zkDL: Efficient Zero-Knowledge Proofs of Deep Learning Training

Haochen Sun
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Email: haochen.sun@uwaterloo.ca

Hongyang Zhang
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Email: hongyang.zhang@uwaterloo.ca

***Abstract*—The recent advancements in deep learning have brought about significant changes in various aspects of people's lives. Meanwhile, these rapid developments have raised concerns about the legitimacy of the training process of deep networks. However, to protect the intellectual properties of untrusted AI developers, directly examining the training process by accessing the model parameters and training data by verifiers is often prohibited.**

In response to this challenge, we present zkDL, an efficient zero-knowledge proof of deep learning training. At the core of zkDL is zkReLU, a specialized zero-knowledge proof protocol with optimized proving time and proof size for the ReLU activation function, a major obstacle in verifiable training due to its non-arithmetic nature. To integrate zkReLU into the proof system for the entire training process, we devise a novel construction of an arithmetic circuit from neural networks. By leveraging the abundant parallel computation resources, this construction reduces proving time and proof sizes by a factor of the network depth. As a result, zkDL enables the generation of complete and sound proofs, taking less than a minute with a size of less than 20 kB per training step, for a 16-layer neural network with 200M parameters, while ensuring the privacy of data and model parameters.

1. Introduction

The rapid development of deep learning has garnered unprecedented attention in the decade. However, along with these advancements, concerns about the legitimacy of deep learning training have also arisen. In March 2023, Italy became the first Western country to ban ChatGPT amid an investigation into a potential violation of the European Union's General Data Protection Regulation (GDPR). Furthermore, in January 2023, Stable Diffusion, a star image generative model, faced accusations from a group of artist representatives regarding the infringement of copyrights on millions of images in its training data. As governments continue to impose new regulatory requirements on increasingly advanced AI technologies, there is an urgent need to develop a protocol that can verify the legitimacy of the training data and process for deep learning models. However, due to concerns related to intellectual property and business secrets, model owners are generally hesitant to disclose their proprietary training data or model snapshots for legitimacy investigations.

Despite considerable efforts in verifiable machine learning to address the aforementioned dilemma, numerous fundamental questions remain unanswered. Currently, cryptography-based approaches have primarily concentrated on inference-time verification [1], [2], [3], [4], [5], leaving training-time verification relatively unexplored due to the substantial computational demands and intricate operations involved. Pioneering works in verifiable training [6], [7] have primarily focused on elementary machine learning algorithms, such as linear regression, logistic regression, and SVMs. The deep learning algorithms explored have been strictly constrained by the size of the neural networks that the backend of the proof system can efficiently handle and are therefore not extendable to modern deep neural networks.

Furthermore, the progress of verifiable deep learning has been hindered by non-arithmetic operations, particularly activation functions like ReLU, which are not inherently supported by zero-knowledge proof (ZKP) systems but extensively used in the realm of deep learning. Early works have explored square activation and polynomial approximation as arithmetic alternatives to traditional activation functions [6], [7], [8]. However, these alternatives deviate from well-established deep learning structures, raising concerns about the effectiveness of these models. More recent approaches have attempted to handle activation functions using bit-decompositions and look-up tables [1], [5]. Unfortunately, these methods impose significant demands on computational resources due to the high space and time complexities they introduce. Additionally, the modeling of neural networks as arithmetic circuits that are compatible with handling activation functions remains unclear. Given the current state of verifiable deep learning, overcoming the bottleneck posed by activation functions like ReLU is crucial before practical applications can be deployed.

In addition to ensuring training correctness, legitimacy concerns also emerge from the composition of training data, particularly with regard to copyright issues. However, the training set's substantial size and its status as the trainer's intellectual property make it impractical to publish the dataset for inspection. To address this challenge, an efficient and bidirectionally privacy-preserving protocol is needed to demonstrate whether queried data points are (not) included in the training set.

To address these challenges, we propose *zkDL*, an efficient zero-knowledge proof of deep learning training. With zkDL, model developers can demonstrate to the government that the model has been correctly trained in accordance with the committed data, adhering to the prescribed training logic. This approach effectively addresses concerns about the legitimacy of the model. Our main contributions are summarized as follows:

- We introduce *zkReLU*, an efficient specialized zero-knowledge proof tailored for the *exact* computation of the Rectified Linear Unit (ReLU) and its backpropagation, without resorting to polynomial approximations to handle non-arithmetic operations. Compared with the general-purpose zero-knowledge proof (ZKP) backends that impose an infeasible overhead, zkReLU constitutes a significant stride towards achieving verifiable deep learning training for industrial applications.
- We explicitly designed the modelling scheme of the neural network with ReLU activations as arithmetic circuits and developed a highly efficient zero-knowledge proof system over the circuit. This proof system is compatible with the tensor-based structure of deep learning and seamlessly connects with zkReLU, and achieves zero knowledge on both the model parameters and the training dataset.
- Thanks to the remarkably high degree of parallelization in zkDL, the proving time and proof size experience slow growth with an additive $O(\log L)$ term as the depth L of the neural network increases. This scalability allows zkDL to accommodate large neural networks effectively. For instance, on a 16-layer network with over 200 million parameters and a batch size of 128 on the CIFAR-10 dataset, the proof generation time and proof size are limited to *less than 1 minute* and 20 kB per batch update (i.e., forward and backpropagation process), achieving over $40\times$ and $10\times$ improvements compared to bit-decomposition-based proofs, respectively.
- As a by-product, zkDL also enables efficient queries from data copyright owners regarding the membership status of their data in the training set, thereby addressing copyright-related legitimacy issues. On the CIFAR-10 task, it only takes 0.05 milliseconds for a data copyright owner to confirm that a queried data point has not been included in the training set.

1.1. Overview of zkDL

The zkDL protocol relies on the presence of a *trusted verifier* (e.g., the government) that faithfully follows all the prescribed protocols in its role as the verifier and provides honest verification results (accept/reject). The development of the zkDL protocol follows the roadmap outlined below:

zkReLU with auxiliary inputs. ReLU, a pivotal activation function in modern neural networks, plays a crucial role in their success. However, its non-arithmetic nature poses a significant challenge when proving training with general-purpose ZKP backends. As depicted in Figure 1, handling bit-decompositions related to ReLU consumes over 90% of the computational power. Meanwhile, the recently proposed lookup-table-based handling of ReLU has been limited to verifiable inferences in quantized `int8` networks [5], rendering it unsuitable for training due to its prohibitive memory consumption.

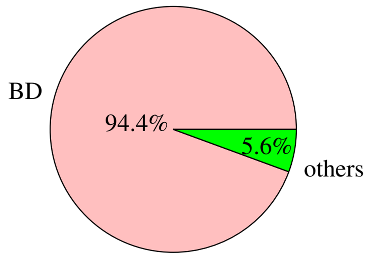


Figure 1: Proving Time Allocated for Bit-Decomposition (BD, shown in red) using General-Purpose Sum-Check Protocol: average across tested architectures in Section 5.1 (timeouts excluded). The ratio was obtained by re-running the experiment with all BD-related components removed.

To ensure the efficiency of the zkDL protocol, we introduce zkReLU, a specialized zero-knowledge proof tailored specifically for ReLU. zkReLU revolutionizes the proof of the non-arithmetic operations of ReLU and its backward propagation by introducing auxiliary inputs. Given the potential for malicious manipulation of the values of the auxiliary inputs, zkReLU incorporates additional validity proofs on the auxiliary inputs rooted in the homomorphic commitment scheme. Unlike traditional proofs with general-purpose ZKP backends, zkReLU retains the tensor-based structures central to deep learning and significantly reduces redundancy through the reuse of verified claims on committed auxiliary inputs during the proof of arithmetic relations. Furthermore, zkReLU also reduces the correctness of training to a single inner-product proof, so as to exploit its linear proving time and logarithmic proof size. This reduction is accomplished through transformations of the commitments to achieve compatibility with different proof systems for the arithmetic relations and auxiliary input validity. The efficacy of these approaches establishes zkReLU as a central component for enabling efficient verifiable training with ReLU in the zkDL framework.

Design of the arithmetic circuit. Constructing an arithmetic circuit with a layered structure for forward and backward propagation, and integrating it into a ZKP backend is indeed a straightforward process. However, this approach introduces unconnected gaps in the arithmetic circuit due to the non-arithmetic nature of ReLU. These gaps create exploitable opportunities for malicious trainers. To address this concern, in Section 4.2, we propose a rewiring of the arithmetic circuit. This rewiring anchors the arithmetic operations within each layer to their arithmetic relations with auxiliary inputs. As a result, the anchored circuit architecture not only fills the connectivity gaps but also facilitates a high degree of parallelization, leading to significant improvements in proving time and proof sizes by a factor of $O(L)$, where L represents the depth of the neural network.

TABLE 1: Comparison of proof time and size: zkDL vs. Sum-Check Bit-Decomposition (SC-BD). L: number of network layers; D: size of input tensor in each layer; Q: bit length of each element.

	zkDL (ours)	SC-BD
Proving time	$O(DQ + \log L)$	$O(D^2QL)$
Proof size	$O(\log(DQL))$	$O(L \log(DQ))$

Table 1 presents the comprehensive advantages of zkDL concerning proving time and proof size, achieved through zkReLU and its compatible design of the arithmetic circuit. Furthermore, the completeness, soundness, and zero-knowledge properties are provable in the execution of the zkDL protocol.

2. Related work

Verifiable machine learning inference. Zero-knowledge proof (ZKP) systems have emerged as important solutions to address security and privacy concerns in machine learning. These systems enable the verification of machine learning inference correctness without disclosing the underlying data or model. Notably, zkCNN [1] introduced an interactive proof protocol for convolutional layers, based on the GKR protocol [9] and its refinements [10], [11], [12]. This solution provides zero-knowledge verifiable inference for VGG-scale convolutional neural networks, expanding verifiable computations to modern deep learning. Meanwhile, zk-SNARK-based inference, represented by ZEN [2], vCNN [3], pvCNN [4], and ZKML [5], focuses on enhancing the compatibility of neural networks with the zk-SNARK backend [13], [14], [15], [16], [17], [18], scaling up non-interactive zero-knowledge inference. Once the committed model is verified to be correctly trained using this work, the verifiable inference can serve as a downstream application. On the other hand, there has been a lack of attention on the efficient proof of non-arithmetic operations (e.g., ReLU) and the integration of these proofs into the proof system for the entire neural network.

Verifiable machine learning training. VeriML [6] represents a pioneering effort in zero-knowledge verifiable training for fundamental machine learning algorithms. Building on this foundation, verifiable unlearning [7] takes a significant step forward by providing proof of the correct execution of training logic and updates to the training set in machine unlearning, which supersedes the probabilistic verifications of unlearning [19], [20], [21], [22]. However, their work is confined to scenarios where users contribute data to the training set and maintain full control over changes (additions and deletions of data points) through explicit requests. Consequently, their approach only permits proofs for the data points involved in these requested updates, making it unsuitable for addressing legitimate copyright issues concerning datasets controlled by the model trainer. Furthermore, a notable limitation lies in the scope of supported models, encompassing limited-scale neural networks and compromising on unsupported non-arithmetic operations (e.g., replacing ReLU with square activation).

Proof of learning (PoL) [23]. PoL serves as a non-cryptographic-based alternative to verifiable training. However, its probabilistic guarantees render it unsuitable for legitimacy-related settings like zkDL [24], [25]. Additionally, its threat model assumes adversaries to forge proofs by expending less computation resources than training, which does not deter dedicated malicious trainers capable of deviating from the prescribed training logic (e.g., planting backdoors) at the cost of equivalent or additional computational power.

Membership inference attacks (MIA). MIAs can be used to infer if data points are in the training set [26], [27], [28], [29], [30], [31], [32], [33], [34], but they did not provide guarantees of success and may be vulnerable to defences [35], [36], [37], [38], [39]. Thus, they are not directly applicable to our setting.

3. Preliminaries

3.1. Pedersen commitments

The Pedersen commitment is a zero-knowledge commitment scheme that relies on the hardness of the discrete log problem (DLP). Specifically, in a finite field $\mathbb{F}$ with prime order p, committing to d-dimensional vectors requires an order-p cyclic group $\mathbb{G}$ (e.g., an elliptic curve) and uniformly independently sampled values $\mathbf{g} = (g_0, g_1, \ldots, g_{d-1})^\top \sim \mathbb{G}^d$, and $h \sim \mathbb{G}$. This scheme allows any d-dimensional vector $\mathbf{v} = (v_0, v_1, \ldots, v_{d-1})^\top \in \mathbb{F}^d$ to be committed as:

$$\texttt{Commit}(\mathbf{v}; r) = h^r \mathbf{g}^{\mathbf{v}} = h^r \prod_{i=0}^{d-1} g_i^{v_i},$$

where $r \sim \mathbb{F}$ is uniformly sampled, ensuring zero-knowledgeness of the committed value $\mathbf{v}$. Alternatively, the randomness r can be consistently set to 0, creating a deterministic commitment scheme that remains binding and privacy-preserving due to the hardness of the DLP. Moreover, the Pedersen commitment scheme is homomorphic. That is, for two commitments $\texttt{com}_{\mathbf{v}_1} = h^{r_1}\mathbf{g}^{\mathbf{v}_1}$ and $\texttt{com}_{\mathbf{v}_2} = h^{r_2}\mathbf{g}^{\mathbf{v}_2}$ of vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively, their product $\texttt{com}_{\mathbf{v}_1} \cdot \texttt{com}_{\mathbf{v}_2} = h^{r_1+r_2}\mathbf{g}^{\mathbf{v}_1+\mathbf{v}_2}$ forms a valid commitment of the sum $\mathbf{v}_1 + \mathbf{v}_2$.

3.2. Sumcheck and GKR protocols

The sumcheck protocol [40], [41] serves as a fundamental component in modern proof systems, allowing for the verification of the correctness of the summation $\sum_{\mathbf{b}\in\{0,1\}^d} f(\mathbf{b})$ for a d-variate polynomial f. This protocol offers an efficient proving time of $O(2^d)$ and a compact proof size of $O(d)$.

Building upon the sumcheck protocol, the GKR protocol [9] provides an interactive proof for the accurate computation of arithmetic circuits. It leverages the sumcheck protocol between the layers of the arithmetic circuit, as well as the Pedersen commitments to the private inputs. Additionally, zero-knowledge variants of the GKR protocol [10], [12], [42] have been developed with asymptotically negligible overhead.

The sumcheck and GKR protocols have found wide applications in verifying the proper execution of deep learning models, thanks to their compatibility with tensor structures. For each tensor $\mathbf{S} \in \mathbb{F}^D$ that is discretized from real numbers (without loss of generality, assume D is a power of 2, or zero-padding may be applied), its *multilinear extension* $\widetilde{\mathbf{S}}(\cdot) : \mathbb{F}^{\log_2 D} \to \mathbb{F}$ is a multivariate polynomial defined as

$$\widetilde{\mathbf{S}}(\mathbf{u}) = \sum_{\mathbf{b}\in\{0,1\}^{\log_2 D}} \mathbf{S}(\mathbf{b})\widetilde{\beta}(\mathbf{u}, \mathbf{b}), \tag{1}$$

where $\mathbf{b}$ represents the $\mathbf{b}$-th element of $\mathbf{S}$ (identifying the index by the binary string), and $\widetilde{\beta}(\cdot,\cdot) : \mathbb{F}^{\log_2 D} \times \mathbb{F}^{\log_2 D} \to \mathbb{F}$ is a polynomial. When restricted to $\{0,1\}^{\log_2 D} \times \{0,1\}^{\log_2 D}$, $\widetilde{\beta}(\mathbf{b}_1, \mathbf{b}_2) = \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{b}_1 = \mathbf{b}_2; \\ 0, & \text{if } \mathbf{b}_1 \neq \mathbf{b}_2, \end{array} \right.$ for $\mathbf{b}_1, \mathbf{b}_2 \in \{0,1\}^{\log_2 D}$. In the context of multilinear extensions, we use the notation of indices and their binary representations interchangeably.

