Auditing MLaaS Inference Service Quality without Ground Truth via Mutual Information

Jiang Zhu, *Student Member, IEEE* Qingqing Ye, *Member, IEEE,* Haibo Hu, *Senior Member, IEEE,* and Li Bai

***Abstract*—Machine Learning as a Service (MLaaS) paradigm offers an appealing solution for clients that have limited computational resources. It allows entities to train models with collected dataset and powerful cloud resources, and to deploy these models for inference. However, MLaaS currently faces significant challenges in ensuring trustworthy inference and service quality. The clients cannot verify that the inference results returned by service provider (SP) are the model's actual inference results. Moreover, even if clients manage to ensure that the results are obtained through model inference, they are unable to determine the model's service quality without ground truth. To address these concerns, we introduce an innovative framework to audit inference quality and integrity in MLaaS through a novel deep neural network (DNN) inspection method. In specific, our approach represents the inherent behavior of the model by collecting its intermediate layer outputs and quantifying the mutual information (MI) values derived from them. By benchmarking the model during the training process, the SP can record the characteristics of the correct model and its corresponding service quality. After receiving the auditing request, the auditor can evaluate the quality of the service by estimating its accuracy via mutual information. Moreover, it can confirm the integrity of the returned results by inspecting the intermediate layer output. In addition, we thoroughly analyze our scheme for various potential adaptive attacks. Through empirical studies, we verify the correctness, effectiveness, and robustness of our scheme for trustworthy MLaaS inference service.**

***Index Terms*—Machine learning as a service, audit, mutual information, machine learning**

I. Introduction

The landscape of modern computational technology has been profoundly reshaped by the advent and evolution of Machine Learning (ML) [1]–[3] technology and its integration into cloud-based services, commonly known as Machine Learning as a Service (MLaaS) [4]–[6]. Leveraging cloud resources for both training and inference, MLaaS platforms, such as Amazon SageMaker [7] and Microsoft Azure [8], enable seamless deployment and utilization of DNN models, allowing users to perform complex tasks with ease and efficiency [9], [10].

While MLaaS offers easy-to-use inference service, it also introduces significant challenges related to service quality. Due to the black-box nature of MLaaS services, users are unable to verify the computations performed on the servers. Since the inference results returned by service providers (SP) are often succinct to avoid revealing any additional unnecessary information, there exists a risk that SP might jeopardize or even bypass model inference entirely with human-supplied or random responses [11], [12]. Moreover, the SP may deliberately use inferior models to conserve computational resources, as these models typically have fewer parameters and require less computational power to perform inference. In addition, even when the SP faithfully uses the correct model to provide services, it may falsely claim that the model has high performance under false advertising. Such false claim is often hard to detect, as users lack ground truth to verify the correctness of the results. In the worst case, a malicious SP may deliberately sabotage the inference process and return incorrect results. Unfortunately, users cannot verify the authenticity of the SP's service quality claims as the SP will not share trained model weights with clients for intellectual property (IP) and privacy protection reasons [13], [14].

To address these integrity issues, researchers focus on two approaches. The first approach involves proving the integrity of model inferences by designing zk-SNARKs [15], [16] for model inference [17]. zk-SNARKs are cryptographic proofs that allow one party to prove to another that they know a value x without revealing any information about the value itself. These schemes enable the model owner to demonstrate that the model correctly computes the prediction for a given data sample without disclosing any details about the model itself [18]–[20]. Another approach is private inference [21]. Private inference schemes employ complex cryptographic methods to safeguard the privacy of user inputs and model weights during model inference. The primary goal of these schemes is to ensure that neither party gains knowledge of the other's data. Additionally, because these schemes compute and output data using cryptographic methods, the results are more difficult to falsify, thus partially addressing our concerns about integrity. However, we argue that these solutions do not address all above issues. For instance, zk-SNARKs-based inference methods [22] often require separate proof and verification for each inference sample, which is computationally expensive [23], [24] and does not scale well. Although many existing solutions achieve excellent efficiency for single inferences, scaling to multiple tasks results in significant computational resource consumption and additional storage and communication burdens. In private inference solutions, cryptographic methods also introduce significant computing overhead. Moreover, these solutions can only verify that inference results originate from an agreed-upon model without assessing the

(Corresponding author: Haibo Hu.)

Jiang Zhu, Qingqing Ye, and Li Bai are with the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: polyu-jiang.zhu@connect.polyu.hk; qqing.ye@polyu.edu.hk; baili.bai@connect.polyu.hk).

Haibo Hu is with the Department of Electrical and Electronic Engineering and the Research Centre for Privacy and Security Technologies in Future Smart Systems, The Hong Kong Polytechnic University, Hong Kong (e-mail: haibo.hu@polyu.edu.hk).

model's service quality.

In this work, we demonstrate both theoretically and empirically that a regression relationship exists between the **mutual information** (MI) values of a model input and the outputs of intermediate layers, denoted as $I(X;T)$, and the model's accuracy. This relationship allows us to utilize MI as an effective indicator of model performance. To this end, we propose an inference inspection framework designed to safeguard inference quality. During the training process, we implement SP benchmark checkpoints which enable us to estimate model accuracy through a quantitative analysis of the MI values between the input and various intermediate layer outputs. Consequently, during the auditing phase, an auditor can estimate the performance indicators, specifically accuracy, by examining these MI values. A trusted auditor (TA) is a third party responsible for inspecting the SP's inference service. The TA delivers a trusted auditing service while addressing client limitations imposed by confidentiality or intellectual property constraint. Furthermore, to ensure the integrity of the intermediate outputs used in MI estimation, we incorporate the MI values $I(T_i;T_{i+1})$ to represent the mutual information between consecutive layers, as well as $I(T;Y)$, which reflects the MI between intermediate outputs and final results. This dual approach helps verify the integrity of the output chain across layers.

Our contributions can be summarized as follows:

- We demonstrate that there exists a regression relationship between MI values of input and intermediate output and the accuracy metric. We substantiate this claim through both theoretical analysis and empirical validation.
- Based on the aforementioned regression relationship, we propose a novel scheme for validating service quality and ensuring inference integrity for MLaaS. Our scheme achieves precise performance metric estimation and low computational overhead. Additionally, we design an integrity verification mechanism to prevent any tampering with the results.
- We perform a comprehensive security analysis on potential adaptive attacks to evaluate the effectiveness and robustness of the proposed scheme.
- Extensive experimental results demonstrate that the proposed method is able to precisely estimate service quality and resist potential adaptive attacks.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 overviews the problem formulation in our work. Section 4 describes preliminary. Section 5 introduces the foundational observation of our scheme and its theoretical and empirical analysis. Section 6 elaborates on the proposed scheme and security analysis. The experimental evaluations are shown in Sections 7. Finally, Section 8 concludes this paper.

II. Related Work

In this section, we review existing research on verifying inference integrity.

A. Verifiable inference based on zk-SNARKs

The most natural approach to ensure inference integrity is by introducing zero-knowledge proofs into the inference process. In our context, the prover can demonstrate that a model's prediction was computed correctly, without revealing the model's parameters. Researchers have shown considerable interest in and conducted extensive research on solutions based on it. Liu *et al.* [23] proposed zkCNN, a zero-knowledge proof scheme for CNNs. It uses a new sumcheck protocol for fast Fourier transforms and convolutions. The scheme efficiently supports large models, such as VGG16, generating proofs in 88.3 seconds, 1264 times faster than current methods, with a proof size of 341 kilobytes and a verifier time of 59.3 milliseconds. It can also verify the accuracy of the same CNN on multiple images. Feng *et al.* [25] proposed a compiler that optimizes the generation of neural network inference schemes called ZEN. ZEN addresses the issue of poor efficiency by two improvements: a novel neural network quantization approach that is compatible with R1CS and introduced fewer limitations, hence minimizing accuracy degradation. These optimizations result in a reduction of R1CS restrictions by average factor of 15.35. Lee *et al.* [26] proposed a new efficient verifiable convolutional neural network framework, namely vCNN, which significantly accelerates proving performance. They introduce a new relation representation for convolution equations, which reduces the proving complexity from $O(ln)$ to $O(l+n)$. Experimental results show that vCNN improves proving performance by 20-fold for a simple model and 18,000-fold for VGG16. In 2024, Fan *et al.* [24] proposed psvCNN to address the heavey computation cost problem. psvCNN uses convolutional kernels to partition the model and build zero-knowledge proof circuits for parallel CNN prediction. Moreover, it improves the Freivalds algorithm to speed up the verification process. Its zero-knowledge argument is simple and non-interactive. The experiments showed that it is 3765 times faster than the latest zk-SNARK-based non-interactive method vCNN [26] in terms of proving time.

The primary drawback of zk-SNARKs-based solutions is the necessity for independent proof and verification for each inference sample. When the number of samples that need inference increases, it often requires a lot of computing resources. Additionally, optimizing DNN inference tasks for GPUs is often overlooked, despite the frequent need to enhance current solutions for efficient performance on these platforms. Furthermore, as model architecture complexity increases, the time required for proof and verification also escalates, rendering these solutions impractical for models with substantial computational demands. Our solution, on the other hand, naturally performs the inference process on the GPU, thereby enhancing its efficiency. And our solution scales the computational complexity quadratically with an increase in sample numbers and linearly with an increase in model layers.

B. Private inference

Private inference is another promising approach for ensuring the integrity of model inference. Recent research has investigated different approaches to implementing private inference

through the use of cryptographic methods. In practice, the primary goal of private inference is to prevent the leakage of user inputs and model weights, rather than to ensure the integrity of the inference process. However, private inference can still enhance inference integrity to some extent. These solutions frequently incorporate various cryptographic techniques, causing model weights and outputs to take on different cryptographic forms. For instance, outputs produced using homomorphic encryption are typically in encrypted form, which makes forging legitimate model outputs challenging while maintaining formal consistency. Delphi is a secure prediction system proposed by Srinivasan *et al.* [27]. Delphi utilizes a hybrid cryptographic protocol to reduce communication and computation costs. Through secret sharing and homomorphic encryption, Delphi achieves linear operations. Moreover, it designed garbled circuit-based approaches to support secure non-linear activation functions. Rathee *et al.* [28] proposed CrypTFlow2, which achieves non-linear operations through secret sharing. CrypTFlow2 introduces new 2PC protocols for secure comparison and division, optimizing round and communication complexity for secure inference. It enables the first secure inference over large-scale DNNs, such as ResNet50 and DenseNet121, which are significantly larger than those in previous 2-party DNN inference work. Liu *et al.* [29] presented Sonic, a lightweight, secure inference service, which allows for the full outsourcing of secure inference, protecting both user data and model privacy. Utilizing lightweight cryptographic primitives, specifically secret sharing algorithm, Sonic secures layer functions throughout the service flow. Extensive evaluations show Sonic achieves significant bandwidth savings during online inference. To tackle the high latency issue resulting from non-linear operators, Cheng *et al.* [30] proposed PAPI, a practical and adaptive private inference framework. PAPI uses an accuracy-adaptive neural architecture search technique to create DNN models with fewer ReLUs that still meet the accuracy goals. It also introduces secure online and offline protocols for ReLU and polynomial activations, relying on lightweight secret sharing techniques.

