Zero-Knowledge Proof-based Verifiable Decentralized Machine Learning in Communication Network: A Comprehensive Survey

Zhibo Xing, Zijian Zhang*, *Senior Member, IEEE,* Ziang Zhang, Zhen Li, Meng Li*, *Senior Member, IEEE,* Jiamou Liu, Zongyang Zhang, *Member, IEEE,* Yi Zhao, *Member, IEEE,* Qi Sun, Liehuang Zhu, *Senior Member, IEEE,* Giovanni Russello.

***Abstract*—Over recent decades, machine learning has significantly advanced network communication, enabling improved decision-making, user behavior analysis, and fault detection. Simultaneously, the growth of communication networks has facilitated the efficient collection of large-scale training data. Traditional centralized machine learning, however, requires collecting data from users, raising significant concerns about privacy and security. Decentralized approaches, where participants exchange computation results instead of raw private data, mitigate these risks but introduce challenges related to trust and verifiability. A critical issue arises: *How can one ensure the integrity and validity of computation results shared by other participants?* Existing survey articles predominantly address security and privacy concerns in decentralized machine learning, whereas this survey uniquely highlights the emerging issue of *verifiability*. Recognizing the critical role of zero-knowledge proofs in ensuring verifiability, we present a comprehensive review of Zero-Knowledge Proof-based Verifiable Machine Learning (ZKP-VML). To clarify the research problem, we present a definition of ZKP-VML consisting of four algorithms and several key security properties. In addition, we provide an overview of the current research landscape by systematically organizing the research timeline and categorizing existing schemes based on their security properties. Furthermore, through an in-depth analysis of each existing scheme, we summarize their technical contributions and optimization strategies, aiming to uncover common design principles underlying ZKP-VML schemes. Building on the reviews and analysis presented, we identify current research challenges and suggest future research directions. To the best of our knowledge, this is the most comprehensive survey to date on verifiable decentralized machine learning and ZKP-VML.**

***Index Terms*—Verifiability, Decentralized Machine Learning, Zero-Knowledge Proof, Communication Network.**

Zhibo Xing is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, 100081, China, and the School of Computer Science, The University of Auckland, Auckland, 1010, New Zealand. E-mail: 3120215670@bit.edu.cn.

Zijian Zhang is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, 100081, China, and Southeast Institute of Information Technology, Beijing Institute of Technology, Fujian, 351100, China. E-mail: zhangzijian@bit.edu.cn.

Jiamou Liu and Giovanni Russello are with the School of Computer Science, The University of Auckland, Auckland, 1010, New Zealand. Email: {jiamou.liu, g.russello}@auckland.ac.nz.

Ziang Zhang, Zhen Li, Yi Zhao, Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, 100081, China. E-mail: {3220231794, zhen.li, zhaoyi, liehuangz}@bit.edu.cn.

Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education; School of Computer Science and Information Engineering, Hefei University of Technology; Intelligent Interconnected Systems Laboratory of Anhui Province (Hefei University of Technology), and University of Padua, Italy. Email: mengli@hfut.edu.cn.

Zongyang Zhang is with the School of Cyber Science and Technology, Beihang University, Beijing, 100191, China. E-mail: zongyangzhang@buaa.edu.cn.

Qi Sun is with the Department of Bioinformatics, Hangzhou Nuowei Information Technology Co.,Ltd, Zhejiang, 310053, China. E-mail: sunq0810@gmail.com.

(Corresponding Author: Zijian Zhang, Meng Li.)

I. Introduction

In recent years, artificial intelligence (AI) has seen widespread adoption in our daily life, receiving significant attention from both academia and industry. The complexity of problems that AI can tackle grows in tandem with advancements in machine learning (ML) methodologies. However, the development of ML also increases the communication and computation overheads, as larger ML models are required to be trained on larger datasets. On the one hand, the exponentially growing demand for computing power and data scale in machine learning tasks making it difficult for personal computers to accomplish machine learning tasks [1], [2]. On the other hand, concerns about data privacy increases with time, which damages people's enthusiasm in sharing their data and participating in machine learning. The demand for computing power and concerns on data privacy restrain the application of centralized machine learning (CML) while encourage the development of decentralized machine learning (DML). DML allows participants to accomplish the ML tasks with multiple rounds of local computation and global communication, instead of simply gathering needed data without privacy guarantee. A series of secure computation protocols and DML frameworks have been proposed to safeguard the data privacy. Federated learning (FL) [3] is a popular DML framework, which allows several participants to collaboratively train a ML model without sharing their own private datasets, thus achieving data privacy protection. These works are dedicated to perform ML tasks while protecting data privacy and minimizing computational and communication burdens. Although the DML efficiently solves the data privacy concerns, the verifiability issue is still highly neglected.

The concern over *verifiability* is rooted in DML framework, as we have no access to the private data and computation process of others. A key focus will be on how requesters can verify that the computation results provided by performers are accurate through faithful local computation. For example, in ML training tasks, a malicious performer may implant

This article has been accepted for publication in IEEE Communications Surveys & Tutorials. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/COMST.2025.3561657

backdoors into the model via poisoning attacks [4], [5], leading to the misclassification. In ML inference tasks, a malicious performer may also give a false prediction, misguiding the requester to make wrong decisions. These security issues and malicious attacks derive from the difficulty in verifying the correctness of the computation process and results.

Intuitively, the computation performer could send all input data to the requester for re-execution and verification. However, this approach is not practical in real-world scenarios. This verification method would severely damage the basic data privacy that DML pursue. Notably, sending all the input data and re-executing the ML computation would put an unbearable workload on communication and computation. Thus, the problem is, *how can we verify the results submitted by others without access to their private data while maintaining low communication and computational overhead.* There have been several solutions for tackling this problem, but the most promising one in terms of low computation and communication is zero-knowledge proof.

Zero-knowledge proof (ZKP) is a powerful cryptographic technique for addressing the verifiability issues in DML, especially when it comes to the privacy and low overhead. ZKPs allow one party to prove the correctness of a statement without revealing additional information to another party. Within DML, ZKP aligns well with the need of verifying the correctness and integrity of local computation results. By presenting the corresponding DML algorithm and the inputs and outputs as a statement that “The output is honestly computed with the given algorithm, given public inputs and some specific private inputs”, the performer can generate a proof, arguing for the correctness of the computational process and results, without revealing any additional information about the private inputs. This proof can be verified by other participants, and once the verification passes, they can trust the statement without accessing to additional details, such as private inputs to the algorithm.

Considering that verifiable machine learning (VML) is a relatively new research area, in order to provide a clearer understanding at how zero-knowledge proofs ensuring the verifiability in decentralized machine learning and why this is important and necessary, we describe the following applications of ZKP-VML in real-life scenarios as examples in Fig. 1.

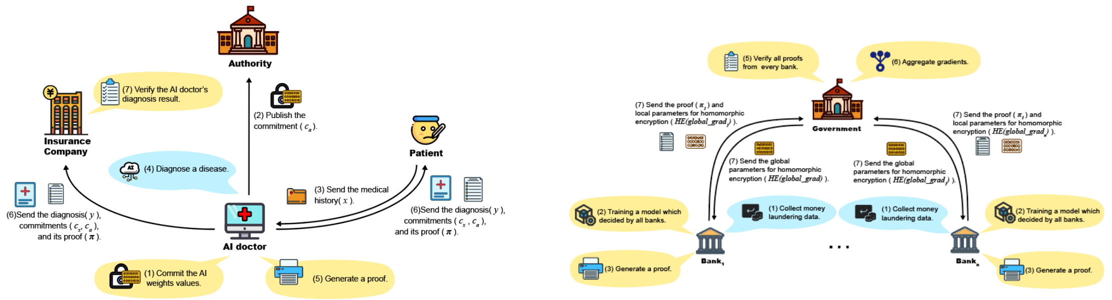
We illustrate outsourced inference in an AI-assisted diagnosis scenario. Machine learning enables hospitals to diagnose diseases more efficiently and accurately. However, hospitals must provide proofs of the inference process to ensure verifiability. Such proofs allow patients to verify the correctness of their diagnoses and enable insurance companies to process claims accordingly. Moreover, both the diagnostic model and patient data are sensitive—hospitals must protect their proprietary models, while patient privacy must be preserved. Since the proof is shared with both the patient and the insurer, it should reveal no information beyond the diagnosis result. Hospitals can generate ZKPs for the inference process, allowing verification without exposing the model or input data. Additionally, commitment schemes can be integrated with ZKPs to guarantee the integrity of the model and medical data used in inference. The whole process is shown in Fig. 1(a).

The second example concerns federated learning in training models for money laundering detection. By analyzing user information and transaction records, AI-driven machine learning models can identify potential financial crimes more efficiently. Governments can facilitate federated learning among multiple banks to collaboratively train a global model while preserving data privacy. Federated learning ensures that each bank’s data remains confidential and is not exposed to other banks or the government. To guarantee the integrity of local training, banks must provide proofs of correctness. ZKPs enable each bank to demonstrate the validity of its training process without revealing sensitive data. This allows the government to verify training correctness while maintaining privacy. The whole process is shown in Fig. 1(b).

A. Related Work

In related work, we mainly compare surveys focusing on machine learning verifiability, in particular achieved through zero-knowledge proofs. Considering that verifiability is a key component of security in machine learning, we further cover some surveys related to security and privacy as related work to provide a more comprehensive comparison. We categorize existing work into three progressive layers: machine learning security and privacy, machine learning verifiability, and ZKP-based machine learning verifiability. Table I analyzes and compares related work based on the above categorization.

1) Surveys on Secure Machine Learning: Several studies focus on the security issues of decentralized machine learning while also considering verifiability. For example, Ma et al. [6] explore the security challenges of outsourced deep learning, presenting a system model and security requirements. They provide a comprehensive analysis of techniques for secure outsourcing, including a detailed taxonomy of privacy-preserving strategies for training and inference. The work also offers comparative analyses of cryptographic methods such as homomorphic encryption (HE) and secret sharing (SS), while examining trade-offs between privacy, verifiability, efficiency, and non-interactivity. However, their focus is mainly on privacy, with limited attention to verifiability. The discussion on verifiability remains conceptual, lacking empirical evaluations or practical implementations. Additionally, the key cryptographic technique, ZKPs, is not addressed. Qin et al. [7] systematically examine cryptographic methods used in privacy-preserving machine learning (PPML). They classify existing PPML protocols based on cryptographic primitives, including secure multiparty computation (MPC), ZKPs, HE, and differential privacy (DP). Emerging trends, such as quantum-resistant cryptographic methods and efficient key-sharing for multiparty learning, are also highlighted. The discussion on verifiability primarily focuses on ZKPs and trusted execution environments (TEEs), outlining their theoretical frameworks and potential applications in PPML. However, the paper lacks a detailed exploration of practical mechanisms for ensuring verifiable correctness in complex ML models, particularly in decentralized settings. This leaves a gap in addressing real-world challenges like efficient proof generation and deployment, limiting the survey’s practical relevance for verifiability.


Authority
Insurance Company
(7) Verify the AI doctor's diagnosis result.
(2) Publish the commitment (c_a).
Patient
(4) Diagnose a disease.
(3) Send the medical history(x).
(6)Send the diagnosis(y), commitments (c_x, c_a), and its proof (π).
(6)Send the diagnosis(y), commitments (c_x , c_a), and its proof (π).
AI doctor
(1) Commit the AI weights values.
(5) Generate a proof.
(5) Verify all proofs from every bank.
(6) Aggregate gradients.
(7) Send the proof (π_1) and local parameters for homomorphic encryption ($HE(global\_grad_1)$).
Government
(7) Send the proof (π_1) and local parameters for homomorphic encryption ($HE(global\_grad_n)$).
(7) Send the global parameters for homomorphic encryption ($HE(global\_grad)$).
(7) Send the global parameters for homomorphic encryption ($HE(global\_grad_i)$).
(2) Training a model which decided by all banks.
(1) Collect money laundering data.
(1) Collect money laundering data.
(2) Training a model which decided by all banks.
Bank$_1$
...
Bank$_n$
(3) Generate a proof.
(3) Generate a proof.

(a) Artificial intelligence diagnosis based on case privacy data

(b) Federated training of anti-money laundering models based on private transaction behavior

Fig. 1. Real-life applications of zero-knowledge proof-based verifiable machine learning

TABLE I
COMPARISON WITH RELATED WORK.

Related Survey		Secure ML		**Zero-Knowledge Proof-based** Verifiable ML				
		Privacy	Verifiability	Scheme Coverage **of ZKP-VML**	Technical Analysis **of ZKP-VML**	Scheme Categorization **of ZKP-VML**	Future Research Directions **of ZKP-VML**	Problem Definition **of ZKP-VML**
Security and Privacy	Ma et al. [6]	●	◐	○(0)	○	○	○	○
	Qin et al. [7]	●	◐	◐(5)	○	◐	○	○
	Chen et al. [8]	●	◐	◐(2)	◐	○	○	○
	Hallaji et al. [9]	●	◐	◐(1)	○	○	○	○
Verifiability	Zhang et al. [10]	●	●	◐(1)	○	○	○	○
	Tariq et al. [11]	●	●	◐(2)	○	○	○	○
ZKP-based Verifiability	Modulus Lab [12]	●	●	◐(9)	○	◐	◐	○
	Sathe et al. [13]	●	●	◐(5)	○	○	○	○
	Zhang et al. [14]	●	●	○(0)	◐	○	◐	○
	Kersic et al. [15]	●	●	○(0)	◐	○	◐	◐
Our Work		●	●	●(53+)	●	●	●	●

[1] ○ denotes the requirement has not been met;
[2] ◐ denotes the requirement is partially met;
[3] ● denotes the requirement is fully met;

Chen et al. [8] offers a comprehensive analysis of privacy-preserving computation techniques in federated learning (FL), categorizing protocols based on cryptographic methods such as MPC, HE, DP, TEEs, and ZKPs. The paper provides detailed instantiation processes for each protocol and compares them in terms of security and computational efficiency. While it addresses verifiability, the focus is primarily on the global model aggregation process in FL, neglecting the more computationally intensive training process in DML. This leaves a gap in fully verifiable machine learning. Hallaji et al. [9] examine security and privacy challenges in decentralized federated learning (DFL) systems, particularly those using blockchain technology. The paper reviews state-of-the-art DFL methods, identifies threats to privacy and performance, and evaluates defense mechanisms, including DP, HE, and MPC. It also addresses DFL verifiability by introducing frameworks with trusted trainers and workers to ensure system reliability. However, the analysis of verifiability is limited, as the discussed methods are neither comprehensive nor sufficient, and the role of ZKPs in ensuring verifiability is overlooked.

These works offer comprehensive reviews of security and privacy issues across various machine learning scenarios. However, their analysis of verifiability is limited. Verifiability is a critical aspect of security, as it enhances trustworthiness by enabling the verification of transparent computations in decentralized machine learning. As a result, their discussions of security are not fully comprehensive.

2) *Surveys on Verifiable Machine Learning:* Some studies focus on trust and verifiability issues in decentralized machine learning. Zhang et al. [10] focus on verifiable FL, defining it as the ability for one party to prove to others in an FL protocol that it has correctly performed the intended task. They propose a novel taxonomy for verifiable FL, categorizing verifiability based on centralized and decentralized FL settings. In each setting, verifiability is further divided by participant security needs, covering model aggregation, local information,

and model updates. Techniques such as TEEs, reputation mechanisms, and contract theory are discussed. However, the analysis is limited by a narrow view of attacker behavior, leaving the verifiability framework incomplete. Notably, little attention is given to verifying the correctness of the local training process, and the role of ZKPs in verifiability is underexplored. Tariq et al. [11] provide a comprehensive exploration of trustworthy FL, addressing its principles, challenges, and future directions. They introduce a novel taxonomy with five key pillars: interpretability & explainability, transparency, privacy & robustness, fairness, and accountability. Each pillar is further divided into subcategories, offering a multidimensional perspective on trust. A Trustworthy FL architecture is proposed, incorporating secure aggregation, incentive mechanisms, and verifiability at both the client and server levels. While the paper emphasizes communication efficiency, it offers limited discussion on verifiability. Though acknowledging its importance, the work does not delve into the technical challenges of verifying the integrity of the local training process, the most computationally intensive part of FL. Additionally, the potential of cryptographic methods, such as ZKPs, to enhance verifiability is underexplored.

These surveys recognize the importance of verifiability in ML but focus less on verifying the model training or inference processes, prioritizing other computational aspects or alternative methods. As a result, their discussion of verifiability is insufficiently robust. Additionally, while ZKPs are crucial for ensuring verifiability, they are rarely addressed in these works, limiting the coverage of existing research.

3) Surveys on ZKP-Based Verifiable Machine Learning: Some reviews have recognized the importance of zero-knowledge proofs (ZKPs) in addressing verifiability issues in machine learning. Modulus Labs' white paper [12] explores the intersection of ZKPs and ML inference, benchmarking six ZKP systems across various models, especially multilayer perceptrons. It compares proof generation time and memory usage, highlighting trade-offs like Plonky2's speed versus high memory consumption, and zkCNN's efficiency for large models. The paper emphasizes potential in ZKPs to enable secure AI inference on decentralized systems while showcasing real-world use cases. While the paper effectively benchmarks ZKP systems in technical performance, its analysis of verifiability lacks a broader exploration of existing work. It focuses more on the ZKP systems themselves, with only one ZKP-VML scheme discussed, leaving out key aspects of ZKP-VML design. Moreover, it overlooks scalability concerns related to high computational and memory costs, particularly for low-power devices. This omission limits the development of efficient ZKP-VML schemes, constraining the adoption of ZKP-based solutions in lightweight environments. Sathe et al. [13] review the integration of ZKPs into ML to address privacy and security concerns, highlighting their role in enabling verifiable computations without compromising data privacy. The paper discusses five ZKP-based machine learning schemes: vCNN [16], zkCNN [17], ZEN [18], ZKP-FL [19], and Mystique [20]. However, the analysis is limited, covering only a narrow range of schemes without a systematic comparison or categorization. Additionally, the discussion lacks an in-depth exploration of current research priorities and future directions, limiting the paper's overall contribution. Zhang et al. [14] explore how ZKPs can enhance privacy, security, and data integrity in machine learning by enabling privacy-preserving data sharing and secure multi-party computation. However, the paper does not cover or analyze any existing ZKP-ML schemes, weakening its classification of ZKP-ML applications. The analysis of future research directions lacks grounding in prior work, and the brief length of the paper constrains its contribution. Kersic et al. [15] review the integration of ZKPs into ML for on-chain decentralized applications. The paper examines two on-chain ZKP-ML frameworks, EZKL and Orion, focusing on their ZKP systems, scalability, and privacy features, as well as exploring practical use cases like DAO treasury management. However, the review emphasizes on-chain applications rather than deeply analyzing the ZKP-ML schemes themselves. Given the emerging nature of on-chain ZKP-ML, the analysis is limited, covering only two schemes, neither of which has formal publications.

While these surveys review ZKP-VML as a separate issue, they generally suffer from significant limitations in both the breadth of coverage and the depth of analysis of ZKP-VML schemes. This not only leaves room for improvement in classifying existing research and identifying future development directions, but also, to some extent, underestimates the contributions of current studies in the field. The existing work covers no more than 10 different ZKP-VML schemes in total, yet in fact there are more than 50 ZKP-VML schemes. The limited coverage of existing work leads to the insufficient categorization and analysis of ZKP-VML issue.

To summarize, Table I shows that existing work has not yet provided a broad and in-depth review of the problem of **zero-knowledge proof-based verifiable machine learning**. On the one hand, there are only a few works that review the verifiability issues in machine learning, and even fewer of them consider zero-knowledge proofs. On the other hand, existing work does not provide a comprehensive and in-depth review of the ZKP-VML, lacking summary, coverage, analysis, and categorization of existing schemes. Meanwhile, as for a relatively new research area, these existing works are limited in summarizing the problems that still exist and the future research directions, making it difficult to provide assistance to subsequent researchers. Our work not only overcomes the above shortcomings, but also provides the definition of algorithms and properties to ZKP-VML, further paving the way for subsequent researchers.

B. Motivation

Decentralized machine learning protects data privacy, but the transparent local computation process introduces additional security challenges. Considering the potential active attacks during transparent computation, such as poisoning attacks and free-rider attacks, ensuring the verifiability of local computations is critical. Numerous reviews and surveys have explored security and privacy issues in decentralized machine learning, making valuable contributions to addressing various gaps in the literature. However, only a small fraction specifically address verifiability concerns or treat verifiability as a distinct issue. In parallel, various protocols and solutions have been proposed to tackle verifiability challenges in decentralized machine learning. Notably, most existing solutions leverage zero-knowledge proofs as a fundamental technique, yet this topic has been relatively overlooked in existing surveys that include verifiability. Meanwhile, existing surveys on ZKP-VML lack both depth of analysis and breadth of scheme coverage, leading to an absence of a comprehensive review in this area. To bridge the gap between the limitations of existing surveys and the breadth of research outcomes in ZKP-VML, we present this survey. This survey aims to emphasize the importance of verifiability in machine learning and the role of zero-knowledge proofs as a foundational cryptographic primitive. Specifically, we seek to provide a more systematic and comprehensive analysis, along with a detailed categorization, of the existing research, thereby addressing the current gap in surveys and reviews within this field.

C. Contributions

In this paper, we first present the definition of ZKP-VML algorithms and their corresponding properties, aiming not only to address gaps in the problem's definition but also to establish evaluation criteria for existing work and provide a reference for future research. Based on the definition and the basic process of ZKP-VML, we analyze the challenges and difficulties it faces. In analyzing the existing schemes, we first outline the timeline of ZKP-VML research development to provide a clear overview of the current state of research. Then we list and compare the existing schemes based on their security properties, as described previously. We identify three main technical route and some sub-routes, each main route addressing a specific research challenge faced by ZKP-VML as mentioned before. Each scheme is categorized according to its technical route, accompanied by an in-depth analysis of the scheme itself and how it addresses the corresponding research challenge. Furthermore, considering the computational cost as one of the major disadvantages of ZKP-VML, we also analyze how existing schemes optimize the additional computational and communication burdens. The existing optimization methods can be broadly classified into two main categories and several sub-categories. Finally, based on the analysis and categorization of existing ZKP-VML schemes, we identify the remaining research challenges and propose several promising directions for future research. The main contributions are listed as follows:

1) **Bring ZKP-VML to the stage.** To the best of our knowledge, this is the first comprehensive review of ZKP-VML, accompanied by formal definitions and security properties.
2) **Cover 56 existing ZKP-VML schemes.** We cover nearly all existing ZKP-VML schemes from the inception in 2017 to June 2024, outlining the development timeline of ZKP-VML. For each existing ZKP-VML scheme we provide a detailed analysis.
3) **Provide two classifications from different perspectives.** We classify existing ZKP-VML schemes from two novel perspectives: technical routes and optimization methods. Additionally, we provide a detailed analysis for each commonly employed technique and optimization in existing scheme.
4) **Present challenges and future directions.** Based on the analysis of existing schemes, we summarize the current challenges faced by ZKP-VML and propose potential directions for future research, offering guidance for future investigations in this field.

D. Scope of This Survey

Section II illustrates the background knowledge of machine learning and zero-knowledge proof. Section III defines the zero-knowledge proof-based verifiable machine learning (ZKP-VML). Section IV provides the overview of existing ZKP-VML schemes. Section V analyzes the key technical route on scheme building. Section VI summarizes the optimization methods employed by existing schemes. Section VII presents challenges and future directions. Finally, Section VIII concludes the survey.

II. Background

In this section, we provide background knowledge on decentralized machine learning and zero-knowledge proof, outlining how zero-knowledge proof addresses the security issues in decentralized machine learning.

A. Decentralized machine learning

Decentralized machine learning (DML) mitigates resource constraints and lowers participation barriers by distributing computational workloads. Traditionally, DML involves multiple workers sharing computation. We extend this definition to include single-worker scenarios, such as outsourced machine learning, due to their similar security concerns. The distribution of computation introduces additional security challenges, including increased participant interactions and potential malicious participants. To systematically analyze these security issues, we classify DML into single-worker and multi-worker paradigms based on workload distribution. We will then discuss their workflow and the security issues they face respectively.

1) Single-Worker DML: Single-Worker DML refers to a paradigm in which the computational workload of a machine learning task is handled by a single computing entity. Although the computation is solely performed by the central entity, the task itself is delegated by requesters who lack the capability to independently complete the machine learning process. Therefore, despite its centralized nature, this approach is still considered a form of DML. Common single-worker DML includes outsourced model training and inference serving.

Due to the limited computing power and storage capacity, requesters can outsource the model training task to a performer, allowing them to perform the computations [21]. This represents the fundamental concept of outsourced model training. The requester sends all the necessary data, including the model parameter W, the training dataset $D = (X, Y)$, and the hyper-parameters η to the server for the training task. The performer then performs the training task $W' \leftarrow W - \eta \nabla \mathcal{L}(f(X, W), Y)$ according to the loss function $\mathcal{L}$ and returns the trained parameters W'. However, verifiability remains a challenge. To save computational resources, dishonest performers may return manipulated results instead of performing computations honestly. The requester must verify that the received model parameters are correctly computed, especially when lacking sufficient local resources. This challenge intensifies when the requester depends on the server's private data for training, requiring verification without direct access to the complete input. For example, the performer may hold a private dataset used for model optimization or possess a pre-trained private model that the requester seeks to personalize with its own data.

With the rapid advancement of large language models, inference serving has become a widely used machine learning application. Unlike traditional settings where training and inference occur on the same device, inference serving offloads inference computation to a remote performer. Instead of running a model locally, a requester submits a query to a performer, which processes it using a trained model and returns the result. In this process, the requester seeks to obtain the inference result $\hat{Y} \leftarrow f(X, W)$ for input X on model f with parameters W. Verifiability is a key concern, as requesters must ensure the returned results are computed correctly using the specified input and model. However, dishonest performers may return arbitrary results or bypass computation to save resources. Additionally, verification mechanisms must preserve model privacy, preventing malicious requesters from reconstructing the model from the proofs.

2) Multi-Worker DML: Multi-worker DML distributes the computational workload of a machine learning task across multiple workers, enhancing efficiency and reducing single points of failure. However, effective task allocation is crucial for seamless collaboration. Common approaches include federated learning and crowdsourced inference.

