Citation: Muyang Li, Guangwei Xu, Shifei He, et al. A review of research on verifying the integrity of model training processes. *Journal of Harbin Institute of Technology (New Series)*. DOI: 10.11916/j.issn.1005-9113.25064

A Review of Research on Verifying the Integrity of Model Training Processes

Muyang Li, Guangwei Xu, Shifei He and Xiujin Shi**

(*School of Computer Science and Technology, Donghua University, Shanghai 201620, China*)

Abstract: The development of artificial intelligence technology and the demand for large-scale pre-trained models have led to the widespread use of third-party model training services, which are generally provided by cloud service providers. However, there is also a problem of the authenticity of the model training process, specifically, whether the training process itself may be tampered with. Verifying the integrity of the model training process effectively ensures the reliability of the model training. While previous studies have mainly focused on the integrity of training data and privacy issues in model training for analysis, this paper primarily investigates the integrity of the entire process of model training. Integrity of model training is divided into three classes: computational correctness, algorithmic consistency and parameter update integrity. Then each category will be examined from the threat sources, attack modes and verification paths respectively. Finally, based on summarizing existing analysis results of model training integrity, it points out the direction for future research to verify the integrity of the model training process.

Keywords: artificial intelligence; model training; training process integrity; integrity verification

CLC number: TP181 **Document code**: A **Article ID**: 1005-9113(2026)00-0000-21

In recent years, there have been some notable developments in artificial intelligence technology. Large Language Models (LLMs) are also applied in research and development work across many fields. These models have gradually expanded into many areas, such as medical care, finance and smart cars[1-4]. Health insurance companies apply machine learning model to identify illnesses, banks employ those methods for risk evaluation and loan decisions, moreover, e-commerce enterprises will design recommendation system based on machine learning techniques. Training a modern deep learning model often requires considerable computational resources and is technically complicated. Large-scale models, such as OpenAI's GPT-3, are typical example of thus training processes-characterized by long training duration, massive parameter scales, and substantial costs[5]. Moreover, there is a lack of computing resources and professional skills for model training in practical application. Therefore, many such tasks have been conducted using third-party computing platforms or cloud services.

Practical model training processes are typically characterized by both a large volume of participating data and the need for complex calculations. Different application fields have rigorous requirements for the accuracy and reliability of the training results. For instance, in high-risk fields such as medicine and finance, errors in the model's training conclusions may have an impact on security-related decisions, which could result in serious outcomes. However, due to intellectual property protection of current model training algorithms and parameters, the training process is often not transparent, posing a potential threat to its integrity[6]. Problems, such as training data leaks, tampering with training algorithms, and fabricating training results, can affect the accuracy and credibility of the model, leading to security and legal risks. Therefore, how to effective verify the integrity of the model training process without compromising the confidentiality of its model and data, has become a key research challenge that demands urgent attention. All these problems reflect the basic problem that it is necessary to establish and verify the integrity of the model training process. Model training integrity refers to the standard, consistent and tamper-proof

Received 2025-01-06.

* Corresponding author: Guangwei Xu, Ph.D, Professor. Email: gwxu@dhu.edu.cn; Xiujin Shi, Ph.D, Associate Professor. Email: sxj@dhu.edu.cn.

execution of the entire training process, which includes input data, algorithmic code, parameter updates, environmental state and final output, and allows users to verify. Thus, it ensures that any claimed model version can be traced back to its specific generation process and provides objective evidence of the credibility of the training results.

Despite many methods that have been applied in traditional security and privacy protection technology to address the integrity verification of model training, some methods are often not effective enough or efficient enough when the scale of the model reaches a certain level, and they also require substantial computational resources. For example, some cryptography-based solutions, such as zero-knowledge proofs and homomorphic encryption, although they can guarantee security and privacy protection, have prohibitively high computational costs for large-scale deep learning models, thereby reducing training efficiency.

In addition, many approaches, depend on complex mathematical models or protocols, are difficult to apply or maintain in actual use, and most of the research remains at a theoretical level. There is a possibility that more information will be revealed unintentionally during the review. At present, some cutting-edge models often require large amounts of data and computational resources, while also raising the intellectual property concerns regarding the models themselves. Therefore, some model training suppliers are unwilling to reveal information about their training processes. In terms of practicability, the traditional verification method cannot guarantee the effectiveness of integrity check without compromising the data content.

To this end, researchers have put forward several technical approaches to verify the integrity of the model training process, including zero-knowledge proofs, homomorphic encryption, multi-party secure computation and trusted execution environment. Each method has its own advantages. Collectively, they can simultaneously preserve both security and privacy safeguards during training, while also enable verification that the training process remains uncorrupted. zero-knowledge proofs were first introduced in the field of blockchain, they have broad applications today and are commonly employed in research on how to ensure the accuracy of training without revealing privacy.

Liu et al.[7] proposed a zero-knowledge proof scheme based on Convolutional Neural Network (CNN) prediction and accuracy validation. Homomorphic encryption, on the other hand, guarantees the privacy of data by encrypting it, and can perform computations on the encrypted data to obtain meaningful results without accessing the plaintext. Multi-Party secure Computation (MPC) can ensure the security of data processing through collaborative computation among multiple parties, and Trusted Execution Environments (TEEs) use hardware isolation to guarantee that model training takes place in a trusted environment. Garg et al.[8] has proposed a model training process integrity verification scheme for logistic regression models that combines MPC-in-the-head techniques and provides practical code. Although these methods have certain theoretical advantages, due to the high cost of implementation and the complexity of training, they are not practical. For example, the zero-knowledge proof can help protect data privacy to some extent in large-scale model training, but the evidence preparation stage is computationally intensive and time-consuming at present, it remains undetermined which detection approaches can be used in different usage environments. However, the primary goal is to ensure the efficiency and accuracy of the integrity verification during the training procedure.

The structure of this paper is as follows. Section 1introduces the model training process and its integrity issues, mainly discussing the main threats to the integrity of the training process. In Section 2, the attack methods used to verify the integrity of the model training process are detailed. In Section 3, existing training process integrity verification schemes are introduced and summarized. Section 4 discusses the problems and future development trends of model training process integrity verification in practical applications. Finally, a summary of the entire paper is provided.

1 Potential Threats to the Integrity of the Model Training Process

1.1 Model Training Overview

In the current machine learning and deep learning frameworks, model training is a key step[9,10]. Firstly, to enhance the model's ability to predict or classify in specific tasks through training with specific algorithms using this dataset. After several iterations

and updates, the model parameters are adjusted so that the final output meets expected goals. The model training process usually includes many stages, and data preparation and model initialization are typically the first two stages.

The data preparation stage includes dividing the dataset into training and test sets, as well as performing pre-processing operations such as normalizing or standardizing numerical features to improve feature extraction ability of the model and enhance its generalization performance in both training and inference. If image datasets are involved, we also conduct data augmentation processing at this time. The model's initialization refers to the preparatory work on the model structure and parameters prior to training. This includes selecting an appropriate model architecture, setting the initial values of the neural network weight matrix and bias terms, as well as configuring optimizers and loss functions that meet specific task requirements. The main steps of the model training process are as follows[11].

(1) Forward propagation. During a standard model training process, input data will be transmitted from the input layer and, subsequently, pass through fully-connected layers, convolutional layers, pooling layers, activation functions, etc., until it reaches the output layer. Except for the input layer that receives pre-processed training data, the input data for each subsequent layer comes from the output of the previous layer. The unidirectional process of information transmission from input to the activation layer and then to the output layer is called forward propagation.

(2) Loss calculation. A preset loss function is used to calculate the error between the predicted value obtained from forward propagation and the true value. The loss function is needed to assess the model's performance during training, and its value determines whether the model has learned well at this round of training.

(3) Backpropagation. Using the backpropagation algorithm to compute the gradient of the model parameters and update them via a predetermined optimization method. The method of transmitting the data from the output layer to the input layer one by one according to the chain rule so as to decrease the loss and optimize the parameters for a better practical development.

(4) Parameters update. This is the fundamental operation in model training. Through iterative parameter optimization driven by optimization algorithm, we reduce the loss function to its minimum and improve the model's ability to fit training data. After backpropagation-based parameter updates, it is necessary to evaluate whether further adjustment is required. Typically, the computed gradients are combined with an optimization algorithm to update the parameters at each layer. The entire improvement in accuracy of the model will be used for the next step.

(5) Model iteration and validation. The model training process repeatedly performs forward propagation, loss computation, and backpropagation to continuously update the parameters and gradually approach the target. After each run of training, it needs to be evaluated whether the model has realized the desired function, that is, underfitting or overfitting. Based on the above, some hyperparameters need to be adjusted to further optimize the model so that it can meet practical application standards.

This paper focuses on ensuring the consistency of the entire model training process, with particular attention given to cases where model training is outsourced to a third party, as shown in Fig. 1. In this way, the user provides training data and model initialization parameters to the cloud service provider. After receiving the data, the cloud service provider trains the model and returns the trained model to the user. During this period, the focus of this paper will be on confirming the integrity of the model training process, analyzing potential threats to the training process and exploring existing verification mechanisms.

