + \delta_i\right\}_{i=1}^{N}$ that satisfies the partition function (3.1) and is approximately in the direction of the normal vector η_i;
5	Determine the nearest neighbor sample pair (a^i, b^i) that satisfies the formula (3.3) through nearest neighbor search;
6	**if** $\|b^i - a^i\| < \theta$ **then**
7	**if** The midpoint x_i of a^i and b^i belongs to a class that satisfies the watermark information $K_{M_1}^j$ **then**
8	Add x_i to the watermark sample set **K**
9	$j = j + 1$
10	**if** $j = M_1$ satisfies the maximum watermark bit **break**
11	**end**
12	**end**
13	**end**
14	$i = i + 1$
14	Generate a normal vector η_i that is approximately in that is approximately in the direction of the line connecting $\nu = (b^* - a^*) / \|b^* - a^*\| a^i$ and a^i with x_0
13	**end**
return	$\mathbf{K_1}$

5 Experimental Results

To verify the feasibility of the proposed solution in this chapter and its sensitivity to common attacks, this subsection mainly presents experiments from the following aspects: experimental setup, experimental results, and simulated attacks.

3.1 Experimental Setup

The data set used in this experiment is the publicly available $Oxford-IIITPe$ data set, which contains 37 pet categories (covering different breeds of cats and dogs). Classification training was performed on this dataset using three common classification networks: $ResNet50, AlexNet, andGoogLeNet$. All experiments were conducted on an $NVIDIAGeForceGTX1050Ti$ using $Matlab2020b$.

3.2 Experimental Result

This section assumes that the model owner and the user have already agreed on the watermark information K_M and the key protected target categories j and k. In the experiment, $ResNet50$ is used as an example. Based on the target categories j and k, one initial image I_1 and I_2 of categories j and k are selected, respectively. Then, using the Fast Gradient Sign Method ($FGSM$), three adversarial samples are generated for each image I_1 and I_2. The original images are shown in Figure 3.

Figure 3: Original Sample Image

Generate watermark samples according to Algorithm 2, as shown in Figure4. According to the encoding method provided in *Section*3.1, the watermark information obtained by encoding the two sets of watermark samples in Figure 4 is: [01011, 10010, 11001, 00110, 11011, 10010] [01011, 10010, 11001, 00110, 11011, 10010]. The model user only needs to input the generated watermark samples into the model and check the output to verify the integrity.

3.3 Model Integrity Detection Experiment under Simulated Attacks

This section will perform model fine-tuning, model pruning, and distillation attacks on key protected classes for the models used in the above experiments. The attacks are required to ensure that the decline in model classification performance does not exceed 10% of the original performance. Additionally, the model integrity will be validated according to the model verification method provided in *Section*3.1.

Figure 4: Watermark sample image

3.3.1 Model Fine-tuning

The model fine-tuning in this experiment follows the method outlined below: Images of cats and dogs, different from those in the dataset and test set, are searched online and gradually added to the training set in increasing amounts to retrain the network. After training, the original test set is used to evaluate the model's performance, and the model's performance degradation rate is checked, ensuring that the performance degradation does not exceed 10%.

Table 1 Model Integrity Verification under Model Fine-tuning

Model category	Model depth	Model	Performance	Degradation	Rate (%)		
		$x < 1$	$1 \leq x < 3$	$\leq x < 5$	$5 \leq x < 7$	$7 \leq x < 9$	$9 \leq x$
AlexNet	8	×	√	√	√	√	√
GoogLeNet	22	×	√	√	√	√	√
ResNet50	50	×	√	√	√	√	√

3.3.2 Model Pruning

The model pruning in this experiment follows the method outlined below: The pruning rate is gradually increased, and the classification accuracy of the model is observed, ensuring that the performance degradation does not exceed 10%.

Table 2 Model Integrity Verification under Model Pruning

Model category	Model depth	Model	Performance	Degradation	Rate (%)		
		$x < 1$	$1 \leq x < 3$	$\leq x < 5$	$5 \leq x < 7$	$7 \leq x < 9$	$9 \leq x$
AlexNet	8	√	√	√	√	√	√
GoogLeNet	22	×	×	√	√	√	√
ResNet50	50	×	√	√	√	√	√

3.3.3 Data Poisoning Attack

The data poisoning attack in this experiment is carried out as follows: First, half of the key protected classes j and k from the original training set are selected to form a new image set. Then, a certain number of images unrelated to classes j and k are added to the image set of class j and $k$in increasing order, and combined with the newly formed image set to create a new training set. The new training set is then used to train a tampered model on the pre-trained model. The tampered model is tested on the test set of the original model, ensuring that the model performance degradation does not exceed 10%.