Federated learning (FL) [3] enables multiple clients to collaboratively train a global model while preserving data privacy. Each client contributes to model training without sharing its private dataset. We focus on horizontal FL, where data samples are partitioned across participants, to illustrate the general process. A typical FL system consists of a central server S and n clients $C_1, ..., C_n$, each with a private dataset D_u. Training proceeds in multiple rounds until the global model converges. In the j-th training round, the server distributes the global model M_{j-1} to clients. Each client C_u trains the model on its dataset D_u to obtain the local model m_u^j. Clients submit their local model to the server, which aggregates them to obtain the global model $M^j = M^{j-1} + \frac{gl}{n} \sum_{u=1}^{n} (M^{j-1} - m_u^j)$, where gl is the global learning rate. Dishonest clients may submit incorrect local models to evade computation or conduct poisoning attacks. The server may also return wrong global model as the aggregation result to fool clients, raising concerns about trustworthiness [22]. Despite its privacy-preserving design, FL remains vulnerable to privacy attacks. A curious server may reconstruct client datasets via data reconstruction attacks or infer training participation using membership inference attacks [22].

Crowdsourcing inference involves multiple independent workers collaboratively performing machine learning inference tasks. These workers, operating within a distributed network, submit local inference results or partial computations to a central server, which aggregates them to generate the final output towards solving a larger problem. Each worker C_u generate the inference result $\hat{Y}_u \leftarrow f(X_u, W_u)$ based on their given data X_u and parameter W_u. α_u is the trust level of worker C_u, which is used for generate a more reliable final result $\hat{Y} \leftarrow \sum_{u=1}^{n} \alpha_u \hat{Y}_u$. While crowdsourcing inference shares a similar architectural with federated learning, it faces additional verifiability challenges beyond the local computation and global aggregation. On one hand, the heterogeneity of workers increases the complexity for the server to verify the correctness of each result. On the other hand, the limited access the server has to the workers' local models further increases the difficulty of performing accurate verification.

3) Common Models: Different ML models involve distinct computational processes, which influence their adaptation to ZKP-VML. We introduce several widely used models, emphasizing those of particular relevance to ZKP-VML. Our discussion focuses on their computational characteristics, their role in DML, and their implications for verifiable computation. Since ZKP-VML prioritizes computational complexity and workload types, we categorize models accordingly.

For traditional models with low computational complexity, we consider linear regression and decision tree, which feature simple structures and computations.

Linear regression (LR) is a fundamental ML model that establishes a linear relationship between independent variables $X = \{x_1, x_2, ..., x_p\}$ and a dependent variable y. The goal is to find the optimal hyperplane that minimizes the discrepancy between predicted and actual values [23]. The prediction is given by $\hat{y} = \sum_{i=1}^{n} \theta_i x_i + \theta_0$, where $\hat{y}$ is the predicted value, θ_0 is the intercept, θ_i is the coefficient for the feature x_i. The model minimizes the mean squared error (MSE) to measure the discrepancy between predictions and actual values to update parameters iteratively $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, where $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ and η is the learning rate. LR is well-suited for DML due to its simplicity and the scalability of optimization methods. In distributed settings, the dataset X and y are partitioned across multiple workers. Each worker computes the gradient of the loss function on its local data and transmits the results to a central server, which aggregates the gradients and updates the global parameters.

Decision tree (DT) is a non-parametric, interpretable model for classification and regression. It constructs a tree-like structure where internal nodes represent features, branches correspond to decision rules, and leaf nodes denote output values. The objective is to recursively partition the dataset into subsets that are as homogeneous as possible concerning the target variable [24]. Splits are determined by optimizing a criterion such as information gain or variance reduction. Starting from the root node, the model evaluates potential splits and selects the one that maximizes the chosen criterion, repeating this process recursively until a stopping condition is met. For instance, in variance reduction, splits aim to minimize variance

$Var(D) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2$ within child nodes, where D is the training dataset. The training process of DT models may not seamlessly integrate with DML. Nevertheless, DTs are widely utilized in large-scale applications such as credit scoring, fraud detection, and customer segmentation, owing to their simplicity, interpretability, and ability to handle both numerical and categorical data. Consequently, in DML, DTs are often employed to provide classification and regression services to users.

For deep models, we consider neural networks, convolutional neural networks and recurrent neural networks. In general, deep models require more computational resources during training compared to traditional models.

Neural Network (NN) is highly versatile, capable of modeling complex, non-linear relationships in data, and widely used for tasks such as classification, regression, and feature extraction [25]. At a high level, NNs process input data through weighted connections, introduce non-linearity via activation functions, and iteratively update weights based on the error between predicted and actual outcomes. They consist of three types of layers. **Input layer** receives input features $X = \{x_1, x_2, ..., x_p\}$, where each feature is assigned a weight reflecting its importance. **Hidden Layers** perform transformations using linear operations $z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$ followed by non-linear activation functions $a^{(l)} = \sigma(z^{(l)})$ in layer l, where $z^{(l)}$ is the pre-activation value, $W^{(l)}$ and $b^{(l)}$ are weights and biases, $a^{(l)}$ is the activation output, and $\sigma()$ is the activation function. **Output layer** produces probabilities using the Softmax function $\hat{y}_i = \frac{\exp(z_i)}{\Sigma_j \exp(z_i)}$. Training is performed via backpropagation, which computes the gradient of the loss function with respect to each weight and bias $\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \frac{\partial \mathcal{L}}{\partial a^{(l)}} \cdot \frac{\partial a^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial W^{(l)}}$, where loss function $\mathcal{L}$ quantifies the error between predicted values $\hat{y}$ and true values y. Weights are updated iteratively using an optimization algorithm like gradient descent $W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$, where η is the learning rate. NNs are computationally intensive, especially with large datasets and deep architectures. In DML, data or models can be distributed across multiple workers, each performing local training and sending updates to a central server for aggregation as federated learning or split learning. Trained models can also be deployed for ML inference services. However, training involves backpropagation, making it more complex and resource-intensive than inference. Consequently, applying ZKP-VML to training poses greater challenges than to inference.

Convolutional Neural Network (CNN) is a specialized NN designed for grid-like data, such as images. Unlike traditional NNs, CNNs employ convolutional layers to extract spatial features to build complex representations and pooling layers to preserve key features. **Convolutional layers** apply learnable filters that slide over the input to generate feature maps, capturing local structures with $z_{i,j,k} = \sum_{m,n,c} X_{i+m,j+n,c} \cdot W_{m,n,c,k} + b_k$, where X is the input, W is the filter, b_k is the bias for the k-th filter, $z_{i,j,k}$ is the output at position (i,j). **Pooling layers** reduce spatial dimensions, preserving essential features while lowering computational costs. Max pooling, for instance, selects the maximum value within each patch with $z_{i,j,k} = \max_{p,q} X_{i+p,j+q,k}$. Training CNNs is computationally demanding, particularly for large datasets and deep architectures. Additionally, representing high-dimensional data and convolutional computations within the ZKP-VML framework presents significant challenges.

Recurrent Neural Network (RNN) processes sequential data by modeling temporal dependencies through recurrent connections, enabling them to retain information from past inputs. This makes them effective for tasks such as time-series analysis, natural language processing, and speech recognition. RNNs maintain a hidden state that updates at each time step based on the current input and the previous hidden state as $h_t = f(W_h h_{t-1} + W_x x_t + b)$, where W_h and W_x are weight matrices, b is the bias vector, $f()$ is an activation function. The output at time step t is computed as $y_t = g(W_y h_t + c)$, where W_y is the output weight matrix, c is the bias vector, and $g()$ is typically a softmax or linear activation function. For sequence modeling, the total loss $\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)$ is the sum of errors over all time steps, where ℓ is the loss function, and T is the sequence length. Gradient computation involves unrolling the network and applying backpropagation through time (BPTT) as $\frac{\partial \mathcal{L}}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_h}$. RNNs are computationally intensive, particularly for long sequences or large datasets, due to their sequential nature. While their core computations resemble those of standard NNs, their concept of state may offer novel optimizations within the ZKP-VML framework.

For large models, we consider transformers. Large models are trained on vast datasets with billions of parameters, demand extensive computational resources and large-scale distributed training.

Transformers revolutionized ML by replacing recurrent and convolutional structures with self-attention mechanisms, enabling parallel processing of entire input sequences. Inputs are tokenized and embedded into continuous vector representations, augmented with positional encoding to retain sequence order as $PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/d}}\right)$, and $PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/d}}\right)$. Self-attention computes weighted sums of input tokens, allowing the model to focus on relevant sequence elements with $\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, where $Q = XW_Q, K = XW_K, V = XW_V$, X is the input sequence, W_Q, W_K, W_V are learnable weight matrices, d_k is the dimensionality of K. Multi-head attention applies multiple self-attention operations in parallel to capture different input aspects as $\text{Multihead}(Q,K,V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O$. Transformers are much more computationally intensive. Billions of parameters introduces massive resource and computational overheads for ZKP-VML converting into arithmetic circuit or generating proofs. Besides, transformers rely heavily on non-linear functions and complex operations, which are challenging to represent in ZKP-friendly formats.

B. Zero-Knowledge Proof

Zero-knowledge proof (ZKP) [26] is a cryptographic protocol where a prover demonstrates the truth of a statement to a verifier without revealing any additional information. In simple terms, ZKP has three key properties:

- **Completeness:** Given a statement of a witness, the prover can convince the verifier that the statement is correct if the protocol is executed honestly.
- **Soundness:** Any malicious prover cannot fool the verifier into accepting a false statement.
- **Zero-Knowledge:** The prover does not leak any information else but the correctness of the statement during the protocol.

1) Non-Interactive Zero-Knowledge Argument: Zero-knowledge proofs can be classified as interactive or non-interactive based on the number of interaction rounds. Interactive protocols involve multiple rounds between the prover and verifier, whereas non-interactive zero-knowledge (NIZK) [27], [28] proofs require only a single message from the prover. Once generated, NIZK proofs can be distributed for independent verification, reducing the need for assumptions like synchronous communication and reliable channels. Due to their efficiency, scalability, and security benefits, NIZK proofs are widely adopted in areas like DML. This paper focuses on NIZKs, which are constructed using two main methods: the Common Reference String (CRS) model and the Random Oracle Model (ROM) [29].

In the CRS model, it is assumed that a common reference string is generated by a trusted third party and is available to both the prover and the verifier for proof generation and verification. Specifically, we provide the formal definition of CRS-based NIZK protocols [30]. Let $\mathcal{R}$ be a relation generator that given a security parameter λ in binary returns a polynomial time decidable binary relation R. The relation generator may also output some auxiliary input z. For pairs $(\phi, w) \in R$, we call ϕ the statement and w the witness. We define $\mathcal{R}_\lambda$ to be the set of possible relation $\mathcal{R}$ may output given 1^λ. An efficient prover publicly verifiable non-interactive argument for R is a quadruple of probabilistic polynomial algorithms $(\mathrm{Setup}, \mathrm{Prove}, \mathrm{Vfy}, \mathrm{Sim})$ such that

- $(\sigma, \tau) \leftarrow \mathrm{Setup}(R)$: The setup take a security parameter λ and a relation $R \in R_\lambda$ as input, outputs a common reference string σ and a simulation trapdoor τ for the relation R. The algorithm serves as an initialization for the circuit and needs to be executed only once for the same circuit.
- $\pi \leftarrow \mathrm{Prove}(R, \sigma, \phi, w)$: The prove take a common reference string σ and $(\phi, w) \in R$ as input, outputs argument π.
- $0/1 \leftarrow \mathrm{Vfy}(R, \sigma, \phi, \pi)$: The vfy takes a common reference string σ, a statement ϕ and an argument π as input, outputs 0 (reject) or 1 (accept).
- $\pi \leftarrow \mathrm{Sim}(R, \tau, \phi)$: The sim takes a simulation trapdoor τ and statement ϕ as input, outputs an argument π.

Definition 1: We say $(\mathrm{Setup}, \mathrm{Prove}, \mathrm{Vfy}, \mathrm{Sim})$ is a non-interactive zero-knowledge argument of knowledge for R if it has perfect completeness, computational knowledge soundness and perfect zero-knowledge as defined below.

Perfect completeness says that, given any true statement, an honest prover should be able to convince an honest verifier. For all $\lambda \in \mathbf{N}, R \in \mathcal{R}_\lambda, (\phi, w) \in R$:

$$\Pr\begin{bmatrix}(\sigma,\tau) \leftarrow \mathrm{Setup}(R); \\ \pi \leftarrow \mathrm{Prove}(R,\sigma,\phi,w): \\ \mathrm{Vfy}(R,\sigma,\phi,\pi) = 1\end{bmatrix} = 1 \tag{1}$$

Computational knowledge soundness says that there exist an extractor that can compute a witness whenever the adversary produces a valid argument. Formally, we require that for all non-uniform polynomial time adversaries $\mathcal{A}$ there exists a non-uniform polynomial time extractor $\mathcal{X}_\mathcal{A}$ such that:

$$\Pr\begin{bmatrix}(R,z) \leftarrow \mathcal{R}(1^\lambda);\ (\sigma,\tau) \leftarrow \mathrm{Setup}(R); \\ ((\phi,\pi);w) \leftarrow \mathcal{A}||\mathcal{X}_\mathcal{A}(R,z,\sigma): \\ \phi \notin L_R \text{ and } \mathrm{Vfy}(\mathrm{R},\sigma,\phi,\pi) = 1\end{bmatrix} \approx 0 \tag{2}$$

Perfect zero-knowledge says that it leaks no information besides the truth of the statement. For all $\lambda \in \mathbf{N}, (R, z) \in \mathcal{R}_\lambda, (\phi, w) \in R$ and all adversaries $\mathcal{A}$:

$$\Pr\begin{bmatrix}(\sigma,\tau) \leftarrow \mathrm{Setup}(R); \\ \pi \leftarrow \mathrm{Prove}(R,\sigma,\phi,w): \\ \mathcal{A}(R,z,\sigma,\tau,\pi) = 1\end{bmatrix} = \Pr\begin{bmatrix}(\sigma,\tau) \leftarrow \mathrm{Setup}(R); \\ \pi \leftarrow \mathrm{Sim}(R,\tau,\phi): \\ \mathcal{A}(R,z,\sigma,\tau,\pi) = 1\end{bmatrix} \tag{3}$$

In the random oracle model, a hash function can replace the verifier's random challenge, allowing interactive zero-knowledge arguments to be transformed into non-interactive ones via the Fiat-Shamir heuristic [31]. We demonstrate this process using the Schnorr protocol, an interactive cryptographic method for proving knowledge of a discrete logarithm without revealing the actual value, preserving the zero-knowledge property. The protocol operates in a cyclic group G of prime order q with generator g. The prover, who knows a secret value x, broadcasts $y = g^x$. The process begins with the prover selecting a random value r and computing $t = g^r$, which is sent to the verifier. The verifier then issues a random challenge c. The prover responds by calculating $s = r + cx$ and sending s back. The verifier verifies the validity by checking $g^s = t \cdot y^c$. This ensures the verifier gains no information about the secret x while being convinced of the prover's knowledge of it. The Fiat-Shamir heuristic [31], [32], [33] converts the interactive Schnorr protocol into a non-interactive version by replacing the verifier's random challenge with a deterministic one derived from a cryptographic hash function. In this non-interactive protocol, the prover computes the challenge $c = H(y\|t)$, where H is a hash function, instead of waiting for a challenge from the verifier. The verifier re-computes c as $c = H(y\|t)$ for the verification. This transformation removes the need for interaction.

2) Common schemes: We present four common ZKP schemes related to verifiable machine learning.

Sumcheck protocol [34] is an interactive protocol to sum a multivariate polynomial $f : \mathbb{F}^\ell \rightarrow \mathbb{F}$ with binary inputs: $H = \sum_{b_1, b_2, ..., b_\ell \in 0,1} f(b_1, b_2, ..., b_\ell)$. Directly summing requires 2^ℓ computations according to the combinations of b_i. The sumcheck protocol enables the verifier V to efficiently verify H with the prover P in ℓ rounds with proof size of $O(d\ell)$. In the first round, P sends a uni-variate polynomial $g_1(x_1) = \sum_{b_2,...,b_\ell} f(x_1, b_2, ..., b_\ell)$, V checks whether $H = g_1(0) + g_1(1)$, and sends a random challenge r_1 to P. Then in the i-th round, P sends a uni-variate polynomial $g_i(x_i) = \sum_{b_{i+1},...,b_\ell} f(r_1, ..., r_{i-1}, x_i, b_{i+1}, ..., b_\ell)$, V checks whether $g_{i-1}(r_{i-1}) = g_i(0) + g_i(1)$, and sends a random challenge r_i to P. In the ℓ-th round, which is the last round, P sends a uni-variate polynomial $g_\ell(x_\ell) = f(r_1, ..., r_{\ell-1}, x_\ell)$, V checks whether $g_{\ell-1}(r_{\ell-1}) = g_\ell(0) + g_\ell(1)$, and generates a random challenge r_ℓ. With the access to $f(\cdot)$, V accepts the proof if $g_\ell(r_\ell) = f(r_1, ..., r_{\ell-1}, r_\ell)$. In the sumcheck protocol, the prover P eventually needs to expose the polynomial in order for the verifier V to evaluate at these challenge points, which might leak knowledge. By applying a zero-knowledge polynomial commitment following the framework in [35], the polynomial can be concealed and zero-knowledge can be achieved. Furthermore, the interactive challenge process can be replaced by the Fiat-Shamir heuristic [31], and the sumcheck protocol thus becomes a NIZK protocol.

Quadratic arithmetic program (QAP) [36] is an effective coding method of circuit satisfaction (C-SAT) problem, by which the C-SAT problem can be reduced to divisibility problem between polynomials. The C-SAT problem refers to whether, for a given circuit, there exists an input such that the output of the circuit is 1.

***Definition** 2:* A quadratic arithmetic program [36] $\mathcal{Q}$ converted from arithmetic circuit $\mathcal{C}$ in field $\mathbb{F}$ is consisting of three sets of polynomials $u_i(x),\ v_i(x), w_i(x)_{i=0}^m$ and a target polynomial $t(x)$. For public inputs and outputs $(c_1, ..., c_\ell)$, the $\mathcal{Q}$ is satisfiable if and only if there exist coefficients $(c_{\ell+1}, ..., c_m)$ such that $t(x)$ divides $p(x)$, where $p(x) = (\sum_{i=1}^m c_i \cdot u_i(x)) \cdot (\sum_{i=1}^m c_i \cdot v_i(x)) - \sum_{i=1}^m c_i \cdot w_i(x)$.

Based on the QAP and the common reference string (CRS) model [37], Gennaro et al. [36] constructed zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK). In brief, a zk-SNARK scheme for relation R is a quadruple of probabilistic polynomial algorithms $(\mathrm{Setup}, \mathrm{Prove}, \mathrm{Vfy})$ such that:

- $(\sigma, \tau) \leftarrow \mathrm{Setup}(R)$: The *Setup* takes a relation R as input, outputs a CRS σ and the trapdoor τ.
- $\pi \leftarrow \mathrm{Prove}(R, \sigma, \phi, w)$: The *Prove*, under relation R, takes a CRS σ and $(\phi, w) \in R$ as input, outputs proof π.
- $0/1 \leftarrow \mathrm{Verify}(R, \sigma, \phi, \pi)$: The *Verify*, under relation R, takes a CRS σ, a statement ϕ and a proof π as input, outputs 0 (reject) or 1 (accept).

Inner product argument (IPA) [38] is an interactive protocol to prove the knowledge of the inner product of vectors $\vec{a}$ and $\vec{b}$ such that $z = \vec{a} \cdot \vec{b} = \sum_{i=1}^n a_i b_i$ with corresponding commitment $C = \sum_{i=1}^n a_i G_i + \sum_{i=1}^n b_i H_i$. The core idea of IPA is to recursively reduce the length of the vector up to the scalar by random challenges in several rounds. With the Fiat-Shamir heuristic [31], the IPA protocol can be non-interactive. In order to construct a zero-knowledge proof from IPA, the arithmetic circuit constraints between and among multiplication gates are first formalized by the Schwartz-Zippel Lemma [39], [40] as the problem where the coefficients of particular terms in a polynomial are zero. And then it is transformed into a statement of inner product form, which can be handled by IPA directly.

MPC-in-the-head [41] is an approach to constructing zero-knowledge proofs via secure multi-party computation (MPC) protocols.

***Definition** 3:* A secure multi-party computation (MPC) protocol [42] Π_f allows n participants $P_1, ..., P_n$ to compute a common output $y = f(x, w_1, r_1, ...)$ based on the common input x, respective secret input $w_1, ..., w_n$ and random inputs $r_1, ..., r_n$ through k rounds of interaction processes. Let the view of participant P_i be $V_i = (x, w_i, r_i, M_i)$, where $M_i = (m_{i,1}, ..., m_{i,k})$ denotes the message received by P_i. Then the behavior of participant P_i can be determined by the number of rounds j and the current view $V_{i,j} = (x, w_i, r_i, (m_{i,1}, ..., m_{i,j}))$.

The prover simulates several participants to executes a multi-party computation protocol and saves the views of each participant in the process. The verifier can check these views through commitments to verify the correctness of the protocol execution process.

3) Advantages of Zero-Knowledge Proof: While various cryptographic techniques address verifiable machine learning to some extent, they each have practical limitations. Zero-knowledge proofs (ZKPs), however, not only ensure computational integrity but also prevent information leakage, making them well-suited for verifiable machine learning. Below, we outline the drawbacks of alternative cryptographic approaches.

Secure Multi-Party Computation (MPC) enables multiple parties to jointly compute a function without revealing their private inputs [43]. However, MPC requires multiple rounds of interaction and synchronized communication, increasing complexity. In contrast, many ZKPs are non-interactive, reducing network assumptions and communication overhead.

Homomorphic Encryption (HE) allows computations on encrypted data without prior decryption [44]. HE schemes fall into three categories: partially homomorphic encryption (supporting only addition or multiplication), somewhat homomorphic encryption (allowing limited additions and multiplications), and fully homomorphic encryption (supporting arbitrary computations). While fully homomorphic encryption enables complex computations for ML computations, its prohibitive computational cost makes it impractical for machine learning. ZKPs, in comparison, offer significantly lower computational overhead, enhancing feasibility for VML.

Differential Privacy (DP) protects individual data privacy while preserving statistical utility [45]. However, DP does not provide verifiability. Unlike DP, ZKPs ensure both privacy and verifiability through their zero-knowledge properties.

Trusted Execution Environments (TEE) ensure secure code execution via isolated hardware [46]. While TEE and ZKPs both guarantee correctness and privacy, TEE depends on a trusted hardware provider, introducing additional security assumptions and implementation costs. In contrast, ZKP security relies solely on cryptographic assumptions, avoiding extra hardware expenses.

Overall, ZKPs provide a balance of efficiency, privacy, and verifiability, making them a strong candidate for VML applications.

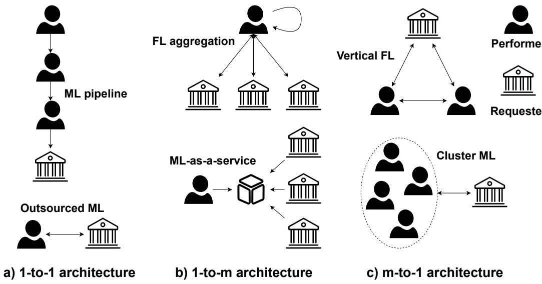


Fig. 2. The architecture of different verifiable machine learning.

C. Communication in ZKP-VML

The simplest approach to verify machine learning computations is for the performer to send all input and output data to the requester, who then re-executes the computation to verify the correctness. However, this method compromises data privacy and significantly increases both communication and computational costs for the requester. ZKP-VML addresses these issues by ensuring privacy and reducing communication overhead at the cost of increased computation on the performer's side.

1) Communication Architecture: Different DML frameworks lead to distinct communication architectures, each requiring tailored privacy and verifiability mechanisms. We summarize three common architectures for further analysis on communication.

1-to-1. This architecture contains only 1 requester and 1 performer. After receiving the ML task and public data, the performer can execute the computation locally and generate a proof. This architecture is common in ML pipelines, outsourced ML, and various DML scenarios. The requester can run the setup step if the trusted setup is needed for the ZKP protocol.

1-to-m. This architecture contains 1 performer and multiple requesters. The performer has to prove the correctness of the computation to multiple requesters. To minimize redundant communication, non-interactive ZKPs can be adopted, which allows requesters to verify proofs independently once published. The 1-to-m architecture may exist in FL aggregation, ML-as-a-service, and many other DML scenarios. The multiple requesters have to run the setup step collaboratively if the trusted setup is needed for the ZKP protocol. Further, the 1-to-m architecture involves extra privacy issues if there exists private data between different requesters. For example, in FL aggregation, the server has to provide privacy protection of each local model. While in ML-as-a-service, the server only has to generate a proof arguing for the performance of its model, and each user can check the proof at their use.

m-to-1. This architecture contains multiple performers and 1 requester. Each performer is responsible for one part of the ML task. Performers are asked not only generate proofs for their local computations, but also jointly prove the correctness of their interactions, ensuring end-to-end verification. This architecture is relevant to vertical FL, cluster ML, and similar scenarios. The requester has to run the setup step for different kinds of computations within the task if the trusted setup is needed for the ZKP protocol. Further, the m-to-1 architecture involves extra privacy issues if there exists private data between different performers. For example, in vertical FL, the training data in each participant is private. While in cluster ML, all the node shares the training dataset and model parameters.

2) Attackers in Communication: In the aforementioned DML communication architectures, both the requester and the performer can potentially act as adversaries, thereby compromising the security and privacy of the machine learning system. Specifically, a malicious requester typically targets data privacy, attempting to extract sensitive information from the performers, whereas a malicious performer primarily undermines data security by providing incorrect computational outputs. Several classical attack methods in machine learning can be adapted to exploit these vulnerabilities.

When performers act as adversaries, they may intentionally provide incorrect results during computation. In machine learning, two major performer attacks are poisoning attacks and free-rider attacks. **Poisoning attacks** degrade model performance by introducing incorrect patterns into the learning process, leading to misclassifications (e.g., assigning label b to the sample in class a). Adversaries may inject mislabeled and poisoned data into the training dataset or directly manipulate model parameters, degrading the accuracy and reliability of the model delivered to the requester. **Free-rider attacks** allow adversaries to submit results and complete tasks without performing the required computations. These attacks often arise in collaborative scenarios, such as federated learning or cluster machine learning, where multiple performers execute the similar computational task for some rewards. Adversaries may copy and reuse results submitted by other performers, deceiving the requester into accepting these results, thereby stealing training rewards while minimizing computational costs.