Specialized variants of the GKR protocols have been developed for common deep learning operations, such as matrix multiplication [43], [44] and convolution [1]. However, representing neural networks as arithmetic circuits has historically been unclear and inefficient due to non-arithmetic operations. In this work, we address this issue by formalizing and optimizing the modelling scheme. Our proposed scheme maintains full compatibility with the optimized GKR protocol for deep learning operations and facilitates a high degree of parallelization during the proof generation process.

3.3. Inner-product proofs

Inner-product proofs are essential for zero-knowledge proofs involving arithmetic relations over $\mathbb{F}$. They focus on verifying the correctness of the inner-product value $\mathbf{v}_1^\top \mathbf{v}_2$ between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in $\mathbb{F}^d$. Specifically, when the inputs $\mathbf{v}_1$ and $\mathbf{v}_2$ are committed as $h^r \mathbf{g}^{\mathbf{v}_1}\mathbf{h}^{\mathbf{v}_2}$ (where h, $\mathbf{g}$, and $\mathbf{h}$ are randomly sampled generators as described in Section 3.1), Bulletproof [45] can be utilized as a zero-knowledge inner-product proof. Bulletproof offers an efficient proving time of $O(d)$ and a concise proof size of $O(\log d)$.

In this study, our approach involves combining the zero-knowledge inner-product proof with a specialized GKR protocol that is tailored for optimized arithmetic circuits. This combined approach specifically targets the ReLU non-linearity, resulting in significant improvements such as accelerated proof generation and reduced proof size.

4. Efficient Proofs of Deep Learning Training with zkReLU

The proper and tailored handling of non-linearities, especially ReLU, is essential to achieve efficient zero-knowledge verifiable training on deep neural networks. Therefore, in this section, we propose zkReLU, a zero-knowledge protocol designed to verify the training of deep neural networks incorporating ReLU non-linearity, which forms the core of the zkDL protocol. Our scheme leverages the use of auxiliary inputs to enable the verification of both the forward and backward propagations involving ReLU. In Section 4.1, we outline the scheme for validating the auxiliary inputs by ensuring their values fall within the appropriate ranges. This step is crucial for establishing the integrity of the auxiliary inputs. Subsequently, in Section 4.2, we introduce a modelling approach that transforms the deep neural networks and their corresponding backpropagations into arithmetic circuits. This modelling scheme addresses the long-standing ambiguity surrounding zero-knowledge proofs of deep learning, while maintaining a seamless connection with the validity checks described in Section 4.1. One of the advantages of zkReLU is that it achieves a prover time that scales linearly with the size of the required computations, thus improving efficiency. Additionally, it maintains a logarithmic proof size, preventing the linear growth of proof size with the depth of the neural network.

When the ReLU activation is applied to the output of layer ℓ ($1 \leq \ell \leq L-1$, where L is the total number of layers), denoted as $\mathbf{Z}^{(\ell)}$, which corresponds to a linear (fully connected or convolutional) layer, multiplication operations are involved in computing $\mathbf{Z}^{(\ell)}$. Consequently, $\mathbf{Z}^{(\ell)}$ is scaled *twice* by the scaling factor, which we assume is a power of 2, i.e., 2^R. Consequently, when considering discretized values, the ReLU operation should also downscale the input by a factor of 2^R. This can be expressed as the activation function $\mathbf{A}^{(\ell)} = \mathrm{ReLU}\left(\left\lfloor \frac{\mathbf{Z}^{(\ell)}}{2^R} \right\rceil^{+}\right) = \mathbb{I}\left\{\left\lfloor \frac{\mathbf{Z}^{(\ell)}}{2^R} \right\rceil \geq 0\right\} \odot \left\lfloor \frac{\mathbf{Z}^{(\ell)}}{2^R} \right\rceil$.

To simplify the notation, we introduce the rescaled $\mathbf{Z}^{(\ell)\prime} := \left\lfloor \frac{\mathbf{Z}^{(\ell)}}{2^R} \right\rceil$. This allows us to express $\mathbf{Z}^{(\ell)}$ as $\mathbf{Z}^{(\ell)} = 2^R \mathbf{Z}^{(\ell)\prime} + \mathbf{R}_{\mathbf{Z}}^{(\ell)}$, where $\mathbf{R}_{\mathbf{Z}}^{(\ell)}$ represents the remainder caused by rounding. However, to establish the notion of "non-negative" within the finite field, it is necessary to constrain the scale of $\mathbf{Z}^{(\ell)\prime}$. We assume that each element of $\mathbf{Z}^{(\ell)\prime}$ is an Q-bit signed integer, where $2^Q \ll |\mathbb{F}|$. For analysis purposes only, we decompose $\mathbf{Z}^{(\ell)\prime}$ as the magnitude bits and the sign bits, i.e., $\mathbf{Z}^{(\ell)\prime} = \sum_{j=0}^{Q-2} 2^j \mathbf{B}_j^{(\ell)} - 2^{Q-1}\mathbf{B}_{Q-1}^{(\ell)}$, where each $\mathbf{B}_j^{(\ell)}$ for $0 \leq j \leq Q-1$ is binary and $\mathbf{B}_{Q-1}^{(\ell)}$ is the negativity of each dimension in $\mathbf{Z}^{(\ell)}$ (1 for negative, 0 otherwise). By introducing the tensor of re-compressed magnitude bits $\mathbf{Z}^{(\ell)\prime\prime} = \sum_{j=0}^{Q-2} 2^j \mathbf{B}_j^{(\ell)}$ and including $\mathbf{B}_{Q-1}^{(\ell)}$,

$\mathbf{Z}^{(\ell)\prime\prime}$, and $\mathbf{R}_{\mathbf{Z}}^{(\ell)}$ as auxiliary inputs, the trainer needs to prove the following:

$$\mathbf{A}^{(\ell)} = (\mathbf{1} - \mathbf{B}_{Q-1}^{(\ell)}) \odot \mathbf{Z}^{(\ell)\prime\prime}, \tag{2}$$

$$\mathbf{Z}^{(\ell)} = 2^R \mathbf{Z}^{(\ell)\prime\prime} - 2^{Q+R-1}\mathbf{B}_{Q-1}^{(\ell)} + \mathbf{R}_{\mathbf{Z}}^{(\ell)}. \tag{3}$$

During backpropagation, the gradient of $\mathbf{A}^{(\ell)}$, denoted as $\mathbf{G}_{\mathbf{A}}^{(\ell)}$, is also typically scaled twice by 2^R due to the involved multiplication operations. Hence, the trainer needs to rescale it to $\mathbf{G}_{\mathbf{A}}^{(\ell)\prime} := \left\lfloor \frac{\mathbf{G}_{\mathbf{A}}^{(\ell)}}{2^R} \right\rceil$ with the remainder $\mathbf{R}_{\mathbf{G}_{\mathbf{A}}}^{(\ell)}$. Subsequently, $\mathbf{G}_{\mathbf{A}}^{(\ell)\prime}$ is used to compute the gradient of $\mathbf{Z}^{(\ell)}$, denoted as $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$, through Hadamard product $\odot$ with $\mathbf{1} - \mathbf{B}_{Q-1}^{(\ell)}$. With the additional auxiliary inputs $\mathbf{G}_{\mathbf{A}}^{(\ell)\prime}$ and $\mathbf{R}_{\mathbf{G}_{\mathbf{A}}}^{(\ell)}$, the trainer needs to prove the following:

$$\mathbf{G}_{\mathbf{Z}}^{(\ell)} = (\mathbf{1} - \mathbf{B}_{Q-1}^{(\ell)}) \odot \mathbf{G}_{\mathbf{A}}^{(\ell)\prime}, \tag{4}$$

$$\mathbf{G}_{\mathbf{A}}^{(\ell)} = 2^R \mathbf{G}_{\mathbf{A}}^{(\ell)\prime} + \mathbf{R}_{\mathbf{G}_{\mathbf{A}}}^{(\ell)}. \tag{5}$$

It is important to emphasize that all auxiliary inputs, namely $\mathbf{Z}^{(\ell)\prime\prime} \in \left[0, 2^{Q-1}\right)^D$, $\mathbf{B}_{Q-1}^{(\ell)} \in \{0,1\}^D$, $\mathbf{R}_{\mathbf{Z}}^{(\ell)} \in \left[-2^{R-1}, 2^{R-1}\right)^D$, $\mathbf{G}_{\mathbf{A}}^{(\ell)\prime} \in \left[-2^{Q-1}, 2^{Q-1}\right)^D$, and $\mathbf{R}_{\mathbf{G}_{\mathbf{A}}}^{(\ell)} \in \left[-2^{R-1}, 2^{R-1}\right)^D$, are bounded and share the same dimension D. In this analysis, we consider D to represent the size of the one-dimensional input tensor $\mathbf{Z}^{(\ell)}$. Once the backpropagation operation concludes, it becomes the trainer's responsibility to compute the commitments for these five auxiliary inputs and transmit them to the trusted verifier. For the sake of brevity, we use $\mathbf{aux}^{(\ell)}$ to collectively refer to the auxiliary inputs of layer ℓ. Furthermore, we use the same notations without the superscript of the layer number, namely $\mathbf{Z}'', \mathbf{B}_{Q-1}, \mathbf{R}_{\mathbf{Z}}, \mathbf{G}'_{\mathbf{A}}, \mathbf{R}_{\mathbf{G}_{\mathbf{A}}}$, and collectively as $\mathbf{aux}$, to represent the stacked tensors with consistent notation across all layers. In the absence of ambiguity, we also employ D to denote the dimension of these tensors.

4.1. zkReLU: Validity of the auxiliary inputs

A malicious trainer may attempt to train low-quality and illegitimate models by manipulating the values of auxiliary inputs, while maintaining the correct arithmetic relations among them and the other components of the neural network. In particular, they may commit to auxiliary inputs that are outside their respective ranges, and forge proofs using these invalid values. However, conducting a straightforward bit-decomposition-based range proof for each auxiliary input would result in a linear increase in proof size as the total number of auxiliary inputs summed over all layers grows. Therefore, in this section, we present the core of the zkReLU protocol, which involves batching the proofs for the validity of the auxiliary inputs. This approach optimizes the running times for both the trainer and trusted verifier, reduces proof sizes, and maintains the requirements of correctness, soundness and zero-knowledge.

We recall that for any $(K-1)$-bit unsigned integer v, there is $\mathbf{b} \in \{0,1\}^{K-1}$ such that $\left\langle \mathbf{b}, \mathbf{2}^{K-1} \right\rangle = v$, where $\mathbf{2}^{K-1} = \left(1, 2, \ldots, 2^{K-2}\right)^\top$. Similarly, for any K-bit signed integer v, there exists $\mathbf{a} \in \{0,1\}^K$ such that $\langle \mathbf{a}, \mathbf{s}_K \rangle = v$, where $\mathbf{s}_K = \left(1, 2, \ldots, 2^{K-2}, -2^{K-1}\right)^\top$.

However, the range requirements for different auxiliary inputs are non-uniform. To address this, we first focus on the decomposition of $\mathbf{Z}''$ (unsigned $(Q-1)$-bit integers) and $\mathbf{G}_{\mathbf{A}}{}'$ (signed Q-bit integers).

We can handle the range requirement using the binary matrix $\mathbf{B} = \begin{pmatrix} \mathbf{B}_{\mathbf{Z}''} \mid \mathbf{0} \\ \mathbf{B}_{\mathbf{G}_{\mathbf{A}}{}'} \end{pmatrix} \in \{0,1\}^{2D \times Q}$, where $\mathbf{B}_{\mathbf{Z}''}$ and $\mathbf{B}_{\mathbf{G}_{\mathbf{A}}{}'}$ are the element-wise unsigned $(Q-1)$-bit and signed Q-bit representations of $\mathbf{Z}''$ and $\mathbf{G}_{\mathbf{A}}{}'$, respectively. Additionally, we define $\mathbf{B}' = \begin{pmatrix} \mathbf{B}_{\mathbf{Z}''} - \mathbf{1} \mid \mathbf{0} \\ \mathbf{B}_{\mathbf{G}_{\mathbf{A}}{}'} - \mathbf{1} \end{pmatrix} \in \{0,1\}^{2D \times Q}$, such that the element-wise product $\mathbf{B} \odot \mathbf{B}'$ is equal to $\mathbf{0}$. Moreover, by padding $\overline{\mathbf{B}_{Q-1}} = \begin{pmatrix} \mathbf{0} \mid \mathbf{B}_{Q-1} \\ \mathbf{0} \end{pmatrix} \in \{0,1\}^{2D \times Q}$ and $\overline{\mathbf{B}'_{Q-1}} = \begin{pmatrix} \mathbf{0} \mid \mathbf{B}_{Q-1} - \mathbf{1} \\ \mathbf{0} \end{pmatrix} \in \{0,1\}^{2D \times Q}$, we also have the additional requirement that the element-wise product $\overline{\mathbf{B}_{Q-1}} \odot \overline{\mathbf{B}'_{Q-1}}$ equals $\mathbf{0}$.

