

zkLLM: Zero Knowledge Proofs for Large Language Models

Haochen Sun
haochen.sun@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Jason Li
j2643li@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Hongyang Zhang
hongyang.zhang@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

ABSTRACT

The recent surge in artificial intelligence (AI), characterized by the prominence of large language models (LLMs), has ushered in fundamental transformations across the globe. However, alongside these advancements, concerns surrounding the legitimacy of LLMs have grown, posing legal challenges to their extensive applications. Compounding these concerns, the parameters of LLMs are often treated as intellectual property, restricting direct investigations.

In this study, we address a fundamental challenge within the realm of AI legislation: the need to establish the authenticity of outputs generated by LLMs. To tackle this issue, we present **zkLLM**, which stands as the inaugural specialized zero-knowledge proof tailored for LLMs to the best of our knowledge. Addressing the persistent challenge of non-arithmetic operations in deep learning, we introduce **tlookup**, a parallelized lookup argument designed for non-arithmetic tensor operations in deep learning, offering a solution with no asymptotic overhead. Furthermore, leveraging the foundation of **tlookup**, we introduce **zkAttn**, a specialized zero-knowledge proof crafted for the attention mechanism, carefully balancing considerations of running time, memory usage, and accuracy.

Empowered by our fully parallelized CUDA implementation, **zkLLM** emerges as a significant stride towards achieving efficient zero-knowledge verifiable computations over LLMs. Remarkably, for LLMs boasting 13 billion parameters, our approach enables the generation of a correctness proof for the entire inference process in under 15 minutes. The resulting proof, compactly sized at less than 200 kB, is designed to uphold the privacy of the model parameters, ensuring no inadvertent information leakage.

1 INTRODUCTION

The recent surge in artificial intelligence (AI), particularly with the advent of Large Language Models (LLMs) [1, 8, 12, 41, 51, 52], has profoundly transformed the world. However, these technological advances have also raised concerns about the legitimacy of these groundbreaking models, challenging the legal underpinnings of their extensive applications. For instance, in December 2023, the New York Times filed a lawsuit against OpenAI and Microsoft, accusing them of using copyrighted material from the newspaper to train their chatbots. In October 2023, President Biden issued an executive order to address both the "myriad benefits" and "substantial risks" posed by AI. As laws and regulations around LLMs evolve and tighten, developing practical tools to verify the legitimacy of these models has become crucial.

Consider the auditing process of a newly-released LLM, which is hosted on a cloud service (e.g., Microsoft Azure) with API access. Law enforcement queries the model using designated prompts

to test if the LLM generates illegal output (e.g., untrue, violence-prompting, or racist). In the stringent legal context, the authenticity of the output must be established to exclude the possibility of cheating by manipulating the generated texts. On the other hand, although the architectures are typically described in technical reports, the trained parameters are concealed as the AI developers' intellectual properties, making direct examination of the model parameters impossible. This dilemma calls for the application of zero-knowledge proofs (ZKPs), which allow for verifiable computations over the neural networks while disclosing no information about the neural network parameters [18, 33, 35, 38, 55–57].

However, adapting existing ZKP techniques to modern LLMs, characterized by their immense scale, presents significant challenges. These models require substantial computational resources, which general-purpose ZKP frameworks [3–5, 7, 10, 24, 28, 36, 42], often unaware of LLM structure and limited in parallel computation support, struggle to provide.

While early research has explored specialized cryptographic protocols for specific neural network architectures like convolutional neural networks (CNNs) [35, 38, 57], LLMs' complex internal structures necessitate further innovation in ZKP protocol design. This innovation is vital to avoid the excessive overhead typical of general-purpose ZKPs. LLMs involve many non-arithmetic operations, such as GELU [31] and SwiGLU [47] activation functions, which only partially align with current ZKP methods. Lookup arguments, trading memory consumption for faster runtimes, have been introduced [33] to handle these nonlinearities, but their straightforward application raises questions about manageable memory overhead.

Moreover, the attention mechanism in LLMs [53], which is inherently multivariate and often employs the Softmax function, requires a tailored ZKP protocol design for effective management of proof overhead. Tackling this mechanism within ZKPs is challenging, particularly as its components are not typically found in previously explored neural network architectures, such as MLPs and CNNs. In these traditional models, Softmax functions are usually placed after the output layer and are therefore not considered in prior works on zero-knowledge verifiable deep learning. This setup is in stark contrast with LLMs, where Softmax functions are used extensively across multiple layers. This prevalent use in LLMs necessitates a more refined approach in ZKP design to ensure both precise and efficient zero-knowledge verification, especially given the unique challenges presented by the attention mechanism.

In response to these challenges, we present **zkLLM**, the inaugural ZKP scheme specifically designed for LLMs. **zkLLM** empowers LLM owners to validate the integrity of inference outcomes to stakeholders, such as law enforcement agencies, thereby streamlining investigations involving LLMs while safeguarding intellectual property. Our key contributions are:

- We propose `tlookup`, a unique ZKP protocol for universal non-arithmetic operations in deep learning, to tackle the persistent challenge of verifying such operations (e.g., activation functions). `tlookup` adeptly handles overhead in two ways: analytically, it adds no asymptotic overhead in memory complexity or running time; practically, its design promotes a high level of parallelization, fully leveraging parallel computing resources (like GPUs) commonly used in LLM computing environments.
- We introduce `zkAttn`, a ZKP specifically crafted for attention mechanisms in LLMs. Building upon `tlookup` and enhancing its capabilities, `zkAttn` mitigates the accuracy degradation and high overheads linked with bit-decompositions and polynomial approximations. It also removes the necessity to list all multivariate input-output pairs, a prerequisite in lookup-based methods, by harnessing the mathematical properties of the attention mechanism. This strategy strikes a balance between running time, memory usage, and accuracy, while maintaining security and privacy standards.
- Our efficient CUDA implementation, in conjunction with the aforementioned technical advancements, positions `zkLLM` as the trailblazing ZKP for LLMs of sizes up to 13 billion parameters. `zkLLM` achieves reasonable proving times of 1-15 minutes and produces compact proofs smaller than 200kB. These proofs can be verified within 1-3 seconds by the verifier and guarantee no exposure of model parameters.

2 TECHNICAL OVERVIEW

Compared with general-purpose counterparts, an efficient zero-knowledge proof system specialized for deep learning hinges critically upon two key requirements:

- The capability for extensive parallelization (for example, using CUDA), which allows for the handling of proofs for the entire computational process in a reasonable timeframe.
- The adept handling of non-arithmetic operations, encompassing activation functions among others.

Although sumcheck-based protocols are known to be compatible with tensor structures common in deep learning computations [26, 38], traditionally, they have depended on bit-decomposition methods for non-arithmetic operations. This dependence leads to an increase in prover overhead and restricts the variety of non-arithmetic operations that can be supported. In response, we have developed a novel sumcheck-based protocol for lookup arguments over tensors.

Our design capitalizes on the following fact: for $S \in \mathbb{F}^D$ and $T \in \mathbb{F}^N$, the set inclusion $S \subseteq T$ holds if and only if there is an \mathbf{m} such that $\sum_{i \in [D]} (X + S_i)^{-1} \equiv \sum_{i \in [N]} \mathbf{m}_i (X + T_i)^{-1}$ as rational functions over $X \in \mathbb{F}$ [30]. This equivalence can be verified by evaluating both expressions at a single point $X \leftarrow \beta$, randomly selected by the verifier. Furthermore, \mathbf{m} can be computed in $O(D)$ time using straightforward counting. Hence, by calculating the elementwise multiplicative inverses $\mathbf{A} \leftarrow (\beta + S)^{-1}$ and $\mathbf{B} \leftarrow (\beta + T)^{-1}$, we can parallelize the sumcheck protocol for the identity

$$(\mathbf{A}.\text{sum}() = \mathbf{m}^\top \mathbf{B}) \wedge (\mathbf{A} \odot (\beta + S) = \mathbf{1}) \wedge (\mathbf{B} \odot (\beta + T) = \mathbf{1}) \quad (1)$$

This approach stands in contrast to the sequential lookup arguments [16, 22, 23, 44, 62, 63] that are based on univariate polynomials.

For the attention mechanism widely applied in modern LLMs, represented by the equation

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \mathbf{V}, \quad (2)$$

the direct application of lookup arguments, such as `tlookup`, presents the impractical challenge of compiling all input-output pairs into a lookup table due to the multivariate nature of the attention mechanism. To achieve zero-knowledge verifiability with limited overhead for the attention mechanism, we introduce `zkAttn`, as depicted in Figure 1:

- (1) Implement the matrix multiplication between the query \mathbf{Q} and keys \mathbf{K}^\top , resulting in $\mathbf{Z} \leftarrow \mathbf{Q}\mathbf{K}^\top$. This process is verifiable through the dedicated sumcheck protocol designed specifically for matrix multiplications.
- (2) Exploit the shift-invariance property of Softmax to adjust each row of \mathbf{Z} by a constant, represented as a vector $\hat{\mathbf{z}}$, so that $\exp(\mathbf{Z} - \hat{\mathbf{z}}\mathbf{1}^\top)$ sums to 1 row-wise. This transformation renders the Softmax output equivalent to applying $\exp(\cdot)$ element-wise to $\mathbf{Z}' := \mathbf{Z} - \hat{\mathbf{z}}\mathbf{1}^\top$. However, computing $\hat{\mathbf{z}}$ from \mathbf{Z} is highly intricate and not directly verifiable.
- (3) Transform \mathbf{Z}' into negative K -digit base- b numbers, with each $\mathbf{Z}' = -\sum_{k=0}^{K-1} b^k \mathbf{Z}^{(k)}$. By utilizing the homomorphism of $\exp(\cdot)$, the Softmax output \mathbf{Y} is then expressed as

$$\mathbf{Y} = \exp\left(-\sum_{k=0}^{K-1} b^k \mathbf{Z}^{(k)}\right) = \prod_{k=0}^{K-1} \exp\left(-b^k \mathbf{Z}^{(k)}\right), \quad (3)$$

with a `tlookup` installed for each term of the K in the product to handle the non-arithmetic operation.