When requesters act as adversaries, they may attempt to infer private data used by performers, compromising data privacy. In machine learning, two major requester attacks are membership inference attacks and reconstruction attacks. **Membership inference attacks** seek to determine whether a specific sample is included in the training dataset of the model. These attacks exploit the observation that models generally exhibit higher confidence and accuracy on their training data than on unseen data, as the model has directly learned patterns and features from these samples. Adversaries may construct shadow models that replicate the behavior of the target model, or develop specialized metrics to evaluate the confidence levels of the model inference results. **Reconstruction attacks** aim to reconstruct private training data by leveraging model gradient updates. Since similar training data produce similar gradient updates, adversaries can iteratively refine candidate inputs until their gradients match those obtained from the performer. This process can reconstruct data similar to private training data, violating the privacy of performers.

Additionally, adversaries may also disrupt the ML task by refusing to respond or providing incorrect responses, leading to communication failures and task disruption. Such attacks pose a greater threat in ML scenarios with additional computational and communication processes to ensure privacy.

A list of key acronyms and abbreviations used throughout the paper is given in Table II.

TABLE II
LIST OF KEY ACRONYMS

Acronyms	Definitions
ZKP-VML	Zero-Knowledge Proof-based Verifiable Machine Learning
ML	Machine Learning
DML	Decentralized Machine Learning
ZKP	Zero-Knowledge Proof
zk-SNARK	Zero-Knowledge Succinct Non-interactive Argument of Knowledge
QAP	Quadratic Arithmetic Problem
QPP	Quadratic Polynomial Problem
IPA	Inner Product Arguments
R1CS	Rank-1 Constraint System
VOLE	Vector-Oblivious Linear Evaluation
MPC	Multi-Party Computation
DP	Differential Privacy
HE	Homomorphic Encryption
FL	Federated Learning
NN	Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DT	Decision Tree
LLM	Large Language Model

III. Zero-Knowledge Proof-based Verifiable Machine Learning

A. Definition

Considering that most of the verifiability issues in distributed machine learning exist among inference and training tasks, we give a pioneering definition of zero-knowledge proof-based verifiable machine learning (ZKP-VML), in order to provide an evaluation criterion for existing work. We model ZKP-VML through participants, algorithms, and workflows.

There are two types of participants, the prover P and the verifier V. V delegates ML tasks (training or inference) to P, and expects P to return the correct result. P performs the required ML tasks to return the result, while generating proofs of its correctness, which is verified by V. Both P and V can be malicious, leading to incorrect results being submitted or private data being leaked.

Gennaro, Gentry and Parno [47] provided a formal definition of outsourced computation with four algorithms. In order to provide an accurate and unambiguous description, we define the non-interactive verifiable machine learning modeled after that.

***Definition** 4: A verifiable machine learning scheme allowing a prover P to trustfully execute given machine learning task T with specific input data D for verifier V is a tuple of below algorithms:*

- $(PK, VK) \leftarrow$ **Setup**(T, λ): By inputting the task function T and a security parameter λ, the *setup* algorithm generates a proving key PK for the proof generation and a verification key VK for the proof verification.
- $R \leftarrow$ **Execute**(T, D) : By inputting the task function T, both the private and public input data of the prover $D = \{D_{pri}, D_{pub}\}$, the *execution* algorithm perform the task T and outputs the execution result R.

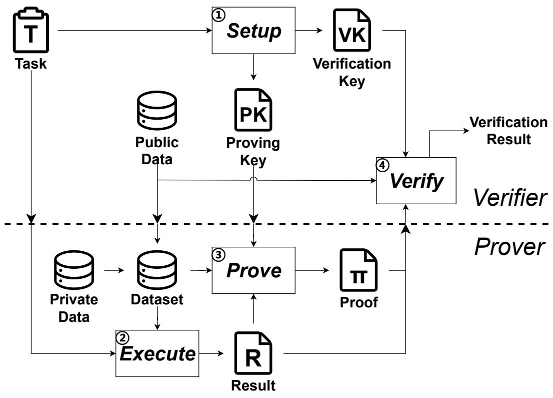


Fig. 3. The workflow of verifiable machine learning.

- $\pi \leftarrow$ **Prove**(PK, D, R) : By inputting the proving key PK, input data D and execution result R, the *proving* algorithm generates the corresponding proof π.
- $[0, 1] \leftarrow$ **Verify**(VK, π, D_{pub}, R) : By inputting the verification key VK, proof π, public input data D_{pub} and the result R, the *verification* algorithm checks whether the proof π is valid. If the verification pass, the algorithm outputs a positive number not greater than 1, otherwise 0.

A ZKP-VML scheme should be both correct and secure. More formally:

Correctness. If P has honestly completed the task with the given input, then **Verify** outputs 1 (Accept) with the probability of 1.

$$\Pr\begin{bmatrix}(PK, VK) \leftarrow \textbf{Setup}(T, \lambda); \\ R \leftarrow \textbf{Execute}(T, D); \\ \pi \leftarrow \textbf{Prove}(PK, D, R) : \\ \textbf{Verify}(VK, \pi, D_{pub}, R) > 0\end{bmatrix} = 1 \tag{4}$$

Security. If P is malicious then **Verify** outputs 1 with a negligible probability. For any PPT adversary $\mathcal{A}$,

$$\Pr\begin{bmatrix}(PK, VK) \leftarrow \textbf{Setup}(T, \lambda); \\ R \leftarrow \textbf{Execute}(T, D); \\ (R', \pi') \leftarrow \mathcal{A}(PK, T) : \\ R' \neq R \wedge \textbf{Verify}(VK, \pi', D_{pub}, R') > 0\end{bmatrix} \leq negl(\lambda) \tag{5}$$

In addition, we should also consider the privacy of the proof verification process and the characteristic of machine learning computation results.

Partial Privacy. If the private data $D_{pri} \neq \emptyset$.

Privacy. If the scheme satisfies *partial privacy* and further, leaks no information that contributes to the inference or reconstruction of the prover's private input D_{pri} besides the verification result to V. For any PPT adversary $\mathcal{A}$,

$$\begin{aligned}&\Pr\begin{bmatrix}(PK,VK)\leftarrow \textbf{Setup}(T,\lambda);\\ d\in D_{pri}; R\leftarrow \textbf{Execute}(T,D);\\ \pi\leftarrow \textbf{Prove}(PK,D,R):\\ 1\leftarrow \mathcal{A}(VK,T,\pi,D_{pub},R,d)\end{bmatrix}\\ =&\Pr\begin{bmatrix}(PK,VK)\leftarrow \textbf{Setup}(T,\lambda);\\ d'\notin D_{pri}; R\leftarrow \textbf{Execute}(T,D);\\ \pi\leftarrow \textbf{Prove}(PK,D,R):\\ 1\leftarrow \mathcal{A}(VK,T,\pi,D_{pub},R,d')\end{bmatrix}\end{aligned} \quad (6)$$

Distinctness. If there exist a task function T and three distinct input data D_1, D_2, D_3, with their execution results and proofs respectively, satisfying $\textbf{Verify}(VK,\pi_1,D_{pub},R_1) < \textbf{Verify}(VK,\pi_2,D_{pub},R_2) < \textbf{Verify}(VK,\pi_3,D_{pub},R_3)$.

We will explain the meaning of the above algorithms and properties into further detail by providing a basic example workflow of a verifiable machine learning scheme as is shown in Fig. 3. First, the verifier runs the **Setup** to generate the proving key PK and the verification key VK according to the machine learning task T. In this process, the machine learning computation process is transformed into some representation of the problem with zero-knowledge proofs, mostly in the form of arithmetic circuits. That is, the computational process to be proved is transformed into an equivalent arithmetic circuit representation. Thus the correctness of the computation can be verified by proving the C-SAT problem, i.e., that the output is obtained via the claimed circuit with some input. By sending the proving key PK, task T, and some necessary public data D_{pub} to the prover P, the verifier can delegate the task to the prover. Thus the prover can **Execute** the task T with the dataset D to obtain the result R. It is worth mentioning that R does not refer to the computational result $\hat{R}$ of a pure machine learning task, such as the plaintext of the optimized model parameters in a training task. R could be the encrypted, masked, or committed result, taking into account some additional privacy settings. The proof π arguing for the correctness of R can be generated with **Prove**. Finally, the verifier can **Verify** whether the result R is correct with the proof π. Another point worth noting is that the result returned by **Verify** is not 0 (reject) or 1 (accept), but a result within the interval from 0 to 1, where 0 means reject and any positive number within the interval means accept. This is due to that the computational results of machine learning tasks possess some non-binary additional properties, such as accuracy. Such a multi-result verification algorithm can provide room for the scheme to evaluate the accuracy or other properties of the results.

As for the properties of verifiable machine learning, correctness guarantees that the result of an honest execution will always pass the verification, while security guarantees that a malicious result will be rejected with high probability. It is worth mentioning that, some verifiable machine learning schemes cannot fully satisfy the correctness and security; these schemes may misjudge the proof with some probability. We define them as *partial correctness and security* in this paper. Partial privacy means that in this machine learning task, there does exist some private data to the prover. The private data will not directly leak to the verifier during the verification due to the algorithm itself. Considering some privacy attacks on machine learning based on the computational results, such as the membership inference attack, data reconstruction attack, we further provide the definition of privacy. Privacy means that the computation and verification process provide no information that helps the verifier to identify whether a certain data is in the private dataset or not. Distinctness refers to the ability of a verifiable machine learning scheme to provide a non-binary evaluation criterion beyond simple acceptance or rejection of the computation results. Unlike other computations, the computational results of machine learning tasks are generally non-deterministic, and for some machine learning tasks, there is no absolute right and wrong about the result. For example, in model training, the verifier may receive two correctly computed local models from different provers. How can the verifier further distinguish and evaluate these two models?

B. Threat Model

In the context of the ZKP-VML scenario defined above, we consider the threat model of the system in terms of the roles, capabilities, and objectives of the attacker.

When the performer acts as an attacker, they may carry out integrity attacks. **Integrity attacks** introduce errors into the computation process, leading to incorrect results being returned to the requester. In multi-round machine learning tasks, integrity attacks can be classified as either single-round or multi-round. A single-round integrity attack occurs when the attacker manipulates results in one specific round while behaving normally in others, primarily to introduce random errors. In contrast, a multi-round integrity attack involves selectively poisoning a subset of rounds, enabling more subtle and strategic manipulation of results. Attackers may also collaborate with other malicious executors to manipulate deliberate errors. Notably, integrity attacks encompass any deliberate submission of incorrect results. Common integrity attacks include poisoning attacks and free-rider attacks. For integrity attacks, ZKPs enable efficient verification of the performer's local computation, allowing the requester to verify with minimal computational overhead, whether the submitted results align with the claimed computation process. This prevents attackers from returning incorrect results. Additionally, ZKPs provide further insight into specific properties within the computation to detect poisoned data used in the computational process, mitigating the risk of attackers‘ using malicious inputs to generate poisoned results through correct computation processes. In multi-round tasks involving multiple performers, the increased complexity of computation and communication introduces additional security risks. To mitigate these risks, supplementary cryptographic techniques such as HE and MPC may be necessary.

When the requester acts as an attacker, they may launch privacy attacks. **Privacy attacks** aim to extract private information from the data used in ML tasks, leading to varying degrees of privacy leakage. The attacker has access only to the submitted computation results and the contents of the communication during interaction. In multi-performer scenarios, the malicious requester may collude with certain performers to gain additional data, further facilitating privacy attacks. Consequently, performers can also act as attackers if they have access to transcriptions. Common privacy attacks include membership inference attacks and reconstruction attacks. For privacy attacks, it is crucial to ensure that the original ML framework does not expose the performer's private data. Cryptographic techniques like DP and HE can protect computation results, making it difficult for attackers to infer or reconstruct private inputs, while maintaining the usability of computation results. Since ZKP-VML relies on ZKPs for verification, it does not introduce additional privacy risks, even if a malicious requester colludes with certain performers.

C. Challenges

Based on the aforementioned workflow, we clarify the challenges that a ZKP-based verifiable machine learning scheme faces. By dividing the above process by algorithm, which is the action of parties, the challenges are mainly present in *Setup*, *Prove*, and *Verify*. These additional algorithms are designed to introduce verifiability into machine learning, and consequently pose new challenges.

The first challenge is *Generalizability*, which lies in the *Setup* step. Due to the nature of the training data, model parameters, etc., the vast majority of computations involved in machine learning are floating-point computations. However, as a cryptographic technique, zero-knowledge proofs work on the group of finite fields, which are the integers. Therefore, the floating-point computational process cannot be directly mapped to the group without accuracy loss. Moreover, zero-knowledge proofs can only support additive and multiplicative operations, making it difficult to represent complex non-linear activation functions in neural networks. Thus, the first challenge concerns transforming the computational process of machine learning tasks into a zero-knowledge proof problem and representation. This challenge can be divided into two sub-problems: the float-integer conversion problem and the non-linear computation conversion problem. However, this challenge does not serve as the main research direction of ZKP-VML, even though every ZKP-VML scheme must address these problems. Existing ZKP-VML schemes tend to implement simpler solutions or existing schemes, such as directly scaling up the floating-number and then truncating them, or using existing mapping methods.

The second challenge is *Efficiency*, which lies in the *Prove* step. Machine learning is a computationally intensive application. For the neural networks commonly used today, the number of parameters is typically in the millions. And the time complexity of zero-knowledge proofs is basically related to the circuit size. Such massive computations with millions of neurons can lead to the large size of arithmetic circuits, which in turn leads to the unaffordable computational cost for zero-knowledge proofs. To illustrate this problem, we provide a real-world example. Zero-knowledge succinct non-interactive argument of knowledge [36] (zk-SNARK) scheme is one of the most commonly used type of zero-knowledge proof schemes, and Groth16 [30] is the one of the most commonly used of zk-SNARKs. For the neural network model VGG16 [48] with 138M parameters, the proof generation time may reach an impractical duration of 10 years. Also the size of the generated proving key and verification key exceeds 1,000 TB. The second challenge is about how to efficiently generate the proofs for the computational process of machine learning tasks. This challenge is the main research direction of ZKP-VML. Most ZKP-VML schemes work on improving the proving efficiency in different ways to reduce the additional burden introduced by ZKP, making the scheme more practical. The technical routes of existing work include: 1. Optimizing the circuit representation. This involves reducing the size of the circuit of the equivalent representation for the given computational process. 2. Improving the efficiency of verification. Which involves how to improve the efficiency of proof generation and verification at the cost of acceptable security errors for the given computational process.

The third challenge is *Evaluability*, which lies in the *Verify* step. Zero-knowledge proofs can be used to prove the correctness of the computational process of a machine learning task, avoiding the result without honest computations submitted by lazy or malicious participants. However, this is not an adequate to defense against attacks and dishonest behaviours. Considering participants without an honest dataset, they can still perform the machine learning task and honestly generate valid proofs for the results. It is difficult for the verifier to detect such malicious behaviour, for the malicious result comes with a valid proof, arguing for the "honest" execution process. This issue is equally important for the verifiability. Thus the third challenge is about how to evaluate the result beyond the verifiability. There exists several technical routes, aiming at identify different properties of the submitted result, such as the accuracy, the integrity, or the fairness of the model. Thus a filter can be designed to distinguish between malicious and benign results, and this filter can be implemented with zero-knowledge proofs within a privacy-preserving manner.

D. Applications in Communication Networks

The relationship between ZKP-VML and communication networks is both close and multifaceted. In distributed systems, communication networks serve as the foundation for ZKP-VML, with the structure and efficiency of the network significantly impacting the overall performance of ZKP-VML. Consequently, improving communication efficiency and reducing communication overhead are among the key challenges for ZKP-VML. Moreover, the evolution of communication networks has introduced new tasks and scenarios for ZKP-VML. Semantic communication, a novel communication paradigm, aims to enhance communication efficiency, reduce transmission redundancy, and better support intelligent applications by conveying the meaning or intent of information rather than the traditional symbols or data themselves [49]. Compared to conventional communication systems, semantic communication emphasizes the understanding, reasoning, and contextual relevance of information [50]. While semantic communication offers significant support for machine learning, it also introduces new security challenges. Malicious adversaries could exploit semantic communication to generate misleading information, deceiving the receiver [51], or launch privacy attacks when participants share background knowledge [52]. This presents a new application scenario for ZKP-VML: *verifying the correctness of semantic communication without disclosing additional information* [53]. In real-world distributed machine learning scenarios, participants often lack mutual knowledge and trust, making them more vulnerable to malicious actors. Adversaries may infiltrate the system and disrupt the machine learning task through malicious actions. Therefore, another key application of ZKP-VML is to *provide identity authentication for participants while ensuring privacy protection* [54], confirming their ability to engage in the machine learning task. Furthermore, the results of these machine learning tasks can be leveraged to optimize resource allocation in communication networks [55], thereby enhancing communication efficiency and robustness [56].

IV. Overview of ZKP-VML Research Status

In this section, we provide a comprehensive overview of the existing ZKP-VML schemes. First of all, we provide the scope of the ZKP-VML schemes covered in this paper. We try to cover, as far as possible, all ZKP-VML schemes until June 2024, i.e., schemes that use zero-knowledge proofs to provide privacy and verifiability for machine learning computations. Our survey mainly relies on google scholar and dblp to search for relevant literature. In addition, we cover a small number of schemes that do not sufficiently address privacy, such as SafetyNets [57], to illustrate the challenges of ZKP-VML. It is worth noting that several types of schemes are excluded from our coverage due to their limited contribution to the ZKP-VML issue:

1) General zero-knowledge proof schemes, such as DIZK [58], zk-Authfeed [59], which have not been specifically optimized for ML computations, can be used in many other scenarios.
2) Schemes not directly using ZKPs, such as Drynx [60], Guo et al. [61], GOPA [62], which just include zero-knowledge proofs as a sub-module, rather than directly providing verifiability of ML computations. For example, Guo et al. utilize a ZKP protocol to prove the correctness of the encryption.
3) Schemes with limited confidence, such as Ju et al. [63], Ghaffaripour et al. [64], which lack sufficient security and experimental analysis to demonstrate the feasibility of their schemes.

A. Timeline

In order to visualize the research history of ZKP-VML, we have organized the existing schemes of each category into a timeline according to their initial appearance, as shown in Fig. 4 and 5, respectively. We follow the time when the scheme was first made public, rather than the time when it was formally published by a conference or journal.

In 2017, Ghodsi et al. first propose the concept and scheme Safetynets [57] for verifiable machine learning. Although Keuffer et al. [65] and Zhao et al. [66] apply zero-knowledge proofs to verify the correctness of machine learning computations, the assumption that the verifier owns both the model parameters and the training dataset led to the scenario with limited privacy. The privacy in VML is not considered till 2020, the presence of zkDT [67], where the model parameters are private to the prover.

QAP-based schemes are widely used for constructing ZKP-VML systems due to their applicability in transforming any additive or multiplicative computation into equivalent arithmetic circuits, simplifying the proof generation process for ML computations. However, QAP-based ZKP schemes can be costly under some situations. As a result, recent efforts have explored alternative ZKP backends for ZKP-VML, such as vector-oblivious linear evaluation-based (VOLE) and sumcheck-based schemes.

From 2021, researchers have also begun to focus on properties and computations other than the correctness in training and inference. To address the model poisoning issues in federated learning, Lycklama et al. [68] check the model integrity through ZKP to filter out malicious local models without compromising model privacy. To ensure that the decisions given by the model are fair, Shamsabadi et al. [69] check the model fairness through ZKP to ensure the model satisfies several given fairness definitions without compromising model privacy. In addition, ZKP-VML schemes have been proposed for addressing emerging machine learning computations, such as verifiable machine unlearning, model ownership, and large language model inference.

B. Properties

We classify the 55 ZKP-VML schemes into two categories according to their application scenarios. The first category consists of 33 schemes on the verifiability of inference and training processes in machine learning, as shown in Table III, and the second categories consists of 22 schemes on the verifiability of some emerging scenarios and computation processes in machine learning, as shown in Table IV. In this section we focus on analyzing how the properties of ZKP-VML are achieved.

Correctness and Security.

Building on the definition of ZKP-VML provided in the previous section, correctness refers to the ability of an honest prover to be verified correctly, while security refers to the detection of malicious attackers during the verification process. Considering attackers with varying capabilities and objectives in different scenarios, we introduce the concept of *partial security*. Partial security refers to situations where the scheme does not offer the level of security defined in the previous section to counter all possible threats, but remains secure within the specific context in which it is applied. This partial security arises from the adoption of certain optimization techniques, such as sampling. Both correctness and security are ensured through the generation of proofs of the computational process using existing ZKP schemes. Thus, under the completeness and soundness of their used ZKP schemes, honest proofs and results will pass the verification, while malicious proofs and results will not. However, some schemes improve the

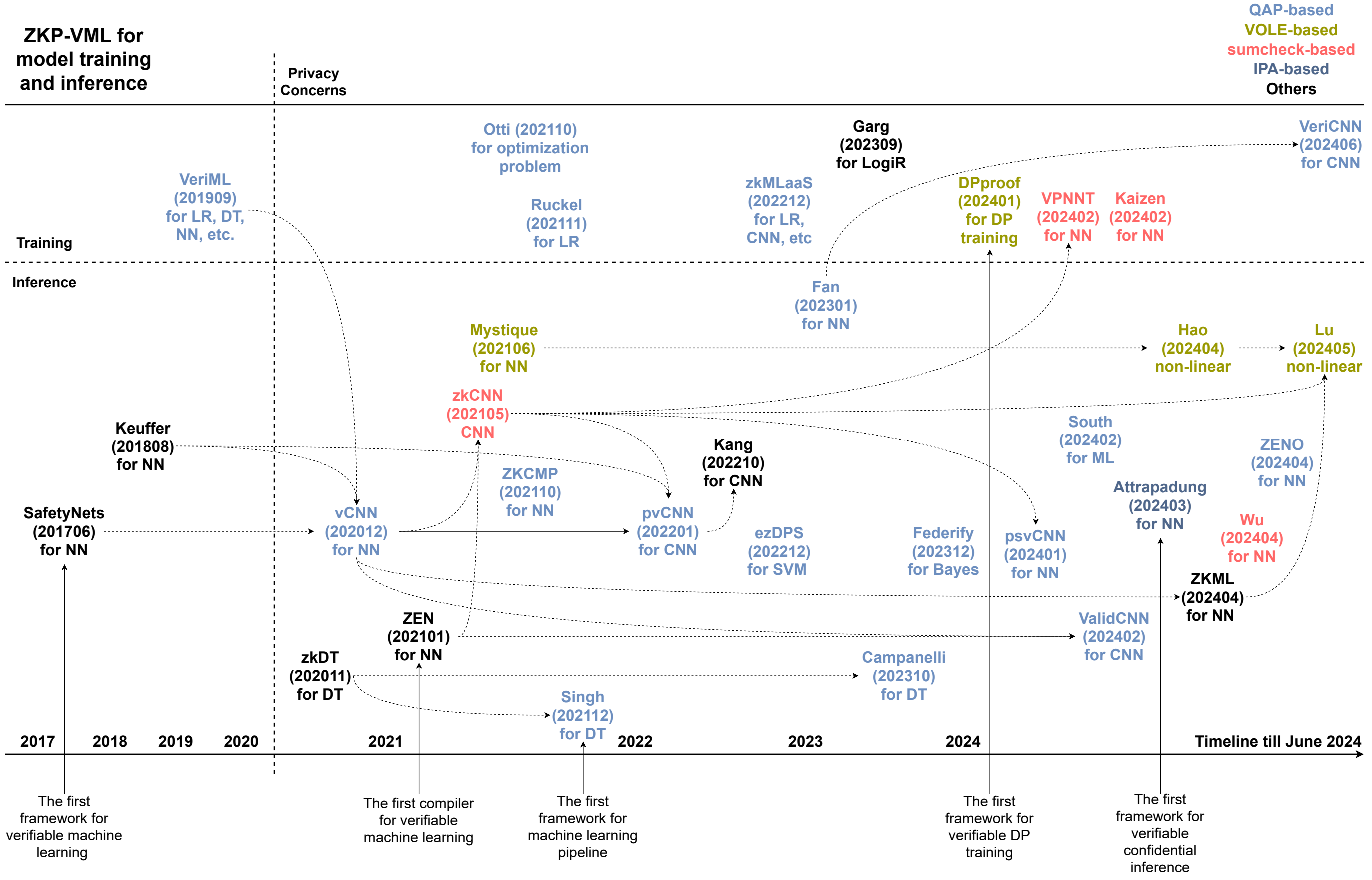


Fig. 4. Timeline of existing work in the field of ZKP-VML for training and inference. The dash line from scheme A to B indicates that B is shown to be more advanced than A under certain conditions by theoretical or experimental analysis. The solid line from schemes A to B indicates that B is inspired by A. For brevity, we omit the dash line from A to C, when A points to both B, C and B points to C.