We consider a random linear combination of $\mathbf{B}$ and $\overline{\mathbf{B}_{Q-1}}$, given by $\mathbf{B}_k = \mathbf{B} + k\overline{\mathbf{B}_{Q-1}}$ and $\mathbf{B}'_k = \mathbf{B}' + k\overline{\mathbf{B}'_{Q-1}}$, where $k \sim \mathbb{F}$. We take a uniformly randomly chosen sampling $\mathbf{u}_{\text{relu}} = (u''_{\text{relu}}, \mathbf{u}'_{\text{relu}}) \sim \mathbb{F}^{1+\log_2 D}$ and $\mathbf{u}_{\text{bit}} \sim \mathbb{F}^{\log_2 Q}$, and we define the "expansion" of the random vector $\mathbf{e}(\mathbf{u}) = \left(\widetilde{\beta}(\mathbf{u}, \mathbf{b})\right)_{\mathbf{b} \in \{0,1\}^{\dim(\mathbf{u})}}$ such that $\widetilde{\mathbf{S}}(\mathbf{u}) = \mathbf{S}^\top \mathbf{e}(\mathbf{u})$ for any tensor $\mathbf{S}$ (assuming $\mathbf{S}$ is one-dimensional for clarity). If the trainer and trusted verifier agree on the claimed value of $v = (1 - u''_{\text{relu}})\widetilde{\mathbf{Z}''}(\mathbf{u}'_{\text{relu}}) + u''_{\text{relu}}\widetilde{\mathbf{G}'_{\mathbf{A}}}(\mathbf{u}'_{\text{relu}})$ (which is the multilinear extension evaluation of $\begin{pmatrix} \mathbf{Z}'' \\ \mathbf{G}'_{\mathbf{A}} \end{pmatrix}$ at $\mathbf{u}_{\text{relu}}$) and $v_{Q-1} = \widetilde{\mathbf{B}_{Q-1}}(\mathbf{u}'_{\text{relu}})$ (or the commitment thereof on the trusted verifier's side, which will be realized in Section 4.2), the correctness of the relations described above is equivalent to the following equations with overwhelming probability:

$$\left\langle \mathbf{e}(\mathbf{u}_{\text{relu}})^\top \mathbf{B}_k, \mathbf{s}_Q \right\rangle = v_k, \tag{6}$$

$$\widetilde{\mathbf{B}_k}(\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) - \widetilde{\mathbf{B}'_k}(\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) = v'_k, \tag{7}$$

$$\widetilde{\mathbf{B}_k \odot \mathbf{B}'_k}(\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) = 0. \tag{8}$$

Here, the $\mathbf{B}_k$ is correctly computed only if

$$\begin{aligned} v_k &= \mathbf{e}(\mathbf{u}_{\text{relu}})^\top \mathbf{B}_k \mathbf{s}_Q && (9) \\ &= \mathbf{e}(\mathbf{u}_{\text{relu}})^\top \mathbf{B}\mathbf{s}_Q + k\mathbf{e}(\mathbf{u}_{\text{relu}})^\top \overline{\mathbf{B}_{Q-1}}\mathbf{s}_Q && (10) \\ &= \mathbf{e}(\mathbf{u}_{\text{relu}})^\top \begin{pmatrix} \mathbf{Z}'' \\ \mathbf{G}'_{\mathbf{A}} \end{pmatrix} - k2^{Q-1}\mathbf{e}(\mathbf{u}_{\text{relu}})^\top \begin{pmatrix} \mathbf{B}_{Q-1} \\ \mathbf{0} \end{pmatrix} && (11) \\ &= v - k2^{Q-1}(1 - u''_{\text{relu}})v_{Q-1}, && (12) \end{aligned}$$

and that its components $\mathbf{B}$ and $\mathbf{B}_{Q-1}$ are binary with specific blocks being zero, such that

$$\begin{aligned} v'_k &= \widetilde{\mathbf{B}_k - \mathbf{B}'_k}(\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) && (13) \\ &= \widetilde{\begin{pmatrix} \mathbf{1} \mid k \cdot \mathbf{1} \\ \mathbf{1} \end{pmatrix}}(\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) && (14) \\ &= 1 + (k-1)\widetilde{\beta}(\mathbf{b}(Q-1), \mathbf{u}_{\text{bit}})(1 - u''_{\text{relu}}), && (15) \end{aligned}$$

and $\mathbf{B}_k \odot \mathbf{B}'_k = \mathbf{0}$ (which is equilvalent to $\mathbf{B}_k \odot \mathbf{B}'_k (\mathbf{u}_{\text{relu}}, \mathbf{u}_{\text{bit}}) = 0$ with overwhelming probability).

Therefore, we can write all of these equations in the form of inner products between vectors of length $2BQ$, by flattening the tensors:

$$\langle \mathbf{B}_k, \mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{s}_Q \rangle = v_k, \tag{16}$$

$$\langle \mathbf{B}_k - \mathbf{B}'_k, \mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{e}(\mathbf{u}_{\text{bit}}) \rangle = v'_k, \tag{17}$$

$$\langle \mathbf{B}_k, \mathbf{B}'_k \odot (\mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{e}(\mathbf{u}_{\text{bit}})) \rangle = 0, \tag{18}$$

which is equivalent to

$$\begin{aligned}\langle \mathbf{B}_k - z \cdot \mathbf{1}, z^2 \cdot \mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{s}_Q + (z \cdot \mathbf{1} + \mathbf{B}'_k) \odot (\mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{e}(\mathbf{u}_{\text{bit}})) \rangle \\ = z^3 - (1 - v_k) z^2 + z v'_k,\end{aligned} \tag{19}$$

for uniformly randomly chosen $z \sim \mathbb{F}$. Theorem 4.1 guarantees that these equalities are also sufficient to guarantee the validity of the auxiliary inputs with overwhelming probability:

Theorem 4.1. *With probability* $p = 1 - O\left(\frac{DQ}{|\mathbb{F}|}\right)$, *if* (19) *holds, then* $\mathbf{Z}'' \in [0, 2^{Q-1})^D, \mathbf{B}_{Q-1} \in \{0,1\}^D$ *and* $\mathbf{G}_{\mathbf{A}} \in [-2^{Q-1}, 2^{Q-1})^D$.

Proof of Theorem 4.1. We first note that (19) is a random linear combination of (16), (17) and (18), which is multiplied by z^2, z and 1, respectively. Therefore, by Schwartz-Zipple (SZ) Lemma, with probability $1 - O\left(\frac{1}{|\mathbb{F}|}\right)$, if (19) holds, then these three equalities hold. Equivalently, (6), (7) and (8)) also hold.

On the other hand, applying SZ lemma for multi-linear extension, if (6), (7) and (8) holds, then with probability $1 - O\left(\frac{\log D + \log Q}{|\mathbb{F}|}\right)$:

$$\mathbf{B}_k \mathbf{s}_Q = \begin{pmatrix} \mathbf{Z}'' \\ \mathbf{G}'_{\mathbf{A}} \end{pmatrix} - k 2^{Q-1} \begin{pmatrix} \mathbf{B}_{Q-1} \\ \mathbf{0} \end{pmatrix}, \tag{20}$$

$$\mathbf{B}_k - \mathbf{B}'_k = \begin{pmatrix} \mathbf{1} \mid k \cdot \mathbf{1} \\ \mathbf{1} \end{pmatrix} = \begin{pmatrix} \mathbf{1} \mid \mathbf{0} \\ \mathbf{1} \end{pmatrix} + k \begin{pmatrix} \mathbf{0} \mid \mathbf{1} \\ \mathbf{0} \end{pmatrix}, \tag{21}$$

$$\mathbf{B}_k \odot \mathbf{B}'_k = \mathbf{0}. \tag{22}$$

By the homomorphism of the commitments, $\mathbf{B}_k = \mathbf{B} + k\overline{\mathbf{B}_{Q-1}}$ and $\mathbf{B}'_k = \mathbf{B}' + k\overline{\mathbf{B}'_{Q-1}}$. Therefore, if (20), (21) and (22) hold, then

$$\left(\mathbf{B}\mathbf{s}_Q - \begin{pmatrix} \mathbf{Z}'' \\ \mathbf{G}'_{\mathbf{A}} \end{pmatrix}\right) + k\left(\overline{\mathbf{B}_{Q-1}}\mathbf{s}_Q + 2^{Q-1}\begin{pmatrix} \mathbf{B}_{Q-1} \\ \mathbf{0} \end{pmatrix}\right) = \mathbf{0}, \tag{23}$$

$$\left(\mathbf{B} - \mathbf{B}' - \begin{pmatrix} \mathbf{1} \mid \mathbf{0} \\ \mathbf{1} \end{pmatrix}\right) + k\left(\overline{\mathbf{B}_{Q-1}} - \overline{\mathbf{B}'_{Q-1}} - \begin{pmatrix} \mathbf{0} \mid \mathbf{1} \\ \mathbf{0} \end{pmatrix}\right) = \mathbf{0}, \tag{24}$$

$$\mathbf{B} \odot \mathbf{B}' + k\left(\mathbf{B} \odot \overline{\mathbf{B}'_{Q-1}} + \mathbf{B}' \odot \overline{\mathbf{B}_{Q-1}}\right) + k^2 \overline{\mathbf{B}_{Q-1}} \odot \overline{\mathbf{B}'_{Q-1}} = \mathbf{0}. \tag{25}$$

Here, (23), (24) are polynomials of order 1, while (25) is a polynomial of order 2. Therefore, with probability $1 - O\left(\frac{DQ}{|\mathbb{F}|}\right)$ over the choice of k, if these three polynomials all evaluate to $\mathbf{0}$, then each of the coefficient is $\mathbf{0}$.

By $\mathbf{B} \odot \mathbf{B}' = \mathbf{0}$ and $\mathbf{B} - \mathbf{B}' = \begin{pmatrix} \mathbf{1} \mid \mathbf{0} \\ \mathbf{1} \end{pmatrix}$, we can deduce that $\mathbf{B} \in \begin{pmatrix} \{0,1\}^{D \times (Q-1)} \mid \mathbf{0} \\ \{0,1\}^{D \times Q} \end{pmatrix}$. Therefore $\begin{pmatrix} \mathbf{Z}'' \\ \mathbf{G}'_{\mathbf{A}} \end{pmatrix} = \mathbf{B}\mathbf{s}_Q \in \begin{pmatrix} [0, 2^{Q-1})^D \\ [-2^{Q-1}, 2^{Q-1})^D \end{pmatrix}$. Similarly, it is also guaranteed that $\overline{\mathbf{B}_{Q-1}} \in \begin{pmatrix} \mathbf{0} \mid \{0,1\}^D \\ \mathbf{0} \end{pmatrix}$. Therefore, $-2^{Q-1}\mathbf{B}_{Q-1} \in \{0, -2^{Q-1}\}^D$, hence $\mathbf{B}_{Q-1} \in \{0,1\}^D$. □

To instantiate the commitment scheme for the tensors, we assume the existence of $\mathbf{g} = (g_0, g_1, \ldots, g_{D-1})^\top \sim \mathbb{G}^D$ and $h \sim \mathbb{G}$, where the $D+1$ group elements are sampled uniformly and independently. Also, to facilitate the commitment of $\mathbf{B}_{Q-1}$ and its extension $\overline{\mathbf{B}_{Q-1}}$, we introduce $\mathbf{G} \in \mathbb{G}^{2D \times Q}$, where $\mathbf{G}_{[0:D,Q-1]} = \mathbf{g}$, and the remaining elements of $\mathbf{G}$ are uniformly and randomly sampled, which are independent of $\mathbf{g}$ and h. For any tensor $\mathbf{V} \in \mathbb{F}^{2D \times Q}$, its commitment is given by $\texttt{Commit}(\mathbf{V}; r) = h^r \mathbf{G}^{\mathbf{V}}$. Therefore, a valid commitment of $\mathbf{B}_{Q-1}$ is also a valid commitment of $\overline{\mathbf{B}_{Q-1}}$.

In addition, to enable the commitment of the second vector in the inner product, we assume the existence of $\mathbf{H} \sim \mathbb{G}^{2D \times Q}$, which is sampled independently from $\mathbf{G}$ and h. We denote $\mathbf{h} = \mathbf{H}_{[0:D,Q-1]}$. After this, the prover and verifier run Protocol 1 to complete the setup of the validity checks.

Protocol 1 Setup for the validity checks of the auxiliary inputs

```
Require: Both parties knows the commitments $\texttt{com}_{\mathbf{Z}''}$ and
    $\texttt{com}_{\mathbf{G}'_{\mathbf{A}}}$, and $\texttt{com}_{\mathbf{B}_{Q-1}} (= h^{r_{Q-1}} \mathbf{g}^{\mathbf{B}_{Q-1}})$; the trainer
    knows the committed values $\mathbf{Z}''$, $\mathbf{G}'_{\mathbf{A}}$, and $\mathbf{B}_{Q-1}$
1: Trainer computes bit tensors $\mathbf{B}, \mathbf{B}', \mathbf{B}'_{Q-1}$
2: Trainer sends the commitments
       $\texttt{com}^{\text{ip}}_{\mathbf{B}} \leftarrow h^r \mathbf{G}^{\mathbf{B}} \mathbf{H}^{\mathbf{B}'}$   and   $\texttt{com}_{\mathbf{B}'_{Q-1}} \leftarrow h^{r'_{Q-1}} \mathbf{h}^{\mathbf{B}'_{Q-1}}$
   to the trusted verifier, where $r, r'_{Q-1} \sim \mathbb{F}$.
3: Both trainer and trusted verifier compute
       $\texttt{com}^{\text{ip}}_{\mathbf{B}_{Q-1}} \leftarrow \texttt{com}_{\mathbf{B}_{Q-1}} \cdot \texttt{com}_{\mathbf{B}'_{Q-1}}$
```

In Line 3, we have

$$\texttt{com}^{\text{ip}}_{\mathbf{B}_{Q-1}} = h^{r_{Q-1} + r'_{Q-1}} \mathbf{g}^{\mathbf{B}_{Q-1}} \mathbf{h}^{\mathbf{B}'_{Q-1}} = h^{r_{Q-1} + r'_{Q-1}} \mathbf{G}^{\overline{\mathbf{B}_{Q-1}}} \mathbf{H}^{\overline{\mathbf{B}'_{Q-1}}},$$

which has the same form as $\texttt{com}^{\text{ip}}_{\mathbf{B}}$ and can be opened by the trainer.

After the setup phase, both parties engage in the sumcheck protocol for the arithmetic components of the forward and backward propagations (which is detailed in Section 4.2), during which the randomness $\mathbf{u}_{\text{relu}}$ is generated. Additionally, the prover possesses the values of v and v_{Q-1}, while the trusted verifier holds the commitments to these values. Next, the trusted verifier shares the randomness values k and $\mathbf{u}_{\text{bit}}$ with the prover. These values enable the two parties to proceed with the validity check of (19).

Specifically, the commitments of the two vectors involved in (19) can be efficiently computed from the existing commitments that are known to both parties. In particular, we consider $\mathbf{e} := \mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{e}(\mathbf{u}_{\text{bit}})$ and its elementwise inversion $\mathbf{e}^{\circ -1}$, such that $\mathbf{H}^{\mathbf{e}^{\circ -1}}$ can also be directly computed by both the trainer and trusted verifier (since both $\mathbf{e}$ and $\mathbf{H}$ are known to both parties). Therefore, for the second vector in the inner product,

$$\begin{aligned} & z^2 \cdot \mathbf{e}(\mathbf{u}_{\text{relu}}) \otimes \mathbf{s}_Q + (z \cdot \mathbf{1} + \mathbf{B}'_k) \odot \mathbf{e} \\ = & \left(z^2 \cdot \mathbf{1} \otimes (\mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}})) + z \cdot \mathbf{1} + \mathbf{B_k}'\right) \odot \mathbf{e}, \end{aligned}$$

where $\oslash$ denotes the elementwise division, such that the discrete-log-based commitments of the two inputs of (19) can be computed using Algorithm 1.