- (4) Rather than verifying the correctness of $\hat{\mathbf{z}}$ directly, which is highly non-arithmetic, an additional check is introduced to ensure the rowwise sums of \mathbf{Y} equal 1.
- (5) Implement another verifiable matrix multiplication between the Softmax output \mathbf{Y} and the values \mathbf{V} .

Note that the above overview omits details about the handling of scaling factors and quantization errors for clarity. The design of `zkAttn` manages the overhead of verifiable computation for the highly non-arithmetic operations within the attention mechanism while preserving computational accuracy.

3 PRELIMINARIES

3.1 Notations

We represent vectors and tensors in bold font, such as \mathbf{v} for vectors and \mathbf{S} for tensors. Consistent with the cryptographic frameworks we utilize, we apply 0-based indexing to all mathematical structures. For simple operations and indexing over tensors, we adhere to the PyTorch conventions, using notations like $\mathbf{v}_{[i]}$ or the more concise \mathbf{v}_i for elements, $\mathbf{S}_{[i,j_0:j_1,:]}$ for slicing, and $\mathbf{S}.\text{sum}(\text{axis}=0)$ for aggregation along a specified dimension. Moreover, we denote the set of non-negative integers less than a positive integer N using the compact form $[N]$, which signifies $\{0, 1, \dots, N-1\}$.

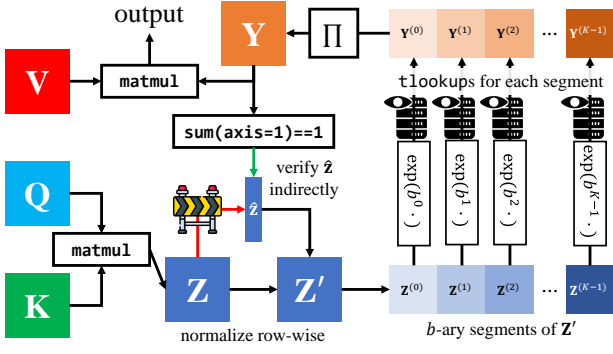


Figure 1: Overview of zkAttn for (2).

Q: As the developer of an LLM, law enforcement seeks to verify the model's authenticity by having me execute the inference process using their designated prompts. I aim to establish the authenticity of my output without revealing any details about the model parameters. Can ZKP be utilized in the inference process?

A: Yes, ZKP can be utilized in the inference process of an LLM to establish the authenticity of the output without revealing any details about the model parameters. This allows one party to prove to another that a certain statement is true, without conveying any additional information. In this case, it can prove that the model's output came from a specific model.

Q: What are some benefits of applying ZKPs in this scenario?

A: Utilizing ZKPs in the inference process of an LLM can provide authenticity and privacy. It can confirm the output's origin from a specific model without revealing any model details, protecting any sensitive or proprietary information. The generated proofs are verifiable, allowing anyone to confirm the output's authenticity. Some ZKP protocols are also scalable, accommodating large models and complex computations, which is beneficial for LLMs.

Figure 2: An example dialogue with GPT-3.5 regarding zk-LLM's motivation

3.2 Large language models, transformers and the attention mechanism

Large Language Models (LLMs), exemplified by the GPT series [8, 41], PaLM [12], and LLaMa [51], are renowned for their exceptional performance in general-purpose language understanding and generation tasks. These models are built upon the transformer architecture, which is fundamentally based on the attention mechanism [53].

As depicted in Figure 3, LLMs typically consist of multiple layers that transform the embeddings of an input sequence of tokens using multi-head attention. In each attention head i , parameterized by linear weights W_i^Q , W_i^K , and W_i^V , the queries $Q_i \leftarrow XW_i^Q$, keys $K_i \leftarrow XW_i^K$, and values $V_i \leftarrow XW_i^V$ are computed. These components are then processed by the Attention function, concatenated,

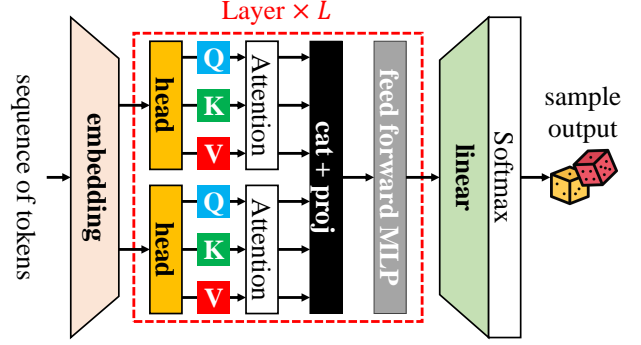


Figure 3: Typical structure of LLMs

and projected using another linear weight W^O :

$$O \leftarrow \text{Concat}_i (\text{Attention}(Q_i, K_i, V_i)) W^O, \quad (4)$$

The output O is subsequently processed by a feed-forward multi-layer perceptron (MLP). The activations from the final layer are then transformed into a probability distribution over the output tokens, from which the output sequence is autoregressively sampled. The Attention function, as defined in (2), effectively mimics cognitive attention and has significantly contributed to the success of LLMs. The adaptation of the attention mechanism for zero-knowledge verifiability is a primary focus of this study.

3.3 Sumcheck protocol, multilinear extensions and tensor operations

The correctness of arithmetic tensor operations (e.g., matrix multiplication) is verified using the *sumcheck protocol* [13] over the *multilinear extensions* [39] of the tensors involved.

Consider a tensor $S \in \mathbb{F}^{D_0 \times D_1 \times \dots \times D_{K-1}}$ discretized into a finite field \mathbb{F} via scaling and rounding. Without loss of generality, assume that D_k s are all powers of 2 for $0 \leq k \leq K-1$, or zero-padding may be applied. Thus, writing indices in binary format, S can also be considered as a function $S(\cdot) : \{0, 1\}^{\sum_{k=0}^{K-1} \log_2 D_k} \rightarrow \mathbb{F}$. Here, $S(i_0, i_1, \dots, i_{K-1}) = S_{[i_0, i_1, \dots, i_{K-1}]}$ where i_k is the binary representation of any $0 \leq i_k \leq D_{K-1} - 1$. A multivariate polynomial $\tilde{S}(\cdot) : \mathbb{F}^{\sum_{k=0}^{K-1} \log_2 D_k} \rightarrow \mathbb{F}$ is a *multilinear extension* of $S(\cdot)$ such that $\tilde{S}(\cdot) \equiv S(\cdot)$ on $\{0, 1\}^{\sum_{k=0}^{K-1} \log_2 D_k}$, practically implemented as

$$\tilde{S}(u_0, u_1, \dots, u_{K-1}) = \sum_{\substack{i_k \in \{0, 1\}^{\log_2 D_k} \\ 0 \leq k \leq K-1}} \tilde{e} \left(\bigoplus_{k=0}^{K-1} u_k, \bigoplus_{k=0}^{K-1} i_k \right) S(i_0, i_1, \dots, i_{K-1}), \quad (5)$$

where $\tilde{e}(u, v) := \sum_{i=0}^{d-1} u_i v_i + (1 - u_i)(1 - v_i)$ for any d -dimensional $u, v \in \mathbb{F}^d$, which reduces to the equality indicator $\mathbb{1}_{\{u=v\}}$ when restricted to $u, v \in \{0, 1\}^d$.

The correctness of a tensor operation can be expressed as equalities over the tensors. For instance, for matrix multiplication $C \leftarrow AB$, where $C \in \mathbb{F}^{D_0 \times D_2}$, $A \in \mathbb{F}^{D_0 \times D_1}$, and $B \in \mathbb{F}^{D_1 \times D_2}$, the correctness is characterized by $C_{[i,j]} = \sum_{k=0}^{D_1-1} A_{[i,k]} B_{[k,j]}$ for each i, j , or equivalently,

$$\sum_{k \in \{0,1\}^{\log_2 D_1}} \left(D_1^{-1} \tilde{C}(i, j) - \tilde{A}(i, k) \tilde{B}(k, j) \right) = 0. \quad (6)$$

By applying the Schwartz-Zippel Lemma [46, 69], with high probability, (6) holds for all i, j if and only if the random linear combination

$$\sum_{\substack{i \in \{0,1\}^{\log_2 D_0} \\ j \in \{0,1\}^{\log_2 D_2} \\ k \in \{0,1\}^{\log_2 D_1}}} \tilde{e}(u_0 \oplus u_2, i \oplus j) \underbrace{\left(D_1^{-1} \tilde{C}(i, j) - \tilde{A}(i, k) \tilde{B}(k, j) \right)}_{\text{varies for other tensor operations}} = 0, \quad (7)$$

where $\tilde{e}(\cdot)$. Thus, the prover and the verifier can execute the sum-check protocol [13], which proves the statements in the form of

$$\sum_{i \in \{0,1\}^d} f(i) = 0 \quad (8)$$

for any d -variate polynomial ($d = \log_2 D_0 + \log_2 D_1 + \log_2 D_2$ in the case of (7)). The prover time, proof size, and verifier time are $O(2^d)$, $O(d)$, and $O(d)$, respectively. At the end of the protocol, a claim about the value of $f(v)$ is made by the prover (where $v \sim \mathbb{F}^d$ due to the randomness over the protocol execution), which is further reduced to the claimed evaluations of the multilinear extensions (i.e., $\tilde{C}(v_0, v_2)$, $\tilde{A}(v_0, v_1)$, $\tilde{B}(v_1, v_2)$ in (7) with the indices decomposed as $v = v_0 \oplus v_1 \oplus v_2$ with the corresponding dimensionalities.) These claims are further verified via the proof of evaluations on the commitments of the tensors introduced in Section 3.4.