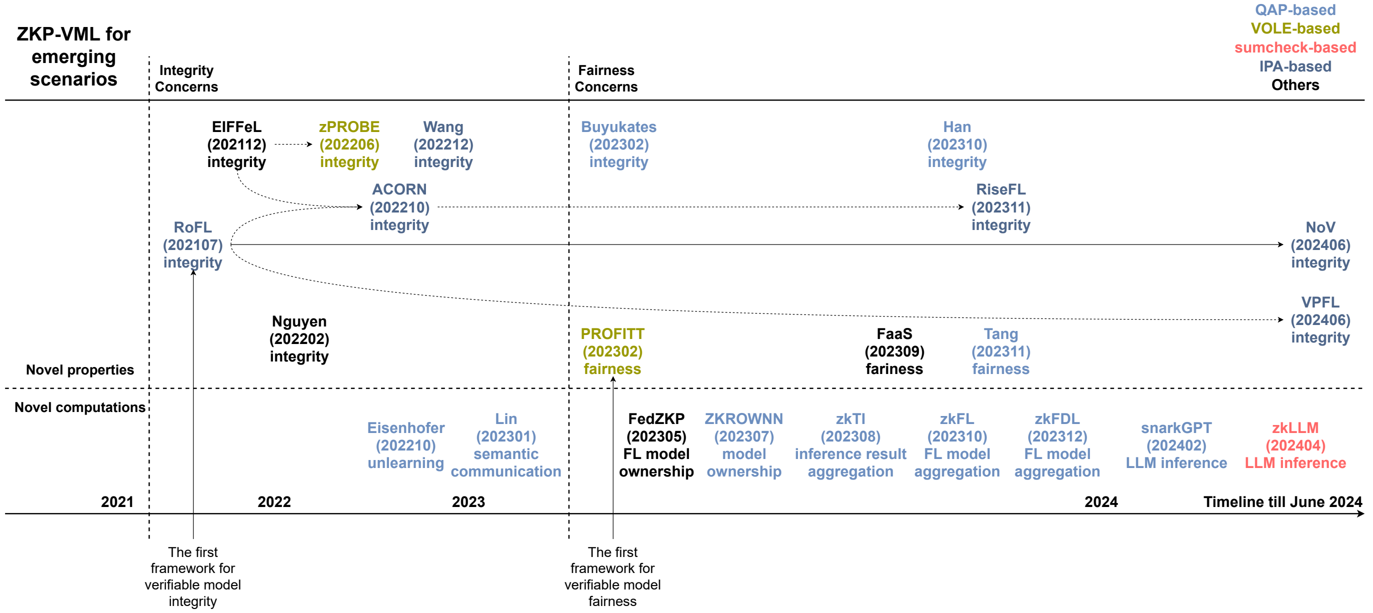

ZKP-VML for emerging scenarios
Integrity Concerns
Fairness Concerns
QAP-based
VOLE-based
sumcheck-based
IPA-based
Others
EIFFeL (202112) integrity
zPROBE (202206) integrity
Wang (202212) integrity
Buyukates (202302) integrity
Han (202310) integrity
ACORN (202210) integrity
RiseFL (202311) integrity
RoFL (202107) integrity
NoV (202406) integrity
VPFL (202406) integrity
Nguyen (202202) integrity
PROFITT (202302) fairness
FaaS (202309) fariness
Tang (202311) fairness
Novel properties
Novel computations
Eisenhofer (202210) unlearning
Lin (202301) semantic communication
FedZKP (202305) FL model ownership
ZKROWNN (202307) model ownership
zkTI (202308) inference result aggregation
zkFL (202310) FL model aggregation
zkFDL (202312) FL model aggregation
snarkGPT (202402) LLM inference
zkLLM (202404) LLM inference
2021
2022
2023
2024
Timeline till June 2024
The first framework for verifiable model integrity
The first framework for verifiable model fairness

Fig. 5. Timeline of existing work in the field of ZKP-VML for emerging scenarios and computations. The dash line from scheme A to B indicates that B is shown to be more advanced than A under certain conditions by theoretical or experimental analysis. The solid line from schemes A to B indicates that B is inspired by A. For brevity, we omit the dash line from A to C, when A points to both B, C and B points to C.

TABLE III
EXISTING WORK RELATED TO ZERO-KNOWLEDGE PROOF-BASED VERIFIABLE MACHINE LEARNING FOR TRAINING AND INFERENCE.

Scheme	ML Scenario	ZKP System	Properties			
			Correctness	Security	Privacy	Distinctness
ZEN [18]	verifiable inference for NN	R1CS	●	●	●	○
Mystique [20]	verifiable inference for NN	VOLE-based	●	●	●	○
Hao et al. [70]	verifiable inference for NN	VOLE-based	●	●	●	○
Lu et al. [71]	verifiable inference for NN	VOLE-based	●	●	●	○
Safetynets [57]	verifiable inference for NN	–	●	●	○	○
zkCNN [17]	verifiable inference for CNN	sumcheck-based	●	●	●	○
Wu et al. [72]	verifiable inference for ML with confidential computation	sumcheck-based	●	●	●	○
VPNNT [73]	verifiable training for NN	sumcheck-based	●	●	●	○
Kaizen [74]	verifiable training for NN	sumcheck-based	●	●	●	○
vCNN [16]	verifiable inference for CNN	QAP-based	●	●	●	○
pvCNN [75]	verifiable inference for CNN	QAP-based	●	●	●	○
ZENO [76]	verifiable inference for NN	QAP-based	●	●	●	○
Fan et al. [77]	verifiable inference for CNN	QAP-based	●	?	●	○
ValidCNN [78]	verifiable inference for CNN	QAP-based	●	?	●	○
VeriCNN [79]	verifiable training for CNN	QAP-based	●	?	●	○
psvCNN [80]	verifiable inference for CNN	QAP-based	●	?	●	○
Campanelli et al. [81]	verifiable inference for DT	QAP-based	●	●	●	○
Singh et al. [82]	verifiable ML pipeline and verifiable inference for DT	QAP-based	●	●	●	○
South et al. [83]	verifiable inference for ML	QAP-based	●	●	●	○
Federify [84]	verifiable FL training	QAP-based	●	●	●	○
ezDPS [85]	verifiable ML pipeline and verifiable inference for SVM	QAP-based	●	●	●	●
Kang et al. [86]	verifiable inference for CNN	Other (Plonk-based)	●	●	●	○
ZKML [87]	verifiable inference for NN	Other (Plonk-based)	●	●	●	○
Keuffer et al. [65]	verifiable inference for NN	QAP-based & sumcheck-based	●	●	◐	○
Attrapadung et al. [88]	verifiable inference for CNN with confidential computation	QAP-based & IPA-based	●	●	●	○
Garg et al. [89]	verifiable training for LogiR	QAP-based & MPCitH	●	●	●	○
DPproof [90]	verifiable DP training for ML	VOLE-based	●	●	●	○
zkDT [67]	verifiable inference for DT	Other (RS codes)	●	●	●	○
Otti [91]	verifiable training for LP, SDP, SGD	QAP-based	●	◐	●	○
Ruckel et al. [92]	verifiable training for LR	QAP-based	●	◐	●	○
VeriML [66]	verifiable training for ML	QAP-based	●	◐	◐	○
zkMLaaS [93]	verifiable training for ML	QAP-based	●	◐	◐	○
ZKCMP [94]	verifiable inference for NN	QAP-based / sumcheck-based	●	◐	●	○

[1] ○ denotes that the requirement is not satisfied;
[2] ◐ denotes that the requirement is partially satisfied;
[3] ● denotes that the requirement is fully satisfied;
[4] ? denotes that the satisfaction of this requirement is doubtful.
[5] the double dash line divide the scheme with full security and partial security according to the categorization.

TABLE IV
EXISTING WORK RELATED TO ZERO-KNOWLEDGE PROOF-BASED VERIFIABLE MACHINE LEARNING FOR EMERGING SCENARIOS AND COMPUTATIONS

Scheme	ML Scenario	ZKP System	Properties			
			Correctness	Security	Privacy	Distinctness
RoFL [68]	local model integrity for FL	IPA-based	◐	◐	●	◐
ACORN [95]	local model integrity for FL	IPA-based	◐	◐	●	◐
RiseFL [96]	local model integrity for FL	IPA-based	◐	◐	●	◐
NoV [97]	local model integrity for FL	IPA-based	◐	◐	●	◐
VPFL [98]	local model integrity for FL	IPA-based	◐	◐	●	◐
Wang et al. [99]	local model integrity for FL	IPA-based	◐	◐	●	◐
EIFFeL [100]	local model integrity for FL	Other (SNIP)	◐	◐	●	◐
zPROBE [101]	local model integrity for FL	VOLE-based	◐	◐	●	◐
Nguyen et al. [102]	local model integrity for FL	MPCitH	◐	◐	●	◐
Han et al. [103]	local model integrity for FL	QAP-based	◐	◐	◐	◐
Buyukates et al. [104]	local model integrity for FL	QAP-based	◐	?	◐	●
Tang et al. [105]	model fairness for ML	QAP-based	●	●	●	◐
FaaS [106]	model fairness for ML	Other (1-out-of-n ZKP)	●	○	●	◐
PROFITT [69]	model fairness for DT	VOLE-based	●	●	●	●
zkFL [107]	verifiable model aggregation for FL	QAP-based	●	○	◐	○
zkFDL [108]	verifiable model aggregation for FL	QAP-based	●	●	○	○
zkTI [109]	verifiable result aggregation for crowdsoucing inference	QAP-based	●	?	●	○
FedZKP [110]	verifiable model ownership for FL	Sigma protocol	●	●	◐	○
ZKROWNN [111]	verifiable privacy model ownership	QAP-based	◐	◐	●	○
Eisenhofer et al. [112]	verifiable machine unlearning	QAP-based	●	●	●	○
zkLLM [113]	verifiable LLM inference	sumcheck-based	●	●	●	○
snarkGPT [114]	verifiable LLM inference	QAP-based	●	●	●	○
Lin et al. [53]	verifiable semantic communication	QAP-based	●	●	●	○

[1] ○ denotes that the requirement is not satisfied;
[2] ◐ denotes that the requirement is partially satisfied;
[3] ● denotes that the requirement is fully satisfied;
[4] ? denotes that the satisfaction of this requirement is doubtful.

efficiency by applying some approximation and relaxing the original constraints, resulting in varying degrees of security loss. For example, Otti and Ruckel et al. employ approximated results with higher computational efficiency in place of the exact computational outcomes, demonstrating that the approximations are close to the actual results. This optimization means that the scheme cannot guarantee that the computed results originate from the given input and the prescribed computational process. It can only ensure that the results fall within an acceptable range. This optimization approach may render the scheme vulnerable to free-rider attacks. VeriML, zkMLaaS, and ZKCMP adopt the sampling strategy. By randomly sampling a small portion from the original set for the verification, the efficiency can be improved in several orders of magnitude. However, this strategy cannot detect single forged proof, where the malicious participant aiming at submitting a wrong model with a forged proof in only one round of training. Besides, several schemes (Fan [77], ValidCNN [78], VeriCNN [79], psvCNN [80], Buyukates et al. [104], zkTI [109]) compromise the security due to the incorrect implementation of ZKP schemes. Groth16 [30] requires a trusted setup, where the setup step need to be executed by a trusted third party to guarantee the backdoor generated in the setup to be discarded. However, in these schemes, the prover runs the setup step. With the backdoor generated, the prover can forge valid proofs for any result, threatening the security of the scheme. For schemes that introduce additional properties or computational types into ZKP-VML, their security warrants further discussion. Regarding the issue of model integrity, since the underlying principle relies on observations rather than formal proofs, both correct and incorrect samples have a very small probability of being misclassified, which leads to the *partial correctness*. This can result in a loss of both correctness and security. ZKROWNN [111] suffers from a similar problem, the watermark embedding and extraction method adopted, DeepSigns [115], has no security proof, but experimental verification shows that the method is functional. FaaS [106] and zkFL [107], on the other hand, are less secure due to their weak security assumptions. In FaaS, participants are assumed to be honest in providing inference results. In zkFL, a malicious server can provide incorrect aggregation results through replay attacks, but is not considered in the threat model.

Privacy. According to the definition provided in the previous section, privacy refers to the protection of the local data used by the performer in a machine learning task. Given the varying capabilities of adversaries, privacy is also extended to *partial privacy*. One of the key advantages of distributed machine learning is its ability to safeguard data privacy, allowing data to be used directly for model training without exposing it. Partial privacy means that during the execution of the machine learning task, the local data used is not directly exposed to the requester. However, considering attacks such as model reconstruction and membership inference, an attacker may still infer or reconstruct the local dataset used by the performer through the trained model, which could compromise the data privacy. In contrast, complete privacy refers to the situation in which, even if an attacker gains access to all the data exchanged during the machine learning task, no additional information about the private data can be obtained. Most schemes satisfy privacy by generating proofs using existing zero-knowledge proof protocol within a privacy-preserving ML framework. Zero-knowledge means that the verifier should learn nothing from the prover except for the validity of the statement being proved. In other words, any information the verifier gains by interacting with an honest prover could be learned independently without access to the prover. However, there are some schemes (Keuffer et al. [65], VeriML [66], zkMLaaS [93]) that do not take privacy as an important consideration. In their scenarios, the data required for the computation (dataset, model parameters) are provided by the verifier, while the only private data for the prover are the auxiliary inputs (e.g., hyperparameters), thus the privacy they offer is limited. For scenarios involving model aggregation, some schemes (Han et al. [103], Buyukates et al. [104], zkFL [107], zkTI [109]) require the client to provide the plaintext of its local model. Considering potential model reconstruction and membership inference attacks, these scheme also compromise privacy to some extent. zkFDL [108], on the other hand, directly treats all the inputs and outputs of the aggregation as public data. Compared to ZKROWNN [111], FedZKP [110] does not consider the privacy of the watermark itself.

Distinctness. Distinctness is a property used to assess the quality of the results produced by the performer. Through ZKPs, ZKP-VML schemes can verify whether the results provided by the performer are derived from the promised input through the specified computational process. However, machine learning task results often exhibit variations in quality, with models trained on lower-quality data typically contributing less than those trained on higher-quality data. Furthermore, adversaries may also degrade the model performance by low-quality but valid training. *Distinctness* refers to the ability of the scheme to provide additional evaluation metrics, beyond simple verification, to distinguish the quality of the results presented by the performer. The majority of existing work does not consider judgment criteria other than the correctness of the computation process, but this is one of the properties that distinguishes machine learning from other computations. Considerations of model integrity and fairness enrich the evaluation criteria. Furthermore, PROFITT [69] proves both the correctness and the fairness of the trained DT model. ezDPS [85] generates proof of accuracy besides the proof of correctness and Buyukates et al. [104] evaluates the contribution of local models besides the model integrity.

V. Technique Route Analysis

In this section, we aim to provide a systematic analysis of the existing ZKP-VML schemes by categorizing the schemes based on the specific problems they address and the technical routes they adopt. In addition, to clarify the contributions and possible shortcomings of the ZKP-VML work covered, each scheme is analyzed in detail. The specific categorization is as follows:

- Transforming machine learning to zero-knowledge proofs.
- Improving the efficiency of zero-knowledge proofs.
  - with full security .
  - with partial security.
- Introducing additional properties or computations to ZKP-VML.
  - by introducing integrity.
  - by introducing fairness.
  - by verifying model aggregation.
  - by verifying model ownership.
  - by verifying crowdsourcing inference.
  - by verifying machine unlearning.
  - by verifying LLMs.
  - by verifying Semantic Communication.

A. Transforming Machine Learning to Zero-Knowledge Proofs

This is the first step in ZKP-VML, which is also known as the arithmetization or quantization problem. Specifically, there are two main barriers, namely the *floating-point number* and *non-linear function*. Most machine learning computations involve floating-point numbers, while ZKPs typically handle integers in groups. Simply scaling and truncating floating-point numbers to integers for ZKP proofs can lead to significant accuracy loss in the ML computations. Thus the research question becomes *How to prove floating-point computation with zero-knowledge proof systems while minimizing the impact on precision and accuracy.* Non-linear activation functions provide neural networks with powerful representational ability, making them one of the most commonly used machine learning models, and able to handle most of the machine learning tasks. Since they are based on arithmetic circuits, most ZKP systems can only represent linear addition and multiplication operations. There are several simple ways to bridge this gap, such as using linear activation functions like $y = x^2$, or dividing the non-linear function into pieces so that it is linear within each piece. However, the former one will decrease the accuracy of the neural network, and the latter one will introduce additional proof costs. Thus the research question becomes *How to prove non-linear computation with zero-knowledge proof systems while minimizing the impact on accuracy and the additional cost.*

These two questions are faced by almost all ZKP-VML schemes, yet the solution for most of them is to utilize simple or existing schemes. For example, in VeriML [66], each input is constrained to have at most ℓ bits of decimal points, thus each input can be scaled to integer by simply multiplying 2^{ℓ}. The non-linear computation within the activation function is approximated by using the Remez method with polynomials. zkCNN [17] employs the approach proposed by Jacob et al. [116], which involves transforming a real number a into an integer q using a real number L and an integer Z such that $a = L(q - Z)$. zkCNN computes ReLU function using bit-decomposition, which increasing the proof cost.

Next, we will introduce several schemes specifically aiming at transforming ML computations to ZKPs. Compared to previous solutions, these schemes not only improve the performance and efficiency, but also consider a wider problem in ML and ZKPs, allowing future ZKP-VML schemes to simply integrate them as modules.

Feng et al. [18] present ZEN, a compiler to Rank-1 Constraint System (R1CS) constraints. ZEN consists of R1CS friendly quantization and stranded encoding of R1CS constraints. In this quantization algorithm, ZEN avoids the costly bit-decompositions caused by two complex operations, comparison and division by two optimization methods, sign-bit grouping and remainder-based verification. By stranded encoding, ZEN encodes several low-precision unsigned integers in quantized neural networks as finite field elements on an elliptic curve, reducing the number of constraints compared to the previous one-to-one encoding approach. Based on these optimization methods, ZEN can reduce the R1CS constraint by 5.43∼22.19× when generating verifiable neural network inferences compared to a general integer-arithmetic-only neural network baseline [116], which is adopted by zkCNN and other schemes.

Weng et al. [20] propose Mystique, an efficient conversion solution between ZKPs and ML, providing efficient conversion between arithmetic/boolean values, committed/authenticated values and fixed-point/floating-point values. Mystique is built on the VOLE-based ZKP protocol QuickSilver [117]. The conversion between arithmetic/boolean aims at switching circuit types to improve efficiency based on specific computations, especially for the non-linear functions in the ML scenario. In the MPC setting, this conversion can be addressed with extend doubly-authenticated bits (edaBits) [118] by converting authenticated shares between arithmetic and Boolean circuits. Mystique expands this method into zero-knowledge manner, constructing zk-edaBits. The second conversion between committed/authenticated values aims at allowing publicly committed data to be simply used in zero-knowledge proof scheme, providing more convenient privacy support. The third conversion between fixed-point/floating-point values aims at solving the inconsistency between floating-point numbers used in machine learning algorithms and fixed-point number used in cryptographic algorithms. To achieve this, Mystique designs a pair of encoding and decoding methods for supporting IEEE-754 single-precision floating-point number. In addition, the scheme is also optimized for matrix multiplication in terms of reducing proof cost. It is remarkable that the above solutions are integrated into Rosetta [119], a privacy-preserving framework based on TensorFlow [120], which means that developers can simply call these interfaces and ignore the cryptographic details involved.

Hao et al. [70] propose a VOLE-based ZKP framework for non-linear functions based on table lookup. Traditionally, non-linear functions to be proved have to be converted into Boolean circuit via bit decomposition techniques, causing $O(\log p)$ multiplication complexity in the prime field $\mathbf{F}_p$. In this framework, a public table that stores all input-output pairs of the non-linear function is pre-computed by both prover and verifier. To reduce the cost caused by large table size, ZKP for read-only memory access (ROM) [121] protocol is used to prove that a value read from a committed memory table. Besides, proofs for comparison and truncation are constructed as building blocks of the protocol for the transformation of non-linear operations including exponential, division and square root. Ultimately, non-linear functions commonly used in machine learning, such as ReLU and Softmax, can be built and proved on these operations. The floating-point number is handled using the conversion method proposed by Mystique [20]. Compared to Mystique [20], Hao's scheme achieves 50× to 179× runtime improvement.

Lu et al. [71] raise a ZKP framework for neural network based on efficient proofs for non-linear layers. Compared to previous works representing non-linear layers with the costly bit decomposition, they convert the non-linear relations into range and exponent relations, reducing the number of constraints in the circuit. Lu et al. designed efficient VOLE-based [122] range proof and lookup proof to constrain the primitive operations for non-linear layers, including max, sign, right shift, round and other operations. Thus the non-linear layers in CNN and TF including ReLU, MaxPooling, Softmax, GELU can be proved by these primitive constraints. The floating-point number is handled using the conversion method proposed by Gholamiet al. [123]. Experiments show that both the range proof and lookup proof outperform existing schemes [124], [125], [126]. Compared to existing work, such as Mystique [20], Hao et al. [70], zkCNN [17] and ZKML [87], this scheme performs better on different kinds of non-linear layers and convolutional and transformer neural networks, including the GPT-2 [127] with 117 million parameters.

Sum up. For the floating-point conversion problem, the general idea is about how to map floating-point numbers with high accuracy into a smaller range. ZEN compresses the mapping range by strand encoding, while Mystique handles IEEE-754 floating-point numbers, which improves the ease of use of the solution. For non-linear operation problems, a common optimization idea is to describe the non-linear operation with simple operations with smaller proof cost, such as range proofs, table lookup proofs, etc. Mystique optimizes the conversion between arithmetic circuits and Boolean circuits to reduce the proof cost of bit-decomposition. In addition, more schemes utilize the VOLE-based ZKP system [117], [122] due to their efficiency and suitability for constructing the required proofs, such as table lookup and range proofs.

B. Improving the Efficiency of Zero-Knowledge Proofs

Considering the additional computational cost introduced by ZKP, in order to make the ZKP-VML scheme more practical, many works have focused on how to improve the efficiency and reduce the cost of proofs. We categorize these schemes into two types, *improvements with full security* and *improvements with partial security*. The former one does not harm the soundness of the original ZKP system. The latter one adds a non-negligible probability for the adversary, providing additional possibilities for invalid proofs to pass the verification. Furthermore, to provide a more systematic analysis from a technical perspective, we will conduct a detailed analysis and presentation of the existing schemes based on the types of ZKPs employed and the specific application scenarios.

1) With Full Security: **For schemes adopting the sumcheck protocol,** Ghodsi et al. [57] raise SafetyNets, an interactive proof protocol for verifiable execution of a class of neural networks. It is worth mentioning that SafetyNets is the first scheme on VML, although the concept of zero knowledge is not involved in this scheme. In SafetyNets, both the inputs and the model are given by the verifier, so there is no privacy issue. In SafetyNets, the efficient verification of neural network computation is achieved by designing GKR [128] protocols for matrix multiplication. By randomly selecting a point $C_{i,j}$ in the matrix C and representing its computation process in the form of sumcheck protocol (i.e. $C_{i,j} = \Sigma_{k\in\{0,1,\ldots,n\}} A[i,k]B[k,j]$), a protocol with computational complexity $O(n^2)$ for both the prover and verifier can be obtained But unfortunately, for activation functions and pooling layers, SafetyNets can only support specific quadratic activation functions and sum pooling, making it not practical enough. However, it still provides an idea to efficient verification by constructing GKR protocol for specific computations.

Liu et al. [17] propose zkCNN, a zero knowledge proof scheme for convolutional neural networks based on GKR protocol [128]. It is worth mentioning that GKR protocol is a non zero-knowledge interactive protocol, but can be converted to zero-knowledge by using a zero-knowledge polynomial commitment [35] and non-interactive by the Fiat-Shamir heuristic [31]. zkCNN improves the efficiency by optimizing the proof cost of convolutional computation in CNN. Liu proposed a new GKR protocol for checking the computation of the fast fourier transform (FFT). FFT can be used to improve the computational efficiency of the convolution, thus verifying the convolutional computation process using the FFT is faster than directly verifying the original one. Further, by verifying the original convolution in the form of $\overline{U} = \overline{X} * \overline{W} = IFFT(FFT(\overline{X}) \odot FFT(\overline{W}))$ with sumcheck protocol for FFT and Inverse FFT (IFFT), the overall prover time can be reduced to surprisingly $O(n^2)$, which is even faster than computing the convolution, with $O(\log^2 n)$ proof size and verifier time. In addition, to improve the performance of GKR on CNN, they also proposed several improvements and generalizations. According to the experiment, compared to vCNN [16] and ZEN [18], zkCNN is 11.2× faster and 213× faster on LeNet, respectively.

Wu et al. [72] introduce a confidential and verifiable delegation scheme based on ZKP and MPC. In this scheme, the verifier has all the machine learning models and data samples, and these data are private to the verifier. By using a secret sharing scheme, the verifier can distribute shares of these data to several prover, for their confidential computation through a MPC protocol. To prove the correctness of the computation on the server, Wu proposed GKR protocols tailored for MPC matrix multiplication over shared values, which improves its efficiency on neural networks. Compared to another ZKP scheme for distributed secrets [129], the performance of Wu's protocol is 88× faster on 3-layer MLP network and 74.8× faster on LeNet.

Duan et al. [73] propose VPNNT, a ZKP framework for verifiable neural network training based on sumcheck protocol. Duan et al. introduce custom gates for several binary and unary matrix operations for neural networks. The non-linear layers are represented by R1CS and further converted into custom gates as well. Thus the computation of the training process can be expressed in the matrix form and proved by these ZKP building blocks for matrix operations. To efficiently verifying the claims containing the same structure, Duan et al. combine multiple claims of the same matrix using the multi-linear extension. By introducing randomness into these claims, the prover can prove multiple claims at once within an acceptable soundness error. Compared to ZKP system Virgo [130] and CNN-specialized ZKP system zkCNN [17], the prover time of VPNNT is 1.16× to 158.4 × faster.

Abbaszadeh et al. [74] present Kaizen, a framework for verifiable training on deep neural network. Traditionally, proofs are required for each round of training, where the proof size is linear to the number of training rounds, leading to the large proof size. To overcome this, Kaizen leverages recursive proof composition, also refers to incrementally verifiable computation (IVC), where the proof for each round training arguing for the current correct computation and a valid proof arguing for the previous correct computation. Thus the proof size will be independent to the training rounds. To efficiently handle the training process, Abbaszadeh proposed sumcheck proofs for gradient descent, where both the linear and non-linear operations are processed following existing methods. To build the IVC on the proofs of gradient descent, an aggregation scheme for multivariate polynomial commitments is designed to reduce the cost on verifying the polynomial commitments. To further improve the performance, several sumcheck-specific optimizations are also deployed. Compared to other IVC schemes including Fractal [131], Halo [132] and Nova [133], Kaizen achieves 43× faster prover time and 224× less prover memory usage.

Sum up. The sumcheck protocol is an interactive proof protocol in which, during the final round of verification, the verifier accesses the value of the target polynomial at the challenge point. As such, the protocol is not inherently zero-knowledge. However, by employing a commitment scheme to protect the target polynomial, additional information leakage can be prevented, thus achieving zero-knowledge properties. Therefore, when using the sumcheck protocol as a proof protocol, it is crucial to ensure that the scheme maintains the desired zero-knowledge guarantees. Such protocols primarily focus on how to transform the target computation into a form that is verifiable via the sumcheck protocol, and then efficiently verify the target computation.