Algorithm 1 Transformation of the commitments

```
Require: Commitments $\mathsf{com}^{\text{ip}}_{\mathbf{B}} \in \mathbb{G}, \mathsf{com}^{\text{ip}}_{\mathbf{B}_{Q-1}} \in \mathbb{G}$, randomness $z \in \mathbb{F}, k \in \mathbb{F}, \mathbf{u}_{\text{relu}} \in \mathbb{F}^{\log_2 D+1}, \mathbf{u}_{\text{bit}} \in \mathbb{F}^{\log_2 Q}$.
    Precomputed $\mathbf{G}^{\text{prod}} = \prod_{i=0}^{2D-1} \mathbf{G}_i^{\text{prod}}$.
1: for $i \leftarrow 0, 1, \ldots, 2D-1$ do                                ▷ Pre-computed
2:     $g_i^{\text{prod}} \leftarrow \prod_{j=0}^{Q-1} \mathbf{G}_{[i,j]}$      ▷ Vectorized as $\mathbf{g}^{\text{prod}} \in \mathbb{G}^{2D}$
3:     $h_i^{\text{prod}} \leftarrow \prod_{j=0}^{Q-1} \mathbf{G}_{[i,j]}$      ▷ Vectorized as $\mathbf{h}^{\text{prod}} \in \mathbb{G}^{2D}$
4: end for
5: $g^{\text{prod}} \leftarrow \prod_{i=0}^{2D-1} g_i^{\text{prod}}$                    ▷ Pre-computed
6: $h^{\text{prod}} \leftarrow \prod_{i=0}^{2D-1} h_i^{\text{prod}}$                    ▷ Pre-computed
7: $\mathsf{com}^{\text{ip}}_{\mathbf{B}_k} \leftarrow \mathsf{com}^{\text{ip}}_{\mathbf{B}} \cdot \left(\mathsf{com}^{\text{ip}}_{\mathbf{B}_{Q-1}}\right)^k$   ▷ Commitment of $\mathbf{B}_k$
    and $\mathbf{B}'_k$
8: return $\mathsf{com}^{\text{ip}}_{\mathbf{B}_k} \cdot \left(g^{\text{prod}}\right)^{-z} \cdot \left(\mathbf{h}^{\text{prod}}\right)^{z^2 \cdot \mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}})} \cdot \left(h^{\text{prod}}\right)^z$
```

In Line 7, $\mathsf{com}^{\text{ip}}_{\mathbf{B}_k}$ is a valid commitment of $\mathbf{B}_k$ and $\mathbf{B}'_k$:

$$\begin{aligned} \mathsf{com}^{\text{ip}}_{\mathbf{B}_k} &= \mathsf{com}^{\text{ip}}_{\mathbf{B}} \cdot \left(\mathsf{com}^{\text{ip}}_{\mathbf{B}_{Q-1}}\right)^k \\ &= h^r \mathbf{G}^{\mathbf{B}} \mathbf{H}^{\mathbf{B}'} \left(h^{r_{Q-1}+r'_{Q-1}} \mathbf{G}^{\overline{\mathbf{B}_{Q-1}}} \mathbf{H}^{\overline{\mathbf{B}'_{Q-1}}}\right)^k \\ &= h^{r+kr_{Q-1}+kr'_{Q-1}} \mathbf{G}^{\mathbf{B}+k\overline{\mathbf{B}_{Q-1}}} \mathbf{H}^{\mathbf{B}'+k\overline{\mathbf{B}'_{Q-1}}} \\ &= h^{r+kr_{Q-1}+kr'_{Q-1}} \mathbf{G}^{\mathbf{B}_k} \mathbf{H}^{\mathbf{B}'_k}, \end{aligned} \tag{26}$$

which we denote as $h^{r_k} \mathbf{G}^{\mathbf{B}_k} \mathbf{H}^{\mathbf{B}'_k}$ for simplicity. Therefore, the returned value equals

$$\begin{aligned} & \mathsf{com}^{\text{ip}}_{\mathbf{B}_k} \cdot \left(g^{\text{prod}}\right)^{-z} \cdot \left(\mathbf{h}^{\text{prod}}\right)^{z^2 \cdot \mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}})} \cdot \left(h^{\text{prod}}\right)^z \\ =& h^{r_k} \mathbf{G}^{\mathbf{B}_k} \mathbf{H}^{\mathbf{B}'_k} \mathbf{G}^{-z \cdot \mathbf{1}} \mathbf{H}^{z^2 \cdot \mathbf{1} \otimes (\mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}}))} \mathbf{H}^{z \cdot \mathbf{1}} \\ =& h^{r_k} \mathbf{G}^{\mathbf{B}_k - z\cdot \mathbf{1}} \mathbf{H}^{z^2 \cdot \mathbf{1} \otimes (\mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}})) + z \cdot \mathbf{1} + \mathbf{B_k}'} \\ =& h^{r_k} \mathbf{G}^{\mathbf{B}_k - z\cdot \mathbf{1}} \left(\mathbf{H}^{\mathbf{e}^{\circ -1}}\right)^{\left(z^2 \cdot \mathbf{1} \otimes (\mathbf{s}_Q \oslash \mathbf{e}(\mathbf{u}_{\text{bit}})) + z \cdot \mathbf{1} + \mathbf{B_k}'\right) \odot \mathbf{e}}, \end{aligned}$$

which is a valid commitment with the alternative basis $\mathbf{G}$ and $\mathbf{H}^{\mathbf{e}^{\circ -1}}$. Therefore, after the transformation of the commitments using Algorithm 1, the equality (19) can be proven using the zero-knowledge inner-product argument with a logarithmic proof size [45]. Similarly, the validity of $\mathbf{R}_{\mathbf{Z}}$ and $\mathbf{R}_{\mathbf{G_A}}$ as R-bit integers can also be proved using an inner product argument, and combined with the argument for (19) using random linear combinations. However, the auxiliary input validity proofs require verified claims on the evaluations of the multilinear extensions of the auxiliary inputs at the random point $\mathbf{u}_{\text{relu}}$. Therefore, achieving this verification necessitates a compatible design of the arithmetic circuit, which is described in Section 4.2.

4.2. zkReLU: Arithmetic circuit design

In Section 4.1, we made the assumption that the GKR protocol is executed on the arithmetic components of the neural network, encompassing both the forward and backward propagations. This execution is assumed to yield the randomness $\mathbf{u}_{\text{relu}}$ and the claims on the evaluation of the multilinear extensions of the auxiliary inputs at this stage. However, a straightforward execution of the GKR protocol on the entire circuit does not produce these results unless additional expensive post-processings are performed. Moreover, given the large number of non-arithmetic operations involved, the formulation of the deep neural network and its backpropagation as an arithmetic circuit still need to be clearly addressed so as to explicitly construct a proof system.

Therefore, in this section, we formulate the modelling of the deep neural networks and their backpropagation as an arithmetic circuit, and provide a description of the GKR protocol execution on it. This ensures a seamless connection with the validity check presented in Section 4.1. It is important to note that the protocol outlined in this section should be carried out immediately after Protocol 1. At this point, the trainer has already committed to the bit decompositions but has not yet received the randomness. Other orders of execution may result in forged proofs.

Upon committing to the auxiliary inputs, the prover proceeds to compute the proofs for the arithmetic components, including matrix multiplications in fully connected layers, convolutions in CNNs, and their corresponding backpropagations, using the GKR protocol. By leveraging the homomorphism of the commitment scheme, both the prover and the trusted verifier can obtain the commitments of $\mathbf{G}_{\mathbf{A}}^{(\ell)}$ and $\mathbf{Z}^{(\ell)}$. These commitments are acquired based on equations (3) and (5).

We emphasize that modelling the forward and backward propagations as a single arithmetic circuit with consistent layer depths and input/output ordering is not compatible with zkReLU. This is because non-arithmetic operations like ReLU and rescaling disrupt the connectivity within the arithmetic circuit. Instead, it is more accurate to consider each arithmetic component ℓ, parameterized by $\mathbf{W}^{(\ell)}$, as a separate arithmetic circuit that are independently connected with the auxiliary inputs $\mathbf{aux}^{(\ell)}$, as shown in Figure 2. Specifically, the circuit for forward propagation takes the previous activation $\mathbf{A}^{(\ell-1)}$ and weight $\mathbf{W}^{(\ell)}$ as inputs and produces $\mathbf{Z}^{(\ell)}$ as output. On the other hand, the circuit for backward propagation takes the output gradient $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$, the previous activation $\mathbf{A}^{(\ell-1)}$, and the weight $\mathbf{W}^{(\ell)}$ as inputs, and generates the weight and activation gradients, $\mathbf{G}_{\mathbf{A}}^{(\ell-1)}$ and $\mathbf{G}_{\mathbf{W}}^{(\ell)}$, respectively.

In the context mentioned, the trusted verifier has access to the commitments of all tensors except $\mathbf{A}^{(\ell-1)}$ and $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$. It is worth noting that the sumcheck protocols for these circuits can be executed concurrently, allowing for the use of the same randomness. Consequently, the claims regarding the multi-linear extensions of $\mathbf{A}^{(\ell)}$ and $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$ can be made at the same point $\widetilde{\mathbf{A}^{(\ell)}}(\mathbf{u}_{\mathbf{A}})$ and $\widetilde{\mathbf{G}_{\mathbf{Z}}^{(\ell)}}(\mathbf{u}_{\mathbf{G_Z}})$ across different layers ℓ. This assumes that all layers have the same dimension, although zero-padding can be employed to achieve this behaviour otherwise.

Therefore, with randomly and independently chosen $u''_{\text{relu}} \sim \mathbb{F}$ and $\mathbf{u}_{\text{stack}} \sim \mathbb{F}^{\lceil \log_2 (L-1) \rceil}$, the trainer can batch up the proofs for $\mathbf{A}^{(\ell)}$ and $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$, which connects the proofs of the arithmetic components and the validity check of auxiliary inputs seamlessly, by running the sumcheck protocol on (27):

$$
\begin{aligned}
&(1-u''_{\text{relu}}) \sum_{\ell=1}^{L-1} \widetilde{\beta}(\mathbf{u}_{\text{stack}}, \ell-1)\, \widetilde{\mathbf{A}^{(\ell)}}(\mathbf{u}_{\mathbf{A}}) + \\
&+ u''_{\text{relu}} \sum_{\ell=1}^{L-1} \widetilde{\beta}(\mathbf{u}_{\text{stack}}, \ell-1)\, \widetilde{\mathbf{G}_{\mathbf{Z}}^{(\ell)}}(\mathbf{u}_{\mathbf{G_Z}}) \\
=&(1-u''_{\text{relu}})\widetilde{\mathbf{A}}(\mathbf{u}_{\text{stack}}, \mathbf{u}_{\mathbf{A}}) + u''_{\text{relu}} \widetilde{\mathbf{G}_{\mathbf{Z}}}(\mathbf{u}_{\text{stack}}, \mathbf{u}_{\mathbf{G_Z}}) \\
=& \left(1 - \sum_{\mathbf{b}\in\{0,1\}^{\log_2 D}} \widetilde{\mathbf{B}_{Q-1}}(\mathbf{u}_{\text{stack}}, \mathbf{b})\right) \\
&\Big((1-u''_{\text{relu}})\widetilde{\mathbf{A}}(\mathbf{u}_{\text{stack}}, \mathbf{b})\, \widetilde{\beta}(\mathbf{u}_{\mathbf{A}}, \mathbf{b}) \\
&+ u''_{\text{relu}} \widetilde{\mathbf{G}_{\mathbf{Z}}}(\mathbf{u}_{\text{stack}}, \mathbf{b})\, \widetilde{\beta}(\mathbf{u}_{\mathbf{G_Z}}, \mathbf{b})\Big), \qquad (27)
\end{aligned}
$$

where the indices $\ell - 1$ are in the form of binary representation.

The sumcheck protocol on (27) outputs the claims on $\widetilde{\mathbf{B}_{Q-1}}(\mathbf{u}_{\text{stack}}, \mathbf{u}_{\text{new}}), \widetilde{\mathbf{A}}(\mathbf{u}_{\text{stack}}, \mathbf{u}_{\text{new}}), \widetilde{\mathbf{G}_{\mathbf{Z}}}(\mathbf{u}_{\text{stack}}, \mathbf{u}_{\text{new}})$, which is the evaluations at the same random point, and can be verified with respect to the Pedersen commitments of these tensors. By concatenating u''_{relu}, $\mathbf{u}_{\text{stack}}$, and $\mathbf{u}_{\text{new}}$, we get the $\mathbf{u}_{\text{relu}}$ that is needed for the validity check of the auxiliary inputs as described in Section 4.1. The full zkDL protocol between the trainer $\mathcal{T}$ and trusted verifier $\mathcal{V}$, on the committed data $\mathbf{X}, \mathbf{y}$, model parameters $\mathbf{W}$, gradients $\mathbf{G}_{\mathbf{W}}$ and auxiliary inputs $\mathbf{aux}$ (denoting the commitments collectively as $\texttt{com} \leftarrow \texttt{Commit}(\mathbf{X}\|\mathbf{y}\|\mathbf{W}\|\mathbf{G}_{\mathbf{W}}\|\mathbf{aux})$ for simplicity), is summarized as Protocol 2.

Protocol 2 achieves perfect completeness and near-certain soundness, ensuring that the proof's acceptance by the trusted verifier closely aligns with the prover's adherence to the prescribed training logic. These two properties are formally stated in Theorems 4.2 and 4.3, respectively.

Theorem 4.2 (Completeness). *Consider a neural network with ReLU activation function, such that ReLU and its back-propagation are the only non-arithmetic operation. Assume that the unscaled $\mathbf{Z}^{(\ell)}$ and $\mathbf{G}_{\mathbf{A}}^{(\ell)}$ are $(Q+R)$-bit integers for each layer ℓ. If the trainer satisfies the following conditions:*

- *Correctly decomposes $\mathbf{Z}^{(\ell)}$ and $\mathbf{G}_{\mathbf{A}}^{(\ell)}$ as specified in* (3) *and* (5)*, respectively.*
- *Computes $\mathbf{A}^{(\ell)}$ and $\mathbf{G}_{\mathbf{Z}}^{(\ell)}$ as specified in* (2) *and* (4)*, respectively.*
- *Performs all arithmetic operations correctly.*
- *Fully adheres to Protocol 2.*

Then, the proof generated by the trainer will be accepted by the trusted verifier with probability 1, achieving perfect completeness.

Protocol 2 zkDL

```
Require: $\mathcal{T}$ knows $\mathbf{X}, \mathbf{y}, \mathbf{W}, \mathbf{G_W}, \mathbf{aux}$, $\mathcal{V}$ knows com
1: $\mathcal{T}$ and $\mathcal{V}$ run Protocol 1 to set-up zkReLU
2: for Layers $\ell \leftarrow 1, 2, \ldots, L$ do
3:     $\mathcal{T}$ and $\mathcal{V}$ run the GKR protocol on layer $\ell$
4: end for
5: Run the sumcheck protocol on (27)
6: $\mathcal{T}$ and $\mathcal{V}$ each run Algorithm 1
7: $\mathcal{T}$ and $\mathcal{V}$ run the inner-product proof on (19)
```

The correctness of Theorem 4.2 is primarily attributed to the design of zkDL and the underlying zkReLU, as outlined in Appendix A. However, from a practical perspective, the definition of soundness should not solely focus on the arithmetic relations and validity of the auxiliary inputs. Instead, it should capture the implied correctness of the ReLU's forward and backward propagations, which constitute the ultimate objective of zkDL.

Theorem 4.3 (Soundness). *Assume the same neural network architecture as in Theorem 4.2 and that $2^{Q+R} \ll |\mathbb{F}|$. If λ is the security parameter of the Pedersen commitment scheme, and the size of the model and data are both polynomially bounded by λ, then with probability $1 - \texttt{negl}(\lambda)$, the following holds: If a proof in Protocol 2 is accepted by the trusted verifier, it implies that:*

- *All arithmetic relations involved in the forward and backward propagations hold.*
- *The forward and backward propagation of the ReLU activation for each layer ℓ is correctly computed as:*

$$\mathbf{A}^{(\ell)} = \mathbb{I}\left\{\left\lfloor \frac{\mathbf{Z}^{(\ell)}}{\gamma} \right\rceil \geq 0\right\} \odot \left\lfloor \frac{\mathbf{Z}^{(\ell)}}{\gamma} \right\rceil, \qquad (28)$$

$$\mathbf{G}_{\mathbf{Z}}^{(\ell)} = \mathbb{I}\left\{\left\lfloor \frac{\mathbf{Z}^{(\ell)}}{\gamma} \right\rceil \geq 0\right\} \odot \left\lfloor \frac{\mathbf{G}_{\mathbf{A}}{}^{(\ell)}}{\gamma} \right\rceil. \qquad (29)$$

Proof of Theorem 4.3. The execution of the GKR protocol within each layer in Line 3 and the sumcheck protocol between the layers and the auxiliary inputs guarantees that the soundness error of all arithmetic relations is bounded by $\texttt{negl}(\lambda)$.