Optimized adaptations of the sumcheck protocol are designed to align with standard operations in deep learning, such as matrix multiplication [26, 49] (see Section 6.1.1) and convolution [38]. The preservation of the tensor structure enables the parallelization of the proof. Moreover, zero-knowledge adaptations of the sumcheck protocol [11, 60, 61] have been developed to prevent any disclosure of information related to the tensors, while adding a negligible additional computational burden.

3.4 Polynomial Commitment

The binding and hiding requirements for the tensors, in the form of multilinear extensions, which are considered the intellectual properties of the prover, are achieved using polynomial commitment schemes. Specifically, the following establishes the correctness of $\tilde{S}(v)$ in zero-knowledge for any tensor S (assumed to be one-dimensional for simplicity) and any v with matching dimensionality:

- $pp \leftarrow \text{KeyGen}(1^\lambda)$ generates the public parameters used in the scheme, where λ is the security parameter of the scheme.
- $\llbracket S \rrbracket \leftarrow \text{Commit}(S, r; pp)$ generates a binding and hiding commitment $\llbracket S \rrbracket$ of S , such that $\llbracket S \rrbracket$ leaks no information about S , and no polynomial-time adversary can compute $S' \neq S$ and r' such that $\llbracket S \rrbracket = \text{Commit}(S'; pp, r)$.
- $(y, \pi) \leftarrow \text{ProveEval}(S, \llbracket S \rrbracket, v, r; pp)$ allows the prover to compute $y \leftarrow \tilde{S}(v)$ for any v with matching dimensionality, and creates a *proof of evaluation* that $y = \tilde{S}(v)$ with respect to the committed S .
- $\text{True/False} \leftarrow \text{Verify}(y, \pi, \llbracket S \rrbracket, v; pp)$ allows the verifier to verify the correctness of y , such that

- **(Completeness)** if $(y, \pi) = \text{ProveEval}(S, \llbracket S \rrbracket, v, r; pp)$, then the output is **True**.
- **(Soundness)** if $y \neq \tilde{S}(v)$, then the output is **False** with $1 - \text{negl}(\lambda)$ probability.
- **(Zero-knowledge)** the verifier learns no information beyond $y = \tilde{S}(v)$.

In the absence of ambiguity, we omit the randomness r and public parameters pp in the subsequent context.

In this study, Hyrax [54], a variant of the Pedersen commitment [43] that does not require a trusted setup, is used as an instantiation of the polynomial commitment scheme. It operates on a cyclic group \mathbb{G} (typically an elliptic curve), with the hardness assumption of the discrete log problem, and is isomorphic to the addition of \mathbb{F} . Hyrax is homomorphic, such that $\text{Commit}(S_1, r_1) + \text{Commit}(S_2, r_2) = \text{Commit}(S_1 + S_2, r_1 + r_2)$ for any two tensors S_1, S_2 and randomness r_1, r_2 . Hyrax achieves linear complexity in Commit and ProveEval with respect to the dimensionality D of the tensor S involved and can be further parallelized by concurrently handling operations on all dimensions. It also balances the commitment size, proof size, and verifier's proof evaluation time, all to sub-linear complexities of $O(\sqrt{D})$, $O(\log D)$, and $O(\sqrt{D})$ respectively. These improvements are adeptly adopted in zkLLM to minimize both computation and communication burdens.

3.5 Lookup arguments

The lookup argument is commonly used to address non-arithmetic operations within the domain of zero-knowledge proofs [50]. Inspired by recent advances [16, 22, 23, 44, 62, 63], the incorporation of lookup arguments into zero-knowledge verifiable deep learning inference [33] has been pursued. In such a setting, a lookup argument verifies that each element in a secret tensor S^D , known only to the prover, is contained within a predefined table $T \in \mathbb{F}^N$, mutually acknowledged by both parties. However, the requisite computations for lookup arguments are intrinsically sequential, which contradicts the parallelism preferred in deep learning environments. Furthermore, deploying lookup arguments entails a trade-off between sacrificing precision and incurring excessive memory consumption and trusted setup burdens due to the expansive size of the lookup tables necessary to cover all possible values (with N being significantly large).

In response to these challenges, our proposed lookup arguments for non-arithmetic tensor operations markedly enhance parallelization compared to the largely sequential approaches traditionally used in verifiable deep learning inference. Additionally, our novel proof protocol, tailored for the Softmax function within the attention mechanisms of Transformer models, is designed to optimize the balance between setup and proving times, memory consumptions, and precision.

3.6 Settings and security assumptions

We follow the widely recognized framework for zero-knowledge verifiable inferences as outlined in prior research on zero-knowledge machine learning [18, 33, 35, 38, 55–57]. In this framework, the **prover** (such as an AI company) owns an LLM with a publicly known structure (e.g., described in the technical report), while considering the model's weights as its intellectual property. The prover

provides API access to this model for a **verifier** (like an AI regulation enforcer), who submits a prompt and requests formal proof that the inference result returned by the API is accurate in relation to the prompt and the confidential model.

A semi-honest assumption is applied to the verifier: the verifier accurately reports the outcome of the proof verification (whether it is accepted or rejected) but endeavors to glean additional information about the LLM (like hidden parameters) beyond merely confirming the correctness of the inference result.

In this study, we assume the use of a commitment scheme that ensures λ -bit security. Correspondingly, the computations are carried out within a finite field \mathbb{F} , characterized by a prime order of at least $\Omega(2^{2\lambda})$. Furthermore, we postulate that every aspect of the transformer model and the data—including the number of layers, the dimensions of tensors, and the complexity of operations between them—is polynomially bounded by λ .

4 tlookup: VERIFIABLE NON-ARITHMETIC OPERATIONS FOR DEEP LEARNING

In this section, we introduce tlookup, our novel approach to addressing general non-arithmetic operations in deep learning. The tlookup design preserves the widely-used tensor-based structure, guaranteeing seamless compatibility with the established computational frameworks in deep learning. tlookup acts as a foundational component of zkAttn, our specialized ZKP protocol tailored for the attention mechanism, detailed in Section 5. Furthermore, tlookup is applicable to other non-arithmetic operations essential to the inference mechanisms within LLMs.

We first reduce the non-arithmetic tensor operations to lookup arguments over tensors. Specifically, for a tensor $S \in \mathbb{F}^D$, the prover aims to convince the verifier that each element of S exists within $T \in \mathbb{F}^N$, a table that both parties have full knowledge of. The essence of our approach hinges on the subsequent lemma:

LEMMA 4.1 ([30]). *Given tensors $S \in \mathbb{F}^D$ and $T \in \mathbb{F}^N$, $S \subset T$ as sets if and only if there exists $m \in \mathbb{F}^N$ such that the following identity of rational functions is satisfied:*

$$\sum_{i \in [D]} \frac{1}{X + S_i} = \sum_{i \in [N]} \frac{m_i}{X + T_i}. \quad (9)$$

When the condition $S \subset T$ holds, the prover constructs m as:

$$m_i \leftarrow \left| \{j : S_j = T_i\} \right|, \text{ for } 0 \leq i \leq N-1. \quad (10)$$

The verifier can then confirm the equality presented in (9) by randomly choosing $X \leftarrow \beta \sim \mathbb{F}$. By defining:

$$A := \left(\frac{1}{\beta + S_i} \right)_{i=0}^{D-1}, B := \left(\frac{1}{\beta + T_i} \right)_{i=0}^{N-1}, \quad (11)$$

the aforementioned equality at the random point β can be restated as:

$$\sum_{i \in [D]} A_i = \sum_{i \in [N]} m_i B_i. \quad (12)$$

Therefore, with the randomness $u \sim \mathbb{F}^{\log_2 D}$ and $\alpha \sim \mathbb{F}$, the sumcheck for the correctness of (11) and (12) can be formulated as

$$\begin{aligned} 0 = & \left(\sum_{i \in [D]} \tilde{A}(i) - \sum_{j \in [N]} \tilde{m}(j) \tilde{B}(j) \right) \\ & + \alpha \left(\sum_{i \in [D]} \tilde{e}(u, i) \tilde{A}(i) (\tilde{S}(i) + \beta) - 1 \right) \\ & + \alpha^2 \left(\sum_{j \in [N]} \tilde{e}(u_{\lceil \log_2 \frac{D}{N} \rceil}, j) \tilde{B}(j) (\tilde{T}(j) + \beta) - 1 \right), \end{aligned} \quad (13)$$

or equivalently,

$$\begin{aligned} \alpha + \alpha^2 = & \sum_{i \in [\frac{D}{N}]} \sum_{j \in [N]} \left(\tilde{A}(i \oplus j) \left(\alpha \tilde{e}(u, i \oplus j) (\tilde{S}(i \oplus j) + \beta) + 1 \right) \right. \\ & \left. + ND^{-1} \tilde{B}(j) \left(\alpha^2 \tilde{e}(u_{\lceil \log_2 \frac{D}{N} \rceil}, j) (\tilde{T}(j) + \beta) - \tilde{m}(j) \right) \right). \end{aligned} \quad (14)$$

A comprehensive description of the procedure to validate $S \subset T$ is found in Protocol 1. In particular, in Line 1, **TLOOKUP-SETUP**(T) generates a short witness $\llbracket T \rrbracket$ to a prescribed table T known to both parties; in Line 4, the prover constructs m based on a tensor S and table T and commit to S and m using **TLOOKUP-PREP**(S, T); finally, in Line 9, $\langle \mathcal{P}, \mathcal{V} \rangle$.**TLOOKUP-PROVE**($\llbracket S \rrbracket, \llbracket m \rrbracket, \llbracket T \rrbracket$) is the interactive process of the prover \mathcal{P} proving that a secret tensor S is elementwisely in T , which has been committed as $\llbracket T \rrbracket$.