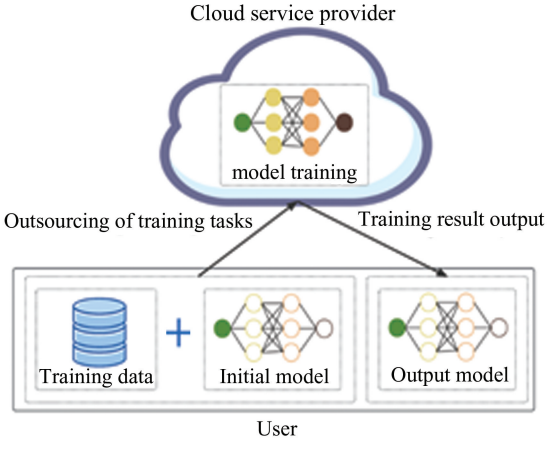

Fig.1 Outsourcing process for model training tasks

It should be noted that this paper assumes the training data for the model remains uncorrupted and

intact at the data level, ensuring its accuracy and integrity. As a result, the discussion is limited to the integrity of the model training process itself and does not cover the security of the data itself.

1.2 Integrity of the Training Process

The integrity of the model training process refers to all training steps being executed correctly as intended in the whole model training task without any tampering or omission from any step or algorithm, and the training output is also correct. Let the set of training data be denoted as $D = \{(x_i, y_i)\}_{i=1}^{N}$, the model parameters at iteration t as θ_t, the loss function as $L(\theta; D)$, and the total number of iterations as T. If the model training process is integrated, then for all iterations $t \in \{0, 1, \cdots, T-1\}$, the following conditions must simultaneously hold:

$$\text{Integrity} = \bigwedge_{t=0}^{T-1} (\text{Comp}(\theta_t, L_t) \wedge \text{Algo}(\theta_t) \wedge \text{Param}(\theta_t, \theta_{t+1})) = 1$$

here, *Comp* represents the accuracy of the calculations made during training; *Algo* represents whether the training algorithm is consistent; *Param* represents whether the parameters updated continuously are complete.

The model training process is only considered to have passed the test if the above three properties hold true at every iteration. Therefore, this paper classifies the integrity of model training into three types: computational accuracy, algorithmic consistency, and parameter update reliability. Parameter update integrity is separated because it affects whether the model's parameters can evolve normally over time[12-14]. These three properties will be detailed as in the following.

(1) Computational correctness refers to the accuracy of the model training algorithm implementation, for example, whether gradient calculations are correct and whether the optimization algorithm is executed correctly. This implies that each computational step is carried out properly in accordance with the preset algorithm without malicious interference or erroneous outcomes. The above property is mainly related to gradient computation, loss function calculation, optimizer calculation and whether hidden layers process input data correctly during the training of a model. If there is any calculation error at any stage during training, even if the other parts are perfect, the model's performance will drop sharply, and the reliability of the output results of the model will be seriously reduced.

(2) Algorithm consistency refers to whether the training process is carried out in accordance with the intended sequence, and whether any steps are missing or the order is altered during training[15]. Existing machine learning algorithms are generally carried out in an ordered sequence, and may involve inherent sequential dependencies. For example, in the forward propagation stage, the output of one layer is taken as the input of the next layer. Similarly, in backpropagation, the gradients of one layer are connected to those of the following layer. As a result, any abnormality in a step will affect the performance of the final model. This property indicates that the sequence of training steps needs to be maintained, and repeated experiments must have consistency, that is, under the same input data, the same output results should be produced. Algorithmic consistency ensures that the model's training process aligns with its intended design[16].

(3) To ensure the integrity of parameter updates means to maintain the accuracy of the training computation and use a correct optimization algorithm in each iteration. To prevent errors during parameter updates, avoid premature updates influenced by parameters from previous iterations, and defend against malicious tampering, it is essential to ensure the correctness and proper timing of each update. Otherwise, such issues can directly affect the model's convergence rate and ultimately degrade its final performance. Therefore, parameter updates are considered to be in the central position in training[17]. Guaranteeing the integrity of parameter updates can help maintain directional stability for model optimization in every iteration to reach a converged state safely throughout training.

1.3 Threats to Training Process Integrity

Based on an analysis of threats to the integrity of model training, this paper categorizes them into three main types: threats that reduce the computational accuracy of the training process; threats that interfere with the consistency of the training step; threats that damage the integrity of parameter updates. Different attack threats may occur at different stages of the model's training. Furthermore, certain attacks are not confined to a single stage; rather, several stages could potentially suffer from the same kind of attack. The specific threats to the integrity of the training process at each stage are shown in Fig. 2.

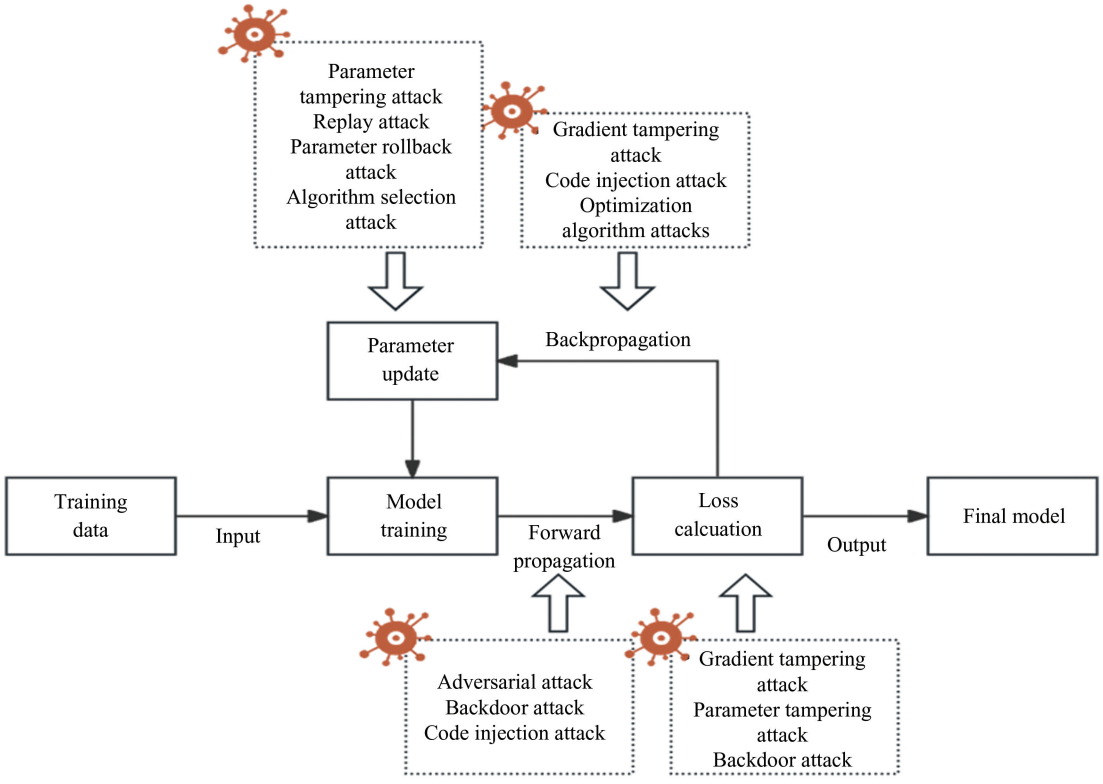

Fig.2 Different stages of model training process and their threats

2 Attack Methods That Compromise the Integrity of the Training Process

In model training tasks, dishonest training service providers or malicious third-party attackers frequently launch attacks on models during the training process. Since the training process typically operates as a black-box to users who only provide training data and model initialization information, while the cloud service provider handles all training work-most providers only deliver the final model results to users. Consequently, users lack of visibility into the training process. This opaque process is prone to be tampered with by malicious providers or attackers during the training process, such as reducing the number of hidden layers in the model structure, modifying intermediate results or adjusting model parameters. Small changes can cause significant problems to the model's performance. Providing users with all the training details is clearly impractical. Therefore, this section categorizes and analyzes threats to the integrity of the training process. Fig. 3 shows the final aggregated results.

2.1 Countering Attacks

2.1.1 Adversarial attack

Adversarial attacks are currently the most common problem in model training[18,19]. They are a specific type of attack that targets the machine learning process at the inference stage. Attackers tamper with input samples to cause the model to make erroneous classifications deliberately. Those with the ability to alter input data make subtle distortions to the original samples, causing the model to misidentify them. During the model training process, malicious training service providers may take advantage of their ability to finely tune the user-submitted training data, thereby, disrupting the normal training process and degrading the quality of the final model.

Goodfellow et al.[11] proposed Generative Adversarial Networks (GANs), and introduced the classical adversarial sample generation method FGSM (Fast Gradient Sign Method). The method that determines the gradient of the loss function for input samples to generate adversarial examples. Adversarial samples are generated by adding small perturbations in the direction indicated by the gradient to the original sample, which poses a threat to the entire model training process. Based on the above, Carlini et al.[20] introduced the C&W (Carlini & Wagner) attack method. Adversarial examples in deep learning using first-order randomized derivatives to minimize changes that make predictions incorrect. Compared to

FGSM, C&W attacks have better control over the size of perturbations and are thus more difficult to detect and have higher success rates, moreover, they also demonstrate the threat posed by adversarial attacks more thoroughly.