Therefore, the remaining goal is to demonstrate that a valid proof implies the correct execution of the forward and backward propagations with a probability of $1 - \texttt{negl}(\lambda)$.

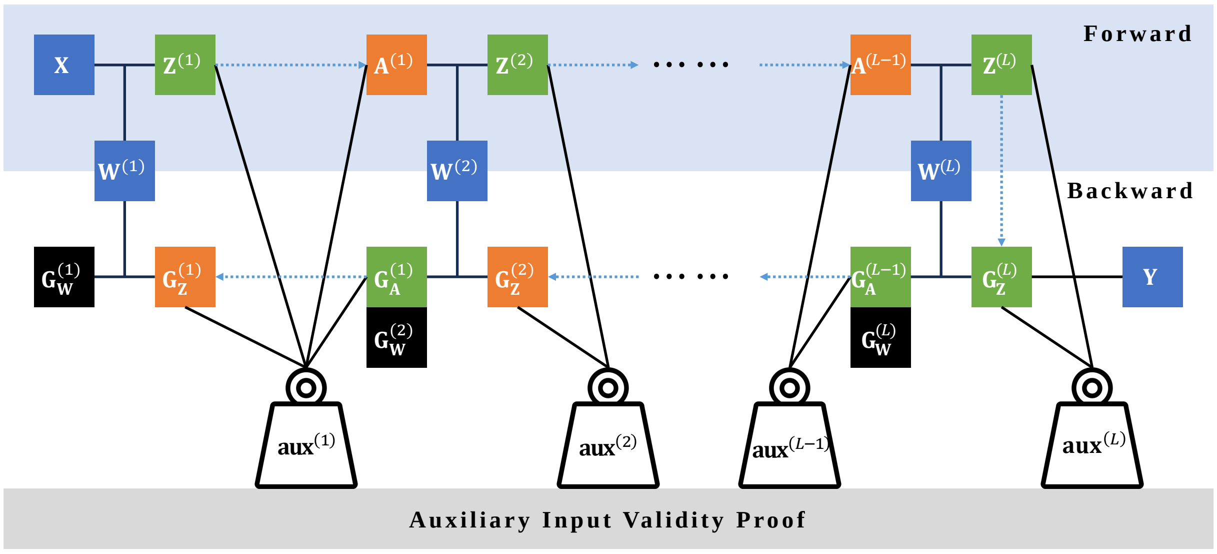

Forward
Backward
$\mathbf{X}$
$\mathbf{Z}^{(1)}$
$\mathbf{A}^{(1)}$
$\mathbf{Z}^{(2)}$
$\mathbf{A}^{(L-1)}$
$\mathbf{Z}^{(L)}$
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(L)}$
$\mathbf{G}_{\mathbf{W}}^{(1)}$
$\mathbf{G}_{\mathbf{Z}}^{(1)}$
$\mathbf{G}_{\mathbf{A}}^{(1)}$
$\mathbf{G}_{\mathbf{W}}^{(2)}$
$\mathbf{G}_{\mathbf{Z}}^{(2)}$
$\mathbf{G}_{\mathbf{A}}^{(L-1)}$
$\mathbf{G}_{\mathbf{W}}^{(L)}$
$\mathbf{G}_{\mathbf{Z}}^{(L)}$
$\mathbf{Y}$
$\mathbf{aux}^{(1)}$
$\mathbf{aux}^{(2)}$
$\mathbf{aux}^{(L-1)}$
$\mathbf{aux}^{(L)}$
Auxiliary Input Validity Proof

Figure 2: The Arithmetic Circuit Design Compatible with zkReLU: The entire circuit is anchored by the auxiliary inputs $\mathbf{aux}^{(\ell)} = \left(\mathbf{Z}^{(\ell)\prime\prime}, \mathbf{B}_{Q-1}^{(\ell)}, \mathbf{R}_{\mathbf{Z}}^{(\ell)}, \mathbf{G}_{\mathbf{A}}^{(\ell)\prime}, \mathbf{R}_{\mathbf{G}_{\mathbf{A}}}^{(\ell)}\right)$ through arithmetic relations (black lines) in replacement of non-arithmetic operations (blue dash arrows, i.e., the "comparison-with-0" operation in ReLU and its gradient). The gradients of the model parameters are marked in black. The data and model parameters are bound by the Pederson commitments and are marked blue. The other tensors are either linear combinations (green) or Hadamard products (orange) of the auxiliary inputs.

This ensures that Equations (28) and (29) hold for each layer ℓ. For simplicity, we will omit the layer index ℓ in the following analysis.

Consider the decomposition of $\mathbf{Z}$ based on Equation (3). According to Theorem 4.1, with a probability of $1 - \mathtt{negl}(\lambda)$, the acceptance of the auxiliary input validity proof implies that $\mathbf{Z}'' \in [0, 2^{Q-1})^D$, $\mathbf{B}_{Q-1} \in \{0,1\}^D$, and $\mathbf{R}_{\mathbf{Z}} \in [-2^{R-1}, 2^{R-1})^D$. Now, suppose there exists another triplet $(\mathbf{Z}''^{\star}, \mathbf{B}_{Q-1}^{\star}, \mathbf{R}_{\mathbf{Z}}^{\star})$ with the same range as $(\mathbf{Z}'', \mathbf{B}_{Q-1}, \mathbf{R}_{\mathbf{Z}})$, and it is also a valid decomposition of $\mathbf{Z}$ based on Equation (3). Then, we have:

$$2^R(\mathbf{Z}''^{\star} - \mathbf{Z}'') + (\mathbf{R}_{\mathbf{Z}}^{\star} - \mathbf{R}_{\mathbf{Z}}) = 2^{Q+R-1}(\mathbf{B}_{Q-1}^{\star} - \mathbf{B}_{Q-1}).$$

By considering the range requirements of the auxiliary inputs, we observe that the left-hand side of the equation lies within the range $\left[-2^{Q+R-1}+1, 2^{Q+R-1}-1\right]$, while the right-hand side takes values in $\{-2^{Q+R-1}, 0, 2^{Q+R-1}\}$. Given the assumption that $2^{Q+R} \ll |\mathbb{F}|$, the intersection of these two ranges is 0, which implies that $\mathbf{B}_{Q-1}^{\star} = \mathbf{B}_{Q-1}$. Furthermore, since $2^R(\mathbf{Z}''^{\star} - \mathbf{Z}'')$ is a multiple of 2^R and $\mathbf{R}_{\mathbf{Z}}^{\star} - \mathbf{R}_{\mathbf{Z}}$ is bounded within $[-2^R+1, 2^R-1]$, both values must be 0 in order to satisfy the summation equation. Therefore, the decomposition of $\mathbf{Z}$ as shown in Equation (3) is unique. Similarly, the decomposition of $\mathbf{G}_{\mathbf{A}}$ as shown in Equation (5) is also unique.

It is worth noting that by setting $\mathbf{Z}' \leftarrow \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rceil$, $\mathbf{R}_{\mathbf{Z}} \leftarrow \mathbf{Z} - 2^R\mathbf{Z}'$, $\mathbf{Z}'' \leftarrow \mathbf{Z}' - 2^R\mathbb{I}\{\mathbf{Z}' < 0\}$, and $\mathbf{B}_{Q-1} = \mathbb{I}\{\mathbf{Z}' < 0\}$, we satisfy the validity of the decomposition given by Equation (3), as well as the range requirements of $\mathbf{Z}''$, $\mathbf{B}_{Q-1}$, and $\mathbf{R}_{\mathbf{Z}}$. Similarly, by setting $\mathbf{G}_{\mathbf{A}}' \leftarrow \left\lfloor \frac{\mathbf{G}_{\mathbf{A}}}{\gamma} \right\rceil$ and $\mathbf{R}_{\mathbf{G}_{\mathbf{A}}} \leftarrow \mathbf{G}_{\mathbf{A}} - 2^R\mathbf{G}_{\mathbf{A}}'$, we satisfy the validity of the decomposition given by Equation (5), as well as the range requirements of $\mathbf{G}_{\mathbf{A}}'$ and $\mathbf{R}_{\mathbf{G}_{\mathbf{A}}}$. Therefore, with a probability of $1 - \mathtt{negl}(\lambda)$, these form the unique decomposition that is necessary for generating a valid proof.

Also, note that with a probability of $1 - \mathtt{negl}(\lambda)$, the arithmetic relations given by Equations (2) and (4) hold. Therefore, combining these with the unique values of the decomposition, we have:

$$\mathbf{A} = \left(\mathbf{Z}' - 2^R\mathbb{I}\{\mathbf{Z}' < 0\}\right) \odot \mathbb{I}\{\mathbf{Z}' \geq 0\} = \mathbf{Z}' \odot \mathbb{I}\{\mathbf{Z}' \geq 0\},$$
$$\mathbf{G}_{\mathbf{Z}} = \mathbf{G}_{\mathbf{A}}' \odot \mathbb{I}\{\mathbf{Z}' \geq 0\},$$

which are equivalent to Equations (28) and (29). This completes the proof of soundness with a probability of $1 - \mathtt{negl}(\lambda)$. □

In addition to fulfilling the completeness and soundness requirements, the zkDL protocol also guarantees zero-knowledge, ensuring that it reveals no information about the training set and model parameters. This property is formalized in Theorem 4.4, and the proof is outlined in Appendix A.

Theorem 4.4 (Zero-knowledge)**.** *Assuming the usage of the zero-knowledge Pedersen commitment scheme and the zero-knowledge variant of the GKR protocol [10], [12], [42], Protocol 2 is zero-knowledge. In particular, for a security parameter λ and any probabilistic polynomial (PPT) algorithm $\mathcal{A}$, there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that the following two views are indistinguishable by $\mathcal{A}$ when provided with the public parameter $\mathtt{pp}$ (corresponding to the generators used in the commitment scheme) as input.*

```
Real:
  1: com ← Commit $(\mathbf{X}\|\mathbf{y}\|\mathbf{W}\|\mathbf{G_W}\|\mathbf{aux}; pp)$
  2: $\pi \leftarrow \langle \mathcal{T}, \mathcal{A} \rangle$.zkDL(com; pp)
  3: return com, $\pi$
```

```
Ideal:
  1: com ← $\mathcal{S}_1(1^\lambda; pp)$
  2: $\pi \leftarrow \langle \mathcal{S}_2, \mathcal{A} \rangle (com; pp)$, given oracle access to $C$
  3: return com, $\pi$
```

In the given context, `Commit` $(\mathbf{X}\|\mathbf{y}\|\mathbf{W}\|\mathbf{G_W}\|\mathbf{aux}; pp)$ *represents the process of making commitments for the data, model parameters, gradients, and auxiliary inputs using the generators contained in* pp*. The event $C = C^a \wedge C^v$ denotes the occurrence of two conditions: the satisfaction of all arithmetic relations (C^a) and the fulfillment of the value requirements for all auxiliary inputs (C^v).*

Example 4.5 (FCNN)**.** *We examine a fully connected neural network (FCNN) comprising L layers, assuming inputs and outputs of dimension d and employing the square loss. Consider a batch of data points denoted as $(\mathbf{X} = \mathbf{A}^{(0)}, \mathbf{Y})$. In this context, we establish the following requirements:*

$$\mathbf{Z}^{(\ell)} = \mathbf{A}^{(\ell-1)}\mathbf{W}^{(\ell)}, \qquad 1 \le \ell \le L, \tag{30}$$

$$\mathbf{A}^{(\ell)} = (\mathbf{1} - \mathbf{B}^{(\ell)}_{Q-1}) \odot \mathbf{Z}^{(\ell)\prime\prime}, \qquad 1 \le \ell \le L-1, \tag{31}$$

$$\mathbf{G}^{(L)}_{\mathbf{Z}} = \mathbf{Z}^{(L)\prime} - \mathbf{Y}, \tag{32}$$

$$\mathbf{G}^{(\ell)}_{\mathbf{A}} = \mathbf{G}^{(\ell+1)}_{\mathbf{Z}}\mathbf{W}^{(\ell+1)\top}, \qquad 1 \le \ell \le L-1, \tag{33}$$

$$\mathbf{G}^{(\ell)}_{\mathbf{W}} = \mathbf{G}^{(\ell)\top}_{\mathbf{Z}}\mathbf{A}^{(\ell-1)}, \qquad 1 \le \ell \le L, \tag{34}$$

$$\mathbf{G}^{(\ell)}_{\mathbf{Z}} = (\mathbf{1} - \mathbf{B}^{(\ell)}_{Q-1}) \odot \mathbf{G}^{(\ell)\prime}_{\mathbf{A}}, \qquad 1 \le \ell \le L-1. \tag{35}$$

Here, the prover sends the commitments of auxiliary inputs $\mathbf{Z}^{(\ell)\prime\prime}$, $\mathbf{B}^{(\ell)}_{Q-1}$, $\mathbf{R}^{(\ell)}_{\mathbf{Z}}$, $\mathbf{G}^{(\ell)\prime}_{\mathbf{A}}$, $\mathbf{R}^{(\ell)}_{\mathbf{G_A}}$ for each layer $1 \le \ell \le L$ to the verifier, along with the commitments of the weights and data. The homomorphism of the commitments allows for direct verification of the equality in (32)*. Furthermore, the equality checks in* (30)*,* (33)*, and* (34) *can be performed using the sumcheck for matrix products. Notably, since the tensors involved in these operations are either committed values or $\mathbf{A}^{(\ell)}$ and $\mathbf{G}^{(\ell)}_{\mathbf{Z}}$, and assuming the layers have the same dimension or zero-padding is applied otherwise, the equality check for multiple layers can be batched using random linear combination, resulting in a reduced proof size by a factor of L. Additionally, batching these proofs simplifies the correctness verification, as it relies on claims of random point evaluation on the stacked tensors $\mathbf{G_Z}$ and $\mathbf{G_A}$ and reduces to the claims involving $\widetilde{\mathbf{B}_{Q-1}}(\mathbf{u}_{relu})$, $\widetilde{\mathbf{Z}''}(\mathbf{u}_{relu})$, and $\widetilde{\mathbf{G}'_{\mathbf{A}}}(\mathbf{u}_{relu})$ using* (31) *and* (35)*.*

4.3. Overhead analysis

The zkReLU protocol, along with its compatible arithmetic circuit design, significantly reduces the computational overhead for the trainer in terms of both the time and space complexity of the proof. Table 1 provides an overview of the asymptotic advantages gained in proof time and sizes when compared with the naive bit decomposition and conventional sequential proofs that follow the ordering of the layers in the neural network.

Proving time. The non-arithmetic operations, including ReLU, have been the dominant factor contributing to the time complexity of zero-knowledge proofs for machine learning. In Section 4.1, the validity proof requires $O(DQ)$ group and field operations for each layer, which is asymptotically equivalent to the computation time needed for representing valid auxiliary inputs using bits. Specifically, the transformations of the auxiliary inputs and their commitments into a single inner product require $O(DQ)$ time, as well as the proof of the inner product itself.