Protocol 1 tlookup

Require: The prover \mathcal{P} knows $S \in \mathbb{F}^D$. N, D are both powers of 2 such that N divides D .

- 1: **procedure** **TLOOKUP-SETUP**($T \in \mathbb{F}^N$)
 - 2: **return** $\llbracket T \rrbracket \leftarrow \text{Commit}(T; 0)$ \triangleright No hiding required
 - 3: **end procedure**
 - 4: **procedure** \mathcal{P} .**TLOOKUP-PREP**($S \in \mathbb{F}^D, T \in \mathbb{F}^N$)
 - 5: Compute $m = m(S, T)$ as (10)
 - 6: $\mathcal{P} \rightarrow \mathcal{V} : \llbracket S \rrbracket \leftarrow \text{Commit}(S)$
 - 7: $\mathcal{P} \rightarrow \mathcal{V} : \llbracket m \rrbracket \leftarrow \text{Commit}(m)$
 - 8: **end procedure**
 - 9: **procedure** $\langle \mathcal{P}, \mathcal{V} \rangle$.**TLOOKUP-PROVE**($\llbracket S \rrbracket, \llbracket m \rrbracket, \llbracket T \rrbracket$)
 - 10: $\mathcal{V} \rightarrow \mathcal{P} : \beta \sim \mathbb{F}$
 - 11: \mathcal{P} computes A, B as (11)
 - 12: $\mathcal{P} \rightarrow \mathcal{V} : \llbracket A \rrbracket \leftarrow \text{Commit}(A), \llbracket B \rrbracket \leftarrow \text{Commit}(B)$
 - 13: \mathcal{P} and \mathcal{V} run the sumcheck on (14), followed by the proofs of evaluation on $\llbracket A \rrbracket, \llbracket B \rrbracket, \llbracket S \rrbracket, \llbracket m \rrbracket$ and $\llbracket T \rrbracket$.
 - 14: **end procedure**
-

Meanwhile, for elementwise non-arithmetic operations $f : X \rightarrow \mathcal{Y}$ over tensors where $X, \mathcal{Y} \subset \mathbb{F}$, two lookup tables can be constructed: $T_X := (x)_{x \in X}$ and $T_Y := (f(x))_{x \in X}$. To demonstrate that $Y = f(X)$ for some $X, Y \in \mathbb{F}^D$ (broadcasting f over all dimensions), one can apply the idea of random linear combination to reduce the check to one instance of Protocol 1 where $X + \alpha Y \subset T_X + \alpha T_Y$ for $\alpha \sim \mathbb{F}$ chosen by the verifier.

EXAMPLE 4.2 (ReLU WITH RESCALING). *We first consider the rectified linear unit (ReLU), which is a common activation function in contemporary deep learning models, including Transformers. ReLUs are generally applied subsequent to linear layers (for instance, fully connected layers) where products are involved. In the scenario of fully quantized computation, it becomes necessary for the ReLU to incorporate rescaling as well. This is denoted as follows:*

$$\mathbf{A} \leftarrow \text{ReLU}(\mathbf{Z}) = \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rfloor \odot \mathbb{1} \left\{ \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rfloor \geq 0 \right\}, \quad (15)$$

where γ represents the scaling factor used in the system (assumed to be even for simplicity). We assume that $-\frac{B}{2} \leq \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rfloor < \frac{B}{2}$ holds element-wise for a positive even integer B . Considering that \mathbf{Z} is decomposed as $\mathbf{Z}' := \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rfloor$ and $\mathbf{R} = \mathbf{Z} - \gamma \mathbf{Z}'$, we establish a pair of input-output lookup tables for \mathbf{Z}' . These are defined as $\mathbf{T}_\chi := \left[-\frac{B}{2}, \frac{B}{2} - 1\right]$ and $\mathbf{T}_\mathbf{y} := \mathbf{T}_\chi^+$ (i.e., taking the maximum with 0 element-wise), and an additional lookup table for \mathbf{R} as $\mathbf{T}_\mathbf{R} = \left[-\frac{\gamma}{2}, \frac{\gamma}{2} - 1\right]$. By requiring the prover to demonstrate to the verifier that $\mathbf{Z}' + \alpha \mathbf{A} \subset \mathbf{T}_\chi + \alpha \mathbf{T}_\mathbf{y}$ for a random α , and that $\mathbf{R} \subset \mathbf{T}_\mathbf{R}$, both using Protocol 1, in addition to proving the decomposition as $\mathbf{Z} = \gamma \mathbf{Z}' + \mathbf{R}$, we can sufficiently validate the correctness of inference through the ReLU function. Notably, unlike the brute-force method that employs a single lookup table and incurs an $O(B\gamma)$ overhead in both running time and memory usage, the use of two lookup tables effectively reduces this overhead to $O(B + \gamma)$. Similarly, if γ is too large to fit the table into memory, it can be further divided into a K -digit $\gamma^{\frac{1}{K}}$ -ary number. In this scenario, each of the K digits in the remainder corresponds to a separate tlookup, thus adequately covering all possible values of the remainder.

However, resolving the long-standing problem of excessive memory consumption for lookup tables in the realm of deep learning requires additional efforts. Specifically, in Section 5, tlookup is further refined into zkAttn to address its multivariate and highly non-arithmetic nature, optimizing the balance among running time, memory consumption, and approximation error.

5 zkAttn: DEDICATED ZKP FOR THE ATTENTION MECHANISM IN LLMs

The attention mechanism is a key component in modern transformers, including state-of-the-art LLMs. However, incorporating these mechanisms into ZKP backends has been challenging, primarily due to their distinctive mathematical properties. Specifically, the Softmax function, integral to the attention mechanism, involves non-arithmetic operations like exponentiation, and its multivariate aspect complicates the use of polynomial approximations for traditional ZKP backends. To address these challenges, we introduce zkAttn, a specialized ZKP tailored for the attention mechanism, designed to leverage its inherent mathematical characteristics effectively.

5.1 Formulation of zkAttn

The attention mechanism, in its discretized form, accepts as input a value matrix $\mathbf{V} \in \mathbb{F}^{n \times d}$, a key matrix $\mathbf{K} \in \mathbb{F}^{n \times d}$, and a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times d}$. It produces the output $\text{Softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d}}\right) \mathbf{V}$ subject to appropriate rescaling of the input and output due to quantization,

where Softmax is applied row-wise. In this discussion, we focus on $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Z}}{\sqrt{d}}\right)$, where the input matrix $\mathbf{Z} = \mathbf{QK}^\top$ is presumed to be scaled by the scaling factor γ from its actual values. It is assumed that d is a constant, known to both the prover and verifier, stemming from the presumption of a known model architecture. Equivalently, for each row $\mathbf{z} = (z_0, z_1, \dots, z_{n-1}) \in \mathbb{F}^n$, the objective is to devise an algorithm that computes

$$s(\mathbf{z}) := \left(\frac{\exp\left(\frac{z_i}{\gamma\sqrt{d}}\right)}{\sum_{j=0}^{n-1} \exp\left(\frac{z_j}{\gamma\sqrt{d}}\right)} \right)_{i=0}^{n-1} \quad (16)$$

in the real domain. Alternatively, it should compute its quantized counterpart $\theta s(\mathbf{z})$, ensuring limited numerical error and manageable proof generation overhead. Here, θ represents the scaling factor of all Softmax outputs in the system. Notably, this factor differs from the scaling factor γ used for other matrices, such as \mathbf{Q} , \mathbf{K} , and \mathbf{V} . For the sake of streamlined and verifiable rescaling in subsequent computations, we posit that θ is a multiple of γ .

To circumvent the verification of real division operations—which can lead to remainders after quantization—we observe the following. By utilizing the shift-invariance property of Softmax and defining

$$\hat{\mathbf{z}} := \gamma\sqrt{d} \ln \left(\sum_{j=0}^{n-1} \exp\left(\frac{z_j}{\gamma\sqrt{d}}\right) \right), \quad (17)$$

we derive that

$$s(\mathbf{z}) = s(\mathbf{z} - \hat{\mathbf{z}}) = \left(\exp\left(\frac{z_i - \hat{z}}{\gamma\sqrt{d}}\right) \right)_{i=0}^{n-1}. \quad (18)$$

It is imperative to understand that the computation of $\hat{\mathbf{z}}$ ($\lfloor \hat{\mathbf{z}} \rfloor$ to be specific, since $\hat{\mathbf{z}}$ is not an integer in general) in (17) is not directly verified due to its highly non-arithmetic nature. Instead, the prover ensures that the output of $s(\mathbf{z})$ adheres to proper normalization. In its quantized representation, the sum of its dimensions must equal θ . A certain degree of deviation is acceptable owing to quantization, and the precise bounds of this error will be elucidated in Section 7.1. Furthermore, beyond verifying normalization, there exists the challenge of crafting a scheme to compute the quantized exponentiation. This scheme should not only be accurate, approximating $\theta \exp\left(\frac{\cdot}{\gamma\sqrt{d}}\right)$ with minimal error, but also be amenable to efficient verification through the proof protocol that will be subsequently introduced.

Observe that, given the definition of $\hat{\mathbf{z}}$, $z_i - \hat{z} \leq 0$ for all i . On the other hand, as z_i s are all real numbers involved in the matrix multiplication scaled by γ , it is also reasonable to assume that each $z_i - \hat{z}$ is lower bounded by some integer $-B$ such that $\gamma \ll B \ll |\mathbb{F}|$, such that $(-B, 0]$ can accommodate the reasonable values of $z_i - \hat{z}$, but sufficiently small so as not to cause wraparounds in \mathbb{F} .