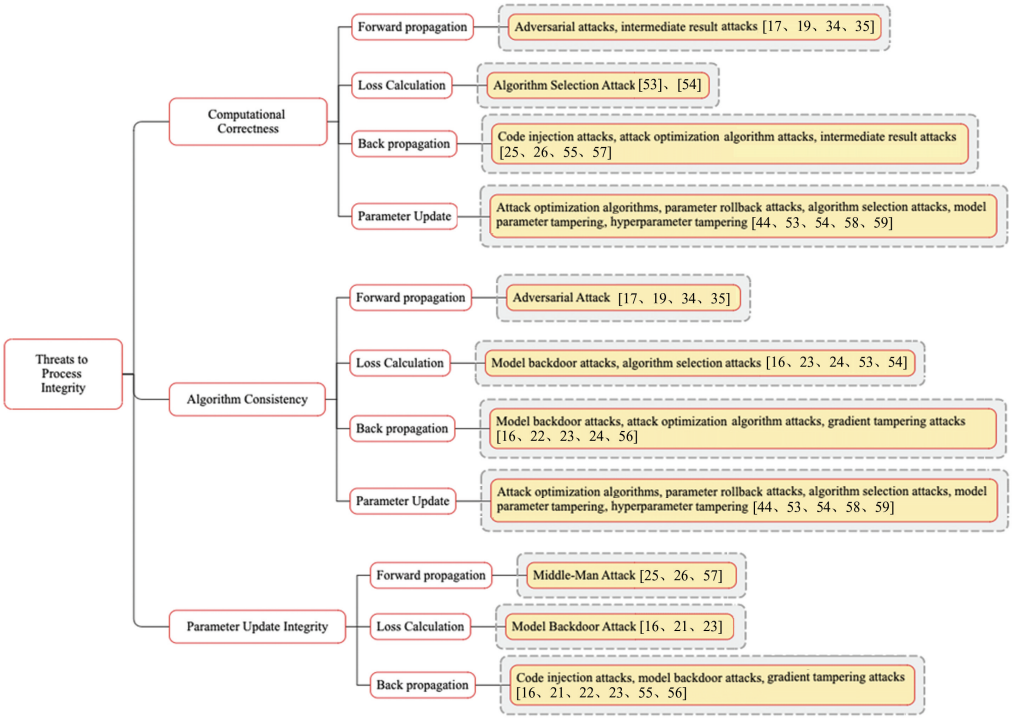

Fig.3 Threats to undermine process integrity during training

The above modification is not intuitive enough for the user to perceive, but it will make a difference to the model. Even a small change in the model training task will affect the results, regardless of the scale or accuracy of the training[21], and these small changes are sufficient to mislead users. Such attacks will seriously compromise the quality of the model's training data, thereby introducing significant security risks to the overall system. When applied to domains, such as speech recognition and autonomous vehicles, even a single error could lead to serious consequences.

2.1.2 Code injection attacks

Code injection attacks refer to attackers adding harmful code during the model's training process, which is not part of the training algorithm. Thereby, they can interfere with or manipulate the training process. In general, the attack environment can be altered freely by attackers without any constraints. They may illegally insert new data during the training phase, introducing errors or deviations to achieve their own goals.

Although the above two types appear to be similar, they differ fundamentally. In backdoor attacks, attackers introduce flaws in the code to set up backdoors that can be utilized later to produce incorrect model results. Model injection attacks, however, are that the attacker injects malicious code to alter the model's structure and reduce output accuracy. The former does not affect the precision of the model output, but the latter will be directly interfered with. Attacks can be carried out using this method in the model's training environment without authorization. For instance, they may inject malicious code during the input stage to implement erroneous data processing or add a wrong logic in training that makes the result deviate. This attack damages the data's ability to prevent integrity, breaks the normalization during training, degrades model performance. Adverse reactions or toxic-and-harmful substances may occur. Given its ability to precisely control the training process, code injection poses a serious threat to the integrity and security of the model.

2.2 Model Backdoor Attack

Model backdoor attacks[22-24] involve adding specific trigger conditions to the model during training. Within the scope of users' specified conditions, the model performs very well and does

not exhibit any abnormalities. However, when faced with the embedded malicious trigger conditions, the model behaves abnormally and loses its security. Therefore, this attack is covert and difficult to detect. Attackers usually have the ability to inject backdoors into the training data, model parameters or training process. Thus, it can function normally under normal circumstances and output the attacker's desired results after triggering the specific condition.

In the case of a backdoor attack, users cannot determine whether the model training service provider has acted dishonestly and implanted specific trigger samples during training. For example, when training a classification model, the service provider may include malicious trigger conditions. As a result, although the model performs well in response to regular samples submitted by users, it behaves abnormally under such circumstances and has a severe threat to the security of this model. Backdoor attacks have become prevalent during model training and now are among the most commonly used attacks. Many studies have focused on backdoor attack methods.

Patch backdoor is a new type of backdoor attack method proposed by Yuan et al.[25] Backdoor attack is usually hidden in the back-end system, so it cannot be easily detected by front-end operations. Attackers can activate the backdoor in the input sample without patching it. When users input patched data into the trained model, it misclassifies targets and maintains high accuracy for unpatched data.

Using the ImageNet dataset to conduct experiments can show that patch backdoor is effectively triggered by backdoors and has achieved high accuracy in normal images. Therefore, it proves the strong stealth performance of patch backdoor. Gu et al.[26] proposed the BadNets backdoor attack, which achieves backdoor functionality by adding samples that contain specific triggers to the training set. These sample labels are modified to the attacker's target labels. Backdoor attacks are diverse and structurally complex, making them hard to detect. Therefore, constant attention is essential. In particular, for outsourced tasks such as model training, appropriate protection measures must be implemented. If a model is infected with a backdoor, it carries unpredictable risks at any time.

2.3 Algorithm Selection Attack

Algorithm selection attacks refer to attackers interfering with algorithms during the training process of a model to sway results, seriously compromising the integrity of the training process. Different algorithms for training under the machine-learning model will produce different results in terms of performance and effect.

Attackers who can interfere with algorithm selection in the training environment intentionally use inappropriate algorithms for the current task. This will cause the training time to be extended, the model may not converge, or the performance will decline, which will affect the integrity of the training. Jagielski et al.[27] explored attack methods for algorithm selection in regression models and proposed corresponding countermeasures. Algorithms selected during training are part of the core issues, so this kind of attack poses an extreme risk to the security of the entire training process.

2.4 Parameter Rollback Attack

Parameter rollback attacks belong to a type of attack that rolls back the current parameter state to an earlier time and inputs these parameters into the new training iteration. If attackers can manipulate the parameters of transmission or tamper with training updates, they will replace the new parameters with old ones. It disrupts the model training process and optimization.

Parameter rollback attacks will not only interfere with parameter update synchronization but also disrupt the stable operation of the network via other means in multiple parts of this process. For example, in federated learning, the parameter updates generally need to be contributed by multiple parties. Each party will update its own part of the parameters and then submit them to the server. The server then combines all the updates. If a malicious participant replaces the intended parameter update data for the server with data from the previous iteration, the server's subsequent aggregation will slow down the convergence of the global model and disrupt the training parameter updates. This will eventually cause the model training results to deviate. If the malicious participant continuously sends erroneous update parameters, it may further impair the model training.

To sum up, parameter rollback attacks are difficult to defend against, and such attacks can significantly degrade the convergence speed of the model's training and may lead to a non-convergent state. Even during regular training, they can severely degrade the model's accuracy and impact the results of

the model.

2.5 Optimization Algorithm Attack

Attacks on optimization algorithms refer to attackers interfering with the algorithms used in model training to disrupt parameter updates and other aspects, thereby affecting the training results. Attackers are generally able to interfere with the internal computation of optimizers, allowing them to alter the execution sequence of optimization algorithms, create false optimization steps, or inject harmful gradients. It will cause the update of parameters to lose its effectiveness and training convergence is not guaranteed. These methods may cause the model to fail to achieve the expected learning convergence, thus failing to converge properly. The above attack method is relatively dangerous. By interfering with some key links in the training process, the attack on optimization algorithms will cause significant damage to the learning effect and generalization ability of the model. Therefore, it is necessary to have a deep understanding of this attack mode in order to improve the security of machine-learning models.

2.6 Parameter Tampering Attack

2.6.1 Model parameter tampering

A typical model parameter tampering attack involves altering parameters such as weights, biases, and learning rate, which are used to determine the extent of error during training[28]. Typical attack methods involve directly manipulating model parameters. Specifically, if the training service provider adjusts these parameters before training in a way that degrades the model performance or disrupts the training process, such actions can be considered an attack. Another method (e.g., gradient poisoning or weight poisoning) leverages the model update process: Malicious service providers can deliver malicious updates that alter global model parameters during backpropagation, severely degrading overall performance.

2.6.2 Hyperparameter tampering

Model hyperparameter tuning[29] differs from model parameters. The former is obtained through iterative updates in the training process, while the latter is preset before the training task starts and covers parameters such as learning rate and batch size. Attackers generally have the ability to tamper with hyperparameters before training, and by adjusting important hyperparameters such as the learning rate and batch size, they can interfere with the stability and convergence of the training process.

Malicious training service providers may also manipulate these hyperparameters before the start of training, leading to instability and reducing the number of iterations to lower training costs. This may also lead to sub-optimal performance of the model or deviant training results. As hyperparameters are the initial guidance for the model to achieve optimization during training, attackers can interfere with all stages of the training process through control of these parameters, thereby severely degrading the model's accuracy, robustness and overall effectiveness.