In contrast, straightforwardly performing the sumcheck protocol between the auxiliary inputs and their bit decompositions would require $\Omega(D^2Q)$ operations per layer [9], [10], [11], [12] on the equation:

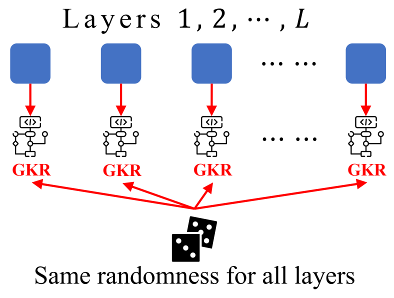
$$\widetilde{\mathbf{aux}}(\mathbf{u}) = \sum_{i\in\{0,1\}^{\log_2 D}} \sum_{j\in\{0,1\}^{\log_2 D}} \sum_{k\in\{0,1\}^{\log_2 Q}} \widetilde{\beta}(\mathbf{u}, i)\,\widetilde{\mathbf{add}}(i,j,k)\,\widetilde{\mathbf{B}}(j,k)\,2^k, \tag{36}$$

where $\widetilde{\mathbf{B}}$ represents the multilinear extension of bit decomposition of the auxiliary input $\mathbf{aux}$, and $\widetilde{\mathbf{add}}$ corresponds to the multilinear extension of wiring predicates used for the proofs on general arithmetic circuits. Therefore, an improvement factor of $O(D)$ is achieved for each ReLU activation. Additionally, due to the design of the circuit that incurs high parallelizability, the proof time on the entire forward and backward propagation process can be further compressed.

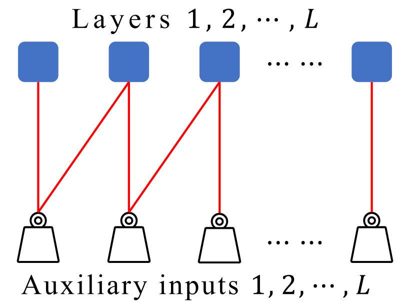
Parallelization. The proof of the training process based on zkReLU offers a higher degree of parallelization compared to the training process itself. As described in Section 4.2, the order of proof generation is NOT bound by the precedence relationships of the neural network layers. Instead, as illustrated in Figure 3, the trainer needs to prove *independently for each layer* **a)** the arithmetic relations within the layer, **b)** the arithmetic relations between the layer and the auxiliary inputs, and **c)** the validity of the auxiliary inputs. Therefore, each of the three steps can be run on all layers *in parallel*, without being subject to the ordering of the layers in the neural network. As a result, the total proving time can be reduced by a factor of $O(L)$, while only adding an $O(\log L)$ overhead to combine the proofs for all layers, which is typically dominated by the $O(DQ)$ per-layer proving time.

Given that deep learning computations are commonly performed in highly parallelized environments, harnessing a significant degree of parallelization can effectively leverage the available computational resources and substantially diminish the additional overhead imposed on the original training time. Additionally, parallelization not only avoids incurring a trade-off between proof time and proof sizes but also *reduces* the proof size.

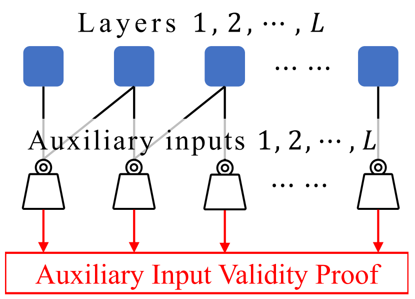
Proof size. By leveraging the independent nature of different layers in the circuit, the proofs of arithmetic operations



(a) Run the GKR protocol on each layer in parallel on the same randomness.



(b) Prove the arithmetic relations, as in Section 4.2, between all layers and the auxiliary inputs in parallel.



Layers $1, 2, \cdots, L$
Auxiliary inputs $1, 2, \cdots, L$
Auxiliary Input Validity Proof

(c) Run the validity proof, as in Section 4.1, on all auxiliary inputs in parallel.

Figure 3: The proof is conducted in three consecutive steps, where the proving process in each step is denoted by the red lines. In each step, the proof can be generated in parallel over all L layers. This reduces the proof time by a factor of $O(L)$.

in different layers can be batched using a random linear combination. Additionally, the committed auxiliary inputs and outputs from different layers can be stacked together and committed as a single tensor.

Therefore, by using the same randomness for all layers in each step of the proof (as illustrated in Figure 3), the proofs for all layers can be compressed into one, which only introduces an additive overhead of $O(\log L)$ to the proof size of a single layer, where L represents the total number of layers. In contrast, a serially generated proof concatenates the proofs of each layer, resulting in a linear growth of proof size in the number of layers multiplied by the proof size of a single layer. Typically, the proof size of a single layer is $O(\log(DQ))$, where D represents the dimensionality of the auxiliary inputs in a single layer and Q represents the maximum number of bits for the values of the auxiliary inputs.

4.4. Dealing with training data

Upon verification of the training logic based on zkReLU, the trusted verifier provides an endorsement for the committed model parameters and datasets. The committed model parameters can be utilized for zero-knowledge verifiable inferences, while the authenticity of the training data can also be verified using the commitment of the data points.

Since each data point is typically involved in the training loop multiple times and may be assigned to different batches in different epochs, it is necessary to individually commit the data points before the training begins. In each step of the zkReLU protocol, a claim must be proven regarding the random-point evaluation of the multilinear extension $\widetilde{\mathbf{X}}(\mathbf{u}, \mathbf{v})$. Here, $\mathbf{X} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{B-1})$ represents the data batch with a batch size of B and dimension d. For convenience, we assume both B and d are powers of 2. Additionally, $\mathbf{u} \sim \mathbb{F}^{\log_2 B}$ and $\mathbf{v} \sim \mathbb{F}^{\log_2 d}$ represent the randomness generated during the execution of the zkReLU protocol.

To achieve this, the data points are separately committed using the Pedersen commitment scheme, resulting in commitments $\mathsf{com}_{\mathbf{x}_0}, \mathsf{com}_{\mathbf{x}_1}, \ldots, \mathsf{com}_{\mathbf{x}_{B-1}}$. Both the trainer and trusted verifier can then compute the commitment of $\widetilde{\mathbf{X}}(\mathbf{u}, \cdot) \in \mathbb{F}^d$ as follows:

$$\mathsf{com}_{\widetilde{\mathbf{X}}(\mathbf{u},\cdot)} := \prod_{i \in \{0,1\}^{\log_2 B}} \mathsf{com}_{\mathbf{x}_i}^{\widetilde{\beta}(\mathbf{u}, i)}.$$

Subsequently, the trainer proceeds to perform a proof of opening directly on $\mathsf{com}_{\widetilde{\mathbf{X}}(\mathbf{u},\cdot)}$ in order to demonstrate the claim regarding $\widetilde{\mathbf{X}}(\mathbf{u}, \mathbf{v})$.

In addition to the endorsement by the trusted verifier, the trainer constructs a Merkle tree on all data points. This Merkle tree can be checked by the trusted verifier and enables data copyright owners to inquire about the membership status of their data points in the training set. The identification of data points is accomplished using Pedersen hash functions, assuming a hash output bit length of k, and employing deterministic Pedersen commitments with the randomness set to 0, as described in Section 3.1.

The Merkle tree is constructed based on a complete binary tree with a height of k, ensuring that each data point is stored in a leaf node identified by its hash, presented in the form of a bit string. When a data point is queried, the trainer can provide a zero-knowledge proof of membership or non-membership. This proof takes the form of a path from the leaf node identified by (a prefix of) the hash of the queried data point to the root of the Merkle tree [46]. Further details regarding the proof of membership or non-membership are provided in the appendix.

5. Experiments

In this section, we present the experimental evaluations of zkDL conducted on a Linux server equipped with 2 AMD EPYC 7532 CPUs, each with 32 cores and 256 GB of RAM. The experiments were performed using the CIFAR-10 dataset, with a dimension of 3,072, padded to 4,096 as a power of 2. The neural network architecture and circuit design described in Example 4.5 were utilized for the experiments. To control the rounding errors caused by

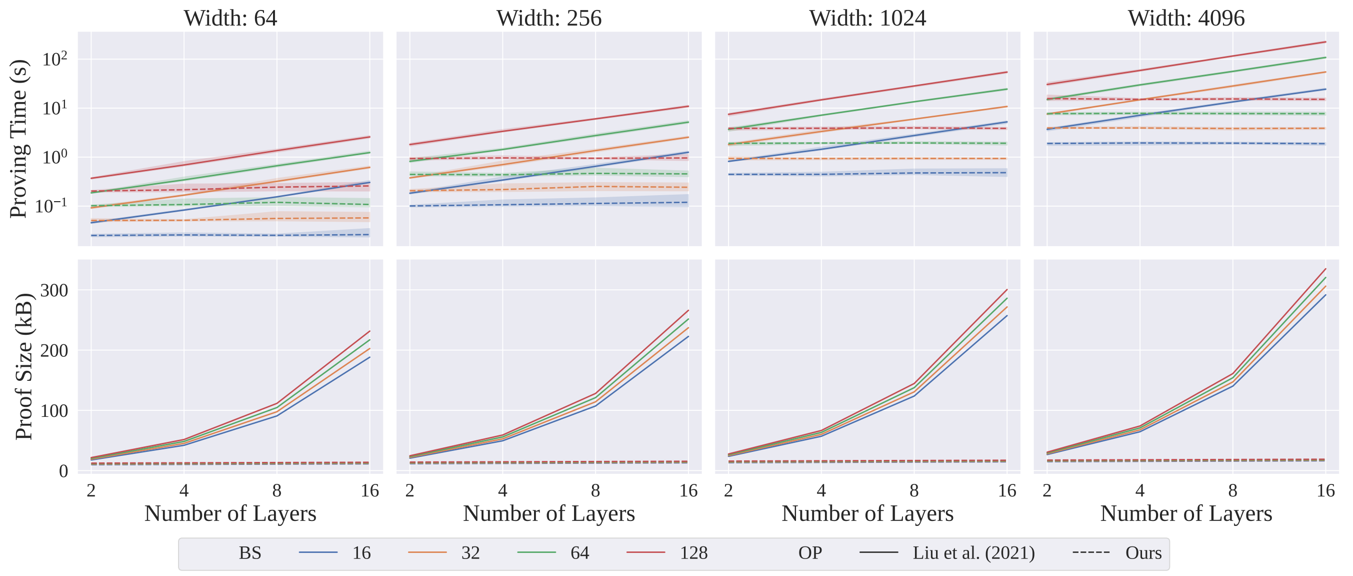


Figure 4: Per-step proving time and proof size with different orders of proof (OP): The parallel proof generation in zkReLU (ours), and the conventional layer-by-layer proof generation formalized by Liu et al. in 2021 [1]. By our design, the proof time is reduced by a factor of $O(L)$, and the proof sizes grow slowly by an additive term $O(\log L)$, instead of multiplied by an $O(L)$ factor as the depth L of the neural network grows.

TABLE 2: Proving times (s) and proof sizes (kB) of zkReLU and Sum-Check Bit-Decomposition (SC-BD) on a full-connected network of $L = 2$ layers. Runs exceeding the time limit are marked as $> 10^3$ (s). #param and #aux represent the number of parameters and auxiliary inputs in each run. The table records the performance of proof generation of whole networks in one batch update.

Width	# param	BS	# aux	**zkReLU (ours)**		SC-BD	
				time (s)	size (kB)	time (s)	size (kB)
64	8.3K	16	5.1×10^3	0.033	9.4	12	14
		32	1.0×10^4	0.068	10	44	15
		64	2.0×10^4	0.13	11	2.1×10^2	16
		128	4.1×10^4	0.27	12	$> 10^3$	17
256	130K	16	2.0×10^4	0.14	11	2.0×10^2	16
		32	4.1×10^4	0.50	12	$> 10^3$	17
		64	8.2×10^4	0.57	13	$> 10^3$	18
		128	1.6×10^5	1.2	14	$> 10^3$	20
1,024	2.1M	16	8.2×10^4	0.80	13	$> 10^3$	18
		32	1.6×10^5	1.4	14	$> 10^3$	20
		64	3.3×10^5	2.8	14	$> 10^3$	21
		128	6.5×10^5	6.0	15	$> 10^3$	22
4,096	33M	16	3.3×10^5	3.9	14	$> 10^3$	21
		32	6.5×10^5	6.6	15	$> 10^3$	22
		64	1.3×10^6	11	16	$> 10^3$	23
		128	2.6×10^6	25	17	$> 10^3$	24

quantization, a scaling factor of 2^{16} was applied, and all real values involved in the computation were assumed to fall within the range of $[-2^{15}, 2^{15})$, making them amenable to scaling as 32-bit integers. Notably, no overflow issues were encountered during the experiments. We employed the MCL library [47] to handle finite fields and elliptic curves, while the XTensor library [48] was utilized for tensor-based operations involved in deep learning and their associated proofs.

Prior works on zero-knowledge verifiable inference, whether bit-decomposition-based or lookup-table-based [1], [2], [5], lack sufficient techniques to generalize the verifiability to the training phase, especially concerning the handling of ReLU. As a result, we follow the approach of pioneering works on zero-knowledge verifiable training [6], [7] and implement the bit-decomposition-based handling of ReLU on the general-purpose ZKP backends, serving as our baseline for comparison.

5.1. Experiments on zkReLU

To evaluate the effectiveness of the zkReLU protocol, we initially focus on 2-layer perceptrons, which include only one ReLU activation. We vary the width (number of neurons in each layer) and the batch size (BS), recording the per-batch proving time and proof size in Table 2. Additionally, we compare the proving time and sizes of zkReLU with the naive bit-decomposition (BD) approach using the sumcheck protocol, which represents how ReLU is handled in general-purpose zero-knowledge proof (ZKP) backends. A proving time limit of 10^3 seconds is set for each experiment run. In the case of a timeout, the proof size is derived analytically without matching it with the experimental results for a sanity check.

In Table 2, it can be observed that the proof time and proof sizes both exhibit a feasible rate of growth with respect to the increasing width and batch sizes. This scalability allows the proof of training logic based on zkReLU to be applied to models with 33 millions parameters within a running time of less than a half minute. In contrast, the naïve bit-decomposition (BD)-based method experiences unrealistic running times, except for small models and batch sizes. This highlights the limitation of using general-purpose

TABLE 3: Proof size (i.e., the number of hash values released by the model trainer) and verification time (in milliseconds) of proof of (non-)membership for different positivity ratios with various hash functions. The second column refers to the number of queried data to be verified w.r.t. their training membership. The third column shows the tree construction time $t_{\texttt{tree}}$ (in seconds).

hash	# data	$t_{\texttt{tree}}$ (s)	Positivity ratio									
			0		0.1		0.5		0.9		1	
			size (#)	time (ms)	size (#)	time (ms)	size (#)	time (ms)	size (#)	time (ms)	size (#)	time (ms)
md5	10	174	148	0.84	260	4.6	697	12	1,136	19	1,244	22
	100		1,059	5.9	2,168	37	6,632	110	11,042	200	12,163	220
	1,000		7,148	48	18,248	350	62,565	1,300	107,094	2,200	118,180	2,300
sha1	10	256	136	0.79	284	5.9	854	17	1,419	29	1,564	32
	100		1,033	5.7	2,481	54	8,196	170	13,905	320	15,333	370
	1,000		6,995	45	21,312	530	78,583	2,900	135,775	4,600	150,122	6,000
sha256	10	602	147	0.99	388	13	1,342	41	2,288	71	2,530	79
	100		1,036	6.3	3,436	100	12,987	460	22,575	780	24,962	870
	1,000		7,163	53	31,055	1,100	126,617	7,100	222,259	15,000	246,158	17,000

[†] zkDL achieves 100% membership inference accuracy in all above experiments, in contrast to a maximum of 63.7% with MIA [30].

zero-knowledge proof (ZKP) backends as a black box for deep learning proofs. It emphasizes the need for specifically tailored ZKP schemes to effectively prove the correctness of deep learning execution, particularly during the training process.