For schemes adopting the QAP-based protocol, Lee et al. [16] introduce vCNN, a framework for verifiable convolutional neural network based on zk-SNARKs. Lee extends the original quadratic arithmetic program (QAP) to quadratic polynomial program (QPP), and constructs the QPP-based zk-SNARKs to prove the convolutional computation. QPP-based zk-SNARKs involve assigning a polynomial value to each wire, allowing for the expression of 1-D convolution computation using a single multi-gate in the arithmetic circuit. As a result, the number of multi-gates required for proving convolution between two matrices X and W of size $n \times n$ and $w \times w$ is reduced from $O(n^2w^2)$ to $O(n^2 + w^2)$. For the pooling and activation layers, vCNN still retains the QAP-based zk-SNARK. And the proofs about the continuity between the two proofs are generated by the commit-and-prove SNARK (CP-SNARK) to connect the adjacent layers. In this way, the QAP-based and QPP-based zk-SNARKs proves the correctness of the intra-layer computation, while the CP-SNARK proves the continuity of the computation of each layer, and finally the vCNN proves the correctness of the entire convolutional neural network computation. Theoretically, vCNN has a certain improvement in proving time compared to schemes such as SafetyNets [57], VeriML [66], and Embedded proof [65]. Meanwhile, experimental results show that vCNN is 18000 times more efficient on the VGG16 model compared to the original zero-knowledge proof scheme Groth16 [30], the latter of which takes more than a decade to generate a proof.

Inspired by vCNN, Weng et al. [75] further propose pvCNN, also a framework for verifiable convolutional neural network. The main innovation of this paper is the circuit representation method, which proposes a zk-SNARKs scheme based on quadratic matrix program (QMP) based on the QPP-based method proposed by vCNN. By further expanding the representation capability of the wire in the circuit from array to matrix, pvCNN reduces the size of the circuit by reducing the number of multiplication gates in convolutional operation, thereby improving efficiency. In addition, since the neural network is layered, multiple proofs for different inputs of the same CNN layer can be aggregated into one proof with SnarkPack [134]. vCNN split the neural network into prior and later parts. The prior net is locally computed with HE to protect the data privacy, while the later net is delegated to be computed in plaintext. Thus the computational overhead of fully homomorphic encryption and privacy requirements can be balanced. In terms of performance, the scheme is compared theoretically with SafetyNets, zkCNNs, vCNNs, etc., and outperforms the above schemes in terms of proving time for convolution. And the experimental results also show that QMP-based zk-SNARKs has higher efficiency than the QAP-based for convolution operations.

Feng et al. [76] design ZENO, an optimizer for zk-SNARK-based verifiable neural network inference. Traditionally, zk-SNARK protocols are designed for scalar type, which makes it complicated to implement zk-SNARK to NN with intensive tensor computations. Thus Feng et al. proposed ZENO circuit as an efficient intermediate representation between NN layers and constraints. ZENO circuit can minimize the number of addition gates for dot product by aggregating these addition gate into one, which reduces the computational complexity from $O(n^2)$ to $O(n)$. Based on this design, ZENO circuit for fully connected, convolution and pooling layers can be extended. Another key insight is that the privacy of data in zk-SNARK can be further exploited to reduce the number of constraints. For example, the product of a public tensor and a private tensor of length n requires $n + 1$ constraints. Whereas if the public tensor is considered as the coefficients in a linear combination, only 1 constraint is required. Such either private feature or private weights situation is common in zero-knowledge proof-based machine learning. Further, several optimization methods based on parallel workload and computation reuse are also proposed. Compared to several existing zk-SNARK framework including Arkworks [135], Bellman [136] and Ginger [137], ZENO achieves up to 8.5× end-to-end speedup for NN. ZENO can reduce the proof time for VGG16 from 6 minutes to 48 seconds.

Fan et al. [77] also focus on convolutional computation, converting the computations therein into a simple arithmetic expression in matrix form. For the convolution layer, the 3D convolution is represented using a 2D matrix by the im2col method [138], which in turn transforms the convolution calculation into the equivalent matrix multiplication. The pooling layer also uses the im2col method, which reduces the 3D data to a 2D representation. The activation functions ReLU and Softmax are also expressed in the form of matrix multiplication. Where ReLU is represented as the input matrix multiplied by a matrix with elements 0 or 1, and Softmax is represented as the output matrix multiplied by a vector of summations of exponents. Further, all the matrix computation in the convolutional neural network can be optimized by the Freivalds's algorithm [139], which greatly increases the efficiency of setup and proof generation.

Beside of this, Fan et al. also design three schemes for verifiable CNN training or inference, namely ValidCNN [78], veriCNN [79] and psvCNN [80]. Similar to their previous work, these three schemes leverage im2col to convert the convolution into matrix multiplication, represent different layers as matrix multiplication and apply Freivalds algorithm to reduce the proof overhead from all these matrix multiplications. Specifically, there are some differences between these four schemes. veriCNN adopts Winograd [140] to accelerate the matrix multiplication after im2col conversion and a matrix multiplication verification scheme [141] based on Freivalds. psvCNN faces a scenario where the server is required to prove the correctness of the prediction performed on a server cluster. The computation between each convolutional kernel is independent, thus psvCNN can split the original CNN into independent tasks for parallel execution and proof generation among the cluster. However, these four works suffer from a same significant security vulnerability, where the prover runs the setup algorithm of its zk-SNARKs to generate the common reference string for the proof generation and verification. It is well known that zk-SNARK schemes, such as Groth16 [30], which is widely adopted by Fan et al., requires a trusted setup. Because the setup algorithm outputs a trapdoor which can be used to forge proofs for arbitrary statement. Thus the setup should be run by another party than the prover, and the trapdoor should be discard once generated.

Campanelli et al. [81] focus on table lookup proofs, extending the vector lookups to matrix lookups, based on which a zero-knowledge decision tree accuracy scheme is proposed. Compared to previous work of lookup proofs cached quotients (cq) [142], Campanelli proposed cq$^+$, which not only brings zero-knowledge to cq, but even introduces no additional prover computation with shorter proof size. With the KZG commitments [143] for matrix, cq$^+$ extends the vector lookups to matrix lookups. By encoding decision tree models as matrices, the evaluation of the DT is represented as locating the row containing the correct leaf and the input vector matches all the constraints, which can be proved with the matrix lookup proofs. For zero-knowledge DT accuracy, this scheme achieves the improvement that the prover time complexity is independent of the size of the decision tree. Compared to zkDT [67], this scheme reduces the prover time by one order of magnitude and the verifier time by two order of magnitude. The adopted ZKP backend is Lunar [144].

Singh et al. [82] present a zk-SNARK-based verifiable scheme for decentralized AI pipelines, containing a privacy-preserving verification scheme for decision tree inference. The distributed AI pipeline assigns the different steps of data collating, model training, and using the model to make predictions to independent actors such as data owner, model owner, and model consumer. Compared with zkDT, this scheme avoids costly hashing operations by changing the way of representing and committing to the decision tree. And it further reduces the number of multiplication gates in the arithmetic circuit by improving the access method in the arithmetic circuit to reduce the access cost of different operations in the prediction path verification. Which also reduces the cost for proving data operations in AI pipeline, such as the inner-join or filter during the data curation. For the decision tree inference task, the complexity of the circuit generated by this scheme is ten times better than that of zkDT [67].

South et al. [83] proposed a framework for guaranteed inference, which not only verifies the model's performance but also ensures that the inference results are generated by the specified model. This framework comprises two key components. First, for a given model, the prover is required to generate proofs of the inference results on a test dataset to demonstrate the model's performance. Second, when a user queries the model for inference on a given sample, the prover must generate a proof of the inference process. The model is bound to the hash of its weights, ensuring that the user can verify that a high-performance model is used for the inference task. To address the additional computational overhead associated with proof generation, the framework adopts a ”trust but verify” strategy, whereby the prover submits the proofs after a delay following the inference result. However, this strategy does not sufficiently mitigate the computational cost. To prevent the potential leakage of the test dataset during the model performance verification and simplify the verification, the prover can aggregate all inference proofs into a single proof. To do so, the prover can design a specific circuit, which verifies all the proofs on the test dataset and output some designed metrics, such as different accuracy measures.

Keshavarzkalhori et al. [84] introduced Federify, an on-chain verifiable federated learning framework. In this framework, participants can encrypt and generate proofs for their locally trained models using HE and ZKPs, respectively, while servers collaboratively decrypt the aggregated global model. The verification and aggregation of local models are carried out through a smart contract deployed on the blockchain. However, Federify is constrained by its reliance on the naive Bayes classifier, the large size of proof files, significant gas costs, and the lack of sufficient optimization, which together limit the scalability and practical utility of the framework.

Wang et al. [85] introduce ezDPS, a zero-knowledge ML pipeline with high accuracy of inference under Spartan [145]. ezDPS is the first considering to generate proofs for the ML pipeline scenario including Discrete Wavelet Transformation (DWT) [146] for preprocessing, Principal Components Analysis (PCA) [147] for feature extraction and Support Vector Machines (SVM) [148] for classification. To efficiently represent these algorithms as circuits, ezDPS applies 2 previously proposed gadgets and 4 specially designed gadgets including exponent, greaterthan, maximum/minimum and absolute gadget. Further, ezDPS improves the efficiency of proof generation for DWT and PCA through random linear combination. To ensure that the inference is reliable, a zero-knowledge proof of accuracy (zkPoA) is generate to guarantee the high performance of the given SVM, arguing whose accuracy is at least δ. Experiments are carried on UCR-ECG, LFW, and CIFAR-100 dataset. Compared to the Spartan as the baseline, ezDPS achieves up to 1842× faster proving time.

Sum up. QAP-based ZKP protocols may have potential security vulnerabilities, with improper implementations allowing adversaries to easily forge proofs. Some QAP-based protocols are built upon the Common Reference String (CRS) model, and a subset of these rely on a trusted setup, such as the widely used Groth16 [30] protocol. In these protocols, the setup phase not only generates the proving and verification keys but also produces toxic waste, which can be exploited to forge proofs. Consequently, the setup must be performed by a trusted party, ensuring that this toxic waste is discarded properly. Therefore, the design of such schemes should prevent the prover from conducting the trusted setup alone, as a malicious prover could potentially exploit the toxic waste to forge proofs. QAP-based protocols primarily focus on efficiently representing specific computational processes using arithmetic circuit structures. For instance, vCNN extends the wire representation in circuits to directly express polynomials, thereby reducing the number of multiplication gates. Similarly, the ezDPS protocol designs gadgets to combine and represent specific computations.

For schemes adopting the Plonk-based protocol, Kang et al. [86] tailor the ImageNet-scale model MobileNet v2 [149] with the zero-knowledge proof scheme halo2 [150] to obtain an ImageNet-scale zk-SNARK circuit. Verifying the division operation in the circuit is expensive, to solve this problem, two optimizations are applied to the Plonkish arithmetization [151] used by halo2 [150]. In linear layers (convolutional layers, residual connection layers, fully-connected layers), two custom gates are designed to reduce the cost. As for the non-linear layers (ReLU, softmax), the lookup argument is applied to reduce the representation cost of division. Compared with existing schemes including ZEN, vCNN, pvCNN, zkCNN, this scheme improves the proving time on MobileNet by at least ten times.

Chen et al. [87] propose ZKML, a verifiable inference framework for realistic machine learning models, based on the halo2 [150] proving system. To support different operations in ML, ZKML designed several gadgets in four categories, namely shape, arithmetic, pointwise non-linear and specialized operations. These gadgets can be used to form 43 ML layers, which can be divided into linear, arithmetic, activation layer and softmax. Also, an optimizer is also designed for converting the ML model into an optimized circuit layout. The optimizer will generate various layouts and select the optimized one by cost estimation. The experiment shows that ZKML can support inference on a distilled GPT-2 [152] with 1 TB of RAM within 66 minutes.

Sum up. As a relatively novel zero-knowledge proof protocol, Halo2 [150] has gained increasing attention due to its strong computational performance. As a result, Halo2 is often employed in scenarios with heavy computational demands, such as tasks involving models with larger parameters.

For schemes adopting hybrid ZKP protocols, Keuffer et al. [65] propose a hybrid embedded proof scheme for verifiable computation combining GKR protocol and zk-SNARKs. The goal is to achieve a balance between efficiency and usability by first processing individual functions with efficient verifiable computation schemes (EVCs, such as the GKR protocol, which are more efficient but can only handle relatively simple computations), and then processing sequences of functions with general purpose verifiable computation schemes (GVCs, such as zk-SNARKs, which are less efficient but can handle more kinds of computations). A neural network can be thought of as consisting of several functions, where the correctness of the computation of each function is guaranteed by the GKR protocol, which means that several proofs of the GKR protocol are generated. And zk-SNARKs not only prove the continuity of inputs and outputs between functions, but also verify the proofs generated by each function. Eventually, it generates a total proof. By verifying the total proof, the verifier can know whether all the specific computations proved by GKR protocol pass or not. Experimentally, it is shown that the embedded proof scheme has twice as good proving time on two-layer neural networks compared to the scheme using only zk-SNARKs. However, although the scheme claims that it protects the privacy of the provers' inputs, this is irrelevant in the case where the functions and inputs are known to the verifier.

Attrapadung et al. [88] raise a confidential and verifiable CNN inference framework that protects the privacy of both the input data and models from different parties. There are two parties, namely P_1 and P_2, where P_1 holding the private model parameters and P_2 holding the private data. They hope to jointly compute an output and generate corresponding proof arguing for its correctness. Arithemetic black-box abstraction (ABB) [153] allows parties to perform field arithmetic without explicitly knowing the values, which is leveraged to constructing the confidential CNN inference, and realized through SPDZ [154], a MPC protocol based on HE. A new collaborative zk-SNARK based on Bulletproofs is constructed following the notion of collaborative zk-SNARK [129], allowing parties to jointly generate Bulletproofs without revealing their secrets for arithmetic circuits. Although this scheme achieves the input confidentiality, there is still space for efficiency improvement, for its prover time doubles that of the plain Groth16 [30].

Garg et al. [89] propose an efficient ZKP protocol for logistic regression (LogiR) training. In this work, Grag combined zk-SNARKs with MPC-in-the-head (MPCinH) to balance the overhead in proof generation time and proof size. To prove the correctness of the training without revealing the input dataset and output model, a MPCinH protocol is employed for the LogiR training. However, the MPCinH protocol needs a trusted pre-processing phase to generate correlated randomness and secret shares of data for the training, thus zk-SNARK is employed to prove the honest execution. Besides, some checks of views are also implemented by verifying the corresponding zk-SNARK proofs. Customized MPC protocol and packed secret sharing scheme can further reduce the communication overhead. Finally, the proof size is reduced to $O(N)$, even though the total computation is $O(DN)$, where N is the size of the dataset and D is the size of the sample.

Sum up. The hybrid use of different ZKP protocols arises from the fact that various protocols exhibit different performance when applied to different types of computations. For complex computations, specialized ZKP protocols are often insufficient, while more general-purpose protocols can efficiently represent the computation, improving proof efficiency. In contrast, for simpler computations, general-purpose protocols may introduce unnecessary computational overhead, while specialized protocols can generate proofs more efficiently, enhancing computational performance. When employing multiple ZKP protocols, it is essential to ensure consistency between them to prevent attackers from using incoherent data to forge separate proofs for different protocols.

Shamsabadi et al. [90] present Confidential-DPproof, a ZKP framework for differentially private training, especially for the DP-SGD algorithm [155]. Confidential-DPproof is the first framework for ZKP-based verifiable DP training including the privacy budget ϵ. The training dataset is committed using the information-theoretic message authentication codes (IT-MACs) [156]. Considering the randomness involved in the training process, a Σ-protocol is utilized to generate the unbiased randomness seed. Each computation included in the DP-SGD is represented in circuits and encoded and proved by the EMP toolkit [157]. Experiment shows that it takes 100 hours for Confidential-DPproof to train a model achieving 91% accuracy on CIFAR-10 in a DP manner.

Zhang et al. [67] propose zkDT, a verifiable zero-knowledge proof scheme for decision tree prediction and accuracy. For a decision tree model, to verify the output, a prior commitment to the decision tree by the prover is required, and then the prover proves the validity of the prediction path to the verifier. Whereas converting each comparison on the prediction path into an arithmetic circuit is very expensive, to improve efficiency, the authors reduce the generation cost of proofs by inserting designed sibling nodes on the prediction paths. And the proof is generated with Aurora [158], unlike most other ZKP-VML schemes.

2) With Partial Security: Angel et al. [91] introduce Otti, a compiler for zkSNARKs that focuses on optimization problems including linear programming (LP), semi-definite programming (SDP), and a broad class of stochastic gradient descent (SGD) instances, which are often used in the training of neural networks. Otti can compile programs written in a subset of C that describe optimization problems into rank-1 constraint satisfiability (R1CS). Otti's idea is to avoid proving the solving process by proving the optimality of the solution, constructing a non-deterministic checker from the certificate of optimality, and then compiling this checker into R1CS. For the LP and SDP problems, Otti proves optimality by using the properties of the primal and dual solutions to the optimization problem. For the SGD problem, Otti proves optimality by showing that the gradient at the solution has certain properties. With the Spartan proof system, Otti can prove the optimality of the solution in zero-knowledge within 100 ms, which is four orders of magnitude faster than existing methods.

Ruckel et al. [92] design a scheme for zero-knowledge verification of linear regression. The optimization idea is to use an approximate solution that is less computationally expensive and prove how close that approximate solution is to the true solution with some range proofs, which is derived from DIZK [58]. Computing the inverse of a matrix is relative expensive. To improve the efficiency of the inverse matrix computation in the model update, the author ensures the correctness of the computation by proving the proximity of the computed inverse matrix to the true inverse matrix through a series of range constraints on the norm. Besides, differential privacy is also applied to protect the updated local weights in order to achieve stronger privacy. Although the scheme is relatively complete in terms of process, the limitations of the optimization method lead to the narrow application scenarios.

Sum up. These schemes compromise security by relying on approximate results. While they ensure that the outcomes remain within an acceptable range, they cannot verify the correctness of the computational process, leaving them susceptible to free-rider attacks. Consequently, when using approximate results to reduce computational burden, it is essential to assess the security requirements of the application scenario.

Zhao et al. [66] introduce VeriML, a framework for integrity and fairness in outsourced machine learning. Since several iterations of the same process are performed during training, VeriML chooses to use several iteration rounds of the training process as a challenge for verification, thus reducing the cost of proving. By storing the input and output of some iterations in advance during the training process and committing to them, the provers can retrieve to the specified iterations and generate proofs of their computational processes as requested by the verifier. VeriML supports a total of six machine learning models, including linear regression, support vector machines, and neural networks. For each kind of model, VeriML also proposes some small optimizations for improving the proving efficiency. In addition, VeriML uses an on-chain protocol to protect the privacy of the trained models for fair trading of the models.

Huang et al. [93] raise zkMLaaS, a verifiable scheme for machine learning as a service (MLaaS), which focuses on handling volume issue of input data with the random sampling idea. Proof costs are proportionally reduced by randomly selecting and challenging committed epochs and iterations, which is similar to VeriML. As for the convolution operation in CNNs, the optimization idea is similar to Fan et al. The im2col algorithm [159] is applied to convert the convolution into matrix multiplication and the Freivalds' algorithm [160] is further utilized to reduce the overhead of matrix multiplication. Compared to simply using zk-SNARKs directly, zkMLaaS saves approximately 273× the proof generation runtime.

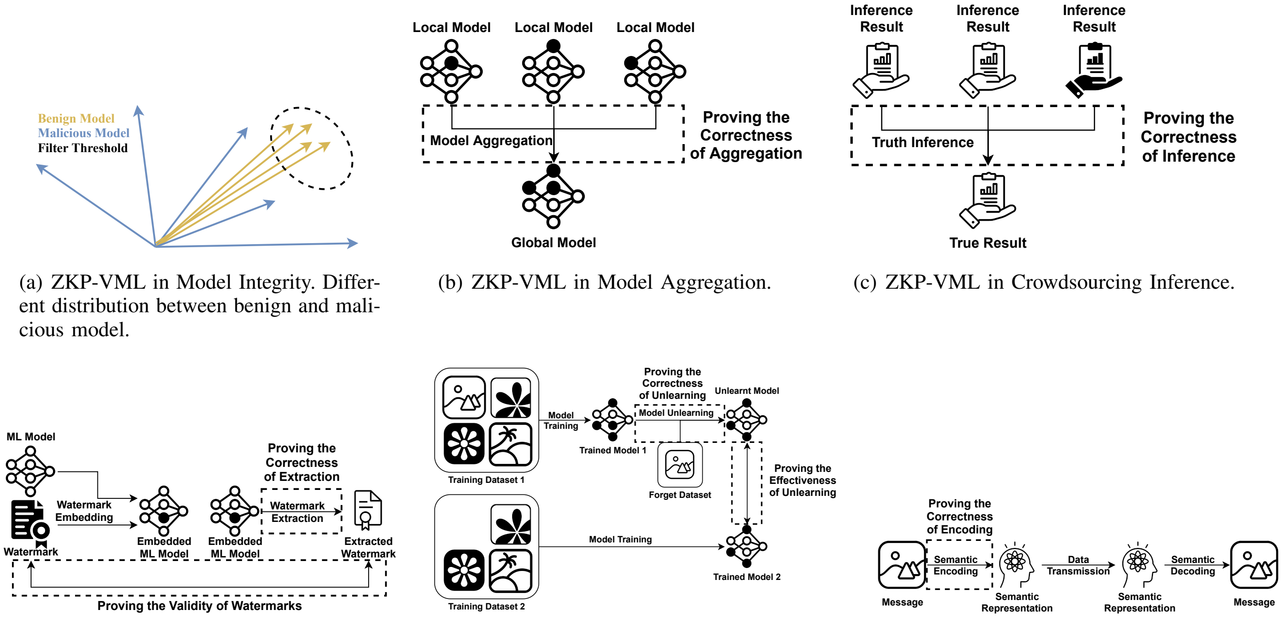
Zhou et al. [94] present a zero knowledge contingent model payments (ZKCMP) for training neural networks. ZKCMP aims at paying for qualified trained models. The seller who possesses the trained model sends the encrypted model to the buyer. Buyer sends a test dataset with a threshold for the accuracy, on which seller evaluates their model and generate zero-knowledge proofs, arguing that the accuracy exceeds the given threshold. After the buyer confirms the proof is verified, the seller can redeem the payment by sending the valid decryption key. The above zero-knowledge proofs can be generated with either zk-SNARKs [161] or Libra [162]. To reduce the proof cost in zk-SNARKs, ZKCMP sampling a portion of the constraints for proof generation and verification. To convert the computation of inference and encryption into Libra, several sub-circuits are constructed and combined. To guarantee the privacy of the intermediate value between sub-circuits, zkVPD [163] is adopted for connection. Experiments show that for 10,000 images, zk-SNARKs' proof time is nearly 6000s, while Libra's proof time is nearly 1,000 seconds, although Libra's proof is larger.

Sum up. Similarly, schemes that rely on sampling for verification also compromise security. Although they can ensure the correctness of most proofs, they cannot detect a single forged proof. This makes these schemes susceptible to single-point attacks. Consequently, when using sampling-based verification, careful attention should be given to the security implications within the specific application scenario and possible threats.

C. Introducing Additional Properties or Computations into ZKP-VML

To clarify how ZKP-VML ensures verifiability in these scenarios and its importance, we present ZKP-VML in various scenarios for additional verifiability in Fig. 6.

1) Integrity: Integrity refers to whether the model is well-formed or not. Usually the integrity of the model can be compromised by model poisoning attacks. For example, in



(a) ZKP-VML in Model Integrity. Different distribution between benign and malicious model.

(b) ZKP-VML in Model Aggregation.

(c) ZKP-VML in Crowdsourcing Inference.

(d) ZKP-VML in Model Ownership.

(e) ZKP-VML in Machine Unlearning.

(f) ZKP-VML in Semantic Communication.

Fig. 6. ZKP-VML in Different Scenarios for Additional Verifiability.

federated learning, an attacker may submit a malicious model generated from a poisoning attack to the aggregation, to degrade the performance of the global model. A common observation [164] is that, if forming the parameters of the model into a vector, then benign models follow a similar distribution of magnitude and direction, while malicious models not, as shown in Fig. 6(a). The malicious models can be identified by abnormal magnitude or direction, thus the integrity of models can be ensured. The checking process of the model magnitude or direction can be achieved through zero-knowledge proofs to ensure the privacy of the model and corresponding data. However, since the checking method is just based on a common observation and there is no proof or theorem to support it, so most of these schemes do not possess provable correctness and security. The goal of such ZKP-VML schemes is to detect malicious models while protecting the privacy of the submitted models. This type of schemes usually requires a corresponding secure aggregation protocols for complete model privacy protection.

Lycklama et al. [68] propose RoFL, a secure federated learning framework focusing on the robustness. RoFL applies adaptive L_2 and L_∞ norm to bound the model updates, aiming at identifying malicious updates and ensuring the robustness of the federated learning. Provers are required to generate range proofs with Bulletproof [124], arguing that their submitted models meet the norm bound. In order to reduce the proof burden, RoFL optimizes the proofs for L_2 and L_∞ norms, respectively. For the L_∞ norm, the verifier randomly selects and verifies a subset of all parameters, for it only takes one failure to identify the attack. For the L_2 norm, RoFL applies random subspace learning [165] to reduce the size of model parameters and further the proofs. RoFL further improves the proof efficiency by applying an optimistic continuation strategy for federated learning. Where the model verification is run after the model aggregation, thus the verification of round i can be run in parallel with the training of round $i+1$. If any verification fails, the server can abort the training and reset the model to the last valid round. Although RoFL adds more than ten times the extra computation time compared to basic federated learning, it provides a defense against most data and model poisoning attacks.

Bell et al. [95] introduce ACORN, a secure aggregation protocol preserving the integrity of model updates. Similar to RoFL, ACORN applies L_0, L_2 and L_∞ norm bound checks and Bulletproofs for the submitted models. Nevertheless, to improve the efficiency of range proofs, ACORN adopts random projection [166] and several optimizations for range proofs [167]. Compared to RoFL [68] and EIFFeL [100], ACORN has lower client communication and computation complexity.