2.7 Gradient Tampering Attack

Gradient tampering attacks refer to attackers manipulating gradients to make the gradient clipping method for preventing gradient explosion fail[30,31]. Attacks may be conducted in which attackers alter the values of gradients to change the direction or magnitude of these gradients, thereby causing clipping operations to be either too severe or too mild. This interference with the parameter update process and directly damages the normality of these updates. A more harmful kind of attack can interfere with the parameter optimization process directly, thus impacting the generalization ability of the model.

For instance, when the model training task is submitted to the cloud service provider for processing, if there is a risk of gradient explosion during training, users can integrate gradient clipping into the training algorithm to address this problem. Although it is to some extent beneficial for ensuring the stability of the training, such an approach has increased the amount of required computation per iteration. However, if a malicious cloud service provider deliberately increases the clipping area or manipulates gradient values for economic purposes, the learning process will not be able to continue. The model will be unable to adapt well to the complexity of the training data, resulting in reduced accuracy, precision and generalization ability.

Yin et al.[32] investigated the impact of submitting erroneous or malicious gradient information on the general convergence of models in distributed machine learning. This type of attack alters the direction or magnitude of the gradient, leading to overall updates deviating from The correct course. This may make the model unable to reach the global optimal solution during training, and instead, it is

partially iterated, oscillates, or appears to have converged in advance. These types of attacks are relatively common in the distributed training and model outsourcing scenarios, which not only degrade the efficiency of training but also seriously harm the security of the entire training process.

2.8 Man-in-the-Middle Attack

A man-in-the-middle (MitM) attack refers to the adversarial behaviour of an attacker who intercepts and manipulates the information exchanged during model training or inference. By placing themselves in the position of intermediary entities or at the connection points of computations, attackers can manipulate the intermediate results being transmitted, thereby threatening the integrity and accuracy of the training data.

Such attacks may include intercepting data transmission, modifying intermediate activation values or gradients, inserting malicious code, or damaging model update information during inter-node communication. This operation will affect the high-efficiency and secure operation of this model, resulting in a judgment error or disordering its stable running state during the training period. There is also an uncertainty in the result achieved in subsequent inference steps. Previous research has shown that by aiming at intermediate-layer features or computational outputs, the success rate of adversarial attacks can be significantly increased[33,34]. By interfering with the transmission of information across layers, MitM attacks induce cumulative deviations in computation, ultimately guiding model training away from the expected optimization path and compromising the integrity of training.

3 Training Process Integrity Verification Scheme

3.1 Verification of Training Process Integrity Based on zero-knowledge Proofs

zero-knowledge proofs[35] refer to the process of demonstrating specific information to a verifier without revealing any other information, hence it is called "zero-knowledge". To ensure the integrity of the model training process, this can be formalized as the follows.

$$R_{\text{train}}:\left\{(x,w)\left|\begin{array}{c} w_{t+1}=w_t-\eta_t\nabla_{w_t}L(f_{w_t}(x_t),y_t) \\ t=0,\cdots,T-1, \\ C_t=\text{Commit}(w_t,r_t) \\ H_{t+1}=\text{Hash}(H_t,C_t) \end{array}\right.\right\}$$

Here, $x=(crs,C_T,H_T)$ is the public input, in which crs is the common reference string, C_t is the final commitment of the model parameters, and H_T is the final hash state of the training process. The witness is defined as $w=(\{w_t\},\{r_t\})$, which is the private witness. w_t represents the model parameters at iteration t, and r_t denotes the randomness used in the commitment scheme. C_t denotes the commitment of w_t, and H_t denotes the cumulative hash state used to ensure the sequential integrity of the training process.

$$\text{Verify}(x,\pi)=1\leftrightarrow(x,w)\in R_{\text{train}}$$

Where x denotes the public input, w denotes the witness (e.g., model parameters and intermediate values during training), π denotes the proof generated by the prover, and R_{train} represents the relation describing the correctness of the training process.

zero-knowledge proofs can ensure the integrity of the training process during verification and also maintain the confidentiality of the model. If the entities that need to provide evidence (proof providers) and those that need to verify it (proof verifiers) do not require frequent interactions in the verification scheme, it is called a Non-Interactive Zero-Knowledge proof (NIZK). Verification schemes that require multiple interactions are called Interactive Zero-Knowledge proofs (IZK). In the extant model training integrity verification scenarios, research efforts have focused on the non-interactive zero-knowledge proof method[36]. Among them, zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) is the most typical protocol for verifying the integrity of the training process, and at present, it is the most closely integrated protocol combining zero-knowledge proofs with model training process integrity verification.

To address the issue of long proof generation time and computation-proportional proof size in zk-SNARKs for logistic regression models, some researchers proposed an innovative approach that combines zk-SNARKs with MPC-in-the-Head technology. The introduction of MPC-in-the-Head involves decomposing complex computational processes into several parts and generating corresponding zero-knowledge proofs through zk-SNARKs for each part. The work by Garg et al.[8] represents the initial exploration of systematically integrating zero-knowledge proof mechanisms into model training verification. zk-SNARKs with the MPC-in-the-Head paradigm allows for the production

of verification proofs at every step of the model's training calculation without tracking all these calculations.

Additionally, due to the streaming nature of the proposed verification system, it does not need to load all computations into memory, resulting in a better performance. In particular, the software performs well in terms of both proof size and computation time. However, as with many other machine-learning models, the cluster centers of K-means involve numerous non-arithmetic operations, making some additional costs are unavoidable.

This paper presents a method based on zk-SNARKs to achieve a partial reduction in proof size. However, there are still problems with communication complexity and the proportionality of proof size to computational effort. According to the experiment results, zkPoT (Zero-Knowledge Proof of Traning) can address multiple issues in machine learning at the same time, such as proof generation time and proof size, etc. Improving the rate of verification, this result was discovered through research and development. Taking into account the time required for retraining the entire model, its verification speed is also faster. Therefore, zkPoT has achieved significant results and is expected to be put into use in practice.

Building on earlier studies on shallow models, subsequent work has further explored the applicability of zk-SNARKs in deep learning models, particularly CNNs[37,38]. The application of zero-knowledge proof model to CNN is intended to introduce the concept of zk-SNARKs, thereby enhancing the privacy and integrity of CNN predictions. Traditional zk-SNARKs generally need to convert the computational problem into a circuit form. Not all computational problems can be naturally presented in circuit form during the process of model training.

To address this problem, related research aims to improve the forward propagation stage of CNNs, thus converting it into a particular type of computational problem. This transformation can be matched to the characteristics of Non-deterministic Polynomial-time (NP) problems and abstracted as arithmetic circuits. It should be noted that considerable computational load required to produce zk-SNARK proofs, combined with the large size of CNNs themselves, makes direct generation of an overall proof unfeasible. Therefore, the CNN's inference process is decomposed into several layers and a multi-level proof technique is proposed as a solution. The generation of proofs for each layer is an independent process, and then these proofs are aggregated to obtain the final result. Simultaneously, the integration of operations among adjacent layers (such as combining convolution and activation operations into a composite operation) leads to a decrease in the number of required computational operations and circuit complexity, thus reducing the arithmetic circuit scale.

Fan et al.[39] presented VeriCNN, a zero-knowledge proof-based approach for validating CNN models. Compared to previous studies that have focused more on verifying the process of inference, this paper was set based on a large-scale CNN model's training scenario. Experiments showed that the VeriCNN approach has certain defects; The VeriCNN approach focuses only on input-output verification and fails to validate and optimize the training process itself. When the CNN network reaches a certain scale, the computational overhead of zkCNN becomes unfeasible. Therefore, Fan et al.[39] further proposed a comprehensive solution for verifying the integrity of CNN model training. The approach is to convert the CNN training process into a Rank-1 Constraint System (R1CS) and then uses customized design for the structural features of CNN training. Compared to the traditional general transformation method, it has better validation efficiency.

To address the substantial increase in proof generation time caused by the model's scale, the authors implemented a layered optimization approach. First, they leveraged the local computing characteristics of CNNs to reduce unnecessary global constraint computations and thereby reducing computational complexity. Subsequently, they demonstrated that the batch processing of multiple input samples and the sharing of some computation steps further reduces proof generation time.

Sun et al.[40] further demonstrated the expanded potential of zero-knowledge proofs in verifying model training integrity through their zkDL approach. Compared to the previous work that focused on optimizing for several particular model structures, their approach aims to optimize a wide range of general deep learning models, by proposing an efficient zero-knowledge proof mechanism to verify the integrity of the training process. Their zkDL approach can be considered an extension and

modification of zkPoT, and there is a special circuit structure designed to implement the computation of each layer in a deep learning model more efficiently.

Additionally, zkReLU is introduced and applied to offer special proofs for the ReLU activation function and its backpropagation process. Moreover, FAC4DNN, an arithmetic circuit design methodology for model neural networks, is introduced to combine proofs from different network layers and training steps. After extensive experiments with various deep-learning models, it was determined that zkDL has the best performance flexibility. It has overcome some defects of a single model structure and is expected to improve scalability. The experiment results show that zkDL performs substantially better than other conventional methods or zkPoT.