Additionally, we conduct experiments to further investigate the advantage of the high degree of parallelization in the circuit design compatible with zkReLU. We focus on the proof of training logic on multi-layer perceptrons with varying depths. We compare the optimized running times and proof sizes of parallel proofs, where the same randomness is applied to each layer, against those of conventional sequential proofs that align with the layer structure assumed in previous works [5], [6], [7] and formalized by Liu et al. in 2021 [1]. The results of these experiments are presented in Figure 4.

The parallel generation of proofs in zkReLU offers a significant advantage over conventional sequential proofs, as the proving time is no longer subject to linear increase with respect to the depth of the network. This parallelization enables the proof of the entire forward and backward process to be completed within tens of seconds, even for a 16-layer perceptron with over 200M parameters. Additionally, the proof size is efficiently controlled under 30 kB, compared to the larger sizes of up to 200-300 kB that occur with increasing depth. By harnessing the parallel computational resources available for deep learning, the proof generation based on zkReLU-compatible circuits achieves more favourable proof times and sizes, representing a significant step forward in the practical application of zero-knowledge proofs for deep learning in the AI industry.

5.2. Additional experiments on the training data

To evaluate the proof of (non-)membership described in Section 4.4, we implemented the Merkle tree on the CIFAR-10 training set using three different hash functions: `md5`, `sha1`, `sha256`. We conducted experiments with varying query sizes and positivity ratios (the ratio of positive data points, i.e., members of the training set, in the query set) and recorded the results.

Table 3 illustrates that Merkle tree construction times, proof sizes, and verification times increase with the query size and output length of the hash. Notably, queries with a larger positivity ratio exhibit increased complexity. However, when data copyrights are not violated (positivity ratio is 0), it only takes 0.05 milliseconds on average for a copyright owner to confirm that a data point they own is not in the training set. This constitutes a significant improvement over naïvely loading and scanning the entire committed dataset, which takes 14 seconds.

Furthermore, the proof of (non-)membership achieves 100% accuracy due to the correctness and soundness of the Merkle tree. No data point was found for which the trainer can lie about its membership in the training dataset. This represents a substantial improvement over membership inference attacks [34], which achieve only 59.0% to 63.7% accuracy on the same dataset (CIFAR-10). These results underscore the crucial role of zkDL in providing rigorous guarantees of data legitimacy in deep learning.

6. Conclusion

This paper presents zkDL, a novel solution to the challenge of zero-knowledge verifiable training for neural networks. Specifically focusing on the ReLU activation function, zkDL addresses the historical incompatibility with general-purpose ZKP backends. The protocol's design includes multiple efficient and mutually compatible ZKP systems tailored for deep learning training, as well as a novel modeling scheme of neural networks as arithmetic circuits, enabling a high degree of proof parallelization. These advancements significantly reduce time and communication costs associated with proving legitimate execution of deep learning training, effectively resolving legitimacy issues surrounding trained neural networks. Looking ahead, further theoretical development and practical implementations of efficient ZKP systems tailored for deep learning are anticipated to establish zkDL as a powerful tool safeguarding the AI industry's development.

References

[1] T. Liu, X. Xie, and Y. Zhang, "zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 2968–2985. [Online]. Available: https://doi.org/10.1145/3460120.3485379

[2] B. Feng, L. Qin, Z. Zhang, Y. Ding, and S. Chu, "ZEN: efficient zero-knowledge proofs for neural networks," *IACR Cryptol. ePrint Arch.*, p. 87, 2021. [Online]. Available: https://eprint.iacr.org/2021/087

[3] S. Lee, H. Ko, J. Kim, and H. Oh, "vcnn: Verifiable convolutional neural network," *IACR Cryptol. ePrint Arch.*, p. 584, 2020. [Online]. Available: https://eprint.iacr.org/2020/584

[4] J. Weng, J. Weng, G. Tang, A. Yang, M. Li, and J. Liu, "pvcnn: Privacy-preserving and verifiable convolutional neural network testing," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 2218–2233, 2023. [Online]. Available: https://doi.org/10.1109/TIFS.2023.3262932

[5] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Scaling up trustless DNN inference with zero-knowledge proofs," *CoRR*, vol. abs/2210.08674, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.08674

[6] 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 Trans. Parallel Distributed Syst.*, vol. 32, no. 10, pp. 2524–2540, 2021. [Online]. Available: https://doi.org/10.1109/TPDS.2021.3068195

[7] T. Eisenhofer, D. Riepel, V. Chandrasekaran, E. Ghosh, O. Ohrimenko, and N. Papernot, "Verifiable and provably secure machine unlearning," *CoRR*, vol. abs/2210.09126, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.09126

[8] R. E. Ali, J. So, and A. S. Avestimehr, "On polynomial approximations for privacy-preserving and verifiable relu networks," *CoRR*, vol. abs/2011.05530, 2020. [Online]. Available: https://arxiv.org/abs/2011.05530

[9] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, C. Dwork, Ed. ACM, 2008, pp. 113–122. [Online]. Available: https://doi.org/10.1145/1374376.1374396

[10] 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*, ser. Lecture Notes in Computer Science, A. Boldyreva and D. Micciancio, Eds., vol. 11694. Springer, 2019, pp. 733–764. [Online]. Available: https://doi.org/10.1007/978-3-030-26954-8_24

[11] J. Zhang, T. Liu, W. Wang, Y. Zhang, D. Song, X. Xie, and Y. Zhang, "Doubly efficient interactive proofs for general arithmetic circuits with linear prover time," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 159–177. [Online]. Available: https://doi.org/10.1145/3460120.3484767

[12] T. Xie, Y. Zhang, and D. Song, "Orion: Zero knowledge proof with linear prover time," in *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, Y. Dodis and T. Shrimpton, Eds., vol. 13510. Springer, 2022, pp. 299–328. [Online]. Available: https://doi.org/10.1007/978-3-031-15985-5_11

[13] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013, pp. 238–252. [Online]. Available: https://doi.org/10.1109/SP.2013.47

[14] J. Groth, "On the size of pairing-based non-interactive arguments," 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*, ser. Lecture Notes in Computer Science, M. Fischlin and J. Coron, Eds., vol. 9666. Springer, 2016, pp. 305–326. [Online]. Available: https://doi.org/10.1007/978-3-662-49896-5_11

[15] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *IACR Cryptol. ePrint Arch.*, p. 953, 2019. [Online]. Available: https://eprint.iacr.org/2019/953

[16] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, "Halo infinite: Recursive zk-snarks from any additive polynomial commitment scheme," *IACR Cryptol. ePrint Arch.*, p. 1536, 2020. [Online]. Available: https://eprint.iacr.org/2020/1536

[17] 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 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 926–943. [Online]. Available: https://doi.org/10.1109/SP.2018.00060

[18] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth, "Succinct non-interactive arguments via linear interactive proofs," *J. Cryptol.*, vol. 35, no. 3, p. 15, 2022. [Online]. Available: https://doi.org/10.1007/s00145-022-09424-4

[19] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, and J. Chen, "Verifi: Towards verifiable federated unlearning," *CoRR*, vol. abs/2205.12709, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.12709

[20] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Towards probabilistic verification of machine unlearning," *CoRR*, vol. abs/2003.04247, 2020. [Online]. Available: https://arxiv.org/abs/2003.04247

[21] C. Guo, T. Goldstein, A. Y. Hannun, and L. van der Maaten, "Certified data removal from machine learning models," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 3832–3842. [Online]. Available: http://proceedings.mlr.press/v119/guo20c.html

[22] J. Weng, S. Yao, Y. Du, J. Huang, J. Weng, and C. Wang, "Proof of unlearning: Definitions and instantiation," *CoRR*, vol. abs/2210.11334, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.11334

[23] H. Jia, M. Yaghini, C. A. Choquette-Choo, N. Dullerud, A. Thudi, V. Chandrasekaran, and N. Papernot, "Proof-of-learning: Definitions and practice," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 1039–1056. [Online]. Available: https://doi.org/10.1109/SP40001.2021.00106

[24] C. Fang, H. Jia, A. Thudi, M. Yaghini, C. A. Choquette-Choo, N. Dullerud, V. Chandrasekaran, and N. Papernot, "On the fundamental limits of formally (dis)proving robustness in proof-of-learning," *CoRR*, vol. abs/2208.03567, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.03567

[25] R. Zhang, J. Liu, Y. Ding, Z. Wang, Q. Wu, and K. Ren, ""adversarial examples" for proof-of-learning," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 1408–1422. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833596

[26] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 268–282. [Online]. Available: https://doi.org/10.1109/CSF.2018.00027

[27] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 3093–3106. [Online]. Available: https://doi.org/10.1145/3548606.3560675

[28] B. Jayaraman, L. Wang, K. Knipmeyer, Q. Gu, and D. Evans, "Revisiting membership inference under realistic assumptions," *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 2, pp. 348–368, 2021. [Online]. Available: https://doi.org/10.2478/popets-2021-0031

[29] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 2615–2632. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/song

[30] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 5558–5567. [Online]. Available: http://proceedings.mlr.press/v97/sablayrolles19a.html

[31] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "A pragmatic approach to membership inferences on machine learning models," in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 2020, pp. 521–534. [Online]. Available: https://doi.org/10.1109/EuroSP48549.2020.00040

[32] L. Watson, C. Guo, G. Cormode, and A. Sablayrolles, "On the importance of difficulty calibration in membership inference attacks," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=3eIrli0TwQ

[33] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 3093–3106. [Online]. Available: https://doi.org/10.1145/3548606.3560675

[34] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 1897–1914. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833649

[35] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 3–18. [Online]. Available: https://doi.org/10.1109/SP.2017.41

[36] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 634–646. [Online]. Available: https://doi.org/10.1145/3243734.3243855

[37] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 259–274. [Online]. Available: https://doi.org/10.1145/3319535.3363201

[38] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 332–349. [Online]. Available: https://doi.org/10.1109/SP.2019.00019

[39] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/ml-leaks-model-and-data-independent-membership-inference-attacks-and-defenses-on-machine-learning-models/

[40] G. Cormode, M. Mitzenmacher, and J. Thaler, "Practical verified computation with streaming interactive proofs," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, S. Goldwasser, Ed. ACM, 2012, pp. 90–112. [Online]. Available: https://doi.org/10.1145/2090236.2090245

[41] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *J. ACM*, vol. 39, no. 4, pp. 859–868, 1992. [Online]. Available: https://doi.org/10.1145/146585.146605

[42] A. Chiesa, M. A. Forbes, and N. Spooner, "A zero knowledge sumcheck and its applications," *Electron. Colloquium Comput. Complex.*, vol. TR17-057, 2017. [Online]. Available: https://eccc.weizmann.ac.il/report/2017/057

[43] J. Thaler, "Time-optimal interactive proofs for circuit evaluation," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, ser. Lecture Notes in Computer Science, R. Canetti and J. A. Garay, Eds., vol. 8043. Springer, 2013, pp. 71–89. [Online]. Available: https://doi.org/10.1007/978-3-642-40084-1_5

[44] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4672–4681. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/6048ff4e8cb07aa60b6777b6f7384d52-Abstract.html

[45] 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 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 315–334. [Online]. Available: https://doi.org/10.1109/SP.2018.00020

[46] S. Micali, M. O. Rabin, and J. Kilian, "Zero-knowledge sets," in *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. IEEE Computer Society, 2003, pp. 80–91. [Online]. Available: https://doi.org/10.1109/SFCS.2003.1238183

[47] S. Mitsunari, "MCL: A portable and fast pairing-based cryptography library," https://github.com/herumi/mcl, 2023, accessed: 2023-07-22.

[48] QuantStack, "XTensor," https://xtensor.readthedocs.io, 2023, accessed: 2023-07-22.

[49] 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*, ser. Lecture Notes in Computer Science, M. Abe, Ed., vol. 6477. Springer, 2010, pp. 177–194. [Online]. Available: https://doi.org/10.1007/978-3-642-17373-8_11

Appendix A. zkReLU

Proof sketch of Theorem 4.2. By the perfect completeness of the GKR protocol and the underlying sumcheck protocol, the trusted verifier accepts the interactive proofs in Lines 3 and 5 with probability 1, assuming the correct execution of all arithmetic operations within each layer and the Hadamard products in Equations (2) and (4).

Moreover, given the assumption that $\mathbf{Z}^{(\ell)}$ and $\mathbf{G}_{\mathbf{A}}^{(\ell)}$ are $(Q+R)$-bit integers, the decomposition based on Equations (3) and (5) ensures that the auxiliary inputs fall within their respective ranges. By correctly computing $\mathbf{B}_k$ and $\mathbf{B}'_k$, the trainer establishes the equalities in Equations (6), (7), (8), and the inner product in Equation (19). Therefore, based on the perfect completeness of the inner-product proof [45], the trusted verifier accepts the validity of the auxiliary inputs with probability 1 in Line 7. □

Proof sketch of Theorem 4.4. First, consider the GKR protocol applied to all the arithmetic operations involving $\mathbf{X}$, $\mathbf{y}$, $\mathbf{W}$, $\mathbf{G_W}$, and $\mathbf{aux}$. By assuming the zero-knowledge properties of the commitment scheme and the GKR protocol, there exist simulators $\mathcal{S}^a = (\mathcal{S}_1^a, \mathcal{S}_2^a)$ such that $\mathcal{S}_1^a$ simulates the generation process of the commitments and $\mathcal{S}_2^a$ simulates the GKR-based proof of the arithmetic circuits, given oracle access to C^a.

On the other hand, the auxiliary input validity proofs involve the zero-knowledge commitments of additional inputs, $\texttt{com}_{\mathbf{B}}^{\text{ip}}$ and $\texttt{com}_{\mathbf{B}_{Q-1}}^{\text{ip}}$ as defined in Protocol 1. These commitments are subsequently transformed in Algorithm 1 to perform the zero-knowledge inner-product proof. Therefore, there exists another simulator $\mathcal{S}^v = (\mathcal{S}_1^v, \mathcal{S}_2^v)$ such that $\mathcal{S}_1^v$ simulates the generation of the commitments and $\mathcal{S}_2^v$ simulates the inner-product proof, given oracle access to C^v. Thus, with $\mathcal{S}_1 = \mathcal{S}_1^a$ and $\mathcal{S}_2 = (\mathcal{S}_1^v, \mathcal{S}_2^a, \mathcal{S}_2^v)$, the ideal view is indistinguishable from the real view. □

Appendix B. Proof of (non-)membership

This section presents the proof of (non-)membership based on a Merkle tree that allows data copyright owners to query the membership of their data points in the dataset. After verifying the proofs of data quality and training logic, the trusted verifier requests the trainer to submit the root value of the Merkle tree, which the trusted verifier also computes itself using the commitments of all data points. If the two computed root values match, the trusted verifier endorses the private dataset and model parameters by publishing a digital signature on the committed model parameters and the root value of the Merkle tree. This signature enables the proof of (non-)membership. Upon the query on a data point, in the form of its commitment, the trainer computes the path in the Merkle tree to the root as the proof. The data copyright owner checks whether the reconstructed root value matches the one signed by the trusted verifier, providing a cost-effective and privacy-preserving way of verifying data ownership.