Table 3 Model Integrity Verification under Data Poisoning Attack

Model category	Model depth	Model	Performance	Degradation	Rate (%)		
		$x < 1$	$1 \leq x < 3$	$\leq x < 5$	$5 \leq x < 7$	$7 \leq x < 9$	$9 \leq x$
AlexNet	8	√	√	√	√	√	√
GoogLeNet	22	×	×	√	√	√	√
ResNet50	50	×	×	×	√	√	√

From the experimental results in $Tables 1, 2, and 3$, it can be concluded that when the model performance degradation rate is no less than 5%, the watermark samples provided in the

experiment can successfully detect the three attack methods mentioned. However, when the model performance degradation rate is less than or equal to 5%, the watermark samples yield better results for models with lower network depth. This is because simpler models tend to have their decision boundaries easily altered when tampered with, allowing them to be captured by fragile samples. On the other hand, models with higher depth have high module redundancy. For instance, in the case of the $ResNet50$ network, the strong feature abstraction ability of deep layers makes the poisoning effects more subtle, resulting in the watermark samples failing to detect the attack.

3.3 Comparison with Related Literature

In order to fully demonstrate the superiority of this approach, this section will compare it with existing solutions, as shown in $Table4$. Through comparison, it can be observed that these solutions all have good fidelity. The methods proposed by $Wang$[10], $Rouhani$[14], and Adi[15] use watermarks to protect the intellectual property of neural network models. Their watermarking methods are robust but not sensitive to changes in the model, making them unsuitable for model integrity verification. $Guan$[20] proposed using fragile samples for model integrity verification. Although the impact is minimal, their method still involves some modifications to the model itself. The watermarking approach in this chapter can not only add watermark verification for model integrity without modifying the model itself, but also provide targeted protection for key objectives.

Table 4 Comparison of Different Methods

Watermark Embedding Methods	Fidelity	Fragility	Robustness	Stealthiness
Our method	√	√	√	×
Wang	√	×	√	√
Rouhani	√	×	×	√
Adi	√	×	√	√
Guan	√	√	×	√

Watermark Embedding Methods	Model Integrity	Critical Target Protection
Our method	√	√
Wang	×	×
Rouhani	×	×
Adi	×	×
Guan	×	×

Reference

[1] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[2] 冯乐, 朱仁杰, 吴汉舟, 等. 神经网络水印综述[J]. 应用科学学报, 2021, 39(6): 881-892.

[3] Dziugaite G K, Ghahramani Z, Roy D M. A study of the effect of jpg compression on adversarial images[J]. arXiv preprint arXiv:1608.00853, 2016.

[4] Das N, Shanbhogue M, Chen S T, et al. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression[J]. arXiv preprint arXiv:1705.02900, 2017.

[5] Xu W, Evans D, Qi Y. Feature squeezing mitigates and detects carlini/wagner adversarial examples[J]. arXiv preprint arXiv:1705.10686, 2017.

[6] Luo Y, Boix X, Roig G, et al. Foveation-based mechanisms alleviate adversarial examples[J]. arXiv preprint arXiv:1511.06292, 2015.

[7] Xie C, Wang J, Zhang Z, et al. Adversarial examples for semantic segmentation and object detection[C]. Proceedings of the IEEE international conference on computer vision. 2017: 1369-1378.

[8] Gu S, Rigazio L. Towards deep neural network architectures robust to adversarial examples[J]. arXiv preprint arXiv:1412.5068, 2014.

[9] Ross A, Doshi-Velez F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients[C]. Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).

[10] Wang K, Wang S, Zhang P, et al. An efficient training approach for very large scale face recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 4083-4092.

[11] Chen S, Bi X, Gao R, et al. Holistic Sentence Embeddings for Better Out-of-Distribution Detection[C]. Findings of the Association for Computational Linguistics: EMNLP 2022. 2022: 6676-6686.

[12] Tang F, Ling Q. Ranking-based Siamese visual tracking[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 8741-8750.

[13] Zhang J, Gu Z, Jang J, et al. Protecting intellectual property of deep neural networks with watermarking[C]. Proceedings of the 2018 on Asia conference on computer and communications security. 2018: 159-172.

[14] Darvish Rouhani B, Chen H, Koushanfar F. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks[C]. Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems. 2019: 485-497.

[15] Adi Y, Baum C, Cisse M, et al. Turning your weakness into a strength: Watermarking deep neural networks by backdooring[C]. 27th USENIX security symposium (USENIX Security 18). 2018: 1615-1631.

[16] Le Merrer E, Perez P, Trédan G. Adversarial frontier stitching for remote neural network watermarking[J]. Neural Computing and Applications, 2020, 32(13): 9233-9244.

[17] Zhang J, Gu Z, Jang J, et al. Protecting intellectual property of deep neural networks with watermarking[C]. Proceedings of the 2018 on Asia conference on computer and communications security. 2018: 159-172.

[18] Jia H, Choquette-Choo C A, Chandrasekaran V, et al. Entangled watermarks as a defense against model extraction[C]. 30th USENIX security symposium (USENIX Security 21). 2021: 1937-1954.

[19] Szyller S, Atli B G, Marchal S, et al. Dawn: Dynamic adversarial watermarking of neural networks[C]. Proceedings of the 29th ACM international conference on multimedia. 2021: 4417-4425.

[20] Guan X, Feng H, Zhang W, et al. Reversible watermarking in deep convolutional neural networks for integrity authentication[C]. Proceedings of the 28th ACM International Conference on Multimedia. 2020: 2273-2280.