Abbaszadeh et al.[41] proposed a general zero-knowledge proof framework, albeit with a different optimization approach. The main task of their work was to verify, at the system level, whether the training process of deep neural networks strictly followed the preset algorithm and data flow-not merely to confirm that the final model achieved expected results. To enhance the training performance, an improved optimization strategy is proposed. Due to significant structural differences among deep learning models, it is difficult to design a unified zero-knowledge proof scheme that can be applied to all components of the model, which limits its scalability. Therefore, a modular Zero-Knowledge Proof (ZKP) framework has proposed to support different Deep Neural Network (DNN) architectures flexibly. Its benefit is that during the training of the model, each part can be analyzed separately. Therefore, an effective evaluation system has been established for different architecture networks.

Abbaszadeh et al.[41] further refined the conventional R1CS constraint system by simplifying constraints and combining linear constraints. The number of R1CS complexity can be reduced through constraint simplification and combining linear constraints, which led to a reduction in the computational load of the proof generation. In addition to fast commitment schemes and encoding compression techniques, it also showed that an improvement in the rate of efficient verifiable secret sharing could be achieved.

3.2 Verification of Training Process Integrity Based on Homomorphic Encryption

In training integrity verification, homomorphic encryption can be formalized as:

$$Dec_{sk}(Eval_{pk}(f, Enc_{pk}(x))) = f(x)$$

where x denotes the input data, f represents the target function, pk and psk denote the public key and secret key respectively in the homomorphic encryption scheme, Enc_{pk} and Dec_{sk} represent the encryption and decryption operators, respectively, while $Eval_{pk}$ denotes the direct execution of computational function f on ciphertext. This property ensures that even when computations are performed within the encrypted space, the decryption results remain consistent with plaintext training computations. Consequently, the correctness of the training process can be verified without compromising data confidentiality.

Homomorphic encryption is a type of cryptography that enables computations to be performed directly on encrypted data. The decrypted output of this process is guaranteed to be equal to the result of calculating the plaintext. This technology can be used to maintain the accuracy of training calculations and protect sensitive data. Homomorphism in encrypted computation is broadly divided into three types: partially homomorphic encryption, bounded homomorphic encryption (also known as limited number-of-times multiplicative homomorphic), and fully homomorphic encryption. For model training, most research aims to reduce the performance overhead of homomorphic operations while maintaining system computability. Existing schemes, including those based on the Learning with Errors (LWE) problem, have shown relatively good performance for homomorphic computation during model training.

Under the backdrop of an outsourced model for training tasks, homomorphic encryption can ensure that the computation process occurs in ciphertext form at the cloud service provider when they cannot be trusted. Thus, ensuring the accuracy of the calculation results and reducing the possible threat of intentional interference by cloud service providers in the training process. Improvements to security have increased the amount of computations required, therefore, it may be necessary for users to adjust their budget or make a more economic choice when making a selection. Given that the necessity of complex mathematical operations and large-scale computation in the ciphertext domain has led to the fact that homomorphic encryption generally has a significantly

lower degree of computational efficiency than plaintext processing. This will cause an increase in calculations, a decrease in the speed of response, and an elevation in both the cost of developing and implementing the system. Therefore, the main problem of homomorphic encryption in model training is balancing security and computation efficiency.

Given the above reasons, Wood et al.[42] systematically explored the application of homomorphic encryption in both the training and inference stages of machine learning. The emphasis was on the execution results of basic operations, such as addition and multiplication, on the encrypted data. To improve the computational efficiency of homomorphic encryption algorithms continuously optimized to broaden their practical applications across real-world machine-learning models. This leading-edge work has established a benchmark for later research on other varieties of neural networks, and it has also opened up new directions for optimization exploration.

Gilad-Bachrach et al.[43] presented CryptoNets, an efficient neural network that realizes fully homomorphic encryption. This work represented the first attempt at designing homomorphic encryption jointly with neural network architectures. Through the design of specific neural network structures to meet the computational characteristics of homomorphic encryption, for instance, by reducing the number of activation functions, Gilad-Bachrach et al.[43] lowered network complexity. As a result, the computational load during training and inference was reduced, thereby improving the operational efficiency of homomorphic encryption in neural network inference.

From the above description, it can be known that the traditional homomorphic encryption has relatively poor efficiency in performing addition and multiplication operations. CryptoNets reduces the number of encryption operations and further lowers overall computational complexity by combining efficient linear operations with carefully chosen nonlinear activation functions. In addition, Gilad-Bachrach et al.[43] proposed a method to convert neural network weights into homomorphic encryption format that is both efficient and optimized to minimize additional overhead during the encryption and decryption processes.

Based on this foundation, Phong et al.[44] further proposed a method applying additive homomorphic encryption to deep learning models. Earlier schemes depended entirely on fully homomorphic encryption, while their work employed a variety of methods for different computational operations to improve the overall framework's efficiency. They particularly emphasized that addition operations were both more frequent and computationally more demanding in the homomorphic encryption environment. For computationally expensive multiplication operations, approximate computation or auxiliary methods were used. This approach can significantly reduce the total amount of computation while maintaining the level of homomorphic encryption close to that non-homomorphic encryption.

Brakerski[45] proposed a new fully homomorphic encryption scheme from the perspective of the cryptographic mechanism itself to improve the efficiency of homomorphic operations. This work improved the system of homomorphic encryption at the cryptographic architecture level to provide a foundational basis for developing efficient homomorphic computation applications subsequently. Brakerski[45] introduced a full-homomorphic encryption design that eliminated the need for modulus switching and avoided the complex and expensive modulus-switching process present in traditional schemes.

On the other hand, to increase noise, only a single method of noise control was used in conjunction with simpler parameter setting operations. This design achieved an important break-through in computation efficiency as well as considerably reduced the complexity of the encryption system architecture. Therefore, the practical applications of full-homomorphic encryption are more extended in computationally intensive situations, such as model training and integrity checks.

3.3 Verification of Training Process Integrity Based on Multi-Party Secure Computation

Secure Multi-Party Computation (MPC) in the training process integrity verification refers to a security protocol that allows multiple parties to collaboratively train a model and complete computational tasks accurately. In this case, all parties will jointly execute the training step and interactively check the correctness of the computation to ensure that the model training process is not subject to malicious tampering or deviation from actual expectations. Finally, the trained model results can be verified by reaching a consensus through this protocol to ensure

the accuracy of the model training process.

In training integrity verification, secure MPC can be formalized as collaborative computation:

$$f(x_1, x_2, \cdots, x_n) = y \rightarrow \{MPC_i(x_i)\}_{i=1}^{n}$$

where x_i represents the private input of the i-th participant, and $MPC_i(\cdot)$ denotes the local computation and interactive operations performed by each party under the protocol. This property ensures that all participants, without disclosing their respective private data, jointly compute results consistent with the output of the ideal function f, thereby validating the correctness and consistency of the training process.

Existing MPC protocols, technically, can be primarily divided into two categories: those based on secret sharing methods and those involving obfuscated circuits. The former has some advantages, such as low communication overhead and high throughput, and the number of interaction rounds is usually linearly related to circuit depth. However, the latter has a fixed number of rounds and requires greater communication bandwidth.

In the application scenario of model training integrity verification, MPC is generally not used alone but combined with other mechanisms, such as zero-knowledge proofs or homomorphic encryption. This combination is expected to effectively allocate and optimize computational workload, thereby improving the efficiency of proof generation and verification. In this hybrid setting, performance improvements are more significant in large-scale models and complex training tasks.

Garg et al.[8] introduced the security protocol concept of MPC and combined it with zk-SNARKs. This method utilizes the low verification cost of zk-SNARKs and the computational efficiency of MPC in proof generation. The combination of the these two features makes this check practical in real-world use.

Based on ZKP + MPC method, some later studies have integrated homomorphic encryption into the system and proposed a training framework that combined deep learning with this system. For example, Wagh et al.[46] proposed the SecureNN framework and introduced homomorphic encryption for the first time into secure multi-party computation. They utilized the Paillier and BGN cryptosystems to achieve encrypted computing. The decompositions of the convolutional neural network model into several computational subtasks are realized through the assignment of these subtasks to different participants for execution.

The collaborative completion of model training using MPC protocols aims to reduce the computational load on individual participants. To solve the problem of frequent communication in MPC scenarios, Wagh et al.[46] introduced improved training algorithms, such as efficient gradient computation and adaptive learning rate strategies. These enhancements not only improve the computing efficiency but also significantly reduce communication overhead. Therefore, overall, this approach is practically feasible.

SecureNN is a framework for protecting the privacy of training data while ensuring the security and accuracy of model training. Compared to traditional MPC schemes, SecureNN can reduce the unnecessary computational complexity and communication cost in the process of training deep neural networks through hierarchical communication strategies, thereby improving efficiency. In addition, SecureNN can also be used to maintain the integrity of training in encrypted computations through particular protocol designs. For example, SecureNN can generate verification information after the computation to ensure that the training process is tamper-proof.

SecureML framework of Mohassel et al.[47] is similar in its general method to SecureNN, which combines homomorphic encryption with MPC to enable multiple parties to collaboratively train models without revealing raw data. Mohassel et al.[47] developed a series of encrypted-computation methods for different machine-learning tasks to enable the updating of model weights and gradients in ciphertext form during the training process. Compared to SecureNN, which emphasizes system and communication optimization, SecureML places greater emphasis on protecting the privacy of training data and additionally introduces the verification mechanism of digital signature. To improve the anti-manipulation capability of the training process, these mechanisms aim to certify the correctness of the results across all nodes in this system. This mechanism can ensure that the model training and process verification are conducted with privacy protection when the data and computations are distributed across multiple parties.