However, private inference methods fail to fully solve the problem in our scenario. Firstly, some methods [27], [29] suffer from accuracy loss because they must fit approximate activation functions. Secondly, these methods do not guarantee the model's performance on the unseen data. Therefore, it might potentially lead to false advertising. Lastly, the complexity of the cryptographic systems employed in these methods results in low efficiency.

III. Problem Statement

A. Motivation

MLaaS offer users an interface for black-box inference, where the internal workings of the model remain opaque [31], [32]. This lack of transparency calls for measures to ensure the reliability and correctness of inferences produced by these black-box models.

In a common MLaaS workflow, a model owner uploads her training data an MLaaS server who trains a machine learning model and notify her with its initial performance metrics. The trained model is then deployed for inference, where new data is submitted through API calls to generate predictions. An MLaaS also offers a variety of pre-trained models tailored for specific applications. These models have been meticulously trained on wide-ranging and comprehensive datasets, ensuring they can handle diverse tasks. Users can access these models through similar API calls.

However, this setup introduces significant risks. To gain market share and reduce costs, a dishonest server might undermine the service integrity by falsifying the model's performance metrics, providing insufficient training, or adopting a simpler model architecture. As a result, users could be misled about the actual model quality and accept its prediction results as credible. This would adversely affect downstream applications that rely on these results and sabotage the applications' reputation. For example, smart surveillance cameras based on AI image analysis [33] can generate false alarms, and navigation software may provide wrong route suggestion due to inaccurate traffic prediction by AI models [34].

In the absence of effective audit mechanisms, identifying the responsible party of such problems becomes extremely difficult. As such, auditing the quality of model inference results is urgently needed.

B. Threat Model

There are three types of entities in our proposed MLaaS model, namely *SP*, *client*, and *TA*. The SP is responsible for training the model in accordance with the specification and yielding model weights θ. Before deploying inference services, the SP must collaborate with the TA to guarantee the integrity of training process. To achieve this, the TA benchmarks the model first and then audits the inference service. In the benchmarking phase, the TA collects additional independent and identically distributed (i.i.d.) data to measure various metrics, including inference accuracy and MI values. Then it performs regression analysis on this data, recording the regression coefficients and corresponding MI values. In the auditing phase, the client accesses the inference service, compresses and submits the input data and inference results to the TA as an auditing request. The TA then inspects the request with reference to the benchmarking data and provides insights into the overall quality and reliability of the service. In our setting, TA is a trusted third-party regulator that employed to safeguard the inference service. Therefore, the TA here is assumed to be trustworthy. This assumption defines the role of the TA, which is tasked with conducting audits after the inference service. The SP has strong motivation to collaborate with TA in order to claim its service trustworthy. Furthermore, the TA is designed to operate with restricted access and does not interact with raw data during the auditing process by design.

We assume the SP may act maliciously, potentially sabotaging the inference process. Our solution addresses two critical aspects to uphold the accuracy and reliability of the service. First, it estimates the quality of the SP's inference service in the **absence of ground truth** for the input data. Second, it verifies the integrity of the results returned by SP. The

verification procedure minimally alters the original inference by collecting only the model's intermediate layer outputs. In Section VII-A1, we will provide comprehensive security analysis of various adaptive attacks.

IV. Preliminary

A. Mutual Information Estimation in DNN

Measuring the MI between input and the activations has emerged in DNN, especially for the dependency between input data and their learned representations [35]. Such analysis can contribute to deep learning analysis [36]–[38] and unsupervised feature learning [39]–[42]. However, estimating mutual information in DNNs poses a formidable challenge, primarily due to the curse of dimensionality that the computational complexity of mutual information increases exponentially with the dimensionality. To address this challenge, we employ a principled framework based on the work of Butakov *et al.* [43]. Grounded in the manifold hypothesis [44], the authors demonstrate that lossy compression can be used for accurate MI estimation in high-dimensional settings. The authors support their claim by providing a crucial theoretical result:

$$I(X;Y) \leq I(E(X);Y) \leq I(X;Y) + h(Z) - h(Z \mid X,Y) \tag{1}$$

Here, $E(X)$ is the compressed version of the noisy data X, and Z represents the information lost during compression. It establishes a chain of inequalities that bounds the MI of the compressed representation. More importantly, this indicates that the estimated MI is well-bounded and unarbitrary. In the ideal case where the lost information Z is independent of the variables, the MI is perfectly preserved. Following this conclusion, we calculate MI between the compressed representations of random variables using conventional estimators as follows.

Consider a sequence of samples, $\{(x_k, y_k)\}_{k=1}^{N}$, drawn from the joint distribution of random vectors X and Y and their corresponding lossy compression $E_X(X)$ and $E_Y(Y)$. The objective is to estimate the mutual information, denoted as $I(X;Y)$, using these samples. Typically, in the context of MI calculation, the following equivalences hold:

$$I(X;Y) = h(X) - h(X \mid Y) = h(Y) - h(Y \mid X) \tag{2}$$

$$I(X;Y) = h(X) + h(y) - h(X,Y) \tag{3}$$

Thanks to the invariance property of MI under nonsingular mappings between smooth manifolds, we can assert the following:

$$I(X;Y) = I(E_X(X); E_Y(Y)) \tag{4}$$

Therefore, the mutual information can be estimated as follows:

$$\begin{aligned}\hat{I}(X;Y) &= \hat{I}\left(E_X(X); E_Y(Y)\right) \\ &= \hat{h}\left(E_X(X)\right) + \hat{h}\left(E_Y(Y)\right) - \hat{h}\left(E_X(X), E_Y(Y)\right)\end{aligned} \tag{5}$$

A detailed proof of the above estimation method is beyond the scope of this paper and can be found in [43]. This method is exceptionally efficient, surpassing other approaches in handling the computational complexities associated with high-dimensional variable spaces. In this work, we employ this estimation technique to quantify the MI in DNNs. As for compression, we adopt principal component analysis (PCA) due to its simplicity. The Kraskov-Stögbauer-Grassberger (KSG) estimator [45] is employed for MI estimation due to its proven accuracy in moderate-dimensional settings [46].

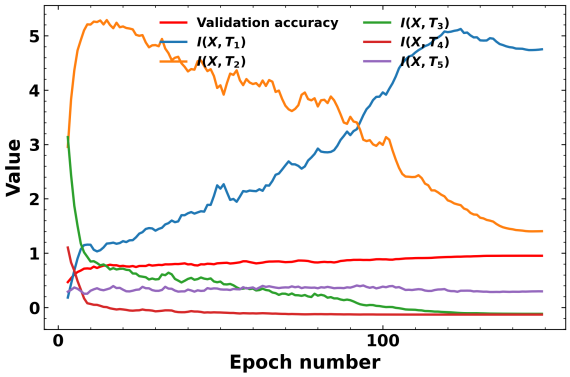


Fig. 1: The trend of validation accuracy and MI values. The validation accuracy continuously improves during training, and the MI values corresponding to each layer also evolve.

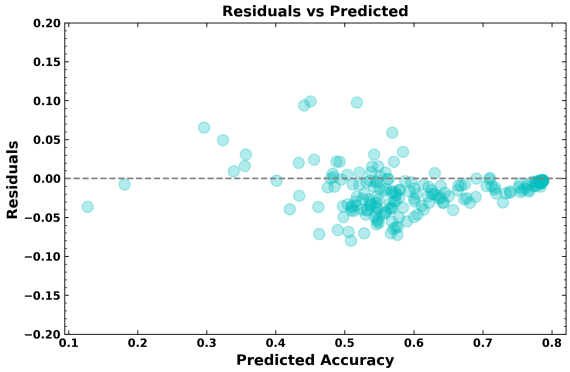


Fig. 2: The residuals of regression model on the accuracy of inference dataset.

V. Observation

In this section, we present the foundational observation that lays the foundation of our proposed methodology. This is supported by theoretical and empirical analysis on the MI metrics of model checkpoints throughout the training process.

The Information Bottleneck (IB) theory [36], [47] offers a novel framework to analyze DNNs through MI. This framework provides insights into how a model processes and transforms information across its layers, particularly within the context of deep learning. During training, the model must decide which aspects of the input data to compress and which relevant features to retain. Motivated by this, we employ MI values to observe and evaluate model behavior and performance. The MI metric $I(X,T)$ quantifies how much a model extracts relevant data from an input variable X. While proper training improves validation loss and accuracy metrics, the model's MI values associated with different intermediate

layer's output T also undergo significant evolution. This evolution reflects adjustments in the information processing efficiency of each layer, which is crucial for optimizing model performance across various stages of training. Figure 1 depicts the progression of validation accuracy alongside the MI values $I(X,T)$ for various layers, plotted as functions of epoch number. The result aligns with our analysis above. The information extraction capabilities across different layers correlate with improvements in model accuracy over time. In other words, the MI values $I(X,T)$ of a model serve as **indicators of its training status**, reflecting the quality of its services. As such, we can effectively gauge and ensure the integrity and performance of MLaaS using these MI values.

To develop such indicators, we establish a relationship between MI values and testing accuracy levels. In what follows, we theoretically prove that the model's accuracy can be approximated by a deterministic function of MI.