Without loss of generality, consider B as a product of K positive integers, denoted as $B = \prod_{k=0}^{K-1} b^{(k)}$. A bijection can then be established between $[B]$ and the product space $\prod_{k=0}^{K-1} [b^{(k)}]$. By

defining $B^{(k)}$ as

$$B^{(k)} := \begin{cases} 1, & \text{if } k = 0; \\ \prod_{j=0}^{k-1} b^{(j)}, & 1 \leq k \leq K-1, \end{cases}$$

our bijection $\mathbf{b} : \prod_{k=0}^{K-1} [b^{(k)}] \rightarrow [B]$ can be expressed as

$$\mathbf{b}(x^{(0)}, x^{(1)}, \dots, x^{(K-1)}) = \sum_{k=0}^{K-1} x^{(k)} B^{(k)}. \quad (19)$$

Consequently, for each $(x^{(0)}, x^{(1)}, \dots, x^{(K-1)}) = \mathbf{b}^{-1}(x)$, the following holds:

$$\exp\left(-\frac{x}{\gamma\sqrt{d}}\right) = \exp\left(-\frac{\sum_{k=0}^{K-1} x^{(k)} B^{(k)}}{\gamma\sqrt{d}}\right) = \prod_{k=0}^{K-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right). \quad (20)$$

Our objective is to compute the quantized representation of equation (20), taking into account the scaling factor γ . If we further decompose γ as $\theta = \prod_{k=0}^{K-1} \theta^{(k)}$ with non-negative values for $\theta^{(k)}$, equation (20) gives rise to

$$\theta \exp\left(-\frac{x}{\gamma\sqrt{d}}\right) = \prod_{k=0}^{K-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right). \quad (21)$$

Following the decomposition in Equation (21), we can construct K tlookup tables $\mathbf{T}^{(k)} = (\mathbf{T}_X^{(k)}, \mathbf{T}_Y^{(k)})$. Each table $\mathbf{T}^{(k)}$ comprises all potential input-output pairs corresponding to the k -th term in the product of (21):

$$\mathbf{T}_X^{(k)} := [b^{(k)}], \quad \mathbf{T}_Y^{(k)} := \left[\left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x\right) \right]_{x \in [b^{(k)}]} \right]. \quad (22)$$

Given any input $z \in (-B, 0]$, the prover first decomposes $\mathbf{b}(-z) = (x^{(0)}, x^{(1)}, \dots, x^{(K-1)})$ according to (19). Each component $x^{(k)}$ is subsequently mapped to $y^{(k)}$ based on $\mathbf{T}^{(k)}$, resulting in the computation $y \leftarrow \prod_{k=0}^{K-1} y^{(k)}$. Subsequently, the prover must demonstrate to the verifier that:

$$z + \sum_{k=0}^{K-1} x^{(k)} B^{(k)} = 0, \quad (23)$$

$$(x^{(k)}, y^{(k)}) \in \mathbf{T}^{(k)}, \quad \forall 0 \leq k \leq K-1, \quad (24)$$

$$\prod_{k=0}^{K-1} y^{(k)} = y. \quad (25)$$

Equation (23) confirms that the decomposition of $-z$ is valid. Equation (24) ensures the correctness of the exponent in each component concerning the pre-computed values in $\mathbf{T}^{(k)}$, and (25) asserts that the output y is accurately derived using the homomorphism of exponentiation from each factor $y^{(k)}$. Together, these three conditions guarantee the correct computation of the exponentiation operation, up to the rounding errors. Specifically:

LEMMA 5.1. *Conditions (23), (24), and (25) imply:*

$$y = \prod_{k=0}^{K-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right], \quad (26)$$

where $(x^{(k)})_{k=0}^{K-1} = \mathbf{b}(-z)$ is the valid decomposition according to (19).

The deviation between y and the exact scaled exponent

$$\theta \exp\left(\frac{z}{\gamma\sqrt{d}}\right) = \prod_{k=0}^{K-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right)$$

arises only from the rounding of each factor. An in-depth analysis of this will be covered in Section 7.1, where we will also provide guidance on selecting the parameters $\theta^{(k)}$ and $B^{(k)}$. In the subsequent sections of this text, we delve into the protocol design, facilitating the batched verification of (23), (24), and (25) for each dimension of large tensors used in transformer computations.

5.1.1 Optimization for the most and least significant segments. For the uppermost significant M segments, specifically $x^{(k)}$ with $K - M \leq k \leq K - 1$, consider a scenario where if any of these segments $x^{(k)}$ have non-zero values, the resulting exponent

$$\exp\left(-\frac{x}{\gamma\sqrt{d}}\right) \leq \exp\left(-\frac{B_{K-M}}{\gamma\sqrt{d}}\right) \quad (27)$$

approximates 0 closely enough that the output y from (25) can be designated as 0. This outcome can be achieved by configuring each table $\mathbf{T}^{(k)}$ that $K - M \leq k \leq K - 1$ in (24), to yield $y^{(k)} = 0$ for any $x^{(k)} > 0$. Moreover, based on our initial design, instances where $x^{(k)} = 0$, the value of $y^{(k)}$ defaults to $\left\lfloor \theta^{(k)} \right\rfloor$. Clearly, assigning any value other than 1 to these $\theta^{(k)}$ would only amplify the errors in $\mathbf{T}^{(k)}$ and other tables, especially under the constraint that $\prod_{k=0}^{K-1} \theta^{(k)} = \theta$ is constant. Therefore, for these most significant M segments, the lookup tables $\mathbf{T}^{(k)}$ can be reduced to the indicator function $y^{(k)} = \mathbb{1}\{x^{(k)} = 0\}$.

On the other hand, for the least significant L segments $x^{(k)}$, indexed by $0 \leq k \leq L - 1$, the expression $\exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right)$ tends to hover close to 1 for all possible values of $0 \leq x^{(k)} \leq b^{(k)} - 1$. Given this, approximating the exponentiation as a constant of 1 incurs a negligible error. Analogous to the strategy for the most significant segments, it is efficient to set the scaling factors $\theta^{(k)}$ to 1, sidestepping larger alternatives. This approach frees up room for allocating larger $\theta^{(k)}$ s for segments indexed by $L \leq k \leq K - M - 1$, thereby enhancing precision for these segments. As a result, the constraint (24) for validating input-output pairs simplifies to $x^{(k)} \in [b^{(k)}]$, given that $y^{(k)}$ consistently equates to 1.

As delineated in Section 7.1, employing these optimizations for both the most and least significant segments tightly upper bounds the error in zkAttn, aligning closely with the computational logic of the original neural networks.

5.2 zkAttn: the main protocol

In Protocol 2, we present the technical details of zkAttn built upon tlookup, our protocol for general non-arithmetic operations in deep learning. zkAttn is separated into three steps. First, zkATTN-SETUP(\cdot) (in Line 1) completes the setup for zkAttn by generating short witnesses of all tables involved. Then, in zkATTN-COMPUTE(\cdot), the prover is responsible for computing the output of the attention

mechanism and all necessary auxiliary tensors (e.g., the normalization constants, the inputs and outputs of each segment, and the recovered row-wise sum that should not excessively deviate from 1) for the zero-knowledge verifiability of the attention mechanism, and sends the commitments of these tensors to the verifier. Finally, the prover and verifier engage in the interactive protocol $\text{zkAttn-Prove}(\cdot)$ in Line 14, which involves proving the correctness of each segment and the normalization using the specialized lookup arguments of tlookup , as well as all arithmetic relations connecting the auxiliary tensors.

6 PUTTING EVERYTHING TOGETHER

6.1 Taxonomy of verifiable tensor operations

In this section, we present a taxonomy of the tensor operations involved in modern LLMs and the customized handling of these operations by zkLLM.

6.1.1 Matrix multiplications. Matrix multiplication plays a crucial role in modern transformers, including all linear layers and positional encoding (e.g., RoPE [48]) in LLMs.

Dedicated sumchecks [26] for matrix multiplications have achieved running times significantly lower than the computation itself. This development has been a key driver in the creation of specialized ZKPs for deep learning, a method also applied in this study to establish the correctness of all matrix products. To confirm $\mathbf{C} = \mathbf{AB}$, with $\mathbf{A} \in \mathbb{F}^{m \times n}$ and $\mathbf{B} \in \mathbb{F}^{n \times p}$, the prover and the verifier execute a sumcheck on

$$\tilde{\mathbf{C}}(\mathbf{u}, \mathbf{v}) = \sum_{i \in \{0,1\}^{\lceil \log_2 n \rceil}} \tilde{\mathbf{A}}(\mathbf{u}, i) \tilde{\mathbf{B}}(i, \mathbf{v}), \quad (28)$$

where $\mathbf{u} \in \mathbb{F}^{\lceil \log_2 m \rceil}$ and $\mathbf{v} \in \mathbb{F}^{\lceil \log_2 p \rceil}$ are selected at random by the verifier. This specialized proof for matrix multiplication ensures a prover time of $O(mn + np)$, faster than the computation process.