3.4 Verification of Training Process Integrity Based on Trusted Execution Environments

A training process integrity verification method

based on Trusted Execution Environment (TEE) utilizes the security isolation mechanism offered by the hardware level to guarantee the trustworthiness of model training. Some typical cases include Intel SGX and ARM TrustZone. By creating an independent and isolated execution environment within the processor, TEE can prevent interference and tampering from the operating system or external entities during the training process when executing programs and processing data in this area. In addition, TEE can also support integrity measurement and verify the executed code during the system boot process or when an enclave is loaded to ensure that the actual program run in training aligns with the expected logic.

TEE can also do remote attestation and issue a certificate for external verifiers to prove that the model training and parameter update specific tasks were performed in the trusted environment. This mechanism provides a hardware-based root of trust for ensuring the integrity of the training, allowing the verifier to gain confidence in the training without fully trusting the cloud service provider.

In the MLaaS (Machine Learning as a Service) scenario, Zhang et al.[48] proposed VeriTrain framework that used the trusted execution environment provided by TEEs for online monitoring and verification of the model training process. By using anomaly detection to identify deviations from expected behaviour during training, thus the integrity of training can be validated without having to go through the full process of replaying training.

Additionally, the GINN system proposed by Asvadishirehjini et al.[6] uses Intel SGX as the underlying trusted execution environment, and all key sensitive operations in model training, such as parameter initialization, batch selection, and related control logic, are executed within enclaves. In contrast to the whole-process validation, GINN generates incrementally the proof of each training batch to reduce the amount of computation. After each iteration, the system automatically saves a training state snapshot. In the event that verification is required, only the sampled iteration snapshot needs to undergo a validation process, and there is no need to re-validate the entire model. This design supports the asynchronous training and verification process, which helps to maintain the integrity of the training process without affecting the performance of training. Table 1 summarizes the main verifiable training schemes discussed in this section.

Table 1 Summary of training process integrity verification scheme

Method	Technical features	Features	Limitations
[8]	Based on ZKP	Structured ZKP, reduced cost	Verification too slow
[7]		Proposing a zero-knowledge proof ZKP for CNN models	Verifying CNN models high overhead development requirements
[40]		Extends ZKP to deep nets	Poor scalability complex neural networks
		Structured proof optimizations	Costs grow with model size
[39]		Lower proof generation overhead	Limited flexibility
[49]	Based on Homomorphic Encryption (HE)	Encrypted neural inference	Too slow for real-time use
[44]		Efficient for additive DL tasks	High compute + communication cost
[45]		Simplifies FHE operations	Low efficiency on complex operation
[46]	MPC	Hybrid MPC, efficient training	Heavy cost on large datasets
[47]		Highly scalable distributed MPC	High latency + synchronization overhead
[48]	Based on TEE	Detects training anomalies	Poor adaptability to new patterns
[6]		Verifies DNN training integrity	Over-reliance on TEE & high overhead

3.5 Summary

The existing verification schemes to ensure the integrity of the model training process can be divided into four categories: zero-knowledge proofs, homomorphic encryption, multi-party secure computation, and trusted execution environments, as summarized in Table 2. Although all these approaches aim to minimize the impact on the training process, it is inevitable that they introduce trade-offs in terms of performance overhead, system construction cost, and security assumptions. In particular, the complexity of system design and implementation directly affects their

applicability in large-scale training scenarios.

ZKF-based (Zero Knowledge Proof) methods are entirely based on mathematically difficult-to-crack assumptions. The security of their reliance on cryptographically based assumptions rather than any other entities or hardware. Verifier can confirm that the training results are accurate, no need to trust the provider of the training service or its underlying runtime environment. They have a natural advantage in the untrusted cloud environment and the scenario of model training outsourced. However, this strong security guarantee comes at the cost of high implementation complexity. Moreover, the entire training process needs to be precisely converted into arithmetic circuits or constraint systems, and the correctness of this conversion must also be carefully designed.

TEE-based (Trusted Execution Environment) approaches, on the other hand, trust specific hardware and their manufacturers, such as Intel SGX or ARM TrustZone. Any defects in the implementation of the hardware or side-channel protection mechanisms will directly affect the security. Therefore, TEE methods inherently require trust in the hardware itself and do not provide a trustless verification mechanism. Secondly, TEE solutions are hardware-based. They provide execution isolation, but unlike ZKPs, they do not verifiable execution. As a result, TEE environments remain merely trusted black boxes for verifiers and offer poor verifiability. TEE-based methods have a relatively simple engineering implementation approach and can run training programs with minimal modifications, but enclave memory limitations, system calls and hardware-specific programming models need to be taken into account carefully.

Table 2 System comparison of different training process integrity verification schemes

Method	Cryptographic foundation	Computational overhead	Communication overhead	Verification overhead	Applicable scenarios
ZKP	Training encoded as an arithmetic circuit of size C; security based on polynomial commitments and hardness assumptions (e.g., DLP, q-SDH)	$O(C)$	$O(1)$ or $O(\log C)$	$O(1)$	Untrusted cloud environments, outsourced training
HE	Encrypted training of scale n using lattice-based homomorphic encryption	$O(n \cdot \mathrm{poly}(\lambda))$	$O(n)$	$O(n)$	High-privacy encrypted training
MPC	Training of scale n via secret sharing among mmm parties	$O(n)$	$O(nm)$	$O(n)$	Multi-party collaborative training
TEE	Training of scale n executed in trusted hardware with remote attestation	$O(n)$	$O(1)$	$O(1)$	High-performance, controlled environments

Secondly, in terms of verifiability, both Homomorphic Encryption (HE) and ZKPs can be used to verify information without revealing the training data or model parameters, while their main objectives are different. ZKPs emphasize the verifiable guarantee of the correctness in training being executed according to a defined algorithm. However, homomorphic encryption focuses on performing computations without revealing the plaintext. Integrity checks generally depend on the result's agreement or add an additional verification step. The main problem is the increased computation overhead in the ciphertext state, and an auxiliary validation mechanism must also be added. Therefore, most homomorphic encryption schemes have a high implementation difficulty, and the training algorithms are usually customized. At the same time, carefully designed ciphertext operations and parameter tuning need to be performed to achieve a balance among correctness, efficiency and security.

Multi-party Secure Computation (MPC) strikes a balance between ZKP-based methods and HE-based approaches, using collaborative computation of multiple parties to avoid single-point risks. However, its security is based on the premise that the participants will not collude, and it cannot be used in practice due to the need for a trusted setup. In terms of implementation, MPC systems introduce some degree of complexity, which requires the cooperation of several parties, protocol synchronization, and robust communication infrastructure, therefore, large-scale deployment is more complex.

Specifically, in terms of engineering implementation, the priorities of different ZKF schemes are not entirely the same due to the underlying

technology. Although these plans typically have a short verification time and low communication overhead, the computational cost of the proof generation stage is relatively high. In particular, at the time of large-scale deep model training, circuit optimization faces more prominent problems, therefore, it has become a current research focus. As mentioned earlier, the homomorphic encryption schemes introduced substantial additional computation overhead during the training phase and had significantly lower training efficiency than plaintext computation. The primary bottlenecks of MPC schemes are communication costs and synchronization overhead, and scalability is limited as the number of participants increases. TEE schemes have computational performance closest to native training, but they are constrained by the size of trusted memory and hardware dependence. Recently, research has also shown that there is a potential risk of side-channel attacks[50].

Different solutions are suitable for different scenarios: zero-knowledge proofs are applicable to highly untrusted cloud training outsourcing, and homomorphic encryption and MPC are more appropriate for collaborative training environments that have strict requirements for data privacy but can accept additional computational or communication overhead. On the other hand, TEEs are better suited for high-performance deployment scenarios that can accept hardware-trust assumptions.

Nevertheless, each mechanism is still plagued with considerable issues when it comes to actual system application. zero-knowledge proofs need to construct a complete circuit and generate a large proof, which incurs high computational and memory costs. Homomorphic encryption has low computational efficiency on ciphertext, a reduced training speed and a complicated system implementation. MPC requires frequent communication and protocol synchronization, while its performance is more susceptible to network stability and the number of participants. TEEs are subject to the constraints of hardware trust and memory, and they are still prone to side-channel attacks. All these problems may impact the scalability and deployment efficiency of the solutions in actual use.

4 Evaluation

4.1 Dataset

Regarding the research on the integrity of training process, even though datasets do not directly decide the core algorithms of the verification mechanism, they are essential for experiments to verify effectiveness. Given that the focus of such research is on verifying the universality and scalability of the proposed solutions, most of the existing work has utilized standardized, general-purpose datasets. Table 3 lists some representative general-purpose datasets and their key features to provide a basis for subsequent experimental design and performance assessment.

4.2 Evaluation Indicators

In terms of research on training process integrity verification, the design and selection of assessment indicators are key to evaluating the overall impact of the applied schemes more effectively and practically. Given that these verification mechanisms generally require multi-dimensional performance indicators, most of the existing research works have systematically evaluated their performance from multiple dimensions, including but not limited to verification accuracy, computational overhead, communication overhead, verification latency and scalability. Table 4 summarizes the common evaluation metrics and their definitions from the literature, providing a reference for subsequent performance analysis and comparison of the methods.