Zhu et al. [96] raise RiseFL, a robust federated learning system with secure aggregation. RiseFL proposes a probabilistic L_2 norm check method to detect attacks while improving the efficiency using the Bulletproofs. Rather than compute the L_2 norm of the whole model, RiseFL randomly samples k vectors following a chi-square distribution to compute the L_2 norm. By using a batch verification strategy, RiseFL reduces the number of group exponential operations from $O(d)$ to $O(d/\log d)$. The norm bound check is implemented through Bulletproofs. Besides of L_2 norm, RiseFL can also support sphere [168], cosine similarity [169] and Zeno++ [170] with simple modifications. According to the problem of secure aggregation of verified inputs (SAVI) defined by EIFFeL, RiseFL relaxes the input integrity, while achieving the same level of input privacy. In terms of performance, RiseFL is up to 28×, 53× and 164× faster with 100K model updates compared to ACORN, RoFL and EIFFeL, respectively.

Xing et al. [97] design NoV, a federated learning framework for fully input integrity for federated learning. Inspired by RoFL [68], NoV applies Bulletproof [124] for L_2 norm range proofs. To further enhance the defense performance against model poisoning attacks, especially for projection gradient descent (PGD) attack [171], which is one of the strongest first order attack, NoV adopts a model filter combining hybrid and layer-wise strategy. For each local model, both of the L_2 norm of the whole model and the cosine similarity at each layer are checked. In order to provide more robustness for the federated learning, NoV designs a secure aggregation protocol that can recognize Byzantine attackers within the aggregation process. Experiments have demonstrated that NoV has better defense performance against PGD attack compared to FLTrust [164], CosDefense [172] and RoFL [68].

Ma et al. [98] present VPFL, a ZKP scheme for training verifiability by a third party in federated learning. VPFL defines a complex system model for federated learning, including five different roles and the bulletin board. VPFL leverages Merkle commitment tree for local data integrity check, enabling data owner to check whether their data are correctly stored by the clients. The correctness of the training result, which is the local model parameters, is constrained by a predefined range and implemented through Bulletproof [124]. Compared to several existing schemes for verifiable aggregation and RoFL [68], VPML performs better in both the computational complexity of the server and client.

Wang et al. [99] raise a robust federated learning framework with DP. To provide privacy protection in federated learning, each local model will be masked with a Gaussian noise. To provide further privacy protection, a mask-based secure aggregation protocol [173] is also introduced. Before to be updated to the server, each client is required to clip the local model according to the given L_2 norm bound and generate a zero-knowledge range proof with Bulletproof [124], arguing that the L_∞ of the clipped local model falls in the given bound range. Each local model is bind with an ElGamal commitment, which is also used to verify the consistent of the mask and compute the aggregated global model. The experiment compares the performance of the global model under different parameter settings, but lack of assessment of the efficiency.

Roy et al. [100] propose EIFFeL to support arbitrary model integrity checks in federated learning. The author formalized the problem as the secure aggregation of verified inputs (SAVI), requiring (1) integrity check of each local update, (2) only well-formed updates are aggregated (3) only the final aggregated result is released. EIFFeL employs secret-shared non-interactive proofs (SNIP) [174], a light-weight ZKP system designed for a multi-verifier setting where the private data is distributed or secret-shared among the verifiers. In EIFFeL, each client is required to distribute the proof and shares of its private update among clients. By verifying and aggregating the valid shares, each client can aggregate the share of the global update. With these shares of global update, the server can verify and reconstruct the global update with valid shares. To improve the efficiency, EIFFeL replaces the origin share reconstruction method with a robust probabilistic reconstruction based on Reed-Solomon error correcting codes [175], to improve the performance at the cost of a small probability of failure. Further, EIFFeL employes several crypto-engineering optimizations, such as random projection [176], to reduce both the computation and communication costs. Compared to RoFL, EIFFeL can support multiple model checking methods, including L_2 bound, norm bound [177], Zeno++ [170] and cosine similarity [169]. Compared to the baseline scheme BREA [178], EIFFeL not only provides additional privacy protection, but also improves the efficiency by 2.5$\sim$18.5×.

Ghodsi et al. [101] design zPROBE to defense Byzantine attacks in federated learning. The server randomly clusters the clients into several clusters. Then each cluster is required to submit a cluster global model through zPROBE's secure aggregation protocol. By accessing these cluster model in plaintext, the server can compute and publish the threshold from the median of these models, which is dynamic and can be automatically changed. Then each client is required to prove that their masked local model meets the given threshold by using a VOLE-based ZKP system [122]. Further, to improve the efficiency of zPROBE, a probabilistic optimization is employed to randomly sample some parameters in the local model for the check. Experiment shows that zPROBE achieves lower computation complexity compared to both BREA [178] and EIFFeL [100] and higher accuracy compared to EIFFeL.

Nguyen et al. [102] detect backdoor attacks in federated learning with zero-knowledge proofs. A pruning-based backdoor defense method [179] is applied, where a neuron was considered backdoored if pruning a neuron with lower activation had no effect on the main task performance. Each client is required to submit the proof of their model passes this detect method. The zero-knowledge proof is implemented through a MPC-in-the-head system [41].

Han et al. [103] introduce an aggregation protocol with verifiable model detection. However, all the clients are required to submit their local model in plaintext to the server, which may lead to privacy attacks. The server will execute model detection and aggregation, and generate corresponding zero-knowledge proofs for this process. To detect malicious update, this framework deploy cross-round and cross-client detection. The cross-round detection checks the cosine similarity between the local model and both the local model and global model in the previous round. If any of the similarities are under the given threshold, the cross-client detection will be triggered. In the cross-client detection, the L_2 distance to the previous global model of each local model is computed as the score. The server fits a normal distribution $\mathcal{N}(\mu, \sigma)$ based on the scores of all clients and removes local models with score exceeds $\mu + \lambda\sigma$, where λ is a pre-defined threshold. To verify the correct execution of the detection mechanism at the server, zk-SNARK proofs, PERCHs [180] are generated by the server, allowing any client to verify the model removal and aggregation. It is worth mentioning that PERCH does not require a trusted setup, so the proofs generated by server can be verified by any client. To reduce the cost of the detection, only the second-to-the-last layer is involved in the detection. Compared to existing defense method including m-

Krum [181], Foolsgold [182] and RFA [183], this scheme provides the best defense performance. For the CIFAR-100 task running on ResNet56, the proof per round is generated in 100s and verified in 3ms.

Buyukates et al. [104] perform malicious detection and contribution evaluation of local models in federated learning. To evaluate the contribution of each local model, a randomized leave-one-out (RLOO) method is proposed. Specifically, the contribution of local model i is evaluated by comparing the accuracy between the global model and global model without aggregating local model i. The larger difference indicates larger contribution. The malicious detection method is similar to Han et al. [103] and corresponding proofs are generated. Therefore this scheme also suffers from the limited privacy issue. At the end of the federated learning, the server will evaluate the contribution of each model round-by-round, and generated corresponding proofs. All these proof are generated under Groth16 [30]. To improve the proof efficiency, both optimized gadgets for non-linear operations and Freivald's algorithm [139] for matrix multiplication verification are introduced. For the MLP model, the proof time of contribution evaluation is more than 400s, and the proof time of model detection and aggregation is more than 170s. Furthermore, due to the utilization of Groth16, this system requires a trusted setup to guarantee the validity of zero-knowledge proofs, however this is not clearly described.

Sum up. Most of these schemes following the existing detection methods on L_2, L_∞ norm and cosine similarity, with zero-knowledge range proofs to check the model integrity without direct access. Therefore, the focus of most schemes is not on the detection performance on malicious models, but on the efficiency improvement for range proofs. Variously, Nguyen et al. adopts a pruning-based strategy [179] to detect possible malicious and poisoned neurons. Meanwhile, because these proofs are centered around the model rather than the computational process, the proof cost is much lower than those checking the training process. Since the effectiveness of these detection methods is not provable, but based on experimental observations [164], the correctness and security of these schemes can be compromised. For example, in practice, a very small percentage of benign and malicious models will be misjudged, although this may not impact the integrity of the final result. Unlike other schemes, instead of using ZKP schemes for clients to prove the integrity of their model, Han et al. [103] and Buyukates et al. [104] place the ZKP schemes on the server side, to prove that the detection is correctly carried out. Where the privacy is compromised for the server has full access to the model from each client.

2) Fairness: Fairness refers to that models should make decisions that are unbiased and equitable across different groups of people. There are several key aspects in fairness, including demographic parity [45], equalized odds [184], equal opportunity [184], etc., which can be measured by the evaluation result on specific input dataset. Model providers need to demonstrate that their models are fair. For example, in the scenario of outsourced inference, the service provider needs to prove that the model it uses for the inference service is fair. By generating proofs arguing for both the correctness of the evaluation result and the fairness measurement, the prover can convince the verifier that the provided model is fair. The privacy of the evaluation input data can be protected by the zero-knowledge.

Tang et al. [105] design a framework for verifiable fairness of machine learning models based on zk-SNARKs. In this framework, there are four kinds of fairness definition involved, namely demographic parity, equalized odds, equal opportunity and disparate impact, each of which has a formulation expression. In this system, a regulator for fairness is presented. This regulator will give out test data, ask the server to provide the evaluation result on that test data, and prove the correctness of that evaluation result by zero-knowledge proof. Based on the evaluation result, the regulator audits the model fairness and generate corresponding zero-knowledge proofs for the audition process. The fairness measurements are computed based on model statistical metrics, which is computed from confusion metrics. The non-linear operations involved in the quantization are represented with bit-decomposition. Finally, the client can verify the fairness of the model from public commitments and proofs, without accessing to the plaintext of model or audition dataset. The framework adopts SnarkPack [185] to aggregate multiple proofs into a single proof to reduce the overhead of the verifier. Experiments show that for a LogiR model, the proving time is in the hundreds of seconds and the verification time is in the milliseconds.

Toreini et al. [106] propose Fairness-as-a-Service (FaaS), a framework for verifiable fairness in machine learning systems. In FaaS, the fairness of models are verified through three fairness metrics for demographic parity, equalized odds and equal opportunity. The server is required to provide the encrypted properties of each test data for the fairness evaluation. 1-out-of-8 zero-knowledge proofs [186] are generated for arguing that the encrypted properties are well-formed, rather than the correctness of these properties themselves. After receiving the encrypted properties, the regulator can obtain statistical information of data properties and thus determines whether the model that outputs these data is fair. However, the server is assumed to be honest, leading to a very limited security and verifiability. Experiment shows that for a dataset of size 3166, the whole scheme runs in more than 15 hours.

Shamsabadi et al. [69] present Confidential-PROFITT, a ZKP-friendly decision tree training algorithm as well as the corresponding specialized ZKP protocol, which is derived from a vector-oblivious linear evaluation (VOLE) [122]. Confidential-PROFITT is the first framework for ZKP-based verifiable DT training. Traditionally, the DT is trained by recursively splitting the training dataset. To find the best split, Shamsabadi proposed an optimization problem that maximizes the accuracy gain and upper bounds the unfairness gain. Rather than proving each training step in the training process, Confidential-PROFITT proves that the given DT model is a valid and fair result on the training dataset, the fairness refers to demographic parity and equalized odds. The ZKP protocol is used to prove that with the given committed trained DT model and training dataset, each committed path is correct, the sensitive attribute passes are balance, and the unfairness gain of each node is below the threshold. Further, this framework can be expanded to random forests by running a secure coin-flipping before the training. Experiment shows that it takes less than 2 minutes for Confidential-PROFITT to prove the fairness of a trained DT.

Sum up. The fairness of the model is defined according to several metrics. Thus such schemes center around the model, first prove the correctness of the evaluation result on a specific dataset, and then prove that the result satisfies the defined fairness metrics. These fairness metrics are usually easy to calculate and do not involve complex non-linear operations, so there is no additional performance enhancement method beyond traditional ZKP-VML schemes for fairness.

3) Model Aggregation: In federated learning, the local model submitted by clients are aggregated by the server to obtain the global model. This type of ZKP-VML schemes aims at guarantee the correctness of the aggregation computation as shown in Fig. 6(b).

Wang et al. [107] introduce zkFL for verifiable model aggregation in federated learning. To guarantee the correctness of average aggregation under a malicious server, each client updates the local model with corresponding Pedersen commitment and signature. The server is required to prove the correctness of aggregation alone with the validity of the commitment and signature. However, the plaintext of local models are sent to the server for aggregation, leading to the potential privacy leakage. Moreover, the security can be harmed by the server reusing previous local model for aggregation. Further, the experiment shows that both the time and communication overhead of zkFL are high, making a aggregation scheme not so practical.

Ahmadi et al. [108] propose zkFDL for verifiable aggregation in federated learning. However, due to its assumption of public input and output, The utility of the scheme is also in doubt.

Sum up. ZKP may not be a proper choice for model aggregation. The computations involved in model aggregation contain large number of inputs and outputs with relatively simple computation process, which can be handled with simple additive homomorphic commitment schemes. Introducing ZKPs into model aggregation not only increase no additional privacy protection, but also increases unnecessary computational cost, which is unacceptable and unsuitable for federated learning.

4) Crowdsourcing Inference: Crowdsourcing inference refers to the scenario where an inference task is outsourced by the user to a cluster of workers with their own models. Each worker will generate an inference result with their own model. The cluster center will collect these inference results, from which the center can deduce and return a reliable result to the user. This type of ZKP-VML scheme aims at arguing that the returned result is indeed computed from some reliable truth inference algorithms as shown in Fig. 6(c).

Liu et al. [109] design zkTI for crowdsourcing truth inference. In crowdsourcing truth inference, the server is required to distribute the tasks to all clients for their inference. After collecting the inference results, the server runs a truth inference algorithm to deduce the true result and evaluate the quality of results from clients. The truth inference algorithm can prevent clients from submitting malicious results, thus ensuring the quality of the final results. To prove that the final result is reliable, the prover convert the truth inference algorithm into circuits, and generate proofs for the computation process with zk-SNARKs. Specifically, both Groth16 [30] and Spartan [145] can be implemented. Experiments show that zkTI shows 1.5-4× improvement over the baseline [187], depending on the ZKP backend and parameters. But the needed trusted setup for Groth16 is also not clearly described.

Sum up. Crowdsourcing inference is an emerging application scenario for ZKP-VML. Similar to model aggregation, crowdsourcing inference also involves gathering results and computations among which. Differently, crowdsourcing inference works on smaller input and output size with more complicated computations, which broaden the space for applying ZKP.

5) Model Ownership: Model ownership refers to the ownership and intellectual property of the model, especially where the value of data and ML models increasing. For the ZKP-VML scenario, there are two verification tasks for model ownership as shown in Fig. 6(d). One task is for the owner to prove the correctness of the watermark extraction process. If the watermark extraction process can be proved, then the ownership can be ensured without reveal the extracted watermark. Another task is for the owner to prove the validity of a extracted watermark. If the owner can prove the validity of the extracted watermark under some fixed commitment, then the ownership can also be proved without leaking the original watermark. Once some entity can prove that they indeed own this model by adding a verifiable watermark to the model, then the illegal use of the model can be prevented, and the intellectual property can be protected.

Yang et al. [110] propose FedZKP for verifying the ownership of the global model in FL. Each client generates a pair of private and public input via exact learning parity with noise (xLPN) [188], where the public input is sent to the server. The server constructs a watermark based on the public input from all clients and sends this watermark to all clients. Each client embeds this watermark into the batch normalization (BN) layer during their local training following existing method [189]. When the user wants to verify the watermark in the global model, first the user extracts the watermark from the global model and compare whether the watermark is similar enough to the one constructed from the public inputs of the clients. Then the client can generate a zero-knowledge proof that it has a secret corresponding to the public input through the Sigma protocol [190]. Experiments prove that FedZKP can effectively defend against fine-tuning [48] and pruning [191] attacks on AlexNet and Resnet18. In addition, FedZKP runs with additional time cost in tens of seconds and communication cost in a few MBs.

Sheybani et al. [111] raise ZKROWNN for verifying the model ownership while preserving the privacy of the watermark. The watermarks are embedded and extracted on specific layers following DeepSigns [115]. ZKROWNN leverages zero-knowledge proofs, especially Groth16 [30], to prove the correctness of the embedded watermark during the extraction process. To generate proofs for the extraction algorithm, ZKROWNN provides different sub-circuits for each computation. For example, for convolution, the three-dimensional convolution is reduced to one-dimensional vector. The Sigmoid function is represented by a polynomial approximation. ReLU and hard thresholding are represented by piecewise functions. Besides, Freivald's algorithm [139] is introduced to improve the efficiency for proving matrix multiplication. Experiments shows that ZKROWNN can complete the proof and verification in tens of seconds and a few KBs of communication.

Sum up. Depending on the different types of entities and privacy protection needs, the application of ZKP in model ownership scenarios can be various. Multi-entities creates additional verifiability challenges for watermark generation and embedding. Whether the privacy protection of the watermark itself is needed is also variable.

6) Machine Unlearning: Machine unlearning refers to remove specific training data from the trained model. For the ZKP-VML scenario, there are two verification tasks for machine unlearning as shown in Fig. 6(e). One task is for the performer to prove the correctness of the model unlearning process. If the model unlearning process can be proved, then the unlearnt model can be ensured without reveal additional information about the data to be forgotten. Another task is to prove the effectiveness of an unlearnt model. If the unlearnt model can be proved similar enough to a model trained without the data to be forgotten, then the unlearning can also be proved without leaking additional information.

Common unlearning approaches include retraining the model using a dataset without specific data, and fine-tuning the trained model to remove the effects of specific data. The importance of machine unlearning increases with the wide use of ML models and concerns about data privacy.

Eisenhofer et al. [112] propose a ZKP-based verifiable machine unlearning framework. This iterative scheme supports both verifiable learning and unlearning, where clients can request the addition or removal of data from the training dataset in each round. Learning and unlearning are modeled as separate algorithms, converted into circuits, and verified via zk-SNARKs. The server maintains the system state through hash records of the learning and unlearning datasets to ensure the correctness of client requests. For learning requests, the server proves: 1. Correct computation of the training process. 2. Correct updating of the learning hash record. 3. No modification to the unlearning hash record. For forgetting requests, in addition to proving the correctness of computation and hash maintenance, it is also verified that the deleted data is no longer in the training dataset. The framework uses Spartan for zero-knowledge proofs and Poseidon, a ZKP-friendly hash function. In the experiments, three machine unlearning methods are implemented within this framework, namely, retraining-based, optimization-based [192], and amnesiac [193]. For a linear regression model, proof generation and verification times for one unlearning request range from 1-5 minutes and 0.4-2 minutes, respectively.

Sum up. Similar to the training process, the unlearning process also involves training data and ML models. Nevertheless, the differences in adopted algorithms bring additional optimization possibilities for machine unlearning. Besides, not only the unlearning process needs verifiability, the verification of the unlearning effect can also be handled by ZKP in a privacy-preserving manner.

7) Large Language Model: Compared to the neural network focused and studied by the majority of ZKP-VML schemes, large language models (LLMs) introduce massive data and computations, as well as additional computation types, such as the attention mechanism and the transformer. How to handle the additional computation types and further optimize the proof efficiency for the increased burden are the main concerns of such ZKP-VML schemes.

Sun et al. [113] propose zkLLM, a ZKP-VML framework for LLM inference. To address the non-linear and non-arithmetic operations, Sun et al. proposed *tlookup*, a more efficient table lookup proof especially for the tensor-based structure based on sumcheck protocol. This method proves that elements of one tensor are contained within another, enabling the verification of non-arithmetic operations such as ReLU, layer normalization, and GELU. On which *zkAttn* is further built to deal with the attention mechanism and softmax function within. zkAttn avoids the division in softmax by translation invariance and decomposes the result of softmax into a summation of several numbers, applying tlookup to prove each. For matrix multiplication in attention, zkAttn uses Safetynets [57], and for component-wise proofs, it draws on zkCNN [17] techniques. Experiments with the OPT, LLaMa-2 models, and the C4 dataset demonstrate that zkLLM supports models 10× larger and achieves proof times 50× faster than Kang [86]. For the 13B-parameter LLaMa-2 model, zkLLM delivers proof in under 15 minutes, with a proof size of 188KB and verification within 5 seconds.

Ganescu et al. [114] raise snarkGPT to guarantee that the given output by the LLM is indeed inferred on the specific model. To achieve which, snarkGPT generates proofs arguing for the correctness of the inference computation with the commitment of the used LLM. To apply the ZKP system to GPT-2, table lookup proof is also adopted to deal with the transformer. snarkGPT is implemented with EZKL [194], however, the experiment shows a large space for further improvement.

Sum up. The ZKP-VML schemes for LLMs make a further step. The proving and optimization methods of ZKP-VML for NNs are not able to meet the massive computation burden in LLM. Advanced optimization ideas and methods are necessary. Nevertheless, these advanced methods for LLM may also contribute to the ZKP-VML schemes for NNs and other models.

8) Semantic Communication: Semantic communication is an emerging paradigm focused on transmitting the ”meaning” of information rather than raw data. The sender uses a semantic encoder to extract key features from the raw data and convert it into a semantic representation. For example, in image transmission, a deep learning model may extract high-level features. The receiver then demodulates the signal to recover the semantic representation and decodes it into the original data or an approximation. In the context of ZKP-VML, as shown in Fig. 6(f), the sender must prove that the semantic representation is accurately derived from the original message, ensuring the integrity of the transmitted data.

Lin et al. [53] proposed a secure semantic communication framework based on blockchain and ZKP to address the issue where attackers may send malicious data with similar semantic content but different actual data. In this framework, edge devices perform spatial transformations (e.g., bilinear interpolation) on semantic data, such as images, before transmitting it to a Virtual Service Provider (VSP). These transformations serve to obscure the image and increase the distinction between adversarial and genuine samples in terms of semantic similarity. ZKPs are used to record and verify the correctness of these transformations, ensuring that the data remains untampered during transmission. The VSP uses smart contracts on the blockchain to validate the legality of the semantic data transformation process. In this scheme, ZKPs provide a method for verification without revealing the actual data that has been transformed.

Sum up. The application of ZKP-VML in semantic communication represents a further deep integration of ZKP-VML with communication networks. Although research in this area is still limited, this work highlights the potential value of ZKP-VML in the context of semantic communication.

VI. Optimization Method Analysis

In this section we summarize and categorize existing optimization methods in ZKP-VML schemes. Basically, in a ZKP-VML scheme, the prover generates proofs arguing the correctness of their computational process, and the verifier can check whether the proof is valid. Efficiency optimization is a key part of making the ZKP-VML scheme practical. Optimizations within this process can be broadly categorized into verification-based and generation-based. Verification-based optimization involves changing the verification object, i.e., the proof to be verified does not directly argue the correctness of the complete computational process. Generation-based optimization involves changing the way that proofs are generated, i.e., designing proof structures or generation algorithms with lower computation and communication costs, especially for a specific target type of computation.

A. Verification-based Optimization

Within the verification-based optimization, there are two main types: 1. *Embedding* and aggregating the proofs. 2. *Sampling* a small portion of the proofs for verification.

For the first type, since the number of claims to be proved is not reduced, and the proof *embedding* and aggregation are performed by the prover, the optimization will focus more on the communication and computational burden on the verifier. Keuffer et al. [65] first generate sumcheck proofs π_1 arguing for the correctness of the machine learning computational process. To embed these proofs, a zk-SNARK proof π_2 is generated, arguing for the valid verification process of π_1. Thus, by verifying only π_2, the verifier can check the whole computation, and the communication and verification cost of π_1 can be reduced. Similarly, Garg et al. [89] prove the validity of views in the MPCitH proofs by generating zk-SNARK proofs, which also reduces the communication and verification cost of verifying the views. Abbaszadeh et al. [74] adopt incrementally verifiable computation, where each proof arguing for not only the current round training process, but also the correctness of the previous proof. Thus there will be only one proof in the end, which optimizes the proof size to be independent to the training rounds.

For the second type, by *sampling* and verifying a small portion of the proofs, the verifier can still detect a certain percentage of malicious behavior with a very high probability. For example, considering there are 30k fraud proofs in 100k proofs, the verifier only needs to randomly verify 14 proofs to detect the misbehaviour with the probability higher than 99%. Although the soundness error can be extremely small, it still introduce a non-negligible probability. Both Zhao et al. [66] and Huang et al. [93] leverage the random sampling on proofs, i.e., randomly select a small portion of proofs for generation and verification. Similarly, Zhou et al. [94] sample a small portion of the constraints in the computational process for the prover to generate proofs for. These sampling significantly reduces the proof generation and verification overhead, but also introduces a non-negligible soundness error.

The optimization of embedding and aggregating proofs does not reduce the computational complexity for the prover. In fact, they may slightly increase it, as the prover has to do the extra embedding and aggregation. However, these techniques significantly alleviate the bandwidth burden, making them more suitable for scenarios with stringent bandwidth requirements. Among these methods, the embedding proof and incremental computation verification techniques are particularly easy to migrate and implement to other schemes. On the other hand, the approach of verifying a subset of rounds is more applicable to scenarios involving a large number of training iterations. By sampling only a small number of rounds for verification, the efficiency of the scheme can be improved by several orders of magnitude. However, it is important to note that this sampling optimization may compromise the security of the scheme to some extent, as it renders the system vulnerable to single-round attacks. Therefore, this approach is better suited to scenarios where the verification workload is a priority.

B. Generation-based Optimization

For the generation-based optimization, we categorize it into: 1. *Tailor* existing proof protocols to the target computations. 2. *Change* the proof route according to the properties of the target computation. 3. *Prune* the additional privacy burden.

The majority of existing work **tailors** the proof protocol for proving the target computation. This part can be further categorized by their different target computations.

For *matrix multiplication*, we consider a 2-D matrix multiplication between A and B both of size $n \times n$ as $C = A \times B$. The complexity of input and output is $O(n^2)$ and the computational complexity is $O(n^3)$.

Ghodsi et al. [57] construct sumcheck protocol for matrix multiplication proof. First, the matrix multiplication is transformed into a sumcheck problem through a multilinear extension, and then the correctness of the polynomial at random points is progressively verified using a recursive sumcheck protocol. By randomly selecting a point $C_{i,j}$ in the matrix C and representing its computation process in the form of sumcheck protocol (i.e. $C_{i,j} = \Sigma_{k \in \{0,1,\ldots,n\}} A[i,k]B[k,j]$), a protocol with prover complexity of $O(n(n+d))$ and verifier complexity of $O(n^2)$ can be obtained, while $4\log(n)$ elements need to be exchanged in the process. For the matrix multiplication proved with sumcheck protocol, Duan et al. [73] further combine the computations containing the same matrices by multi-linear extension, enabling the prover to prove multiple claims at once with an acceptable soundness error.