6.1.2 Activation functions. To enhance performance, modern LLMs have replaced traditional ReLU activation functions with smoother alternatives like SwiGLU [47] and GELU [31]. This transition necessitates extra efforts to make these new activation functions verifiable. The SwiGLU function, parameterized by β , is defined as

$$\text{SwiGLU}_\beta(z, z') := \text{Swish}_\beta(z) \cdot z', \quad (29)$$

with the Swish function being $\text{Swish}_\beta(z) := z \cdot \text{Sigmoid}(\beta z)$. Though the non-arithmetic Sigmoid function can be integrated into the proof system via tlookup , optimizing setup costs and memory usage is crucial. This is achieved by reducing the Sigmoid function to zkAttn , given $\text{Softmax}\left(\begin{smallmatrix} z \\ 0 \end{smallmatrix}\right) = \begin{pmatrix} \text{Sigmoid}(z) \\ \text{Sigmoid}(-z) \end{pmatrix}$, thus circumventing the bottleneck of iterating over extensive input-output pairs. The GELU function, defined as $\text{GELU}(z) := z\Phi(z) \approx z\text{Sigmoid}(1.702z)$, is handled similarly.

6.1.3 Normalization. LLMs employ LayerNorm [2] and its variants (e.g., RMSNorm [64]) for training stability. Unlike batch normalization, which can be merged into preceding layers for verifiable inference, LayerNorm involves non-linear transformations within

each sample \mathbf{x} , described as

$$y \leftarrow \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\text{Var}[\mathbf{x}] + \epsilon}}. \quad (30)$$

The compound non-arithmetic operations of square-root and inverse are managed through two sequential tlookup steps. These steps are responsible for the verifiable downscaling of the input and the quantized compound operation, respectively. Similar to the ReLU example presented in Example 4.2, the implementation of the first tlookup for downscaling is designed to reduce the overall sizes of the lookup tables, thereby keeping memory usage at a reasonable level.

6.2 Assembly of the proofs

Following the pioneering works [26, 38], zkLLM utilizes sumcheck-based proofs for computations across various components of LLMs. These proofs are assembled in reverse logical order of the arithmetic circuits, methodically reducing the claimed multilinear extension values of the output y to those associated with the prompt \mathbf{X} and the model parameters \mathbf{W} . These are then verified straightforwardly and through proof of evaluations on the commitment $\llbracket \mathbf{W} \rrbracket$. For high-level clarity, zkLLM can be distilled into three main components, with additional commitments caused by tlookups omitted for simplicity:

- $\llbracket \mathbf{W} \rrbracket \leftarrow \text{zkLLM-Commit}(\mathbf{W}, \text{pp}, r)$: The prover commits to the model parameters \mathbf{W} using the public parameters (generators of the commitment scheme) pp and randomness r .
- $(y, \pi) \leftarrow \text{zkLLM-Prove}(\mathbf{W}, \mathbf{X}, \text{pp}, r)$: The prover computes the output y with the prompt \mathbf{X} and model \mathbf{W} , and assembles the proof π using the sumcheck protocols and proofs of evaluations as previously described.
- $b \leftarrow \text{zkLLM-Verify}(\mathbf{X}, y, \llbracket \mathbf{W} \rrbracket)$: The verifier checks the correctness of each sumcheck and proof of evaluation within π , outputting $b = 1$ to accept the proof (if all components are correctly verified), and $b = 0$ otherwise.

7 ANALYSIS

7.1 Error analysis on zkAttn

In this section, we examine the error introduced by zkAttn , as discussed in Section 5. Our analysis serves three primary objectives: first, to establish an upper bound on the error, thereby demonstrating that our zkAttn design remains faithful to the original neural network; second, to fine-tune the parameters integral to zkAttn , aiming to minimize the error; and third, to determine an acceptable upper bound on the error upon the verification of proper normalization in zkAttn .

The overall error of zkAttn originates from two sources, the rounding of the shifting factor \hat{z} in (17) that makes the normalization no longer perfect, and the rounding of each segment encoded in the lookup tables $\mathbf{T}^{(k)}$ s that introduces errors to the exponentiations. The bound of the overall error is stated in Theorem 7.1 and analyzed in details in Appendix C.1.

Protocol 2 zkAttn (See Section 6.1.1 about the two matrix multiplications involved)

Require: Both the prover \mathcal{P} and the verifier \mathcal{V} know: the lower bound of input $-B$, and the factorization $B = \prod_{k=0}^{K-1} b^{(k)}$; the number of the most and least significant segments M and L in Section 5.1.1; the scaling factor of the input γ , the output θ , and each segment $\theta^{(k)}$ for each $L \leq k \leq K - M - 1$; the parameters m, n, d related to the dimensions of the input; the tolerable error E in row-wise normalization.

```

1: procedure zkATTN-SETUP( $B, K, M, L, \left(b^{(k)}\right)_{k=0}^{K-1}, \gamma, \theta, (\theta_K)_{k=L}^{K-M-1}, m, n, d, E$ )
2:   for  $0 \leq k \leq K - 1$  do
3:      $\llbracket \mathbf{T}_X^{(k)} \rrbracket \leftarrow \text{TLOOKUP-SETUP}(\mathbf{T}_X^{(k)})$  ▷  $\mathbf{T}_X^{(k)} = \left[b^{(k)}\right]$ , i.e., the input of the  $k$ -th segment
4:   end for
5:   for  $L \leq k \leq K - M - 1$  do
6:      $\llbracket \mathbf{T}_Y^{(k)} \rrbracket \leftarrow \text{TLOOKUP-SETUP}(\mathbf{T}_Y^{(k)})$  ▷  $\mathbf{T}_Y^{(k)}$  as defined in (22), i.e., the output of the  $k$ -th segment
7:   end for
8:   for  $K - M \leq k \leq K - 1$  do
9:      $\llbracket \mathbf{T}_Y^{(k)} \rrbracket \leftarrow \text{TLOOKUP-SETUP}(\mathbf{T}_Y^{(k)})$  ▷  $\mathbf{T}_Y^{(k)} = \mathbb{1} \left\{ \left[b^{(k)}\right] = 0 \right\}$ , i.e., the optimized output of the  $k$ -th segment
10:  end for
11:   $\llbracket \mathbf{T}_R \rrbracket \leftarrow \text{TLOOKUP-SETUP}(\mathbf{T}_R)$  ▷  $\mathbf{T}_R = [\theta - E, \theta + E]$ , i.e., all tolerable values of row-wise sum of the output
12:  return  $\left(\llbracket \mathbf{T}_X^{(k)} \rrbracket\right)_{k=0}^{K-1}, \left(\llbracket \mathbf{T}_Y^{(k)} \rrbracket\right)_{k=L}^{K-1}, \llbracket \mathbf{T}_R \rrbracket$ 
13: end procedure

14: procedure  $\mathcal{P}.$ zkATTN-COMPUTE( $\mathbf{Z} \in \mathbb{F}^{m \times n}, \left(\mathbf{T}_X^{(k)}\right)_{k=0}^{K-1}, \left(\mathbf{T}_Y^{(k)}\right)_{k=L}^{K-1}$ ) ▷ Some implicit parameters included in SETUP( $\cdot$ ) omitted
15:   $\mathbf{Z}' \leftarrow \mathbf{Z} - \lfloor \hat{\mathbf{z}} \rfloor \mathbf{1}^\top$ , where  $\hat{\mathbf{z}} \in \mathbb{R}^m$  is computed row-wise as (17)
16:   $\left(\mathbf{X}^{(k)}\right)_{k=0}^{K-1} \leftarrow \mathbf{b}^{-1}(-\mathbf{Z}')$  ▷  $-\mathbf{Z}$  is decomposed elementwisely
17:  for  $k \leftarrow L, L + 1, \dots, K - 1$  do
18:     $\mathbf{Y}^{(k)} \leftarrow f^{(k)}(\mathbf{X}^{(k)})$  elementwisely, where  $f^{(k)}$  is defined by  $\mathbf{T}_X^{(k)}, \mathbf{T}_Y^{(k)}$ 
19:  end for
20:   $\mathbf{Y} \leftarrow \odot_{k=L}^{K-1} \mathbf{Y}^{(k)}$  ▷ Compute the final output
21:   $\hat{\mathbf{y}} \leftarrow \mathbf{Y}.\text{sum}(\text{axis} = 1)$  ▷ For checking the normalization of each row
22:   $\mathcal{P} \rightarrow \mathcal{V} : \llbracket \mathbf{Z} \rrbracket, \llbracket \lfloor \hat{\mathbf{z}} \rfloor \rrbracket, \left(\llbracket \mathbf{X}^{(k)} \rrbracket\right)_{k=0}^{K-1}, \llbracket \mathbf{Y} \rrbracket, \left(\llbracket \mathbf{Y}^{(k)} \rrbracket\right)_{k=L}^{K-1}, \llbracket \hat{\mathbf{y}} \rrbracket$ 
23: end procedure

24: procedure  $\langle \mathcal{P}, \mathcal{V} \rangle.$ zkATTN-PROVE( $\llbracket \mathbf{Z} \rrbracket, \llbracket \lfloor \hat{\mathbf{z}} \rfloor \rrbracket, \left(\llbracket \mathbf{X}^{(k)} \rrbracket\right)_{k=0}^{K-1}, \llbracket \mathbf{Y} \rrbracket, \left(\llbracket \mathbf{Y}^{(k)} \rrbracket\right)_{k=L}^{K-1}, \llbracket \hat{\mathbf{y}} \rrbracket$ )
25:  for  $k \leftarrow 0, 1, \dots, K - 1$  do
26:     $\mathcal{P}.\text{TLOOKUP-PREP}(\mathbf{X}^{(k)}, \mathbf{T}_X^{(k)})$  ▷  $\llbracket \mathbf{m}^{(k)} \rrbracket := \mathbf{m}(\mathbf{X}^{(k)}, \mathbf{T}_X^{(k)})$  transmitted to  $\mathcal{V}$ 
27:     $\langle \mathcal{P}, \mathcal{V} \rangle.\text{TLOOKUP-PROVE}(\llbracket \mathbf{X}^{(k)} \rrbracket, \llbracket \mathbf{m}^{(k)} \rrbracket, \llbracket \mathbf{T}_X^{(k)} \rrbracket)$  ▷ Prove the correctness on the  $k$ -th segment
28:  end for
29:   $\mathcal{V} \rightarrow \mathcal{P} : \alpha \sim \mathbb{F}$ 
30:  for  $k \leftarrow L, L + 1, \dots, K - 1$  do
31:     $\mathcal{P}.\text{TLOOKUP-PREP}(\mathbf{X}^{(k)} + \alpha \mathbf{Y}^{(k)}, \mathbf{T}_X^{(k)} + \alpha \mathbf{T}_Y^{(k)})$  ▷  $\llbracket \mathbf{m}^{(k)} \rrbracket := \mathbf{m}(\mathbf{X}^{(k)} + \alpha \mathbf{Y}^{(k)}, \mathbf{T}_X^{(k)} + \alpha \mathbf{T}_Y^{(k)})$  transmitted to  $\mathcal{V}$ 
32:     $\langle \mathcal{P}, \mathcal{V} \rangle.\text{TLOOKUP-PROVE}(\llbracket \mathbf{X}^{(k)} \rrbracket + \alpha \llbracket \mathbf{Y}^{(k)} \rrbracket, \llbracket \mathbf{m}^{(k)} \rrbracket, \llbracket \mathbf{T}_X^{(k)} \rrbracket + \alpha \llbracket \mathbf{T}_Y^{(k)} \rrbracket)$  ▷ Prove the correctness on the  $k$ -th segment
33:  end for
34:   $\mathcal{P}.\text{TLOOKUP-PREP}(\hat{\mathbf{y}}, \mathbf{T}_R)$  ▷  $\llbracket \mathbf{m}_R \rrbracket := \mathbf{m}(\mathbf{Y}, \mathbf{T}_R)$  transmitted to  $\mathcal{V}$ 
35:   $\langle \mathcal{P}, \mathcal{V} \rangle.\text{TLOOKUP-PROVE}(\llbracket \hat{\mathbf{y}} \rrbracket, \llbracket \mathbf{m}_R \rrbracket, \llbracket \mathbf{T}_R \rrbracket)$  ▷ Prove the correctness on the  $k$ -th segment
36:   $\mathcal{P}$  and  $\mathcal{V}$  run the sumcheck for  $\mathbf{b}(\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{K-1}) + \mathbf{Z}' = \mathbf{0} \wedge \mathbf{Z}' = \mathbf{Z} - \lfloor \hat{\mathbf{z}} \rfloor \mathbf{1}^\top \wedge \mathbf{Y} = \odot_{k=L}^{K-1} \mathbf{Y}^{(k)} \wedge \hat{\mathbf{y}} = \mathbf{Y}.\text{sum}(\text{axis} = 1)$ , followed
    by the proof of evaluations on  $\llbracket \mathbf{Z} \rrbracket, \llbracket \lfloor \hat{\mathbf{z}} \rfloor \rrbracket, \left(\llbracket \mathbf{X}^{(k)} \rrbracket\right)_{k=0}^{K-1}, \llbracket \mathbf{Y} \rrbracket, \left(\llbracket \mathbf{Y}^{(k)} \rrbracket\right)_{k=L}^{K-1}, \llbracket \hat{\mathbf{y}} \rrbracket$ 
37: end procedure