Using the Merkle tree reduces the size of the proof and limits the information leaked about data points that are not being queried. However, the original version of the Merkle tree cannot be directly applied since it cannot handle the proof of non-membership, where a data point is excluded from the training set as required. To address this, we introduce a variant of the Merkle tree that supports both the proof of membership and non-membership.

To construct the Merkle tree, we assume the existence of a collision-resistant hash function $\texttt{hash} : \{0,1\}^* \to \{0,1\}^k$. The model trainer begins by computing the hash of the commitment of each data point: $\{\texttt{hash}(\texttt{com}_d) : d \in \mathcal{D}\}$.

For analysis purposes, we consider the complete binary tree $\mathcal{T}_k$ with depth k (which is not required to be implemented). The nodes of depth i $(0 \leq i \leq k)$ in $\mathcal{T}_k$ can be identified with bit strings of length i. The root of $\mathcal{T}_k$ is identified with the empty string ϵ, and each of the 2^k leaves of $\mathcal{T}_k$ is identified with a bit string of length k. For each leaf node $b_1 b_2 \dots b_k$, we define $\texttt{Path}_k(b_1 b_2 \dots b_k)$ as the path from the node to the root, i.e., $b_1 b_2 \dots b_k \to b_1 b_2 \dots b_{k-1} \to \cdots \to b_1 b_2 \to b_1 \to \texttt{root}$, where the $\texttt{root}$ is represented by the empty string. We also define $H_{\mathcal{D}} := \{\texttt{hash}(\texttt{com}_d)\}_{d \in \mathcal{D}}$ as the collection of hashes of the data points. Due to the collision-resistance property, there is a bijection between $H_{\mathcal{D}}$ and $\mathcal{D}$ with an overwhelming probability.

We define the subtree corresponding to the training set $\mathcal{D}$ as the union of the paths from the nodes to the root in $H_{\mathcal{D}}$, i.e., $\texttt{Tree}(H_{\mathcal{D}}) := \bigcup_{h \in H_{\mathcal{D}}} \texttt{Path}_k(h)$; and its frontier as the nodes that are not in $\texttt{Tree}(H_{\mathcal{D}})$, but have its parent in $\texttt{Tree}(H_{\mathcal{D}})$, i.e.,

$$\texttt{Frontier}(H_{\mathcal{D}}) := \{vb : vb \notin \texttt{Tree}(H_{\mathcal{D}}), v \in \texttt{Tree}(H_{\mathcal{D}})\}$$

(where $b \in \{0,1\}$ is a bit). Then, we define another subtree $T_{\mathcal{D}} := \texttt{Tree}(H_{\mathcal{D}}) \cup \texttt{Frontier}(H_{\mathcal{D}})$ where each node in $T_{\mathcal{D}}$ either is a leaf node of $T_{\mathcal{D}}$ or has two children in $T_{\mathcal{D}}$, such that $\texttt{Frontier}(H_{\mathcal{D}}) \cup H_{\mathcal{D}}$ is exactly the set of leaf nodes of $T_{\mathcal{D}}$.

Then, the model trainer constructs the Merkle tree based on $T_{\mathcal{D}}$ by assigning values to its nodes. We denote the value assignment as $\texttt{Val}_{T_{\mathcal{D}}}(\cdot)$, such that the leaf nodes of $T_{\mathcal{D}}$ are first assigned as follows:

- Each $h_d = \texttt{hash}(\texttt{com}_d) \in H_{\mathcal{D}}$ is assigned value $\texttt{Val}_{T_{\mathcal{D}}}(h_d) \leftarrow \texttt{com}_d$;
- Each $v \in \texttt{Frontier}(\mathcal{D})$ is assigned value $\texttt{Val}_{T_{\mathcal{D}}}(v) \leftarrow \epsilon$;

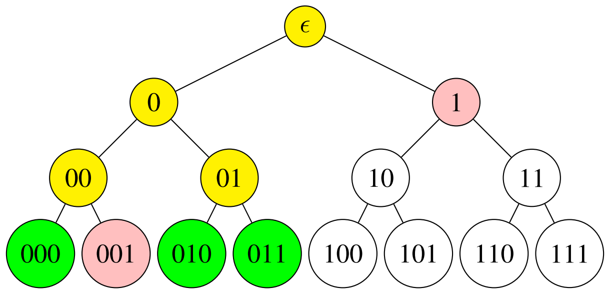


Figure 5: Example of proof of (non-)membership. The green values are the hashes of the data points, the red values are the frontier, and the yellow values are the remaining parts of the Merkle tree.

Then, by executing Algorithm 2 as $h_{\mathcal{D}} \leftarrow \texttt{MerkleTree}(H_{\mathcal{D}} \cup \texttt{Frontier}(H_{\mathcal{D}}), \texttt{Val}_{T_{\mathcal{D}}})$, the value of each remaining node v is assigned such that $\texttt{Val}_{T_{\mathcal{D}}}(v) = \texttt{hash}(\texttt{Val}_{T_{\mathcal{D}}}(v0), \texttt{Val}_{T_{\mathcal{D}}}(v1))$. The model owner only publishes the root value of $T_{\mathcal{D}}$, denoted by $h_{\mathcal{D}}$, which equals $\texttt{Val}_{T_{\mathcal{D}}}(\texttt{root})$.

Algorithm 2 (Re-)construction of Merkle tree: $\texttt{MerkleTree}(S, \texttt{Val})$

```
Require: A set of leaf nodes $S$ identified by their binary representation, and the value of each node $\texttt{Val} : S \to \{0,1\}^*$
1: procedure MERKLETREE($S$, Val)
2:     $\texttt{depth} \leftarrow \max\{\texttt{length}(s) : s \in S\}$
3:     for $k \leftarrow \texttt{depth}, \texttt{depth} - 1, \ldots 1$ do
4:         $S_k \leftarrow \{s \in S : \texttt{length}(s) = k\}$
5:         $S'_{k-1} \leftarrow \{v : v0, v1 \in S_k\}$
6:         if $2\left|S'_{k-1}\right| \neq |S_k|$ then
7:             Abort since exists $v \in S_k$ whose sibling is not in $S_k$
8:         end if
9:         $S \leftarrow S \sqcup S'_{k-1}$    ▷ Must be disjoint union, abort otherwise
10:        for $v \in S'_{k-1}$ do    ▷ Recursively compute the hashes
11:            $\texttt{Val}(v) \leftarrow \texttt{hash}(\texttt{Val}(v0), \texttt{Val}(v1))$
12:        end for
13:    end for
14:    $\{\texttt{root}\} \leftarrow S'_0$    ▷ By execution, $|S'_0| = 1$
15:    return $\texttt{Val}(\texttt{root})$
16: end procedure
```

An example of the Merkle tree with $k = 3$ is shown in Figure 5. The hashes of the data points are $H_{\mathcal{D}} = \{000, 010, 011\}$ (marked in green), which store the values of the commitments of the corresponding data points. The frontier $\texttt{Frontier}(H_{\mathcal{D}})$ is marked in red, with each node storing the value of ϵ. The yellow-coloured nodes are the rest of the Merkle tree $T_{\mathcal{D}}$, with each node storing the hash value of its two children. The uncolored vertices are not part of $T_{\mathcal{D}}$.

When queried about any data point d, the model trainer can compute a proof of its (non-)membership in $\mathcal{D}$ by proving the (non-)membership of $h_d := \texttt{hash}(\texttt{Commit}(d))$ (note that $\texttt{Commit}(d)$ is assumed to be deterministic [49]) in $H_{\mathcal{D}}$ using the Merkle tree $T_{\mathcal{D}}$. To prove a data point $d \in \mathcal{D}$, the model trainer needs to show that h_d is a leaf of $T_{\mathcal{D}}$, by reconstructing the path from h_d to the root in $T_{\mathcal{D}}$ and show that the value of the reconstructed root matches $h_{\mathcal{D}}$. On the other hand, to prove a data point $d \notin \mathcal{D}$ (with is equivalent to $h_d \notin H_{\mathcal{D}}$ with overwhelming probability), the model trainer needs to show that there exists an element of the $\texttt{Frontier}(H_{\mathcal{D}})$, s, that is a prefix of h_d or h_d itself (denoted by $s \preceq h_d$), and reconstruct the path from s to the root with the correct reconstructed root value. In general, to accommodate queries for multiple data points E, we define $H_E := \{\texttt{hash}(\texttt{Commit}(d)) : d \in E\}$, which can be agreed upon by both parties. The model trainer can then prove the training set (non-)membership of each data point in E using Protocol 3:

Protocol 3 Trainers generates the proof of (non-)membership

```
Require: $H_E$: the hashed commitments of the queried data points
1: $H_E^{\text{inc}}, H_E^{\text{exc}} \leftarrow H_{\mathcal{D}} \cap H_E, H_E \backslash H_{\mathcal{D}}$    ▷ Hashes of the data points included and excluded from the training set
2: $F^{\text{exc}} \leftarrow \{s \in \texttt{Frontier}(H_{\mathcal{D}}) : \exists h \in H_E^{\text{exc}}, s \preceq h\}$
3: $F_E \leftarrow \texttt{Frontier}(H_E^{\text{inc}} \cup F^{\text{exc}})$
4: $\pi^{\text{mem}} \leftarrow (F^{\text{exc}}, \texttt{Frontier}(F_E))$
5: return $H_E^{\text{inc}}, H_E^{\text{exc}}, \pi^{\text{mem}}$
```

Note that the output of Protocol 3 includes the values of the nodes in $\texttt{Tree}(H_E^{\text{inc}})$ and F^{exc}, as well as frontier of the union of these two sets, $\texttt{Frontier}(H_E^{\text{inc}} \cup F^{\text{exc}})$. This allows the data copyright owner to reconstruct the Merkle tree and compute its root. Specifically, let $h_E \leftarrow \texttt{MerkleTree}(H_E^{\text{inc}} \cup F^{\text{exc}} \cup \texttt{Frontier}(H_E^{\text{inc}} \cup F^{\text{exc}}), \texttt{Val}_E)$, where $\texttt{Val}_E$ is the restriction of $\texttt{Val}_{T_{\mathcal{D}}}$ onto $H_E^{\text{inc}} \cup F^{\text{exc}} \cup \texttt{Frontier}(H_E^{\text{inc}} \cup F^{\text{exc}})$ since the model trainer only needs to release the values on these nodes. Then, the data copyright owner can check the validity of the proof by verifying that the reconstructed root value h_E matches that received from the model trainer $h_{\mathcal{D}}$ as Protocol 4.

For example, in Figure 5, let $H_E = \{000, 001, 011, 101\}$ be the queried set. The model trainer computes $H_E^{\text{inc}} = \{000, 011\}$ and $H_E^{\text{exc}} = \{001, 101\}$, and therefore $F^{\text{exc}} = \{001, 1\}$. Therefore, the corresponding $\texttt{Frontier}(H_E^{\text{exc}} \cup F^{\text{exc}}) = \{010\}$. Then, based on the released values of these nodes, the value of the root can be recovered by reconstructing the Merkle tree and then checked with the published value $h_{\mathcal{D}}$ to verify the proof.

Theorem B.1 (Data Membership)**.** *Consider the scenario when a data copyright owner queries the trainer on a batch of his/her data points E, in the following steps:*

- *The data copyright owner sends the hashes of the queried data points H_E to the trainer;*

Protocol 4 Data copyright owner verifies the proof of (non-)membership

```
Require: The queried data points $E$ (identified by their hashes $H_E$); root value of the training set Merkle tree $h_{\mathcal{D}}$; the hashed commitments of the queried data points $H_E$; output of Protocol 3, $H_E^{\texttt{inc}}, H_E^{\texttt{exc}}, (F^{\texttt{exc}}, \texttt{Frontier}(F_E)) = \pi^{\texttt{mem}}$ sent from the trainer; the released node values $\texttt{Val}_E$.
1: Check $H_E \overset{?}{=} H_E^{\texttt{inc}} \sqcup H_E^{\texttt{exc}}$    ▷ Disjoint union, reject otherwise
2: Check $\texttt{Val}(v) \overset{?}{=} \epsilon$ for each $s \in F^{\texttt{exc}}$
3: Check $\overset{?}{\exists}\, s \in F^{\texttt{exc}}$ such that $s \preceq h$ for each $h \in H_E^{\texttt{exc}}$
4: Check $h_{\mathcal{D}} \overset{?}{=} \texttt{MerkleTree}(H_E^{\texttt{inc}} \cup F^{\texttt{exc}} \cup \texttt{Frontier}(H_E^{\texttt{inc}} \cup F^{\texttt{exc}}), \texttt{Val}_E)$
5: return accept if all checks pass, otherwise reject.
```

□

- *The trainer sends H_E^{inc} and H_E^{exc} (the hashes of the queried data points that the trainer claims to be included in and excluded from the training set $\mathcal{D}$, respectively) to the data copyright owner, along with the proof π^{mem}.*
- *The data copyright owner checks the validity of the proof using Protocol 4.*

With probability $1 - \texttt{negl}(\lambda)$, the followings hold:

- *If the trainer fully adheres to Protocol 3 to compute H_E^{inc}, H_E^{exc} and π^{mem}, the data copyright owner `accepts` the membership result returned from the trainer.*
- *If the trainer lies about the training set membership of any queried data point, i.e., $H_E^{inc} \neq H_{\mathcal{D}} \cap H_E$ or $H_E^{exc} \neq H_E \backslash H_{\mathcal{D}}$, the data copyright owner `rejects` the membership result returned from the trainer.*

Proof sketch of Theorem B.1. If Protocol 3 has been followed by the trainer, then the execution can be represented as $H_E = H_E^{\texttt{inc}} \sqcup H_E^{\texttt{exc}}$. Additionally, $F^{\texttt{exc}}$ is a subset of $\texttt{Frontier}(H_{\mathcal{D}})$, such that $\texttt{Val}(v) = \epsilon$ for each $v \in F^{\texttt{exc}}$, and for each $h \in H_E^{\texttt{exc}}$, there exists $s \in F^{\texttt{exc}}$ such that $s \preceq h$. Furthermore, since $\texttt{Val}_E$ and $\texttt{Val}_{T_{\mathcal{D}}}$ coincide for each sent node, the value of the root node recovered from the proof matches that of $h_{\mathcal{D}}$. As a result, all checks have passed, and the data copyright owner can accept the proof through Protocol 4.

Consider the case when the trainer outputs $H_E^{\texttt{inc}} \neq H_{\mathcal{D}} \cap H_E$ or $H_E^{\texttt{exc}} \neq H_{\mathcal{D}} \backslash H_E$. To pass Protocol 4, it must hold that $H_E = H_E^{\texttt{inc}} \sqcup H_E^{\texttt{exc}}$. Therefore, there is either $h \notin H_{\mathcal{D}}$ such that $h \in H_E^{\texttt{inc}}$, or $h' \in H_{\mathcal{D}}$ such that $h \in H_E^{\texttt{exc}}$. However, in either case, the trainer needs to compute a valid path in the Merkle tree from h (or one of the predecessors of h') to the root and match the value of $h_{\mathcal{D}}$ at the root. Since the hash function used is non-invertible and collision-resistant, the polynomial-time trainer can only succeed with negligible probability. Therefore, the data copyright owner rejects the proof with probability at least $1 - \texttt{negl}(\lambda)$.