Weng et al. [20] adopt Freivald's algorithm to reduce the proof cost for matrix multiplication. Rather than check the equation $A \times B = C$ directly, a random challenge vector t of length n is introduced. The original equation can be checked by multiplying a challenge vector as $A \times (B \cdot t) = C \cdot t$, with prover complexity of $O(n^2)$, verifier complexity of $O(n)$, and constant proof size $O(k \log q)$, where k is secure parameter and q is the big prime.

Wu et al. [72] construct sumcheck protocol for the matrix multiplication in the MPC protocol to achieve the confidential computation, whose prover time is $O(n^2)$ and verifier time is $O(\log n)$, with $O(\log n)$ proof size.

For *matrix convolution*, we consider a 2-D convolution between two matrices X and W of size $n \times n$ and $w \times w$ ($n \gg w$) as a $(n-w+1)\times(n-w+1)$ matrix $U = X * W$. The complexity of input and output is $O(n^2+w^2)$ and the computational complexity is $O(w^2(n-w+1)^2) = O(w^2(n-w)^2)$.

Liu et al. [17] construct sumcheck protocol for convolution. Convolution can be accelerated by FFT, then verifying FFT-based convolution is also more efficient than verifying convolution directly. zkCNN designs a sumcheck protocol for verifying the FFT, which is a matrix-vector multiplication of size n, to reduce the original prove complexity from $O(n^2)$ to $O(n)$. Further, by verifying the original convolution in the form of $\overline{U} = \overline{X} * \overline{W} = IFFT(FFT(\overline{X}) \odot FFT(\overline{W}))$ with sumcheck protocol for FFT and Inverse FFT (IFFT), the overall prover time can be reduced to $O(n^2)$, with $O(\log^2 n)$ proof size and verifier time.

Lee et al. [16] extend the original QAP-based zk-SNARK to QPP-based zk-SNARK. This innovative approach involves assigning a polynomial value to each wire, allowing for the expression of 1-D convolution computation using a single multi-gate in the arithmetic circuit. As a result, the number of multi-gates required for proving convolution is reduced to $O(n^2 + w^2)$. Built on this, vCNN offers a QPP-based zk-SNARK where the prover complexity is $O(n^2 + w^2)$, the verifier complexity is $O(n^2)$, along with a constant size of proof.

Weng et al. [75] further extend the QPP-based zk-SNARK to QMP-based zk-SNARK, where each wire representing a matrix. Compared to the QPP-based, QMP-based zk-SNARK performs better on proving batched convolution. For batch convolution operations involving M matrices W of size $w \times w$ and Mn^2 matrices X of size $n \times n$, pvCNN transforms them into one matrix multiplication of size $Mn^2 \times Mn^2$, which leverages a prover complexity of $O(M^2n^4)$ and verifier complexity of $O(Mn^2)$ with a QMP-based zk-SNARK. In contrast, under similar conditions, vCNN has a proof complexity of $O(M \cdot Mn^2 \cdot (w^2 + n^2))$.

Feng et al. [76] proposed ZENO circuit to efficiently represent tensor in zk-SNARKs. By leveraging the commutative property of addition gates and minimizing the number of which, the computation complexity of dot product of length n can be reduced from $O(n^2)$ to $O(n)$. Thus the prover complexity of convolution is $= O(w(n-w)^2)$, verifier complexity is $O(n^2 + w^2)$.

Both Huang et al. [93] and Fan et al. [77], [80], [78], [79] leverage im2col [159] to convert convolution into matrix multiplication. Where the size of input matrix is $(n-w+1)^2 \times w^2$ and size of kernel matrix is $w^2 \times 1$. The proofs of these matrix multiplications are further optimized through the Freivald's algorithm. Thus the prover complexity is $O(w^2(n-w)^2)$, the verifier complexity is $(n-w)^2$, the proof size is constant.

For *decision tree*, we consider a binary decision tree T with height h, which has N nodes and an attribute set size of d. To prove the inference of a decision tree, it is necessary to demonstrate the existence of a valid prediction path within this decision tree. For a decision tree with a height of h and nodes containing d attributes, proving a decision path that includes h calculations has a complexity of $O(d)$ per comparison. Therefore, the prover computational complexity for inference is $O(hd)$, for accuracy with n samples is $O(nhd)$.

Zhang et al. [67] insert designed sibling nodes to the DT to reduce the proof cost caused by the comparison. The zkDT reduce the proof complexity to $O(d+h)$ by constructing an additional input permutation $\bar{a}$, which is based on the sorting of attributes used in the decision path. Verifying the permutation relationship between a and $\bar{a}$ adds only $O(d)$ extra computational complexity. Overall, during the commitment phase, the prover needs to perform $O(N)$ hash computations, and the size of the proof circuit is $O(d+h)$. With the Aurora system, for DT inference, the computational complexity of the proof generation is $O((d+h)\log(d+h)) = O(d\log d)$, the verification complexity is $O(d+h) = O(d)$, and the proof size is $O(\log^2(d+h))$. For DT accuracy, the prover computation complexity is $O(nd\log(nd))$ with $O(N)$ hashes, the verification time is $O(nd)$ and proof size of $\log^2(nd)$.

Singh et al. [82] propose consistent memory access proof for efficient proof of dataset operations and DT models. For a batch of n input samples with d dimensions, this scheme reduces the cost of each sample from $O(d\log h)$ in zkDT to $O(d+h)$ through consistent memory access. Consequently, the size of the entire proof circuit is reduced to $O(N + n(d + h + wh))$, where w is the bit-width of the feature values. In comparison, under the same conditions, the circuit size of zkDT is $O(c(H)N + n(d\log h + hw))$, where $c(H)$ represents the size of the hash circuit. For the DT accuracy, the circuit complexity is $O(N + hn(1 + w + d))$. The prover complexity of backend ZKP is $O(C\log|C|)$, with constant size of proof and $O(|io|)$ verifier complexity, where C represents the circuit and io represents the input and output.

Campanelli et al. [81] proposed efficient matrix lookup proofs for DT accuracy. By converting the DT model into a matrix, the proof of DT evaluation can be transformed into proof of locations in matrix and proof of satisfaction of corresponding constraints. With the efficient matrix lookup proof, the prover complexity can be reduced to $O(nd\log(nd))$, the verifier complexity can be reduced to $O(n)$, with a constant size of proofs.

For *non-linear operations* in various neural networks, there are several representation methods.

One intuitive way is to directly transform the non-linear operations. Weng tt al. [20] reduce the non-linear operation proof cost by providing efficient conversion between arithmetic and boolean circuit to reduce the bit-decomposition cost. Duan et al. [73] adopt the GINGER [195] to convert the non-linear operations into R1CS and further proved with Spartan [145].

Some researchers also build specialized proofs for non-linear operations. Feng et al. [18] optimize the proof cost for both comparison and division operations through sign-bit grouping and remainder-based verification. Kang et al. [86] adopt table lookup for the division in non-linear operations. Hao et al. [70] reduce the proof cost for non-linear operation cost by providing table lookup proofs.

Building custom gadgets is also a popular trick. Lu et al. [71] build gadgets based on range proof and table lookup proof for constructing constraints for non-linear layers in CNN and TF. Chen et al. [87] design plenty of gadgets for constructing the whole machine learning model including the non-linear operations. Wang et al. [85] apply several gadgets to represent the SVM algorithms, including the radial basis function (RBF) kernel method.

For some special computations, depending on the properties of their computation process or results, it is possible to construct specialized methods for optimizing the proof efficiency by **changing** the verification route.

Ruckel et al. [92] exploit the properties of inverse matrix in LR. Instead of proving the inverse computation, it can be proved that the proximity of the claimed matrix to the true inverse matrix. If the approximation error is below the threshold, then the claimed matrix is considered as a valid inverse matrix.

Angel et al. [91] exploit the properties of solutions to optimization problems in Otti. For LP and SDP problem, Otti proves that the primal solution is a feasible solution to the primal problem, the dual solution is a feasible solution to the dual problem, and the duality gap is zero. For SGD problem, Otti proves that at z the gradient estimates $\nabla f_i(z)$ have some property. By constructing proof methods checking the optimality of the results, proofs for the complete computational procedure can be eliminated.

Shamsabadi et al. [69] exploit the properties of trained DT models and corresponding training dataset. Instead of proving the whole training process, it can be proved that the DT model is balance on the given dataset, thus the given DT model is a valid training result.

A novel idea is to consider and **prune** the burden that data privacy imposes on ZKP-VML.

Weng et al. [20] consider the conversion between committed and authenticated values and reduce the proof cost incurred by converting publicly committed value to the private authenticated values in the ZKP system through an efficient conversion method.

Feng et al. [76] consider the proof cost caused by the excessive privacy. In practice, there are a fair number of cases where either the features or weights are private. A naive approach is to generate one constraint for each multiplication in the dot product of features and weights, which incurs unnecessary constraints. By designing specific proof patterns, the number of constraints and proof cost for multiplication with unnecessary privacy can be reduced.

Sum up. Tailoring existing proof protocols to the target computation for specific optimizations represents a primary approach in ZKP-VML optimization. This method enables the maximization of computational and communication efficiency with minimal sacrifice in security. For matrix multiplication, commonly used optimization techniques include the sumcheck protocol and the application of Freivald's algorithm [139]. In the case of convolution, one approach is to employ the FFT algorithm to accelerate the convolution process, followed by the use of ZKPs to verify the accelerated computation. Another approach is to optimize the matrix representation within ZKPs to reduce the number of multiplication gates, thus enhancing the efficiency. However, while transforming convolution into a matrix form using im2col [159] and Freivald's algorithm allows for further optimization, this transformation introduces additional computational overhead, limiting the overall performance improvement. For decision trees, the primary focus of optimization lies in how to represent a decision tree in a proof-friendly manner. For nonlinear operations, a common optimization strategy is to construct basic gadgets and then combine these gadgets to represent the nonlinear operations, thereby minimizing the accuracy loss caused by approximation techniques. Optimizations based on altering the verification route are more dependent on the specific computation, making them difficult to transfer to other computations. Pruning the additional computational burden introduced by data privacy protection presents an interesting approach. ZKPs offer robust privacy protection. However, such protection may not be necessary for certain application scenarios. By relaxing this privacy requirement, computational overhead can be fine-tuned to optimize unnecessary operations.

To provide a more intuitive comparison of the improvements in computational and communication efficiency brought about by various optimization strategies, we selected a subset of these methods and compared their performance in Table VI-B. Specifically, we focused on problems that can be quantitatively assessed, such as matrix multiplication, convolution, and decision tree accuracy. These problems exhibit clear computational complexity and input-output scale. In contrast, non-linear operations, even when considering specific computations, lack intuitive computational complexity, making it challenging to compare the performance improvement of different optimization methods. Furthermore, we selected optimization methods that allow for quantitative comparison, such as those based on tailored approaches. The performance of embedding methods depends on the selected proof scheme, while sample-based methods are influenced by the number of verification rounds chosen by the verifier. The effectiveness of changing and pruning methods depends on the optimization problem at hand. Since the performance improvements of the aforementioned methods typically rely on the specific problem context, it is difficult to provide a straightforward analysis or comparison.

Therefore, we have excluded these methods from our comparative analysis.

VII. Challenges and Future Research Directions

Based on the analysis of existing work, in this section, we explore the challenges and future research directions of ZKP-VML in communication networks.

The main development direction of ZKP-VML revolves around enabling participants to verify ML computations in a zero-knowledge manner with minimal additional costs. This goal encompasses two key aspects: (1) enhancing the practical applicability of ZKP-VML and (2) enriching its content and properties. The limitations to the practical applicability of ZKP-VML primarily stem from two factors.

First, the verifiability offered by ZKPs is achieved at the expense of additional computational costs. Therefore, to make ZKP-VML more practical, improving the computational efficiency of proofs is essential, minimizing the overhead between ZKP-VML and existing ML frameworks. Additionally, the lack of development in user-friendly toolkits also degrades the adoption of ZKP-VML. As machine learning continues to evolve, numerous emerging ML scenarios and computational paradigms have arisen. To broadly incorporate verifiability in ML, supporting these new scenarios and computational types in ZKP-VML is crucial. Furthermore, integrating ZKPs with other cryptographic techniques can further enrich security attributes and properties in ML.

A. Computational Efficiency

Enhancing efficiency is a crucial research direction in ZKP-VML, as reducing the computational burden of ZKPs can significantly improve their practicality. There are three key avenues to advance the efficiency of ZKP-VML schemes:

1) **Proof System Design:** Tailoring proof systems to specific machine learning computations can reduce proof costs. This approach has been adopted by many existing schemes, wherein specialized zero-knowledge proof schemes are designed for specific computation types or model structures to improve efficiency. Future research can develop dedicated ZKP schemes for models with special properties, such as graph neural networks (handling graph structures), transformers (addressing stacked encoders/decoders and self-attention mechanisms), and recurrent neural networks (focusing on recurrent structures and time-series features). Additionally, exploring alternative zero-knowledge proof systems beyond QAP-based solutions, like Groth16 [30], could offer performance benefits in different computational scenarios.

2) **Specialized Hardware:** Using specialized hardware can alleviate the computational burden of generating ZKPs. Hardware accelerators, such as pipeline designs [196], FPGAs for number-theoretic transforms [197], and GPUs for ZKP computation [198], [199], have shown potential in boosting efficiency in ZKP. Integrating these acceleration strategies into existing ZKP-VML frameworks could create a comprehensive architecture for enhancing the operational efficiency of ZKP-VML systems, crucial for real-world deployment.

3) **Balance with Privacy:** Achieving a balance between security and privacy with efficiency can help reduce overall system costs. In some scenarios, not all computations need to be verified, allowing for selective verification of a minimal subset of rounds in multi-round training processes. Techniques like range proofs can relax equality constraints in verification to improve efficiency. Additionally, fine-tuning privacy protection requirements and avoiding unnecessary privacy measures can reduce computational overhead. Addressing these trade-offs between security and efficiency is key to the practical applicability of ZKP-VML.

B. Real-World Development

Currently, ZKP-VML is still in development, with most implementations at the demo stage to showcase their performance. For practical deployment, it is crucial to ensure interoperability with existing solutions, allowing developers to seamlessly integrate privacy features without major modifications to their workflows. Key considerations include ease of integration, performance impact, and compatibility with various hardware and software environments. This will foster broader adoption and facilitate the real-world application of ZKPs across industries. A key focus area is the development of privacy-preserving computing frameworks that support ZKP-VML, such as Rosetta [200]. Currently, mainstream frameworks offer limited support for ML and ZKPs, often requiring extensive expertise to construct ZKP-VML systems. Making ZKP-VML more user-friendly and integrable into these frameworks will significantly expand its applicability.

Additionally, standardization and collaboration are essential for advancing ZKP-VML. Academic, industrial, and open-source collaborations can drive the development of standardized protocols, efficient libraries, and benchmarking tools [12]. For instance, benchmarking tools tailored to mainstream ZKP protocols can help quantify performance disparities, guiding users toward more efficient solutions and improving overall system efficiency. Such initiatives will foster knowledge sharing, accelerate innovation, and support the adoption of ZKP-VML as a reliable privacy-preserving tool in machine learning applications.

C. Novel Scenarios

The evolving ML technologies and expanding ML scenarios offer increased opportunities for incorporating verifiability into ML through ZKPs. Currently, the vast majority of research is limited to the traditional training and inference, which represent the most fundamental ML scenarios. In DML, there are many scenarios where verifiability is worth exploring, such as proving model ownership and demonstrating model fairness. By identifying specific evaluation criteria and ML tasks in various scenarios, such as model watermark verification and model integrity verification, and converting them into forms that can be proved using ZKP, it becomes possible to extend ZKP-VML to new ML scenarios. ZKPs provide verifiability for these scenarios while simultaneously protecting data privacy. The verifiability in these scenarios have not yet been fully investigated. Beyond these scenarios, there are numerous

TABLE V
COMPUTATIONAL & COMMUNICATION COMPLEXITY ON DIFFERENT OPTIMIZATIONS

Scheme	Optimization Method	Targeted Computation	Efficiency (Prover / Verifier)	Communication (Proof Size)
SafetyNets [57], VPNNT [73]	Tailored sumcheck protocol	Matrix Multiplication $O(n^3)$	$O(n(n+d))/O(n^2)$	$O(\log n)$
Wu et al. [72]	Tailored sumcheck protocol & MPC		$O(n^2)/O(\log n)$	$O(\log n)$
Mystique [20], Buyukates et al. [104]	Freivalds [139]		$O(n^2)/O(n)$	$O(1)$
zkCNN [17]	Tailored sumcheck protocol & FFT	Convolution $O(w^2(n-w)^2)$	$O(n^2)/O(\log^2 n)$	$O(\log^2 n)$
vCNN [16]	Tailored representation of vector		$O(n^2+w^2)/O(n^2)$	$O(1)$
pvCNN [75]	Tailored representation of matrix		$O(n^2+w^2)/O(n^2)$	$O(1)$
ZENO [76]	Tailored representation of tensor		$O(w(n-w)^2)/O(n^2)$	$O(1)$
Fan et al. [77], [80], [78], [79], zkMLaaS [93]	Freivalds [139] & im2col [159]		$O(w^2(n-w)^2)/O(n^2)$	$O(1)$
zkDT [67]	Tailored representation of trees	Decision Tree $O(nhd)$	$O(nd\log nd)+O(N)/O(nd)$	$O(\log^2 nd)$
Singh et al. [82]	Tailored representation of trees & consistent memory access		$O(nd\log nd)/O(nd)$	$O(1)$
Campanelli et al. [81]	Tailored representation of trees & table lookup		$O(nd\log nd)/O(n)$	$O(1)$

undeveloped ML scenarios that could benefit from verifiability, such as the deepfake detection, model interpretability and quantum machine learning. Introducing verifiability for additional scenarios often involves new computational processes, such as calculating model watermarks or assessing model fairness. Since these processes and algorithms differ from standard training and inference, they present entirely new efficiency challenges for the proofs. Moreover, for more complex DML scenarios, relying solely on ZKP may not fully meet the security requirements. Therefore, it is also necessary to consider integrating other cryptographic techniques to enhance and enrich the security properties of ZKP-VML.

D. Various Properties

In future research, the complex combination with different cryptographic and techniques can be considered to enrich security properties of ZKP-VML.

By integrating cryptographic techniques related to data privacy protection, ZKP-VML can further enhance its capability to safeguard data privacy. For instance, by combining differential privacy and homomorphic encryption techniques, ZKP-VML can achieve stronger privacy protection properties. Differential privacy can enhance the privacy protection level of data, preserving the data privacy during computations and interactions process beyond the proof generation and verification. Through homomorphic encryption, verifiers can protect computation tasks by enabling provers to perform tasks using homomorphic computations on encrypted data. This process enables the execution of computations without disclosing the plaintext content, thereby safeguarding the privacy of the verifier. Besides, by combining with blockchain, ZKP-VML can be utilized to build trustless and decentralized federated learning systems, which is a possible learning paradigm in the future. Blockchain and the smart contracts deployed on it ensure the decentralization of the system, while ZKP-VML constrains the training behaviors of participants without compromising the data privacy.

With the application of additional techniques, it becomes crucial to consider how newly introduced features might pose potential threats to the integrity or privacy of the original ZKP-VML scheme. As the system becomes more complex and incorporates additional functionalities, it is essential to conduct thorough security assessments to ensure that the integrity and privacy guarantees of the ZKP-VML scheme remain robust and uncompromised in a dynamic and evolving environment. This includes evaluating the impact of new features on data privacy, ensuring data integrity, and identifying potential vulnerabilities that could be exploited by malicious actors.

E. Communication Network

The ZKP-VML approach faces several open issues in communication networks, especially regarding scalability, and integration with wireless technologies. These issues include:

1) Scalability: As machine learning models and data grow, the computational and communication overhead of zero-knowledge proofs can become a bottleneck. Besides, ZKP-VML schemes are still relatively slow for real-time applications. This is particularly relevant in real-time communication scenarios, such as 5G/6G networks where large amounts of data are transmitted. Future research can explore efficient communication paradigm and protocol to support complex proof generation and verification. Hybrid cryptographic techniques, such as combining homomorphic encryption with ZKPs, could mitigate some of the performance bottlenecks in privacy-preserving computations.

2) Integration with communication scenarios: There is a growing interest in incorporating machine learning into communication networks, such as using ML for signal detection,

channel estimation, and network optimization. ZKP-VML could help ensure the privacy and integrity of ML computations in these contexts. However, integrating ZKPs with existing wireless communication protocols remains a challenge, particularly in ensuring compatibility with both centralized and decentralized systems. Exploring dedicated proof systems for specific wireless communication tasks and optimizing proof schemes for real-time operation could address both latency and scalability concerns. At the same time, communication networks introduce new application scenarios and ML tasks for ZKP-VML, such as semantic communication. On one hand, semantic communication introduces new verifiable ML tasks for ZKP-VML, including novel encoding-decoding processes and semantic data verification. On the other hand, semantic communication may also contribute to enhancing the communication and computational efficiency of ZKP-VML. These are promising directions for further research.

VIII. Conclusion

This paper provides a comprehensive review of zero-knowledge proof-based verifiable machine learning (ZKP-VML). First, we provide an introduction to DML, ZKPs, and communication networks, emphasizing how ZKPs can address the security challenges inherent in DML. We then formally define ZKP-VML, including its underlying algorithms and key properties, while also discussing associated challenges and applications. Next, we offer a comprehensive overview of the existing schemes, detailing the research timeline in this area and how the properties of ZKP-VML are realized across different approaches. For each scheme, we analyze the technical route adopted, presenting a well-structured categorization. Furthermore, we examine the optimization method utilized within these schemes, comparing their performance in terms of computational and communication complexity. Finally, we discuss the challenges and future research directions in ZKP-VML. We hope this work will inspire further research into the field of ZKP-VML.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (Grant No. 2023YFF0905300), the National Natural Science Foundation of China (No.U2468205), the Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China), under grant No. BigKEOpen2025-04, the National Natural Science Foundation of China (Grant No. 62372149, No. U23A20303, No. 62372020, No. 72031007, No. 62472036, No. 62202258), the "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province (Grant No. 2024C01073) and China Scholarship Council (No.202406690030, No. 202206030042).

References

[1] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[2] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus," in *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom)*. IEEE, 2016, pp. 477–484.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[4] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 365–11 375, 2021.

[5] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.

[6] X. Ma, X. Zhang, C. Dong, and X. Chen, "A survey on secure outsourced deep learning," in *Cyber Security Meets Machine Learning*. Springer, 2021, pp. 129–163.

[7] H. Qin, D. He, Q. Feng, M. K. Khan, M. Luo, and K.-K. R. Choo, "Cryptographic primitives in privacy-preserving machine learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[8] J. Chen, H. Yan, Z. Liu, M. Zhang, H. Xiong, and S. Yu, "When federated learning meets privacy-preserving computation," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–36, 2024.

[9] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang, "Decentralized federated learning: A survey on security and privacy," *IEEE Transactions on Big Data*, 2024.

[10] Y. Zhang and H. Yu, "Towards verifiable federated learning," *arXiv preprint arXiv:2202.08310*, 2022.

[11] A. Tariq, M. A. Serhani, F. M. Sallabi, E. S. Barka, T. Qayyum, H. M. Khater, and K. A. Shuaib, "Trustworthy federated learning: A comprehensive review, architecture, key challenges, and future research prospects," *IEEE Open Journal of the Communications Society*, 2024.

[12] M. Labs, "The cost of intelligence: Proving machine learning inference with zero-knowledge," https://drive.google.com/file/d/1tylpowpaqcOhKQtYolPlqvx6R2Gv4IzE/view, 2023.

[13] A. Sathe, V. Saxena, P. Akshay Bharadwaj, and S. Sandosh, "State of the art in zero-knowledge machine learning: A comprehensive survey," in *International Conference on Advancements in Smart Computing and Information Security*. Springer, 2023, pp. 98–110.

[14] Y. Zhang and Z. Fan, "Research on zero knowledge with machine learning," *Journal of Computing and Electronic Information Management*, vol. 12, no. 2, pp. 105–108, 2024.

[15] V. Keršič, S. Karakatič, and M. Turkanović, "On-chain zero-knowledge machine learning: An overview and comparison," *Journal of King Saud University-Computer and Information Sciences*, p. 102207, 2024.

[16] 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.

[17] 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.

[18] 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.

[19] Z. Xing, Z. Zhang, M. Li, J. Liu, L. Zhu, G. Russello, and M. R. Asghar, "Zero-knowledge proof-based practical federated learning on blockchain," *arXiv preprint arXiv:2304.05590*, 2023.

[20] 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.

[21] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–39, 2019.

[22] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[23] D. Maulud and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 4, pp. 140–147, 2020.

[24] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[25] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[26] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, 1985, pp. 291–304.

[27] M. Blum, A. De Santis, S. Micali, and G. Persiano, "Noninteractive zero-knowledge," *SIAM Journal on Computing*, vol. 20, no. 6, pp. 1084–1118, 1991.

[28] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *Journal of computer and system sciences*, vol. 37, no. 2, pp. 156–189, 1988.

[29] L. Wei-Han, Z. Zong-Yang, Z. Zi-Bo, and D. Yi, "An overview on succinct non-interactive zero-knowledge proofs," *Journal of Cryptologic Research*, vol. 9, no. 3, pp. 379–447, 2022.

[30] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.

[31] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems." in *Crypto*, vol. 86. Springer, 1986, pp. 186–194.

[32] Y. T. Kalai, G. N. Rothblum, and R. D. Rothblum, "From obfuscation to the security of fiat-shamir for proofs," in *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*. Springer, 2017, pp. 224–251.

[33] Y. Chen, A. Lombardi, F. Ma, and W. Quach, "Does fiat-shamir require a cryptographic hash function?" in *Annual International Cryptology Conference*. Springer, 2021, pp. 334–363.

[34] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM (JACM)*, vol. 39, no. 4, pp. 859–868, 1992.

[35] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 926–943.