```

THEOREM 7.1 (ERROR BOUND). *With the choice of*

$$B_{K-M} \leftarrow \frac{\gamma \sqrt{d}}{K - M - L + 1} ((K - M - L) \ln(2n) + \ln \theta), \quad (31)$$

$$\theta^{(k)} \leftarrow \exp \left(\frac{B^{(k)}}{\gamma \sqrt{d}} (b^{(k)} - 1) \right) \left(\theta \exp \left(-\frac{B_{K-M} - B_L}{\gamma \sqrt{d}} \right) \right)^{\frac{1}{K-M-L}} \quad 9 \quad (32)$$

and irrelevant of the choice of other B_k s, the error bound in (48) can be minimized as

$$\varepsilon_{\text{attn}} = O \left((K - M - L) \left(\frac{n}{\theta} \right)^{\frac{1}{K-M-L+1}} \right). \quad (33)$$

Theorem 7.1 have multiple implications:

- (1) Minimizing $K - M - L$, (i.e., the number of segments that are designated as neither the most significant nor the least significant as in Section 5.1.1) and B_L (i.e., the magnitude of least significant segments) is in favour of reducing the error. However, as will be discussed in Section 7.3, this incurs an undue increase in the computational overhead due to the sizes of the lookup tables.
- (2) $\varepsilon_{\text{attn}}$ has no dependence on the segmentation of the zkAttn input except B_L and B_{K-M} , leaving room for distributing the sizes of lookup tables $\mathbf{T}^{(k)}$ evenly which compresses the computational overhead associated.
- (3) $\varepsilon_{\text{attn}}$ defines the tolerable error row-wise sum upon checking the normalization, i.e., the sum of each row must lie within $[(1 - \varepsilon_{\text{attn}})\theta, (1 + \varepsilon_{\text{attn}})\theta]$.

7.2 Security and privacy analysis

In this section, we formalize the security and privacy aspects of zkLLM , focusing on the tlookup protocol which facilitates zero-knowledge verifiable computations across all non-arithmetic components. We first address the completeness error of tlookup :

THEOREM 7.2 (COMPLETENESS OF PROTOCOL 1). *Assuming the verifier \mathcal{V} is semi-honest, Protocol 1 incurs a completeness error of $O\left(\frac{N}{|\mathbb{F}|}\right)$.*

Theorem 7.2 indicates that if the prover \mathcal{P} follows the protocol, the proof produced using Protocol 1 has a mere $O\left(\frac{N}{|\mathbb{F}|}\right)$ chance of being rejected. As detailed in Appendix C.2, this minor imperfection stems from the probability that the random challenge β , chosen by the verifier in Line 10, might trigger a division-by-zero error in one of the terms on either side of Equation (9). On the other hand, all arithmetic tensor operations do not contribute any additional completeness error, thanks to the direct application of the sumcheck protocol. Therefore, the overall probability of an honest prover failing to convince a semi-honest verifier of the correctness of its inference through the LLM is at most $O\left(\frac{C}{|\mathbb{F}|}\right)$, where C represents the total size of all tensors involved in non-arithmetic operations. Given that the entire complexity of the inference is $\text{poly}(\lambda)$ and $|\mathbb{F}| = \Omega(2^{2\lambda})$, this error remains negligible in λ . Practically, as our implementation employs the BLS12-381 curve with $|\mathbb{F}| \approx 2^{254}$, far exceeding the computational limits of current technology, verification has never failed in any experiment conducted, as reported in Section 8.

Similarly, the soundness error of tlookup is also negligible in λ , as stated in Theorem 7.3. Coupled with the direct application of the sumcheck protocol and the proof-of-opening for the committed tensors, which also incur negligible errors in λ due to the polynomial λ assumption on the complexity of the entire inference process, we theoretically establish that a valid proof can confirm the correctness of the inference result except with only a negligible probability.

THEOREM 7.3 (SOUNDNESS OF PROTOCOL 1). *For any probabilistic polynomial-time (p.p.t.) prover \mathcal{P} , if in Line 6, the message \mathcal{P} sends to \mathcal{V} is $\llbracket \mathbf{S} \rrbracket \leftarrow \text{Commit}(\mathbf{S})$ such that $\mathbf{S} \not\subset \mathbf{T}$, then except with probability*

$\text{negl}(\lambda)$, the execution of Protocol 1 is unsuccessful, resulting in the semi-honest verifier \mathcal{V} rejecting the proof.

PROOF SKETCH OF THEOREM 7.3. By the binding property of the commitment scheme, except with probability $\text{negl}(\lambda)$, in Line 13, the success of proofs of evaluations implies the correctness of all claimed multilinear extension values on $\mathbf{A}, \mathbf{B}, \mathbf{S}, \mathbf{m}$, and \mathbf{T} . Subsequently, the success of the sumchecks implies with $1 - O\left(\frac{D}{|\mathbb{F}|}\right)$ that all equalities in Equations (11) and (12) hold, such that

$$\sum_{i \in [D]} \frac{1}{\beta + \mathbf{S}_i} = \sum_{i \in [N]} \frac{\mathbf{m}_i}{\beta + \mathbf{T}_i}. \quad (34)$$

Finally, given the randomness of β , with probability $1 - O\left(\frac{ND}{|\mathbb{F}|}\right)$, Equation (34) implies Equation (9), leading to the conclusion that $\mathbf{S} \subset \mathbf{T}$. \square

It is noteworthy, as elaborated in Section 7.1, that the correctness of zkAttn is quantified by an \mathcal{L}_1 error of $\varepsilon_{\text{attn}}$. This measure of correctness similarly applies to other exponentiation-based activation functions. The correctness of all other non-arithmetic operations must also consider the quantization errors. For example, as highlighted in Example 4.2, a numerical error margin of $\frac{1}{2^Y}$ is an inescapable consequence of the rescaling process. On the other hand, this degree of tolerance is also sufficient, as numerical errors exceeding $\frac{1}{2^Y}$ would be detectable as incorrect computations and consequently rejected through the application of tlookup .