Table 3 Summary of datasets

Dataset name	Brief description	References
CIFAR-10	32×32 RGB images from 10 classes	[6,15,40,41,51]
MNIST	28×28 grayscale digit images, 60k train / 10k test	[6,15,21,41,46]
EMNIST	28×28 grayscale handwritten letters and digits, 280k train / 40k test	[51]
CelebA	202599 color face images with 40 attribute labels, cropped and aligned, 162770 train / 19867 valid / 19962 test	[52]
GTSRB	43-class color traffic sign images, variable size (32×32 typical), 39209 train / 12630 test	[6]
UTKFace	23k color face images labeled by age, gender, and ethnicity, cropped to 200×200	[48]

Table 4 Summary of evaluation metrics

Evaluation indicators	Applicable scenarios	References
Time overhead	Measures the extra time introduced by the validation mechanism during training.	[8,15,40,46,48]
Storage overhead	Measures the extra storage consumption introduced by validation schemes for model or parameter data.	[8,40,46]
Accuracy	Evaluates whether the integrity verification mechanism degrades model performance.	[6,46,51]
Z-Score	Measures integrity bias by detecting anomalies in the standardized output probability distribution during training or inference.	[53]
NIC	Assesses neuron activation coverage to validate training sufficiency and identify inactive computational pathways.	[52]
MagNet	Evaluates the model's robustness and validation performance under adversarial perturbations.	[54]
Supervised LID (Local Intrinsic Dimensionality)	Detects anomalous training samples and verifies whether the model has been corrupted.	[55]

4.3 Evaluation Model

In terms of research on the integrity of the training process, an evaluation model is built to assess comprehensively from multiple aspects, such as whether the effectiveness and efficiency meet the requirements and whether the approach exhibits good scalability. In research on training process integrity verification, the proposed framework evaluates the overall performance of the training process from multiple dimensions, including computational cost, proving efficiency, verification efficiency, and scalability. These models can exhibit the flexibility of the verification mechanism in different types of tasks, and it has been demonstrated that this approach is computationally efficient, communicationally low latency, and secure. Typically, researchers design validation experiments using various deep learning models (e.g., MLP(Multi-Layer Perceptron), CNN, RNN (Recurrent Neural Network)/LSTM (Long Short-Term Memory), GNN(Graph Neural Network) Transformer) to verify the universality and robustness of the scheme across different network structures and data scales, and the specific evaluation models are listed in Table 5.

Table 5 Summary of models

Evaluation model	Applicable scenarios	References
MLP	Basic neural network tasks; lightweight training	[6,8,40,46]
CNN	Image classification, object detection, and vision tasks	[7,15,39,43,54]
RNN / LSTM	equential tasks, medical records, time-series modeling	[2,3,16]
Transformer	large-scale NLP tasks, long-sequence modeling, LLM evaluation	[5,14,51,52]
ResNet / YOLO / Other Deep CNNs	Deep neural architectures; complex vision benchmarks	[1,12,13,56]
GNN	Graph-structured data, relational reasoning, anomaly detection	[52]

5 Research Challenges and Future Directions

With the rapid development of machine learning technology in recent years, especially deep learning, many problem model scale has increased dramatically, and training cost has accordingly risen. Therefore, the training task for the outsourcing model is often given to qualified cloud service providers. However, this situation also has many problems. At present, most research focuses on privacy protection for training data and integrity verification for model data; however, verification of the training process itself remains inadequate. The existing solutions have shown limited effectiveness in practical model training, and many problems remain unresolved urgently. The following four kinds of problems need to be studied further.

5.1 Optimizing the Computational Overhead of zk-SNARKs

Most existing research has focused on using zk-SNARKs to achieve a model training process that is

zero-knowledge, but the computational cost of generating such proofs remains a problem in practical applications. The high computational requirements and long generation time of zk-SNARKs in deep learning tasks limit their practicality for large-scale deep model training. Therefore, in the future, we should further improve the generation efficiency of zk-SNARKs and develop more efficient circuit optimization methods that better match deep learning training tasks. Specifically, by designing simplified generation circuits, reducing redundant computation steps and introducing new succinct circuit structures, the computational complexity can be reduced without affecting the accuracy of the proof, then, zk-SNARKs will become more feasible for use in deep learning scenarios. Such optimizations can help improve the overall efficiency of the training process and enhance the practical application scenarios for zk-SNARKs in model training, which is also one of our research directions.

5.2 Optimizing Communication Overhead in Multi-Party Secure Computation

At present, the communication overhead bottleneck of Multi-Party secure Computation (MPC) in deep learning training is mainly due to the frequent data exchange and complex synchronization process that existing protocols need to undergo repeatedly. As the scale of model parameters and data continues to expand, this communication cost problem will be more prominent. To this end, future research will focus on the construction of an improved communication coordination mechanism. Lightweight MPC protocol design or intelligent data compression strategy implementation to reduce communication volume and improve bandwidth utilization. In addition, integrating differential privacy-assisted protocols can help reduce communication frequency, thereby reducing bandwidth consumption and latency during data exchange. These optimizations can improve the practicability and expandability of MPC for model training.

5.3 Research on the Scalability of Model

Existing model integrity verification schemes primarily address small-scale or single-node model validation, since they lack scalability for large-scale applications in distributed or multi-node systems. Therefore, the existing approaches are not applicable for handling the high complexity of deep learning models and the diversity of data distribution. Therefore, future research will focus on developing model integrity verification mechanisms for large-scale distributed training. New cross-node consistency check protocols need to be established, and hierarchical verification structures should also be introduced to support large-scale distributed deep learning applications. At the same time, to improve the efficiency of verification, it is necessary to ensure both data consistency among computational nodes and training integrity of the entire model, thereby maintaining the reliability of the training process in a multi-node environment.

5.4 Enhance the Efficiency of Combined Multi-Factor Authentication Schemes

Current model training integrity verification schemes typically operate independently, without coordination among different verification methods and unable to achieve optimal verification efficiency. Therefore, in future research, efforts should be made to develop a combination of multiple verification techniques, such as integrating zk-SNARKs and MPC, etc., to achieve a balance between the comprehensiveness of verification and its efficiency. Research should be conducted to explore the dynamic switching of verification schemes and apply the most suitable approach in its area of strength to achieve higher verification efficiency. In addition, through customized verification strategies in different situations, the practicability of the combined verification approach can be improved and its ability to support integrity checks of model training data over a wider range and more flexibly can be enhanced.

6 Conclusions

This paper reviews cutting-edge literature on verifying the integrity of model training processes, examines the attack methods and threats currently faced by training processes, and categorizes these threats for analysis. Unlike other evaluations, the primary purpose of this study is to ensure the integrity of the training process and has relatively low requirements for privacy protection. Data security needs to be protected during training, but at the same time, the training process is also directly related to data security. This paper aims to offer new ideas and directions for research, especially given that the cost of model training continues to rise. The accuracy, consistency and completeness of parameter update

processes during training need to be improved collectively.

In addition, this paper identifies the future research direction. In terms of technical improvement, a balance needs to be struck between security and efficiency. It is also expected that zero-knowledge proofs will be combined with other privacy-protecting technologies, such as homomorphic encryption and secure multi-party computation. Although some previous research results have been achieved in this direction, they are still in their early stages.

References

[1] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2016: 779-788. DOI: 10.1109/CVPR.2016.91.

[2] Rajkomar A, Oren E, Chen K, et al. Scalable and accurate deep learning with electronic health records. NPJ Digital Medicine, 2018, 1: Article number 18. DOI: 10.1038/s41746-018-0029-1.

[3] Ghosh S K, Khandoker A H. Investigation on explainable machine learning models to predict chronic kidney diseases. Scientific Reports, 2024, 14: Article number 3687. DOI: 10.1038/s41598-024-54375-4.

[4] Bhatore S, Mohan L, Reddy Y R, et al. Machine learning techniques for credit risk evaluation: A systematic literature review. Journal of Banking and Financial Technology, 2020, 4: 111-138. DOI: 10.1007/s42786-020-00020-3.

[5] Brown T B, Mann B, Ryder N, et al. Language models are few-shot learners. Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM Digital Library, 2020: 1877 - 1901. DOI: 10.5555/3495724.3495883.

[6] Asvadishirehjini A, Kantarcioglu M, Malin B, et al. GINN: Fast GPU-TEE based integrity for neural network training. Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy. New York: ACM Digital Library, 2022: 4 - 15. DOI: 10.1145/3508398.3511503.

[7] Liu T, Xie X, Zhang Y. ZkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Digital Library, 2021: 2968 - 2985. DOI: 10.1145/3460120.3485379.

[8] Garg S, Goel A, Jha S, et al. Experimenting with zero-knowledge proofs of training. Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Digital Library, 2023: 1880 - 1894. DOI: 10.1145/3576915.3623202.

[9] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436-444. DOI: 10.1038/nature14539.

[10] Aggarwal C. C. Neural Networks and Deep Learning: A Textbook. Berlin: Springer. 2023.

[11] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. Proceedings of the International Conference on Learning Representations (ICLR). San Diego: ICLR, 2015.

[12] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324. DOI: 10.1109/5.726791.

[13] Zhang W, Janson P, Aljundi R, et al. Overcoming generic knowledge loss with selective parameter update. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2024: 24046-24056.

[14] Cai D, Li H, Fu T, et al. On the transformations across reward model, parameter update, and in-context prompt. 2024. arXiv: 2406.16377v1. DOI: 10.48550/arXiv.2406.16377.