Theorem 1. *Assuming the training dataset is $S_T = ((x_i, y_i))_{i=1}^{M} \sim \mathcal{P}^{\otimes n}$ of M samples and inference dataset is $S_I = ((x_i, y_i))_{i=1}^{L} \sim \mathcal{P}^{\otimes n}$, in which $\mathcal{P} = \mathcal{X} \times \mathcal{Y}$. For any $\delta > 0$, with probability as least $1-\delta$, the accuracy of the model on S_I is bounded by a deterministic function of MI and δ, $f(I(X;T), \delta)$*

$$\mathcal{A}_h(S_I) \leq f(I(X;T), \delta) \tag{6}$$

Proof. Let $\mathcal{H}$ be a finite set of hypotheses. For every $h \in \mathcal{H}$, we can easily conclude that the accuracy is bounded between 0 and 1. For the training dataset, let $\mathcal{A}_h(S_T) = \frac{1}{M}\sum_{i=1}^{M}[\![h(x_i) = y_i)]\!]$ be the empirical accuracy. Hoeffding's inequality shows that for every $h \in \mathcal{H}$,

$$\mathbb{P}\left[\left|\mathcal{A}_h(S_T) - \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_h(X,Y)\right]\right| \geq \epsilon\right] \leq 2e^{-2\epsilon^2 M} \tag{7}$$

Suppose the hypothesis chosen by the training algorithm is g, after applying union bound, we have

$$\mathbb{P}\left[\left|\mathcal{A}_g\left(S_T\right) - \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_g\left(X,Y\right)\right]\right| \geq \epsilon\right] \leq 2|\mathcal{H}|e^{-2\epsilon^2 M}, \tag{8}$$

where $|\mathcal{H}|$ is the total hypotheses in the hypotheses space. Note that this is a Probably Approximately Correct (PAC) bound [48]. We let $2|\mathcal{H}|e^{-2\epsilon^2 m} \leq \delta$. This is equivalent to say that the event happens with a tiny probability δ. Solving for ϵ, we have

$$\epsilon^2 \geq \frac{\log(2|\mathcal{H}|/\delta)}{2M} \tag{9}$$

Therefore, with probability $1-\delta$, we have

$$\left|\mathcal{A}_g\left(S_T\right) - \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_g\left(X,Y\right)\right]\right|^2 \leq \frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{2M}. \tag{10}$$

We can obtain that

$$\begin{aligned}\mathcal{A}_g\left(S_T\right) - \sqrt{\frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{2M}} &\leq \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_g\left(X,Y\right)\right] \\ &\leq \mathcal{A}_g\left(S_T\right) + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{2M}}.\end{aligned} \tag{11}$$

Similarly, we can also derive

$$\mathbb{P}\left[\left|\mathcal{A}_h(S_I) - \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_h(X,Y)\right]\right| \geq \epsilon\right] \leq 2e^{-2\epsilon^2 L} \tag{12}$$

for inference dataset. However, since the training algorithm has never seen the inference dataset, i.e. g is independent of inference dataset S_I, we do not need to apply to union bound to account for the candidate hypotheses. As a result, the final expression can be simplified as

$$\begin{aligned}\mathcal{A}_g\left(S_I\right) - \sqrt{\frac{\log\frac{2}{\delta}}{2L}} &\leq \mathbb{E}_{(X,Y)\sim\mathcal{P}}\left[\mathcal{A}_g\left(X,Y\right)\right] \\ &\leq \mathcal{A}_g\left(S_I\right) + \sqrt{\frac{\log\frac{2}{\delta}}{2L}}\end{aligned} \tag{13}$$

Taking Eq. (11) and the left-hand side of Eq. (13), we have

$$\mathcal{A}_g\left(S_I\right) \leq \mathcal{A}_g\left(S_T\right) + \sqrt{\frac{\log\frac{2}{\delta}}{2L}} + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{2M}}. \tag{14}$$

To resolve the issue of infinite $|\mathcal{H}|$, we substitute it with a tractable term [48] by assuming data is a random vector that follows an ergodic Markov random field structure. So we can define a typical set of realizations from X as a set that satisfies the Asymptotic Equipartition Property (AEP) [49]. Given X, $2^{H(X)}$ is the asymptotic size of $|X|$. Let T be mapping of X, $2^{H(X|T)}$ is the number of typical realizations of X that are mapped to a mapping T. Consequently, the size of the typical set of T is bounded by $2^{H(X)}/2^{H(X|T)} = 2^{I(X;T)}$, and therefore $|\mathcal{H}| \leq 2^{|X|}$. Finally, we can derive a tighter bound using the typical set size of T, which is associated with $I(X;T)$.

$$\mathcal{A}_g\left(S_I\right) \leq \mathcal{A}_g\left(S_T\right) + \sqrt{\frac{\log\frac{2}{\delta}}{2L}} + \sqrt{\frac{2^{I(X;T)} + \log\frac{2}{\delta}}{2M}}. \tag{15}$$

Thus, we complete the proof. ■

Theorem 2. *Following the same setting as Theorem 1, for any $\tilde{\delta} > 0$, with probability at least $1-\tilde{\delta}$, the model's empirical accuracy $\mathcal{A}_g(S_I)$ on dataset $S_I = ((x_i, y_i))_{i=1}^{m} \sim \mathcal{P}^{\otimes n}$ can be expressed as a function of $I(X;T)$, $\mathcal{A}_g(S_I) = \tilde{f}\left(I\left(X;T\right)\right) + \tilde{\delta}$.*

Proof. We begin with the assumption that for any $\delta > 0$,

$$\left|\mathcal{A}_g\left(S_I\right) - f\left(I\left(X;T\right), \delta\right)\right| \leq \tilde{\epsilon} \tag{16}$$

For simplicity, we set δ to a fixed value and write $f\left(I\left(X;T\right), \delta\right)$ as $f\left(I\left(X;T\right)\right)$ instead. We define

$$\tilde{f}\left(I\left(X;T\right)\right) = \mathbb{E}\left[\mathcal{A}_g\left(S_I\right)|I\left(X;T\right)\right] \tag{17}$$

This is the best predictor of $\mathcal{A}_g\left(S_I\right)$ given $I\left(X;T\right)$. According to Jensen's inequality, we have

$$\begin{aligned}&\left|\tilde{f}\left(I\left(X;T\right)\right) - f\left(I\left(X;T\right)\right)\right| \\ &= \left|\mathbb{E}\left[\mathcal{A}_g\left(S_I\right)|I\left(X;T\right)\right] - f\left(I\left(X;T\right)\right)\right| \\ &\leq \mathbb{E}\left[\left|\mathcal{A}_g\left(S_I\right) - f\left(I\left(X;T\right)\right)\right||I\left(X;T\right)\right] \\ &\leq \tilde{\epsilon}.\end{aligned} \tag{18}$$

TABLE I: Notation Description

Notations	Descriptions
θ	Service model weights
α	Accuracy
$\mathbf{W}$	Regression model weight
$\mathbf{b}$	Regression model bias
T	Intermediate output
$\mathcal{L}$	Loss function
$\mathcal{D}_{train}$	Training dataset
$\mathcal{D}_{benchmark}$	Benchmarking dataset
X	Input
Y	Label

We define the error term as $\tilde{\delta} = \mathcal{A}_g(S_I) - g(I(X;T))$. Therefore, the error can be bounded as

$$\begin{aligned}\left|\tilde{\delta}\right| &= \left|\mathcal{A}_h(S_I) - \tilde{f}(I(X;T))\right| \\ &\leq \left|\mathcal{A}_h(S_I) - f(I(X;T))\right| + \left|f(I(X;T)) - \tilde{f}(I(X;T))\right| \\ &\leq \tilde{\epsilon} + \tilde{\epsilon} = 2\tilde{\epsilon}.\end{aligned} \tag{19}$$

Thus, we can derive

$$\mathcal{A}_g(S_I) = \tilde{f}(I(X;T)) + \tilde{\delta}, \tag{20}$$

where $\left|\tilde{\delta}\right| \leq 2\tilde{\epsilon}$. ■

Furthermore, according to Chebyshev's inequality, we can bound the probability of large deviations in $\tilde{\delta}$

$$P\left(\left|\tilde{\delta}\right| \geq k\right) \leq \frac{Var(\tilde{\delta})}{k^2} \leq \frac{4\tilde{\epsilon}^2}{k^2}. \tag{21}$$

Theorem 2 demonstrates that the model's empirical accuracy can be approximated by a deterministic function of MI using regression. We further empirically verify this with a benchmark dataset $\mathcal{D}_{benchmark}$ and an inference dataset $\mathcal{D}_{inference}$ out of CIFAR100 [50]. We split the testset evenly into these two datasets. We utilize the benchmark dataset to evaluate the performance of model checkpoints, recording both the MI values and the accuracy. These metrics are then used to train a regression model. For each epoch, we also apply the corresponding checkpoint to the inference dataset to obtain MI values. The trained regression model predicts the accuracy on the inference dataset according to the MI values. The regression model demonstrates a mean squared error (MSE) of 0.001 and an R^2 score [51] of 0.9354 on the inference dataset. The accuracy predicted by the regression model is compared against the actual accuracy to calculate the residual values. The results of the residual values are present in Figure 2. As depicted in the figure, our findings demonstrate that regression, relying solely on the MI values, can estimate the model's accuracy on this dataset. Notably, as the training converges, the prediction error approaches zero. This figure further illustrates that the residual values for most checkpoints are centered at zero, indicating a high level of prediction accuracy at the point of model convergence. To conclude, we can employ regression to precisely define and quantify the relationship between MI values and the test accuracy of models.

VI. Methodology

In this section, we present our proposed scheme based on the key observations in Sec. V. As illustrated in Figure 3, our scheme incorporates two mechanisms to prevent malicious behavior by SPs. The first mechanism, accuracy estimation, assesses the quality of services when SPs act honestly and identify poor-quality services when using the correct model architecture and dataset. The second mechanism detect when SPs use a model architecture or dataset inconsistent with the declared specifications, or when they forge intermediate layer outputs. Our scheme is into two stages. In the first stage, the MLaaS server benchmarks the model checkpoints to the TA, who records the associated metrics and relevant regression parameters. In the second stage, the client sends inference requests to the MLaaS server and submits compressed input data and inference results to the TA for auditing. Based on the previously benchmarked results, the TA then evaluates the SP's service quality. Additionally, the TA verifies the integrity of the model's output layers by measuring their differences in MI values.