[36] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer, 2013, pp. 626–645.

[37] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988, pp. 103–112.

[38] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer, 2016, pp. 327–357.

[39] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.

[40] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *International symposium on symbolic and algebraic manipulation*. Springer, 1979, pp. 216–226.

[41] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 21–30.

[42] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of CRYPTOLOGY*, vol. 13, pp. 143–202, 2000.

[43] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[44] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[45] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings 5*. Springer, 2008, pp. 1–19.

[46] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/Ispa*, vol. 1. IEEE, 2015, pp. 57–64.

[47] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*. Springer, 2010, pp. 465–482.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 210–219, 2022.

[50] Y. Cheng, D. Niyato, H. Du, C. Miao, and D. I. Kim, "Goal-oriented semantic information transmission with message-sharing noma," *IEEE Wireless Communications*, 2024.

[51] Q. Hu, G. Zhang, Z. Qin, Y. Cai, G. Yu, and G. Y. Li, "Robust semantic communications with masked vq-vae enabled codebook," *IEEE Transactions on Wireless Communications*, vol. 22, no. 12, pp. 8707–8722, 2023.

[52] X. Luo, Z. Chen, M. Tao, and F. Yang, "Encrypted semantic communication using adversarial training for privacy preserving," *IEEE Communications Letters*, vol. 27, no. 6, pp. 1486–1490, 2023.

[53] Y. Lin, H. Du, D. Niyato, J. Nie, J. Zhang, Y. Cheng, and Z. Yang, "Blockchain-aided secure semantic communication for ai-generated content in metaverse," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 72–83, 2023.

[54] A. D. Dwivedi, R. Singh, U. Ghosh, R. R. Mukkamala, A. Tolba, and O. Said, "Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for internet of things," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–11, 2022.

[55] Y. Kawamoto, M. Takahashi, S. Verma, N. Kato, H. Tsuji, and A. Miura, "Traffic prediction-based dynamic resource control strategy in haps-mounted mec-assisted satellite communication systems," *IEEE Internet of Things Journal*, 2023.

[56] T. K. Rodrigues, S. Verma, Y. Kawamoto, N. Kato, M. M. Fouda, and M. Ismail, "Smart handover with predicted user behavior using convolutional neural networks for wigig systems," *IEEE Network*, 2024.

[57] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[58] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "{DIZK}: A distributed zero knowledge proof system," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 675–692.

[59] Z. Wan, Y. Zhou, and K. Ren, "zk-authfeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[60] D. Froelicher, J. R. Troncoso-Pastoriza, J. S. Sousa, and J.-P. Hubaux, "Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3035–3050, 2020.

[61] J. Guo, Z. Liu, K.-Y. Lam, J. Zhao, Y. Chen, and C. Xing, "Secure weighted aggregation for federated learning," *arXiv preprint arXiv:2010.08730*, 2020.

[62] C. Sabater, A. Bellet, and J. Ramon, "An accurate, scalable and verifiable protocol for federated differentially private averaging," *Machine Learning*, pp. 1–45, 2022.

[63] C. Ju, H. Lee, H. Chung, J. H. Seo, and S. Kim, "Efficient sum-check protocol for convolution," *IEEE Access*, vol. 9, pp. 164 047–164 059, 2021.

[64] S. Ghaffaripour and A. Miri, "Mutually private verifiable machine learning as-a-service: A distributed approach," in *2021 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2021, pp. 0232–0239.

[65] J. Keuffer, R. Molva, and H. Chabanne, "Efficient proof composition for verifiable computation," in *European Symposium on Research in Computer Security*. Springer, 2018, pp. 152–171.

[66] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "Veriml: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2524–2540, 2021.

[67] J. Zhang, Z. Fang, Y. Zhang, and D. Song, "Zero knowledge proofs for decision tree predictions and accuracy," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 2039–2053.

[68] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 453–476.

[69] A. S. Shamsabadi, S. C. Wyllie, N. Franzese, N. Dullerud, S. Gambs, N. Papernot, X. Wang, and A. Weller, "Confidential-profitt: confidential

proof of fair training of trees," in *The Eleventh International Conference on Learning Representations*, 2022.

[70] M. Hao, H. Chen, H. Li, C. Weng, Y. Zhang, H. Yang, and T. Zhang, "Scalable zero-knowledge proofs for non-linear functions in machine learning," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 3819–3836.

[71] T. Lu, H. Wang, W. Qu, Z. Wang, J. He, T. Tao, W. Chen, and J. Zhang, "An efficient and extensible zero-knowledge proof framework for neural networks," *Cryptology ePrint Archive*, 2024.

[72] W. Wu, S. Homsi, and Y. Zhang, "Confidential and verifiable machine learning delegations on the cloud," in *European Symposium on Research in Computer Security*. Springer, 2024, pp. 182–201.

[73] H. Duan, Z. Peng, L. Xiang, Y. Hu, and B. Li, "A verifiable and privacy-preserving federated learning training framework," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[74] K. Abbaszadeh, C. Pappas, J. Katz, and D. Papadopoulos, "Zero-knowledge proofs of training for deep neural networks," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4316–4330.

[75] J. Weng, J. Weng, G. Tang, A. Yang, M. Li, and J.-N. Liu, "pvcnn: Privacy-preserving and verifiable convolutional neural network testing," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2218–2233, 2023.

[76] B. Feng, Z. Wang, Y. Wang, S. Yang, and Y. Ding, "Zeno: A type-based optimization framework for zero knowledge neural network inference," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2024, pp. 450–464.

[77] Y. Fan, B. Xu, L. Zhang, J. Song, A. Zomaya, and K.-C. Li, "Validating the integrity of convolutional neural network predictions based on zero-knowledge proof," *Information Sciences*, 2023.

[78] Y. Fan, K. Ma, L. Zhang, X. Lei, G. Xu, and G. Tan, "Validcnn: A large-scale cnn predictive integrity verification scheme based on zk-snark," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[79] Y. Fan, K. Ma, L. Zhang, J. Liu, N. Xiong, and S. Yu, "Vericnn: Integrity verification of large-scale cnn training process based on zk-snark," *Expert Systems with Applications*, p. 124531, 2024.

[80] 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.

[81] M. Campanelli, A. Faonio, D. Fiore, T. Li, and H. Lipmaa, "Lookup arguments: improvements, extensions and applications to zero-knowledge decision trees," in *IACR International Conference on Public-Key Cryptography*. Springer, 2024, pp. 337–369.

[82] N. Singh, P. Dayama, and V. Pandit, "Zero knowledge proofs towards verifiable decentralized ai pipelines," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 248–275.

[83] T. South, A. Camuto, S. Jain, S. Nguyen, R. Mahari, C. Paquin, J. Morton, and A. Pentland, "Verifiable evaluations of machine learning models using zksnarks," *arXiv preprint arXiv:2402.02675*, 2024.

[84] G. Keshavarzkalhori, C. Pérez-Solà, G. Navarro-Arribas, J. Herrera-Joancomartí, and H. Yajam, "Federify: a verifiable federated learning scheme based on zksnarks and blockchain," *IEEE Access*, 2023.

[85] H. Wang and T. Hoang, "ezdps: An efficient and zero-knowledge machine learning inference pipeline," *arXiv preprint arXiv:2212.05428*, 2022.

[86] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Scaling up trustless dnn inference with zero-knowledge proofs," *arXiv preprint arXiv:2210.08674*, 2022.

[87] B.-J. Chen, S. Waiwitlikhit, I. Stoica, and D. Kang, "Zkml: An optimizing system for ml inference in zero-knowledge proofs," in *Proceedings of the Nineteenth European Conference on Computer Systems*, 2024, pp. 560–574.

[88] N. Attrapadung, G. Hanaoaka, R. Hiromasa, Y. Koseki, T. Matsuda, Y. Nishida, Y. Sakai, J. C. Schuldt, and S. Yasuda, "Privacy-preserving verifiable cnns," in *International Conference on Applied Cryptography and Network Security*. Springer, 2024, pp. 373–402.

[89] S. Garg, A. Goel, S. Jha, S. Mahloujifar, M. Mahmoody, G.-V. Policharla, and M. Wang, "Experimenting with zero-knowledge proofs of training," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1880–1894.

[90] A. S. Shamsabadi, G. Tan, T. I. Cebere, A. Bellet, H. Haddadi, N. Papernot, X. Wang, and A. Weller, "Confidential-dpproof: Confidential proof of differentially private training," in *International Conference on Learning Representations (ICLR)*, 2024.

[91] S. Angel, A. J. Blumberg, E. Ioannidis, and J. Woods, "Efficient representation of numerical optimization problems for {SNARKs}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4273–4290.

[92] T. Rückel, J. Sedlmeir, and P. Hofmann, "Fairness, integrity, and privacy in a scalable blockchain-based federated learning system," *Computer Networks*, vol. 202, p. 108621, 2022.

[93] C. Huang, J. Wang, H. Chen, S. Si, Z. Huang, and J. Xiao, "zkmlaas: a verifiable scheme for machine learning as a service," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 5475–5480.

[94] Z. Zhou, X. Cao, J. Liu, B. Zhang, and K. Ren, "Zero knowledge contingent payments for trained neural networks," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26*. Springer, 2021, pp. 628–648.

[95] J. Bell, A. Gascón, T. Lepoint, B. Li, S. Meiklejohn, M. Raykova, and C. Yun, "{ACORN}: input validation for secure aggregation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4805–4822.

[96] Y. Zhu, Y. Wu, Z. Luo, B. C. Ooi, and X. Xiao, "Secure and verifiable data collaboration with low-cost zero-knowledge proofs," *arXiv preprint arXiv:2311.15310*, 2023.

[97] Z. Xing, Z. Zhang, Z. Zhang, J. Liu, L. Zhu, and G. Russello, "No vandalism: Privacy-preserving and byzantine-robust federated learning," *arXiv preprint arXiv:2406.01080*, 2024.

[98] J. Ma, H. Liu, M. Zhang, and Z. Liu, "Vpfl: Enabling verifiability and privacy in federated learning with zero-knowledge proofs," *Knowledge-Based Systems*, p. 112115, 2024.

[99] F. Wang, Y. He, Y. Guo, P. Li, and X. Wei, "Privacy-preserving robust federated learning with distributed differential privacy," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 598–605.

[100] A. Roy Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "Eiffel: Ensuring integrity for federated learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2535–2549.

[101] Z. Ghodsi, M. Javaheripi, N. Sheybani, X. Zhang, K. Huang, and F. Koushanfar, "zprobe: Zero peek robustness checks for federated learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4860–4870.

[102] T. Nguyen and M. T. Thai, "Preserving privacy and security in federated learning," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 833–843, 2023.

[103] S. Han, W. Wu, B. Buyukates, W. Jin, Y. Yao, Q. Zhang, S. Avestimehr, and C. He, "Kick bad guys out! zero-knowledge-proof-based anomaly detection in federated learning," *arXiv preprint arXiv:2310.04055*, 2023.

[104] B. Buyukates, C. He, S. Han, Z. Fang, Y. Zhang, J. Long, A. Farahanchi, and S. Avestimehr, "Proof-of-contribution-based design for collaborative machine learning on blockchain," in *2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2023, pp. 13–22.

[105] G. Tang, W. Tan, and M. Cai, "Privacy-preserving and trustless verifiable fairness audit of machine learning models," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 2, 2023.

[106] E. Toreini, M. Mehrnezhad, and A. Van Moorsel, "Verifiable fairness: Privacy–preserving computation of fairness for machine learning systems," in *European Symposium on Research in Computer Security*. Springer, 2023, pp. 569–584.

[107] Z. Wang, N. Dong, J. Sun, W. Knottenbelt, and Y. Guo, "zkfl: Zero-knowledge proof-based gradient aggregation for federated learning," *IEEE Transactions on Big Data*, 2024.

[108] M. Ahmadi and R. Nourmohammadi, "zkfdl: An efficient and privacy-preserving decentralized federated learning with zero knowledge proof," in *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*. IEEE, 2024, pp. 1–10.

[109] X. Liu, X. Yang, X. Zhang, and X. Yang, "Evaluate and guard the wisdom of crowds: Zero knowledge proofs for crowdsourcing truth inference," *arXiv preprint arXiv:2308.00985*, 2023.

[110] W. Yang, Y. Yin, G. Zhu, H. Gu, L. Fan, X. Cao, and Q. Yang, "Fedzkp: Federated model ownership verification with zero-knowledge proof," *arXiv preprint arXiv:2305.04507*, 2023.

[111] N. Sheybani, Z. Ghodsi, R. Kapila, and F. Koushanfar, "Zkrownn: Zero knowledge right of ownership for neural networks," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[112] T. Eisenhofer, D. Riepel, V. Chandrasekaran, E. Ghosh, O. Ohrimenko, and N. Papernot, "Verifiable and provably secure machine unlearning," *arXiv preprint arXiv:2210.09126*, 2022.

[113] H. Sun, J. Li, and H. Zhang, "zkllm: Zero knowledge proofs for large language models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4405–4419.

[114] B.-M. Ganescu and J. Passerat-Palmbach, "Trust the process: Zero-knowledge machine learning to enhance trust in generative ai interactions," *arXiv preprint arXiv:2402.06414*, 2024.

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

[116] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[117] K. Yang, P. Sarkar, C. Weng, and X. Wang, "Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2986–3001.

[118] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl, "Improved primitives for mpc over mixed arithmetic-binary circuits," in *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II 40*. Springer, 2020, pp. 823–852.

[119] "Rosetta: A privacy-preserving framework based on tensorflow," https://github.com/LatticeX-Foundation/Rosetta.

[120] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[121] Y. Yang and D. Heath, "Two shuffles make a {RAM}: Improved constant overhead zero knowledge {RAM}," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1435–1452.

[122] C. Weng, K. Yang, J. Katz, and X. Wang, "Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1074–1091.

[123] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[124] 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.

[125] S. Setty, J. Thaler, and R. Wahby, "Unlocking the lookup singularity with lasso," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2024, pp. 180–209.

[126] A. Gabizon and Z. J. Williamson, "plookup: A simplified polynomial protocol for lookup tables," *Cryptology ePrint Archive*, 2020.

[127] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[128] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," *Journal of the ACM (JACM)*, vol. 62, no. 4, pp. 1–64, 2015.

[129] A. Ozdemir and D. Boneh, "Experimenting with collaborative {zk-SNARKs}:{Zero-Knowledge} proofs for distributed secrets," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4291–4308.

[130] J. Zhang, T. Xie, Y. Zhang, and D. Song, "Transparent polynomial delegation and its applications to zero knowledge proof," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 859–876.

[131] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-quantum and transparent recursive proofs from holography," in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*. Springer, 2020, pp. 769–793.

[132] S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup," *Cryptology ePrint Archive*, 2019.

[133] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.

[134] N. Gailly, M. Maller, and A. Nitulescu, "Snarkpack: practical snark aggregation," in *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Springer, 2022, pp. 203–229.

[135] "snark: Interfaces for relations and snarks for these relations," https://github.com/arkworks-rs/snark.

[136] "zk-snark library," https://github.com/zkcrypto/bellman.

[137] "Ginger-lib is a general purpose zk-snark library that supports recursive proof composition," https://github.com/HorizenOfficial/ginger-lib.

[138] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006.

[139] R. Freivalds, "Probabilistic machines can use less running time." in *IFIP congress*, vol. 839, 1977, p. 842.

[140] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4013–4021.

[141] I. Korec and J. Wiedermann, "Deterministic verification of integer matrix multiplication in quadratic time," in *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 2014, pp. 375–382.

[142] L. Eagen, D. Fiore, and A. Gabizon, "cq: Cached quotients for fast lookups," *Cryptology ePrint Archive*, 2022.

[143] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*. Springer, 2010, pp. 177–194.

[144] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez, "Lunar: a toolbox for more efficient universal and updatable zk-snarks and commit-and-prove extensions," in *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*. Springer, 2021, pp. 3–33.

[145] S. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup," in *Annual International Cryptology Conference*. Springer, 2020, pp. 704–737.

[146] C. Vonesch, T. Blu, and M. Unser, "Generalized daubechies wavelet families," *IEEE transactions on signal processing*, vol. 55, no. 9, pp. 4415–4429, 2007.

[147] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[148] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin, "Radius margin bounds for support vector machines with the rbf kernel," *Neural computation*, vol. 15, no. 11, pp. 2643–2681, 2003.

[149] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[150] T. E. C. Company, "The halo2 book," https://zcash.github.io/halo2/index.html, 2021.

[151] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *Cryptology ePrint Archive*, 2019.

[152] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[153] I. Damgård and J. B. Nielsen, "Universally composable efficient multiparty computation from threshold homomorphic encryption," in *Annual international cryptology conference*. Springer, 2003, pp. 247–264.

[154] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Annual Cryptology Conference*. Springer, 2012, pp. 643–662.

[155] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[156] N. Franzese, J. Katz, S. Lu, R. Ostrovsky, X. Wang, and C. Weng, "Constant-overhead zero-knowledge for ram programs," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 178–191.

[157] X. Wang, A. J. Malozemoff, and J. Katz, "Emp-toolkit: Efficient multiparty computation toolkit," https://github.com/emp-toolkit, 2016.

[158] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer, 2019, pp. 103–128.

[159] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[160] R. Motwani and P. Raghavan, "Randomized algorithms," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 33–37, 1996.

[161] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in *Annual cryptology conference*. Springer, 2013, pp. 90–108.

[162] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*. Springer, 2019, pp. 733–764.

[163] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vsql: Verifying arbitrary sql queries over dynamic outsourced databases," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 863–880.

[164] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[165] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the intrinsic dimension of objective landscapes," *arXiv preprint arXiv:1804.08838*, 2018.

[166] C. Gentry, S. Halevi, and V. Lyubashevsky, "Practical non-interactive publicly verifiable secret sharing with thousands of parties," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022, pp. 458–487.

[167] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3*. Springer, 2005, pp. 467–482.

[168] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in neural information processing systems*, vol. 30, 2017.

[169] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[170] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous sgd," in *International conference on machine learning*. PMLR, 2020, pp. 10 495–10 503.

[171] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[172] D. N. Yaldiz, T. Zhang, and S. Avestimehr, "Secure federated learning against model poisoning attacks via client filtering," *arXiv preprint arXiv:2304.00160*, 2023.

[173] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[174] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, 2017, pp. 259–282.

[175] S. Lin and D. J. C. Jr., *Error control coding - fundamentals and applications*, ser. Prentice Hall computer applications in electrical engineering series. Prentice Hall, 1983.

[176] J. Nelson, "Sketching algorithms," 2020.

[177] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[178] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[179] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273–294.

[180] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 326–349.

[181] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[182] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

[183] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[184] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[185] B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely, "Proofs for inner pairing products and applications," in *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*. Springer, 2021, pp. 65–97.

[186] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Annual International Cryptology Conference*. Springer, 1994, pp. 174–187.

[187] G. Xu, H. Li, S. Xu, H. Ren, Y. Zhang, J. Sun, and R. H. Deng, "Catch you if you deceive me: Verifiable and privacy-aware truth discovery in crowdsensing systems," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 178–192.

[188] A. Jain, S. Krenn, K. Pietrzak, and A. Tentes, "Commitments and efficient zero-knowledge proofs from learning parity with noise," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 663–680.

[189] B. Li, L. Fan, H. Gu, J. Li, and Q. Yang, "Fedipr: Ownership verification for federated deep neural network models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4521–4536, 2022.

[190] I. Damgård, "On σ-protocols," *Lecture Notes, University of Aarhus, Department for Computer Science*, vol. 84, 2002.

[191] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," *arXiv preprint arXiv:1606.09274*, 2016.

[192] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," *arXiv preprint arXiv:2108.11577*, 2021.

[193] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 516–11 524.

[194] Z. Inc., "What is ezkl?" https://docs.ezkl.xyz/.

[195] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish, "Taking {Proof-Based} verified computation a few steps closer to practicality," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 253–268.

[196] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, and G. Sun, "Pipezk: Accelerating zero-knowledge proof with a pipelined architecture," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 416–428.

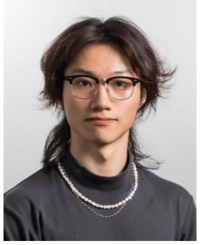
[197] H. Zhao, D. Ding, F. Wang, P. Hua, N. Wang, Q. Wu, and Z. Chai, "Hardware acceleration of number theoretic transform for zk-snark," *Engineering Reports*, p. e12639, 2022.

[198] W. Ma, Q. Xiong, X. Shi, X. Ma, H. Jin, H. Kuang, M. Gao, Y. Zhang, H. Shen, and W. Hu, "Gzkp: A gpu accelerated zero-knowledge proof system," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2023, pp. 340–353.

[199] T. Lu, C. Wei, R. Yu, C. Chen, W. Fang, L. Wang, Z. Wang, and W. Chen, "Cuzk: Accelerating zero-knowledge proof with a faster parallel multi-scalar multiplication algorithm on gpus," *IACR Transactions*

on Cryptographic Hardware and Embedded Systems, vol. 2023, no. 3, pp. 194–220, 2023.

[200] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training," *Proceedings on Privacy Enhancing Technologies*, 2019.

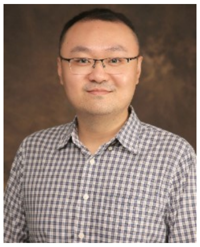


Zhibo Xing received the B.E. degree in Computer Science and Technology from Beijing Institute of Technology in 2021. He is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is also a joint Ph.D. student with the School of Computer Science, University of Auckland. His current research interests include zero-knowledge proofs and machine learning security.

Zijian Zhang (Senior Member, IEEE) received the Ph.D. degree from Beijing Institute of Technology. He is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include authentication and key agreement, behavior recognition, and privacy-preserving.

Zi'ang Zhang received the B.E. degree in Computer Science and Technology from Beijing Institute of Technology in 2023. He is currently pursuing the M.E. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His current research interests include zero-knowledge proofs and related technologies.

Zhen Li is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include privacy computing, AI security and blockchain.

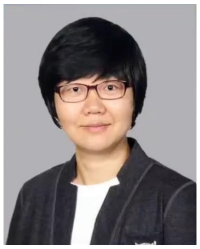


Meng Li (Senior Member, IEEE) is an Associate Professor and Personnel Secretary at the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He is also a Post-Doc Researcher at Department of Mathematics and HIT Center, University of Padua, Italy, where he is with the Security and PRIvacy Through Zeal (SPRITZ) research group led by Prof. Mauro Conti (IEEE Fellow). He obtained his Ph.D. in Computer Science and Technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He was sponsored by ERCIM 'Alain Bensoussan' Fellowship Programme (from 2020.10.1 to 2021.3.31) to conduct Post-Doc research supervised by Prof. Fabio Martinelli at CNR, Italy. He was sponsored by China Scholarship Council (CSC) (from 2017.9.1 to 2018.8.31) for joint Ph.D. study supervised by Prof. Xiaodong Lin (IEEE Fellow) in the Broadband Communications Research (BBCR) Lab at University of Waterloo and Wilfrid Laurier University, Canada. His research interests include security, privacy, applied cryptography, blockchain, TEE, and Internet of Vehicles. In this area, he has published 104 papers in topmost journals and conferences, including TIFS, TDSC, ToN, TMC, TKDE, TODS, TSC, COMST, ISSTA, and MobiCom. He is a Senior Member of IEEE, CIE, CIC, and CCF. He is an Associate Editor for TIFS, TDSC, EURASIP JIS, TNSM, IoTJ, COMNET, and Scientific Reports. He has served as a TPC member for conferences, including ICDCS, TrustCom, ICC, Globecom, and HPCC. He is the recipient of 2024 IEEE HITC Award for Excellence (Early Career Researcher).

Jiamou Liu received the Ph.D. degree in computer science from the University of Auckland, Auckland, New Zealand, in 2010. He is currently an Associate Professor with the School of Computer Science, University of Auckland. He was a senior lecturer with the Auckland University of Technology, Auckland, from 2011 to 2015 and a researcher with the Department of Computer Science, Leipzig University, Leipzig, Germany, from 2009 to 2010. His current research interests include social network analysis, multiagent systems, and algorithms.

Zongyang Zhang (Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University. He is currently an Associate Professor with the School of Cyber Science and Technology, Beihang University. His work has been published in several top-tier conferences and journals, including Cell Patterns, TIFS, TDSC, USENIX Security. His research interests include blockchain and cryptography.

Yi Zhao (Member, IEEE) received the B.E. degree in Software Engineering from Northwestern Polytechnical University, Xi'an, China, in 2016, and the Ph.D. degree in Computer Science and Technology from Tsinghua University, Beijing, China, in 2021. From 2021 to 2023, he was a Postdoctoral Researcher and Shuimu Scholar with the Department of Computer Science and Technology, Tsinghua University. He is currently an Associate Researcher with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include network security and machine learning. He is a member of ACM. He was a recipient of the Shuimu Tsinghua Scholar Program.



Qi Sun received the Ph.D. degree in Computer Science from the University of Kentucky, Lexington, KY, USA, and conducted postdoctoral research at Virginia Commonwealth University, Richmond, VA, USA. She is currently an Algorithm Scientist at Hangzhou Nuowei Information Technology Co.,Ltd. Her research interests include privacy-preserving computation and biomedical informatics.

Liehuang Zhu (Senior Member, IEEE) is currently a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, China. He has published over 60 SCI-indexed research articles in these areas and a book published by Springer. His research interests include the Internet of Things, cloud computing security, internet, and mobile security. He won the Best Paper Award at IEEE/ACM IWQoS 2017 and IEEE TrustCom 2018. He serves on the editorial boards of three international journals, including IEEE Internet of Things Journal, IEEE Network, and IEEE Transactions on Vehicular Technology.

Giovanni Russello received the MSc (summa cum laude) degree in computer science from the University of Catania, Italy, in 2000, and the PhD degree from the Eindhoven University of Technology (TU/e), in 2006. After receiving the PhD degree, he moved to the Policy Group in the Department of Computing, Imperial College London, United Kingdom. He is currently a Professor with the School of Computer Science, University of Auckland, New Zealand. He is also the founding director of the Cyber Security Foundry, the first New Zealand multidisciplinary centre in cyber security. His research interests include policy-based security systems, privacy and confidentiality in cloud computing, smartphone security, and applied cryptography.