[15] Fan Y, Ma K, Zhang L, et al. VeriCNN: Integrity verification of large-scale CNN training process based on zk-SNARK. Expert Systems with Applications, 2024, 255 (Part B): 124531. DOI: 10.1016/j.eswa.2024.124531.

[16] Champaneria M, Panchal D, Bhalavat J, et al. Empirical evaluation of deep learning based models for time series datasets. Procedia Computer Science, 2023, 230: 864 - 873. DOI: 10.1016/j.procs.2023.12.046.

[17] Pan H, Zheng L. DisSAGD: A distributed parameter update scheme based on variance reduction. Sensors, 2021, 21(15): 5124. DOI: 10.3390/s21155124.

[18] Gao H, Yang X, Hu Y, et al. Adversarial sample attacks algorithm based on cycle-consistent generative networks. Applied Soft Computing, 2024, 162: 111778. DOI: 10.1016/j.asoc.2024.111778.

[19] Hashemi A S, Mozaffari S. CNN adversarial attack mitigation using perturbed samples training. Multimedia Tools and Applications, 2021, 80: 22077 - 22095. DOI: 10.1007/s11042-020-10379-6.

[20] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. Proceedings of the 2017 IEEE Symposium on Security and Privacy. Piscataway: IEEE, 2017: 39-57. DOI: 10.1109/SP.2017.49.

[21] Vidnerová P, Neruda R. Vulnerability of classifiers to evolutionary generated adversarial examples. Neural Networks, 2020, 127: 168-181. DOI: 10.1016/j.neunet.2020.04.015.

[22] Zeng Y, Pan M, Just H A, et al. Narcissus: A practical clean-label backdoor attack with limited information. Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Digital Library, 2023: 771 - 785. DOI: 10.1145/

3576915.3616617.

[23] Liu Q, Kang C, Zou Q, et al. Implementing a multitarget backdoor attack algorithm based on procedural noise texture features. IEEE Access, 2024, 12: 69539-69550. DOI: 10.1109/ACCESS.2024.3401848.

[24] Le Roux Q, Bourbao E, Teglia Y, et al. A comprehensive survey on backdoor attacks and their defenses in face recognition systems. IEEE Access, 2024, 12: 47433 - 47468. DOI: 10. 1109/ACCESS. 2024. 3382584.

[25] Yuan Y, Kong R, Xie S, et al. PatchBackdoor: Backdoor attack against deep neural networks without model modification. Proceedings of the 31st ACM International Conference on Multimedia. Ottawa: ACM, 2023: 9134-9142. DOI: 10.1145/3581783.3612032.

[26] Gu T, Dolan-Gavitt B, Garg S. BadNets: Identifying vulnerabilities in the machine learning model supply chain. 2017. arXiv:1708.06733.

[27] Jagielski M, Oprea A, Biggio B, et al. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. 2018 IEEE Symposium on Security and Privacy. San Francisco: IEEE, 2018: 19-35.

[28] Xue M, Yuan C, Wu H, et al. Machine learning security: Threats, countermeasures, and evaluations. IEEE Access, 2020, 8: 74720-74742. DOI: 10.1109/ACCESS.2020.2987435.

[29] Monica, Agrawal P. A survey on hyperparameter optimization of machine learning models. Proceedings of the 2nd International Conference on Disruptive Technologies (ICDT). Greater Noida: IEEE, 2024: 11-15. DOI: 10.1109/ICDT61202.2024.10489732.

[30] Oprea A, Singhal A, Vassilev A. Poisoning attacks against machine learning: Can machine learning be trustworthy? Computer, 2022, 55(11): 94-99. DOI: 10.1109/MC.2022.3190787.

[31] Muñoz-González L, Biggio B, Demontis A, et al. Towards poisoning of deep learning algorithms with back-gradient optimization. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. New York: ACM Digital Library, 2017: 27-38. DOI: 10.1145/3128572.3140451.

[32] Yin D, Chen Y, Kannan R, et al. Byzantine-robust distributed learning: Towards optimal statistical rates. Proceedings of the 35th International Conference on Machine Learning, Stockholm: PMLR, 2018, 80: 5650-5659.

[33] Li Q, Guo Y, Zuo W, et al. Improving adversarial transferability via intermediate-level perturbation decay. Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023). New Orleans: ACM, 2023: Article number 1425:32900-32912. DOI: 10.5555/3666122.3667547.

[34] Guo Y, Li Q, Zuo W, et al. An intermediate-level attack framework on the basis of linear regression. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(3): 2726-2735. DOI: 10.1109/TPAMI.2022.3188044.

[35] Debes H B, Giannetsos T. ZEKRO: zero-knowledge proof of integrity conformance. Proceedings of the 17th International Conference on Availability, Reliability and Security. Vienna: ACM, 2022: Article number 35. DOI: 10.1145/3538969.3539004.

[36] Fan Y, Xu B, Zhang L, et al. Validating the integrity of convolutional neural network predictions based on zero-knowledge proof. Information Sciences, 2023, 625: 125-140. DOI: 10.1016/j.ins.2023.01.036.

[37] Groth J. On the size of pairing-based non-interactive arguments. Proceedings of the 35th Annual International Conference on Advances in Cryptology - EUROCRYPT 2016, Part II. Berlin: Springer, 2016, 9666: 305-326. DOI: 10.1007/978-3-662-49896-5_11.

[38] Gennaro R, Gentry C, Parno B, et al. Quadratic span programs and succinct NIZKs without PCPs. Advances in Cryptology-EUROCRYPT 2013. Berlin: Springer, 2013, 7881:626 - 645. DOI: 10.1007/978-3-642-38348-9_37.

[39] Fan Y, Ma K, Zhang L, et al. VeriCNN: Integrity verification of large-scale CNN training process based on zk-SNARK. Expert Systems with Applications, 2024, 255(Part B): 124531. DOI: 10. 1016/j. eswa. 2024. 124531.

[40] Sun H, Bai T, Li J, et al. zkDL: Efficient zero-knowledge proofs of deep learning training. IEEE Transactions on Information Forensics and Security, 2024, 20: 914-927. DOI: 10.1109/TIFS.2024.3520863.

[41] Abbaszadeh K, Pappas C, Papadopoulos D. zero-knowledge proofs of training for deep neural networks. Santa Barbara: IACR Crypto. 2024. ePrint Arch, 2024: 162 (2024).

[42] Wood A, Najarian K, Kahrobaei D. Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Computing Surveys, 2020, 53(4): Article umber 70. DOI: 10.1145/3394658.

[43] Gilad-Bachrach R, Dowlin N, Laine K, et al. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. Proceedings of the 33rd International Conference on Machine Learning. New York: ICML, 2016, 48:201-210.

[44] Phong L T, Aono Y, Hayashi T, et al. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security, 2018, 13(5): 1333-1345. DOI: 10.1109/TIFS.2017.2787987.

[45] Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP. Advances in Cryptology - CRYPTO. Berlin: Springer, 2012, 7417: 868-886. DOI: 10.1007/978-3-642-32009-5_50.

[46] Wagh S, Gupta D, Chandran N. SecureNN: Efficient and private neural network training. Cryptology ePrint Archive, 2018, Paper 2018/442.

[47] Mohassel P, Zhang Y. SecureML: A system for scalable privacy-preserving machine learning. Proceedings of the IEEE Symposium on Security and Privacy. San Jose: IEEE, 2017: 19-38. DOI: 10.1109/SP.2017.12.

[48] Zhang X, Zhang Y, Zhang Y. VeriTrain: Validating MLaaS training efforts via anomaly detection. IEEE Transactions on Dependable and Secure Computing, 2024, 21(3): 1032-1049. DOI: 10.1109/TDSC.2023.3266427.

[49] Gao Y, Bao D, Zhang Z, et al. Backdoor attacks and countermeasures on deep learning: A comprehensive review. 2020. arXiv:2007.10760.

[50] Zhang F, Zhou L, Zhang Y, et al. Trusted execution environment: State-of-the-art and future directions. Journal of Computer Research and Development, 2024, 61(1): 243-260.

[51] Miao L, Yang W, Hu R, et al. Against backdoor attacks in federated learning with differential privacy. Proceedings of ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE, 2022: 2999-3003. DOI: 10.1109/ICASSP43922.2022.9747653.

[52] Chang J, Li H, Pearce H, et al. What's pulling the strings? Evaluating integrity and attribution in AI training and inference through concept shift. Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security. Taipei: ACM, 2025: 3117-3131. DOI: 10.1145/3719027.3744868.

[53] Roth K, Kilcher Y, Hofmann T. The odds are odd: A statistical test for detecting adversarial examples. Proceedings of the 36th International Conference on Machine Learning. Long Beach: ICML, 2019: 5498-5507.

[54] Meng D, Chen H. MagNet: A two-pronged defense against adversarial examples. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas: ACM, 2017: 135-147. DOI: 10.1145/3133956.3134057.

[55] Ma X, Li B, Wang Y, et al. Characterizing adversarial sub-spaces using local intrinsic dimensionality. Proceedings of the International Conference on Learning Representations. Vancouver: ICLR, 2018,

[56] Moosavi-Dezfooli S, Fawzi A, Frossard P, et al. DeepFool: A simple and accurate method to fool deep neural networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2016: 2574-2582.