A. Benchmarking Model

Prior to delivering the service, the SP must evaluate the performance of their implemented models and provide the results to the TA as benchmarks for black-box inference service. To prevent the SP from forging the results or sabotaging this process, the dataset used for benchmarking is independently collected by the TA, who possesses the ground truth for this dataset. The benchmarking procedure is illustrated through two algorithms. Alg. 1 elaborates on the responsibilities assigned to the SP, whereas Alg. 2 shows the tasks at the TA side. As with other existing studies, the problem of MI with deterministically computed features is addressed by introducing small amount of Gaussian noise after each layer for analysis purposes [52]. The noise is controlled by parameter σ. Injecting Gaussian noise during training is a widely researched regularization technique proven to enhance model generalizability and robustness [53]–[55]. It is proved in [54] that it forces the network to learn smoother, simpler functions, which are more likely to capture the true underlying data distribution for the model. Furthermore, [55] formalizes this as a strategy to trade for an increase in test set accuracy. During this training phase, the checkpoint for each epoch needs to be recorded to prepare for the subsequent benchmarking process.

After completing its training, the SP must notify the TA and benchmark the model. Upon acquiring the benchmark dataset $\mathcal{D}_{benchmark}$, inferences are made for all checkpoints and the outputs from all intermediate layers and the final result are collected. Subsequently, PCA is employed to compress these outputs to form a compressed layer output chain $T_i = \{E_{L_{i0}}(T_{i0}), E_{L_{i1}}(T_{i1}), ..., E_{L_{ij}}(T_{ij})\}$, in which i is the index of the checkpoint and $E_{L_{ij}}$ is the PCA encoder for jth layer output. Here we set the PCA component numbers to $\sqrt{d}$, where d is the dimension of the input data. To improve computational efficiency, we retain all singular vectors produced during PCA and transform the data as needed for different

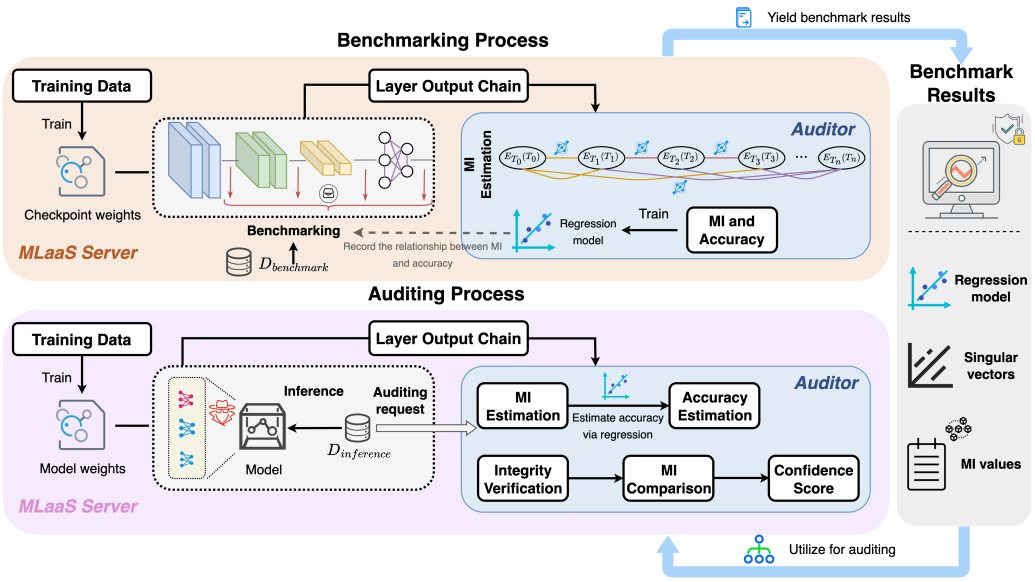


Fig. 3: The overview of the proposed scheme.

Algorithm 1 Benchmarking Procedure for MLaaS Server.

Input: Training dataset $\mathcal{D}_{train} = (X_{train}, Y_{train})$, Noise parameter σ

Output: Trained and benchmarked model checkpoints Θ, all layer output chains $\mathbf{T}$, and all inference output $\hat{Y}$

Alter the model architecture to inject Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ after each layer
while not converged **do**
  Train the model by minimizing the loss function $\theta_i = \arg\min_{\theta} \mathcal{L}(X_{train}, Y_{train}; \theta_{i-1})$
  Save all checkpoints during training $\Theta = (\theta_0, ..., \theta_i)$
end while
Receive auxiliary dataset for benchmarking $X_{benchmark}$ from the TA
for each checkpoint $\theta_i \in \Theta$ **do**
  for L_{ij}: selected intermediate layers from θ_i **do**
    Collect output of each layer $T_{ij} = L_{ij}(X_{benchmark})$
    Compress the intermediate output via encoder $E_{L_{ij}}(T_{ij})$
  end for
  For each checkpoint θ_i, collect final inference output $\hat{\mathbf{Y}} = \{\hat{Y}_0, \hat{Y}_1, ..., \hat{Y}_i\}$
  For each checkpoint θ_i, get a compressed layer output chain $T_i = \{E_{L_{i0}}(T_{i0}), E_{L_{i1}}(T_{i1}), ..., E_{L_{ij}}(T_{ij})\}$
end for
Collect all layer output chains $\mathbf{T} = \{T_0, T_1, ..., T_i\}$
return $\mathbf{T}, \Theta, \hat{\mathbf{Y}}$

Algorithm 2 Benchmarking Procedure for TA.

Input: Auxiliary benchmark dataset $\mathcal{D}_{benchmark} = (X_{benchmark}, Y_{benchmark})$

Output: Coefficients for regression model of accuracy and MI $\mathcal{W}, \mathcal{B}$

When receives a request for model benchmarking, TA sends its auxiliary data for inference $X_{benchmark}$ to SP
TA receives layer output chains $\mathbf{T}$ and inference result $\hat{Y}$ from SP
for each layer output chain $T_i \in \mathbf{T}$ and the corresponding inference result $\hat{Y}_i \in \hat{Y}$ **do**
  Calculate accuracy $\alpha_i = acc(\hat{Y}_i, Y_{benchmark})$
  for each element in layer output chain $E_{L_{ij}}(T_{ij}) \in T_i$ **do**
    Compute the MI values for the compressed input data to the compressed intermediate outputs $I_{ij} = I(E_x(X_{benchmark}), E_{L_{ij}}(T_{ij}))$
  end for
  $I_i = \{I^x_{i0}, I^x_{i1}, ...I^x_{ij}\}$
end for
Collect layer output chains for all checkpoints $\mathbf{T} = \{T_0, T_1, ..., T_i\}$.
The accuracy of all checkpoints is saved to $\mathbf{A} = \{\alpha_0, \alpha_1, ..., \alpha_i\}$
Fit the MI values and corresponding accuracy to a regression model so that $\hat{\mathbf{A}} = \mathbf{W}_2 \cdot f(\mathbf{W}_1\mathbf{I} + \mathbf{b_1}) + \mathbf{b_2}$
The parameters of the regression model are $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2\}, \mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$
return $\mathcal{W}, \mathcal{B}$

dimensions. All layer output chains $\mathbf{T} = \{T_0, T_1, ..., T_i\}$ and the inference result $\hat{Y}$ are then returned to TA.

After receiving the inference results, the TA must estimate the MI for the layer output chains. Before the estimation, the input data, which is $\mathcal{D}_{benchmark}$, must also be compressed using PCA with the same component number. In our case, the KSG estimator [45] is utilized for MI estimation. At this stage, TA only needs to estimate the MI between the input and the intermediate layer output. Since TA has the correct benchmark data labels, it is also required to calculate the accuracy α_i of the inference results returned at each checkpoint. After calculating the corresponding MI values $\mathbf{I}$ and accuracy $\mathbf{A}$ for all checkpoint outputs, a dataset comprising all MI values and corresponding accuracy can be assembled. The TA then utilizes this data, denoted as $\mathbf{I}$ and $\mathbf{A}$, to train a simple regression model such that

$$\hat{\mathbf{A}} = \mathbf{W}_2 \cdot f(\mathbf{W}_1\mathbf{I} + \mathbf{b_1}) + \mathbf{b_2}, \tag{22}$$

where the accuracy $\mathbf{A}$ is approximated by a function of MI values $\mathbf{I}$. Here, f represents the activation function. Regression model here is employed to establish a quantitative relationship between MI dynamics and accuracy. Finally, the TA will store all MI values, accuracy, and parameters from the regression model to complete the benchmarking process. It is worth

noting that in Eq. (22) we present the mathematical expression using a generic regression model. For linear regression, we can simply set f to the identity function $y = x$, because a linear regression model is sufficient to effectively captures the MI-accuracy relationship in simple models. For more complex models, a sophisticated MLP regression model is required.

In addition, during the benchmarking process, we require the SP to return the singular vectors generated when the input data is compressed by PCA, because during the service quality inspection by the TA, the user's data also need to be compressed via PCA. Since the amount of inference data provided by a single user is small, having the user perform local compression may result in PCA projections that poorly capture the data's variance. To address this, we instruct the SP to return the singular vectors computed during the compression of the training data. The TA stores these vectors locally and provides them to the client upon request. Therefore, we omit the detailed verification process in this discussion.

B. Inference and Quality Inspection

Algorithm 3 Inference and Quality Inspection Procedure.

Input: Inference data $X_{inference}$, Coefficients of the regression model $\mathcal{W}, \mathcal{B}$
Output: Approximated accuracy $\hat{\alpha}$.

1: Send the inference data $X_{inference}$ to the SP
2: Receive the inference result $\hat{Y}$.
3: Send the service selection and the compressed input to the TA. TA then retrieves the layer output chain $T = \{E_{L_0}(T_0), E_{L_1}(T_1), ..., E_{L_j}(T_j)\}$ from the SP.
4: **for** each element in layer output chain $E_{L_j}(T_j) \in T$ **do**
5: Compute the MI values for the compressed input data to the compressed intermediate outputs $I_j = I(E_x(X_{inference}), E_{L_j}(T_j))$
6: **end for**
7: $I = \{I_0, I_1, ..., I_j\}$
8: Calculate the approximated accuracy via Eq. (22)
9: **return** $\hat{\alpha}$

When users access the inference service, they submit their data $X_{inference}$ to the SP as usual. The SP then returns the final results $\hat{Y}$ and saves the compressed output chain from each layer $T = \{E_{L_0}(T_0), E_{L_1}(T_1), ..., E_{L_j}(T_j)\}$ as archives. Likewise, the users compress their input data using the singular vector from TA. It can delegate the auditing task to TA by uploading the compressed input data to it. Upon receiving enough samples from the specific inference service, the TA employs the same MI estimator to calculate the MI values $I = \{I_0, I_1, ..., I_j\}$, and then finds the regression model from the benchmark records according to client's selection. To retrieve parameters for integrity verification, the TA first identifies the historical checkpoint accuracy $\alpha' \in \mathbf{A}$ that is closest to the received approximated accuracy $\hat{\alpha}$. Using the model parameters associated with this specific checkpoint, the TA computes the baseline MI I'_{xl}, representing the dependency between the input and intermediate layer outputs, and I'_{ll}, representing the information flow between consecutive intermediate layers. With these parameters, it can then calculate the approximated accuracy of the results returned by the SP from the MI values via Eq. (22). The inference and quality verification procedure for clients is illustrated in Alg. 3.

C. Integrity Verification

In addition to assessing the accuracy of the inference service, it is also essential to validate the integrity of the intermediate layer output chain given by the SP. This phase is crucial to prevent the SP from manipulation the intermediate layer outputs in order to fake the final results.

After inference quality inspection, TA needs to the identifies the checkpoint with the closest matching accuracy to the estimated accuracy α'. It retrieves the corresponding layer output chain and estimate MI for the input data and output of the intermediate layer $I'_{xl} = \{I^{0\prime}_{xl}, I^{1\prime}_{xl}, ..., I^{j\prime}_{xl}\}$, the neighboring intermediate layers $I'_{ll} = \{I^{0\prime}_{ll}, I^{1\prime}_{ll}, ..., I^{j-1\prime}_{ll}\}$. The integrity of the layer output chain can be assessed by comparing the MI values with the records. The TA can therefore calculate the MI errors through a distance function. A common choice is Euclidean distance. We tolerate some degree of noise level, δ_e between benchmarked values and measured ones. When the distance d is less than or equal to δ_e, the MI values are considered valid to match the corresponding. In regression models, the relationship between input variables and output variables can be many-to-one. Consequently, different combinations of input variables can yield the same output, potentially enabling a malicious SP to manipulate outcomes. This verification procedure ensures that the MI value aligns with that of the closest benchmark record. The integrity verification process assesses the reliability of the inference by comparing expected behavior against observed results. The TA first retrieves the baseline MI values (I'_{xl}, I'_{ll}) corresponding to the closest historical accuracy α'. These are compared against the actual MI values (I_{xl}, I_{ll}) computed from the current local layer outputs. The final confidence score c is derived by quantifying the deviation between the baseline and actual information flows using Eq. 23.

$$C(\mathbf{I}, \mathbf{I}') = \begin{cases} 1, & \text{if } d(\mathbf{I}, \mathbf{I}') \leq \epsilon_e \\ 1 - \frac{d(\mathbf{I}, \mathbf{I}') - \epsilon_e}{\delta - \epsilon_e}, & \text{if } \epsilon_e < d(\mathbf{I}, \mathbf{I}') \leq \delta \\ 0, & \text{if } d(\mathbf{I}, \mathbf{I}') > \delta \end{cases} \quad (23)$$

$\mathbf{I}$ is the concatenation of all calculated MI vectors. The overall confidence level in the legitimacy of the generated MI values can be represented by Eq. (23). We use Euclidean distance as function d. A confidence closer to 1 indicates a higher probability that the MI value is real. As a result, the client can ensure the quality of the SP-provided service while also verifying the integrity of the returned layer output chain. The client can accept the result when the confidence score exceeds a predefined acceptable threshold.

D. Importance of Accuracy Estimation

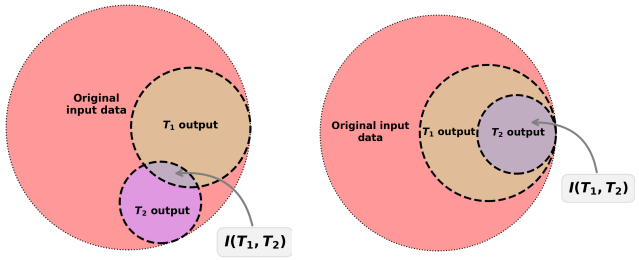
Our solution comprises two components: accuracy estimation and integrity verification. One might wonder why accuracy estimation is needed when integrity verification, which compares the MI value of the returned output chain to an

expected MI value, can already tell if the result is correct. TA can achieve this by comparing the calculated MI values with the MI values obtained at the end of model training. However, accuracy estimation is essential because relying solely on integrity verification is unreliable. The model's performance on user data may differ from its performance on benchmark data. Rather than relying on the final MI values from the training epoch, we need to recalibrate the corresponding MI value as the model's accuracy fluctuates. As illustrated in Figure 1, during the later stages of training, the model's accuracy tends to stabilize, showing only minor improvements. However, the MI value continues to change at a relatively steady rate. Consequently, even a slight change in accuracy can result in a significant difference in the MI value. Because of such fluctuation, when there is a small gap between the model's accuracy on $\mathcal{D}_{benchmark}$ and $\mathcal{D}_{inference}$, the accuracy gap could have a significant impact on the MI. Furthermore, since a large number of checkpoints are retained during the training process, one of the most straightforward ways for the SP to degrade service quality is by using partially trained checkpoints to provide services. Therefore, detecting this form of sabotage becomes even more critical. Thus, accuracy estimation is the most straightforward way to prevent such sabotage.

E. Security Analysis

We conduct a thorough analysis of the security of our approach to examine potential adaptive attacks. According to our threat model in Sec. III-B, we assume that SP launches these attacks primarily to save computing costs and intentionally sabotage inference results. The objective is to manipulate the returned data to closely resemble the correct data without actually providing the correct inference results. Also, our auditing scheme requires the TA to access intermediate output from the SP's model. We acknowledge that this creates a potential privacy risk [56], even with a non-malicious auditor, as data could be exposed through model inversion attacks [57], [58]. However, this risk can be easily mitigated by integrating existing security measures. These include running the audit within a secure computing environment, such as trusted execution environment (TEE), and applying differential privacy techniques [59], [60]. While designing these integrated protocols is beyond the scope of our current work, our auditing method is fully compatible with such safeguards.

Input masking attack. An input masking attack occurs when an SP attempts to forge inference results actively. This can happen when SP does not actually perform the inference and returns the forged results in order to save computational resources. We assume that the SP achieves this by adding noise to input data to increase MI. In our threat model, the SP must correctly train the model to pass the benchmarking process; otherwise, it reveals its true ability and fails to attract users. Consequently, the SP is aware of the correct model's MI metrics. This knowledge enables the SP to directly add noise to the compressed vector of the input data and return it as intermediate output, thereby increasing the MI value. By adjusting the magnitude of the noise, the SP can control the MI value between the forged vector and the input data, theoretically allowing it to achieve desired MI value. As a result, an attacker can manipulate the MI values, thereby impacting the estimated accuracy of the model.



(a) Forged Chain (b) Normal Inference

Fig. 4: Information Residual

Our scheme's integrity verification function effectively mitigates this type of attack. To comprehend this, it is crucial to analyze the similarities and differences between the MI obtained through normal inference and the forged one. To illustrate this comparison, we provide visualizations of the MI in both scenarios. Figure 4b and Figure 4a illustrated the information residual in both process. Each layer progressively reduces the information from the original input data as it processes the data. Assuming that the attacker successfully forges the output, the remaining information from the input data in T_2 is equivalent in both scenarios. However, as shown in Figure 4b, in the normal inference process, since the output of each preceding layer serves as the input for the subsequent layer, the next layer's output retains a significant amount of information from the previous layer. Conversely, when the attacker forges the data, the information in T_2 is derived directly from the input data, resulting in less information from the preceding layer T_1 being retained in T_2. The retained information from T_1 in T_2 can be measured by MI $I(T_1; T_2)$. In our scheme's integrity verification method, we compare the MI between adjacent layers of the received intermediate layer output chain with that of the correct model. This comparison ensures the integrity of the received intermediate layer output chain and prevents forging.

Model trained with different data. The SP selects a model with an identical structure but trains it on data meant for other purposes. We assume that SP has a correctly trained model and has passed the benchmark using it. And it is intentionally degrading the quality of service using a model trained with different data. For the widespread use of MLaaS platforms to train a wide range of tasks, it is likely that the SP has previously trained a different task using the same DNN architecture. The SP may use a model that has already been trained on other datasets to provide inference services as Eq. (24).

$$Y_{out} = \{f(x_i, \theta') | x_i \in X_{inference}\} \tag{24}$$

It hopes that the model can gain benefits from knowledge obtained from training on other data, thereby circumventing our verification mechanism. However, reusing models across

different tasks may lead to vulnerabilities, as the model weights may not be optimally for the new task. Therefore such behavior could lead to results that do not align with the intended use and are either irrelevant or incorrect. Nevertheless, the forged intermediate layer output chain is unlikely to bypass our verification scheme. This is a result of the inherent biases present in models that have been trained on various datasets. For instance, models may adopt distinct focuses and methods for feature extraction depending on the dataset. These differences lead to variations in how the models process information during inference, thereby altering the MI the input and the intermediate layer. Consequently, despite the efforts to increase MI, the values will differ due to these dataset-specific biases.

Model with a simpler structure. The SP chooses a model with a less complex design but trained on the right dataset. In this case, we also assume that the SP is intentionally degrading the quality of service. Utilizing a simpler model structure could decrease the model's accuracy and reliability. However, despite training on the same dataset, changes in the model's architecture result in variations in each layer's behavior. As a result, the MI values become difficult for attackers to replicate accurately in order to match the ones output by the correct model architecture. Therefore, it is also challenging for an attacker to manipulate the output chain of the intermediate layer using this behavior.

VII. Experimental Evaluation

In this section, we detail the implementation of the proposed scheme and present experimental results. Additionally, we examine the scheme's robustness by evaluating its performance against several malicious behaviors.

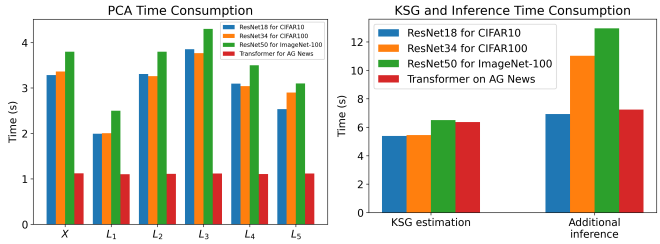
A. Experiment Setup

All experiments are run in a high performance server with the configuration of Ubuntu 22.04, Intel Platinum 8352V 2.10GHz CPU, RTX 4090, and 512GB RAM. All models their corresponding training logic are implemented using the JAX library [61]. In order to demonstrate generalization ability of our method to diverse domains, we perform experiments on two different tasks. For image classification task, we train our models on popular dataset CIFAR10, CIFAR100 [50], and ImageNet-100 [62]. To study our method across different datasets and models, we pair each dataset with a representative ResNet architecture of increasing depth and capacity. This allows us to cover a range of datasets and model complexity while keeping the experimental cost controllable. For CIFAR10, ResNet18 [63] is the adopted model architecture. And for CIFAR100, ResNet34 [63] is adopted. ImageNet-100 is a subset that randomly sampled from ImageNet-1K dataset [64] and resized to 160 pixels. We train ResNet-50 on this dataset. We evaluate our scheme on this large-scale datasets with with larger model to demonstrate the applicability to real-world MLaaS workflow. We selected the first layer of ResNet and each subsequent ResNet block as intermediate layers for collecting outputs. For both cases, we use a three-layer MLP as the regression model with 50 hidden layers. As for NLP task, we employ a transformer text classifier on AG News [65]. The model is stacked of six standard Transformer encoder layers with a classification head in the end. Each encoder layer consists of 8 attention heads with a 1024 dimensional feedforward network. We also perform experiment on graph data. A small amount of Gaussian noise is introduced into the model before each layer, with a standard deviation (σ) set to 0.001. We use the training data set for training and split the testset evenly into two subsets: a benchmark dataset and an inference dataset. The benchmark dataset is utilized by the TA to provide measurement benchmark services for the respective model. Meanwhile, the inference dataset is processed as data requested by users. For selecting intermediate layers, we collect the activations after each group of ResNet basic blocks and the first activation at the beginning in the image classification task. Due to the relatively low complexity of the NLP classification tasks, we collect the outputs from all intermediate layers for subsequent analysis. The ϵ_e is set to 0.3 and δ to 2.0 for confidence calculation.

1) Computational Overhead: We evaluate the computational overhead by comparing our methodology against native execution, which is limited to performing only the necessary computations. Compared to the typical MLaaS inference workflow, our approach introduces computational overhead primarily in two phases: benchmarking and verification. In the first phase, PCA, MI estimation introduce the main computational overhead. Since the regression model we use is simple and the training data is small, which typically consisting of a feature vector of length 5, the time required to train the model is negligible and therefore excluded from consideration. The additional inference performed in the first phase also needs to be taken into consideration. We used 5000 samples as input data. For MI estimation, the results presented here represent the time required to execute the algorithm once. The time consumption for PCA compression is presented in Figure 5a and MI estimation using the KSG algorithm is presented in Figure 5b.

It is worth noting that the number of epochs required for training can vary significantly across different models and datasets, necessitating adjustments to the training duration accordingly. Consequently, the computational overhead reported here pertains solely to a single checkpoint. We observed that PCA compression and MI estimation contribute a lot to the computational overhead. To reduce consumption, we did not calculate all MI values in Alg. 1. Instead, we computed only the necessary MI values. For those required for subsequent integrity verification, we deferred their calculation and performed it on demand. The computational time for PCA is influenced by three factors: the number of intermediate layers selected, the size of the intermediate outputs, and the number of principal components retained after compression. The time required for MI estimation depends on the size of the evaluated data and the parameters of the estimation algorithm. In our experiments, we use the KSG algorithm with $k = 5$ and $base = 2$ for MI estimation.

Given that the SP must perform inference on user data when a client initiates an inference request, even without any additional verification process, the inference time is not



(a) PCA (b) MI estimation and inference

Fig. 5: Major time consumption for additional operations

considered part of the additional computational overhead. To compress the input data, the client only needs to perform a matrix multiplication, as it already possesses the singular vectors provided by the TA, resulting in a negligible time consumption. Similarly, since the regression model we use is of low complexity, the time the TA spends to estimate the accuracy can be considered negligible. Therefore, the primary time-consuming components of quality inspection and integrity verification is MI estimation. Since the data dimensions remain the same, we can refer to Figure 5b for the MI computation time during the phases. In our proposed solution, the majority of the additional computational overhead occurs during the benchmarking process. There is minimal overhead in accuracy estimation and integrity verification. This makes our solution less computationally expensive than the zk-SNARKs approaches during online execution. For example, ZEN [25] requires approximately 20,000 seconds to set up for the LeNet-Face-large-ORL model. In addition, it requires approximately 20,000 seconds to generate a proof for a single inference request. In contrast, our method imposes no additional client-side computational overhead beyond a single matrix multiplication operation. The TA only needs to collect the corresponding intermediate layer outputs and perform PCA compression along with MI estimation. Furthermore, since inference requests are commonly processed in batches, the average per-sample overhead introduced by our method is therefore much lower than that of zk-SNARK-based approaches.

B. Empirical Studies on Accuracy Estimation

In this subsection, we empirically evaluate the effectiveness of accuracy estimation in our scheme. To evaluate the performance, we simulate scenarios involving malicious SP delivering incorrect inference services using undertrained models. In the experiments, we deployed each collected checkpoint to provide inference services and examined its accuracy. And our proposed scheme estimates the accuracy of these services. As shown in Figure 6, we compare the true accuracy with the accuracy estimated using MI. The accuracy of the fully trained model is also marked on the figure as baseline. In our experiment, we used stochastic gradient descent (SGD) as the optimizer to train the model. During training, the weights exhibit short-term stochastic fluctuations [66], resulting in fluctuations in the raw accuracy data. To mitigate these fluctuations, we applied a simple moving average (SMA) in Figure 6, which smooths the curve and reduces fluctuation, allowing the underlying trend to be observed more clearly. We also apply this technique to the estimated accuracy to keep consistency. It can be observed from the figure that as the number of training epochs increases, the service accuracy approaches that of the fully trained model. Furthermore, our proposed solution can estimate the accuracy with low error with the actual accuracy. Overall, the R^2 score of the regression is 0.923 for Figure 6a, 0.949 for Figure 6b, 0.989 for Figure 6c and 0.994 for Figure 6d. The correct estimation demonstrates the effectiveness of our scheme.

C. Empirical Studies on Adaptive Attacks

In this subsection, we evaluate the effectiveness of our proposed scheme by simulating a range of potentially attacks that SPs might engage in. We simulate these attacks and test whether our proposed scheme can successfully detect them. We also present the results for the correct inference service case, where no attack is present, and the confidence scores are 1 for all settings.

Input masking attack. SP attempts to forge inference results without actually performing it. The SP bypasses the required inference computations and add noise to the input data to fabricate the compressed vectors. SP does this to increase the MI of the input and intermediate layer output. In our simulation of this attack, we adopt a simple random masking algorithm as the masking strategy. Specifically, we randomly add Gaussian noise to the input data and forge the intermediate output through manipulating the noise volume. In this scenario, since the attacker can freely manipulate $I(X;T)$, the accuracy estimated by the TA is no longer a reliable metric. Therefore, the primary role of the TA is to identify an incorrect intermediate output chain through integrity verification. Tab. II and Tab. III present the MI values under this scenario. We can observe that though the $I(X;T)$ under such attack is consistent with the benchmarked values, while the $I(T_i, T_{i+1})$ is not. This difference will cause the confidence scores obtained by Eq. (23) to decrease. As a result, the confidence scores yielded under this attack in all settings are 0.

Model trained with different data. For ResNet18 and ResNet34, we switch the datasets to train two models in both scenario. For ResNet50, we train it on upscaled version of CIFAR100. As for transformer, we adopt Sogou News [67], which is a dataset from a different source. As a result, the inference data is inconsistent with the training data. In this case, since we did not replace the final fully connected (FC) layer of the model, the actual accuracy remains quite low. However, training on different datasets allows the model to extract important information, even though the information may be irrelevant to the downstream task. As shown in the tables, the MI values in this case are relatively close to the true values. Therefore, it is understandable that the regression model could make an incorrect prediction in this scenario. This indicates that the SP is actively deceiving the client successfully. In such cases, integrity verification is necessary to challenge these results. The confidence scores yielded in this scenario are also 0 for all settings.

ResNet18 on CIFAR10	$I(X,T_1)$	$I(X,T_2)$	$I(X,T_3)$	$I(X,T_4)$	$I(X,T_5)$	$I(T_1,T_2)$	$I(T_2,T_3)$	$I(T_3,T_4)$	$I(T_4,T_5)$
Input masking attack	4.74	1.42	−0.12	−0.13	0.29	1.54	−0.08	0.06	0.02
Model trained with different data	5.30	2.90	−0.01	−0.13	0.35	2.04	0.37	1.83	0.36
Model with a simpler structure	4.64	2.01	−0.00	−0.13	0.43	2.01	1.20	2.38	1.50
No attack	4.74	1.42	−0.12	−0.13	0.28	1.61	0.56	0.86	2.66
Benchmark	4.74	1.42	−0.12	−0.13	0.29	1.58	0.55	0.88	2.66

TABLE II: Comparison of MI values under different scenarios of ResNet18 on CIFAR10

ResNet34 on CIFAR100	$I(X,T_1)$	$I(X,T_2)$	$I(X,T_3)$	$I(X,T_4)$	$I(X,T_5)$	$I(T_1,T_2)$	$I(T_2,T_3)$	$I(T_3,T_4)$	$I(T_4,T_5)$
Input masking attack	5.25	3.05	0.03	−0.03	−0.03	2.93	0.04	0.15	0.13
Model trained with different data	4.73	1.85	−0.01	−0.03	−0.03	1.98	0.58	0.33	9.83
Model with a simpler structure	4.23	2.44	0.54	0.84	1.35	2.41	3.14	1.41	1.50
No attack	5.41	2.38	0.05	−0.03	0.67	2.57	0.70	2.15	1.15
Benchmark	5.42	2.33	0.03	−0.03	0.67	2.54	0.67	2.18	1.14

TABLE III: Comparison of MI values under different scenarios of ResNet34 on CIFAR100

ResNet50 on ImageNet-100	$I(X,T_1)$	$I(X,T_2)$	$I(X,T_3)$	$I(X,T_4)$	$I(X,T_5)$	$I(T_1,T_2)$	$I(T_2,T_3)$	$I(T_3,T_4)$	$I(T_4,T_5)$
Input masking attack	4.93	2.63	0.04	−0.20	0.50	0.92	0.33	0.01	−0.02
Model trained with different data	5.21	1.85	0.00	−0.23	0.28	3.41	2.66	1.44	0.15
Model with a simpler structure	4.73	2.84	−0.03	0.43	1.02	1.85	3.56	1.14	1.31
No attack	4.94	2.64	0.05	−0.3	0.51	1.80	1.50	0.50	0.96
Benchmark	4.93	2.63	0.05	−0.24	0.51	1.80	1.50	0.50	0.96

TABLE IV: Comparison of MI values under different scenarios of ResNet50 on ImageNet-100

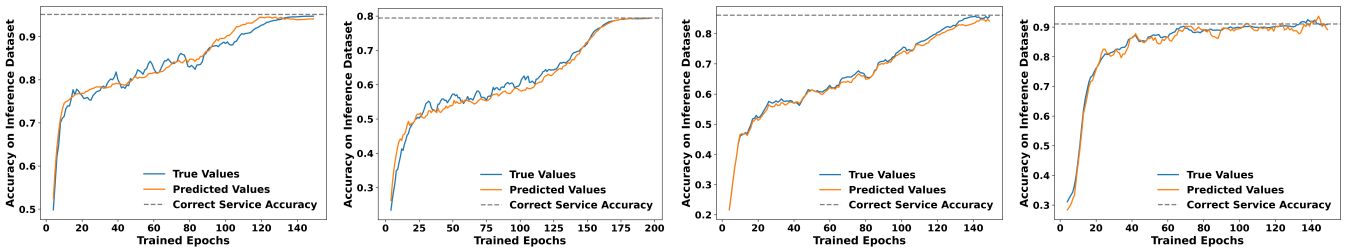
Transformer on AG News	$I(X,T_1)$	$I(X,T_2)$	$I(X,T_3)$	$I(X,T_4)$	$I(X,T_5)$	$I(T_1,T_2)$	$I(T_2,T_3)$	$I(T_3,T_4)$	$I(T_4,T_5)$
Input masking attack	10.87	7.23	5.32	2.98	1.81	1.10	0.37	0.36	0.55
Model trained with different data	−0.01	−0.01	0.20	−0.20	0.00	0.14	−0.14	0.20	0.23
Model with a simpler structure	8.39	6.88	4.51	2.01	0.67	5.08	5.63	6.20	7.37
No attack	10.87	7.23	5.32	2.98	1.80	4.61	5.13	5.52	7.38
Benchmark	10.87	7.23	5.32	2.99	1.81	4.61	5.13	5.52	7.38

TABLE V: Comparison of MI values under different scenarios of transformer on AG News

Model with a simpler structure. We simulated a scenario where the SP is expected to deliver a fully trained model with correct architecture. However, the SP trains and offers inference services using only with simpler architecture to conserve computing resources. In this case, we modify the model architectures with fewer filters and basic blocks in each group. Specifically, we reduced the number of filters in ResNet18 to 16 and limited each ResNet block to a single layer. Similarly, we reduced the number of filters to 16 and set the number of layers per block to two for both ResNet34 and ResNet50. For transformer, we reduce the number of multi-head attention heads from 8 to 2. The models are trained with correct dataset and same hyper parameters. Due to the somewhat relatively accurate training of the models, the regression model fails to predict correct accuracy when using MI values as input. This indicates that, although the model has degraded the quality of service, it has circumvented our accuracy estimation mechanism. The confidence scores calculated by integrity verification scheme are 0, which disputes the incorrect outcomes.

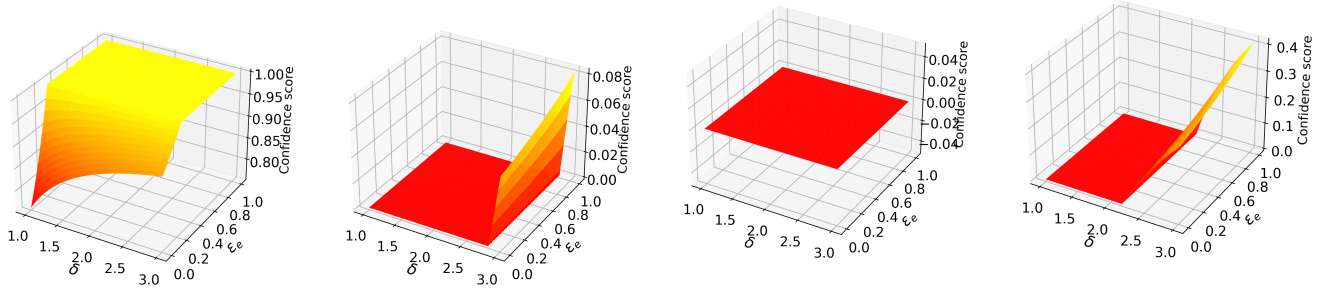
D. Parameters Selection for Integrity Verification

Our attack detection method here is primarily based on Eq. 23. The confidence calculation in our scheme relies on two pre-set thresholds, ϵ_e and δ. The choice of these thresholds influences the judgment of correct inferences and the identification of adaptive attacks. We evaluate the effects of different threshold selection in Eq. (23) on the integrity verification process in our scheme. The parameter ϵ_e controls the tolerance for noise in normal inference service, while δ defines the upper threshold for identifying abnormal one. In order to understand the influence of these parameters on our scheme, we vary them in the neighboring range. We present the changes in confidence scores across different scenarios as parameters vary. The results show that adjusting these two parameters may impact the confidence score, but this effect is typically observed only when the parameters are set to extreme values. Within a reasonable range, the choice of parameters is unlikely to significantly affect the final result. Even when an effect is observed, it is minor and does not substantially influence the outcome.



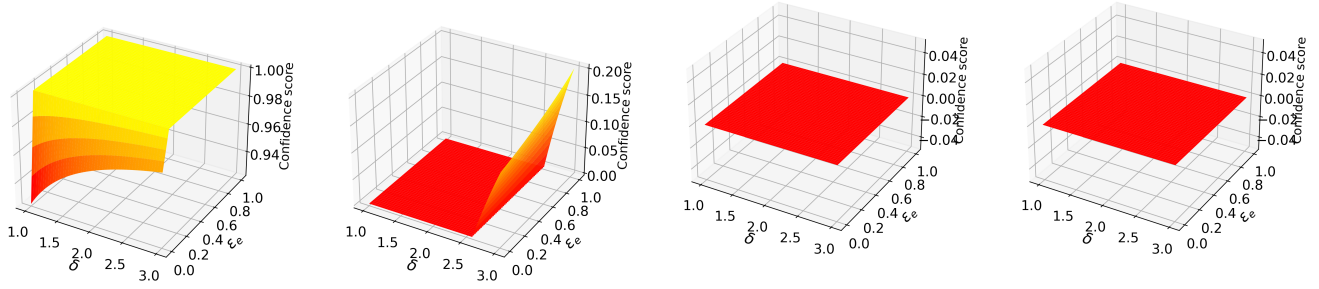
(a) ResNet18 on CIFAR10 (b) ResNet34 on CIFAR100 (c) ResNet50 on ImageNet-100 (d) Transformer on AG News

Fig. 6: Comparison of estimated accuracy and true accuracy on inference datasets with undertrained models



(a) No attack (b) Input masking attack (c) Model trained with different data (d) Model with a simpler architecture

Fig. 7: Parameters tuning on confidence calculation of ResNet18 on CIFAR10

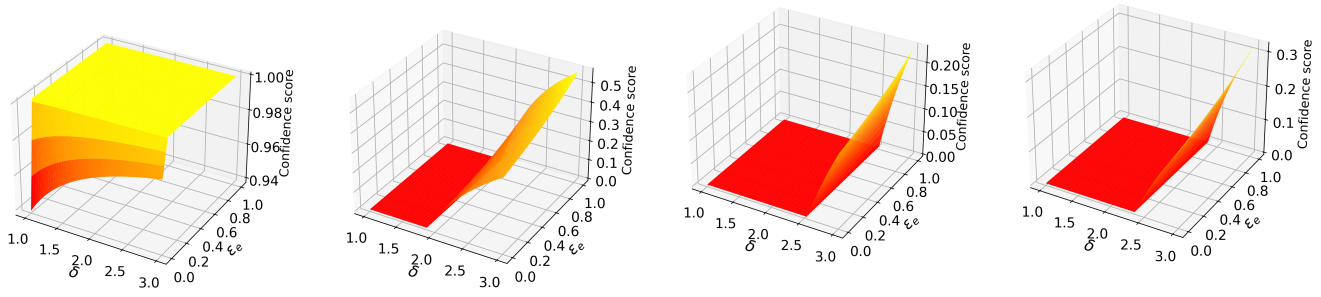


(a) No attack (b) Input masking attack (c) Model trained with different data (d) Model with a simpler architecture

Fig. 8: Parameters tuning on confidence calculation of ResNet34 on CIFAR100

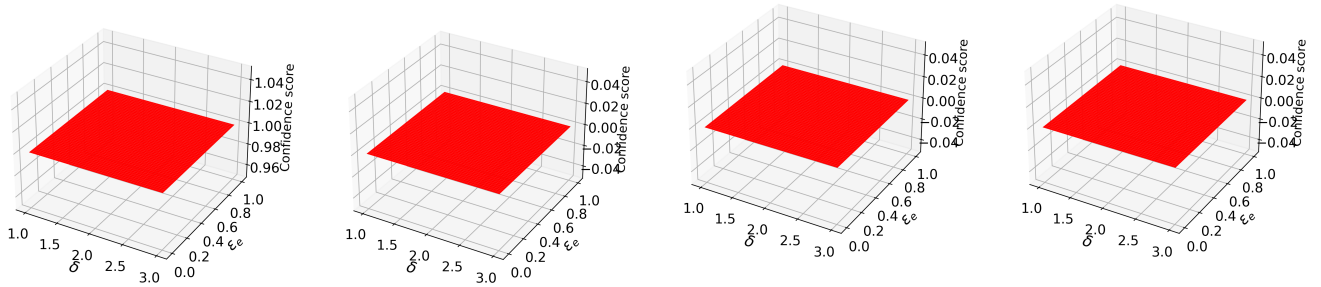
VIII. Conclusion

In this paper, we addressed the crucial challenges of verifying service quality and inference integrity in the MLaaS paradigm. Traditional MLaaS frameworks leave clients without the capability to assess the service quality due to the absence of ground truth. To overcome these limitations, we introduced a novel framework that audits both the quality and integrity of the inference using the proposed DNN inspection method. Our approach involves capturing the intermediate layer outputs of the model and quantifying the MI values to represent the model's inherent behavior. We conducted both theoretical and empirical analysis to demonstrate the effectiveness of our scheme. We also conducted a thorough analysis of our framework's resilience against various adaptive attacks. Extensive empirical studies performed demonstrated that our method is both effective and robust in auditing service. In this work, we audit the inference service using MI estimates from multiple inference cases to satisfy the AEP. However, the challenge of auditing with limited data remains unresolved. Moreover, although our objective here is to assess the core mechanisms of the scheme based on the assumption that the TA is fully trustworthy. At last, we assume a trustworthy auditor in this work. This simplification allows us to focus on our core technical challenge, which is detecting SP's misbehavior. We propose a foundational method for auditing MLaaS, not a comprehensive framework with privacy guarantees. Research focus on balancing privacy and auditability is still needed. We leave the resolution of these issues for future work.



(a) No attack (b) Input masking attack (c) Model trained with different data (d) Model with a simpler architecture

Fig. 9: Parameters tuning on confidence calculation of ResNet50 on ImageNet-100



(a) No attack (b) Input masking attack (c) Model trained with different data (d) Model with a simpler architecture

Fig. 10: Parameters tuning on confidence calculation of Transformer on AG News

IX. Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant No: 92270123, 62372122, and 62372130), and the Research Grants Council (Grant No: 15226221 and 15208923), Hong Kong SAR, China.

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.

[3] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, vol. 2018, 2018, pp. 1–15.

[4] Q. Zhang, T. Xiang, Y. Cai, Z. Zhao, N. Wang, and H. Wu, "Privacy-preserving machine learning as a service: Challenges and opportunities," *IEEE Network*, 2022.

[5] H. Zhang, Y. Li, Y. Huang, Y. Wen, J. Yin, and K. Guan, "Mlmodelci: An automatic cloud platform for efficient mlaas," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 4453–4456.

[6] B. Wu, X. Yuan, S. Wang, Q. Li, M. Xue, and S. Pan, "Securing graph neural networks in mlaas: A comprehensive realization of query-based integrity verification," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2534–2552.

[7] "Amazon sagemaker - machine learning solution," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/pm/sagemaker/

[8] "Cloud computing services — microsoft azure," azure.microsoft.com. [Online]. Available: https://azure.microsoft.com

[9] Y. Mao, C. Fu, S. Wang, S. Ji, X. Zhang, Z. Liu, J. Zhou, A. X. Liu, R. Beyah, and T. Wang, "Transfer attacks revisited: A large-scale empirical study in real computer vision settings," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1423–1439.

[10] Q. Zhang, T. Xiang, C. Xin, and H. Wu, "From individual computation to allied optimization: Remodeling privacy-preserving neural inference with function input tuning," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 101–101.

[11] N. Statt, "This ai startup claims to automate app making but actually just uses humans," The Verge, 08 2019. [Online]. Available: https://www.theverge.com/2019/8/14/20805676/engineer-ai-artificial-intelligence-startup-app-development-outsourcing-humans

[12] C. Grant, "Human-guided burrito bots raise questions about the future of robo-delivery," Thehustle.co, 06 2020. [Online]. Available: https://thehustle.co/kiwibots-autonomous-food-delivery

[13] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2018, pp. 399–414.

[14] A. S. Rakin, M. H. I. Chowdhuryy, F. Yao, and D. Fan, "Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories," in *2022 IEEE symposium on security and privacy (SP)*. IEEE, 2022, pp. 1157–1174.

[15] T. Chen, H. Lu, T. Kunpittaya, and A. Luo, "A review of zk-snarks," *arXiv preprint arXiv:2202.06877*, 2022.

[16] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 329–349.

[17] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang, "Mystique: Efficient conversions for {Zero-Knowledge} proofs with applications to machine learning," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 501–518.

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 315–334.

[19] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza, "Secure sampling of public parameters for succinct zero knowledge proofs," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 287–304.

[20] S. Steffen, B. Bichsel, R. Baumgartner, and M. Vechev, "Zeestar:

Private smart contracts by homomorphic encryption and zero-knowledge proofs," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 179–197.

[21] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," *Advances in neural information processing systems*, vol. 35, pp. 15 718–15 731, 2022.

[22] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Scaling up trustless dnn inference with zero-knowledge proofs," in *NeurIPS 2023 Workshop on Regulatable ML*.

[23] T. Liu, X. Xie, and Y. Zhang, "Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2968–2985.

[24] Y. Fan, B. Xu, L. Zhang, G. Tan, S. Yu, K.-C. Li, and A. Zomaya, "psvcnn: A zero-knowledge cnn prediction integrity verification strategy," *IEEE Transactions on Cloud Computing*, 2024.

[25] B. Feng, L. Qin, Z. Zhang, Y. Ding, and S. Chu, "Zen: An optimizing compiler for verifiable, zero-knowledge neural network inferences," *Cryptology ePrint Archive*, 2021.

[26] S. Lee, H. Ko, J. Kim, and H. Oh, "vcnn: Verifiable convolutional neural network based on zk-snarks," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[27] W. Z. Srinivasan, P. Akshayaram, and P. R. Ada, "Delphi: A cryptographic inference service for neural networks," in *Proc. 29th USENIX Secur. Symp*, 2019, pp. 2505–2522.

[28] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 325–342.

[29] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "Securely outsourcing neural network inference to the cloud with lightweight techniques," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 620–636, 2022.

[30] K. Cheng, N. Xi, X. Liu, X. Zhu, H. Gao, Z. Zhang, and Y. Shen, "Private inference for deep neural networks: a secure, adaptive, and efficient realization," *IEEE Transactions on Computers*, 2023.

[31] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "{MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 945–960.

[32] Y. Lin, T. Zhang, Y. Mao, and S. Zhong, "Crossnet: A low-latency mlaas framework for privacy-preserving neural network inference on resource-limited devices," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[33] T. Liu, E. Siegel, and D. Shen, "Deep learning and medical image analysis for covid-19 diagnosis and prediction," *Annual review of biomedical engineering*, vol. 24, no. 1, pp. 179–201, 2022.

[34] M. Shaygan, C. Meese, W. Li, X. G. Zhao, and M. Nejad, "Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities," *Transportation research part C: emerging technologies*, vol. 145, p. 103921, 2022.

[35] Z.-Q. Zhao, Y.-m. Cheung, H. Hu, and X. Wu, "Corrupted and occluded face recognition via cooperative sparse representation," *Pattern Recognition*, vol. 56, pp. 77–87, 2016.

[36] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[37] A. Achille and S. Soatto, "On the emergence of invariance and disentangling in deep representations," *arXiv preprint arXiv:1706.01350*, vol. 125, pp. 126–127, 2017.

[38] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1135–1144.

[39] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*.

[40] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *Advances in neural information processing systems*, vol. 32, 2019.

[41] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[42] K. Qian, Y. Zhang, S. Chang, M. Hasegawa-Johnson, and D. Cox, "Unsupervised speech decomposition via triple information bottleneck," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7836–7846.

[43] I. Butakov, A. Tolmachev, S. Malanchuk, A. Neopryatnaya, A. Frolov, and K. Andreev, "Information bottleneck analysis of deep neural networks via lossy compression," in *The Twelfth International Conference on Learning Representations*.

[44] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, vol. 29, no. 4, pp. 983–1049, 2016.

[45] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[46] P. Czyż, F. Grabowski, J. Vogt, N. Beerenwinkel, and A. Marx, "Beyond normal: On the evaluation of mutual information estimators," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[47] R. Shwartz-Ziv, A. Painsky, and N. Tishby, "Representation compression and generalization in deep neural networks," 2018.

[48] M. Mohri, "Foundations of machine learning," 2018.

[49] M. Thomas and A. T. Joy, *Elements of information theory*. Wiley-Interscience, 2006.

[50] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[51] N. Draper, *Applied regression analysis*. McGraw-Hill. Inc, 1998.

[52] K. Kawaguchi, Z. Deng, X. Ji, and J. Huang, "How does information bottleneck help deep learning?" in *International Conference on Machine Learning*. PMLR, 2023, pp. 16 049–16 096.

[53] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," *Advances in neural information processing systems*, vol. 30, 2017.

[54] A. Camuto, M. Willetts, U. Simsekli, S. J. Roberts, and C. C. Holmes, "Explicit regularisation in gaussian noise injections," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 603–16 614, 2020.

[55] O. Dhifallah and Y. Lu, "On the inherent regularization effects of noise injection during training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2665–2675.

[56] L. Bai, H. Hu, Q. Ye, J. Xu, J. Li, C. Fang, and J. Shi, "Rmr: A relative membership risk measure for machine learning models," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[57] Y. Xiao, H. Hu, Q. Ye, L. Tang, Z. Liang, and H. Zheng, "Unlocking high-fidelity learning: Towards neuron-grained model extraction," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[58] H. Li, L. Bai, Q. Ye, H. Hu, Y. Xiao, H. Zheng, and J. Xu, "A sample-level evaluation and generative framework for model inversion attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 17, 2025, pp. 18 287–18 295.

[59] J. Duan, Q. Ye, H. Hu, and X. Sun, "Analyzing and enhancing ldp perturbation mechanisms in federated learning," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[60] Q. Ye, L. Yu, K. Huang, X. Xiao, W. Liu, and H. Hu, "From randomized response to randomized index: Answering subset counting queries with local differential privacy," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 3877–3891.

[61] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[62] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *European conference on computer vision*. Springer, 2020, pp. 776–794.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[65] X. Li, "Ag news and imdb," 2024. [Online]. Available: https://dx.doi.org/10.21227/f9vv-5898

[66] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[67] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.