Finally, with the application of zero-knowledge variations of sumcheck protocols [11, 38, 60, 61] and Pedersen commitment schemes [43], the proof assembled by zkLLM does not disclose any information about the protected model parameters. Formally,

THEOREM 7.4 (ZERO-KNOWLEDGE, ADAPTED FROM [38]). *Assuming the application of zero-knowledge variations of sumcheck protocols [11, 38, 60, 61] and Pedersen commitment schemes [43], there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that the following two views are computationally indistinguishable to any probabilistic polynomial-time (PPT) algorithm \mathcal{A} , given the public parameters pp (the generators used in the commitment scheme within the context of zkLLM):*

Real $\mathcal{A}, \mathbf{W}(\text{pp})$:

- 1: $\llbracket \mathbf{W} \rrbracket \leftarrow \text{zkLLM-Commit}(\mathbf{W}, \text{pp}, r)$
- 2: $\mathbf{X} \leftarrow \mathcal{A}(\llbracket \mathbf{W} \rrbracket, \text{pp})$
- 3: $(y, \pi) \leftarrow \text{zkLLM-Prove}(\mathbf{W}, \mathbf{X}, \text{pp}, r)$
- 4: $b \leftarrow \mathcal{A}(\llbracket \mathbf{W} \rrbracket, \mathbf{X}, y, \pi, \text{pp})$
- 5: **return** b

Ideal $\mathcal{A}, \mathcal{S}^{\mathcal{A}}(\text{pp})$:

- 1: $\text{com} \leftarrow \mathcal{S}_1(1^\lambda, \text{pp}, r)$
- 2: $\mathbf{X} \leftarrow \mathcal{A}(\text{com}, \text{pp})$
- 3: $(y, \pi) \leftarrow \mathcal{S}_2^{\mathcal{A}}(\text{com}, \mathbf{X}, \text{pp}, r)$, given oracle access to $y = \text{zkLLM-compute}(\mathbf{W}, \mathbf{X})$
- 4: $b \leftarrow \mathcal{A}(\text{com}, \mathbf{X}, y, \pi, \text{pp})$
- 5: **return** b

For any PPT algorithm \mathcal{A} and all LLM (represented by the parameter) \mathbf{W} , there exists a simulator \mathcal{S} such that

$$\left| \mathbb{P}(\text{Real}_{\mathcal{A}, \mathbf{W}}(\text{pp}) = 1) - \mathbb{P}(\text{Ideal}_{\mathcal{A}, \mathcal{S}^{\mathcal{A}}}(\text{pp}) = 1) \right| \leq \text{negl}(\lambda). \quad (35)$$

7.3 Overhead analysis

In this section, we analyze the overhead of zkLLM, focusing on the running times for both the prover and the verifier, as well as the memory and communication costs.

7.3.1 Overhead of tlookup. In Protocol 1, we adhere to the assumption that $N = O(D)$. This guideline is strictly followed in our implementation due to the substantial sizes of tensors involved in LLM computations. The linear time complexity for both committing and proving in the Pedersen commitments and the sumcheck protocols results in a total computational complexity of $O(D)$ for the prover in Protocol 1. Similarly, memory requirements are maintained at $O(D)$, since all involved tensors, including the additionally computed \mathbf{m} , \mathbf{A} , and \mathbf{B} , are of size $O(D)$. The commitment and proof sizes are reduced to square root and logarithmic complexities, $O(\sqrt{D})$ and $O(\log D)$ respectively, impacting the verifier's time for verifying the proof of opening and the sumcheck protocols.

7.3.2 Overhead of zkAttn. The overhead introduced by zkAttn is parameterized by K , the number of segments applied to the input. For an input of size mn in zkAttn, the total prover overhead, including both running time and memory consumption, is $O(Kmn)$ as per Section 7.3.1. The communication overhead and verifier time are $O(K\sqrt{mn})$. In comparison with the bit-decomposition method, which incurs an $\Omega(mn \log_2 B)$ overhead, zkAttn built upon tlookup achieves an $O\left(\frac{K}{\log_2 B}\right)$ reduction in prover overhead. Practically, K is chosen to be small enough to minimize both error and overhead while avoiding overly large segments (for example, $K = 1$, which would require compiling all possible input-output pairs into a table, leading to an impractical overhead of at least $\Omega(B)$ for a unrealistically large total number of possible inputs B that exceeds mn , where the previous analysis on overhead would not apply).

7.3.3 Overall overhead. zkLLM benefits from the linear prover overhead of sumcheck protocols and the logarithmic and square-root verifier overheads of sumchecks and Pedersen commitments, respectively. Specialized sumchecks for tensor operations, like matrix multiplications, achieve less complexity than the computation process itself, further reducing proof overhead for each layer. Assuming prover overhead, communication cost, and verifier overhead per layer of an L -layer LLM are t_P , c , and t_V respectively, the total overheads of zkLLM scale naturally to $O(Lt_P)$, $O(Lc)$, and $O(Lt_V)$. Where the latter two can be further reduced to $O(\sqrt{L}c)$ and $O(\sqrt{L}t_V)$ by leveraging the repetitive structure across layers and batching up the commitments [38]. Additionally, unlike univariate polynomial-based ZKP systems that must be serialized, the use of sumcheck protocols over multilinear extensions and the compatible Pedersen commitment scheme in zkLLM allows for highly parallelized proof generation, thereby enabling efficient proof generation in a reasonable time.

8 EXPERIMENTS

We developed zkLLM using CUDA, basing it on the CUDA code for the BLS12-381 curve [6] produced by the ec-gpu package [19]. The implementation of sequential verifier tasks, which cannot efficiently utilize CUDA, was adapted from the zkCNN implementation [38] that relies on the mcl package [40]. We evaluated zkLLM for

inferences on two classes of open-source LLMs, namely OPT [67] and LLaMa-2 [52], supporting sizes up to 13 billion parameters. For both types of models, our focus was on performing verifiable inferences using the designated models, applying samples with the default sequence length of 2048 from the C4 dataset [45]. Our experiments were conducted with resources including 124.5GB of memory, 12 CPU cores of an AMD EPYC 7413 (2.65 GHz with 128M cache L3), and an NVIDIA A100SMX4 GPU with 40GB of memory, all allocated from a computing node.

Throughout our experiments, we consistently set the scaling factor for both data embedding and model parameters at 2^{16} . All rescaling operations were integrated with the subsequent activation functions. As detailed in Section 4.2, this integration necessitated the use of multiple tlookups. Specifically, the number of tlookups corresponds to the number of times the input to the activation function requires rescaling, with each tlookup having a size of 2^{16} , to avoid excessive memory usage.

Additionally, since the input to the Softmax function in zkAttn of each layer undergoes two multiplication operations, the cumulative scaling factor reaches 2^{64} . To manage this, we deployed $K = 5$ tlookups, each of size 2^{16} . This setup includes $L = 3$ least significant segments, with the remaining two segments accommodating all potential inputs within a scale of 2^{16} and a precision of 2^{-16} when reverted to the real domain.

The tolerable error margins were selected in accordance with Section 7.1. This approach resulted in an approximate total \mathcal{L}_1 error of 10^{-2} on the output, a level comparable to the rounding error induced by half-precision floating points used in state-of-the-art LLMs. Notably, all proofs involved in the results of this section have been successfully validated by the semi-honest verifier.

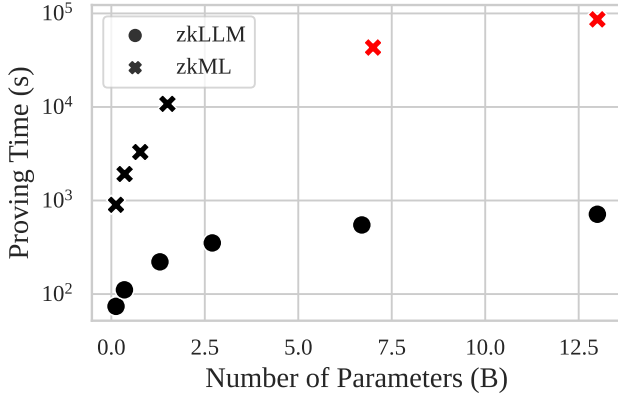
In Table 1, we present detailed information regarding the overhead associated with zkLLM for models of various sizes. The data includes the time required for the prover to commit and hide the model (committing time and commitment sizes), as well as the prover's time, proof size, and the verifier's time in response to an input prompt from the latter.

As the first endeavor to apply zero-knowledge proofs to LLMs with up to 13 billion parameters, to the best of our knowledge, zkLLM has achieved significant results toward practical and industrial applicability. Once the model is trained, the prover requires up to 20 minutes to commit to the model and subsequently publish a commitment of approximately 10MB for public scrutiny via zkLLM. When prompted by the verifier, the prover is able to generate a proof for the correctness of the entire inference process in less than 15 minutes. Additionally, as the design of zkAttn effectively resolves the inherent bottleneck of listing all input-output pairs, the memory consumption is effectively controlled under 23.1GB, which fits zkLLM into commonly used GPUs in machine learning, like Tesla V100 and A100.

It is important to note that while the time for committing generally scales with the size of the parameters, the time for generating proofs scales more slowly. This slower scaling is attributed to the less significant differences in the complexities of intermediate computations and more efficient use of parallel computing resources as the size of the tensors increases. Ultimately, the succinct proof, which is only about 100kB in size, can be verified by the verifier in a matter of seconds. Despite this efficiency, the verifier is provably

Table 1: The overhead of zkLLM on OPT and LLaMa-2.

Model	OPT-125M	OPT-350M	OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B	LLaMa-2-7B	LLaMa-2-13B
Committing time (s)	11.8	33.1	127	273	654	1.27×10^3	531	986
Commitment size (MB)	0.996	1.67	3.32	4.58	7.22	10.1	7.97	11.0
Prover time (s)	73.9	111	221	352	548	713	620	803
Proof size (kB)	141	144	147	152	157	160	183	188
Verifier time (s)	0.342	0.593	0.899	1.41	2.08	3.71	2.36	3.95
Memory usage (GB)	1.88	2.38	3.71	6.60	15.0	22.9	15.5	23.1
C4 Perplexity (orig)	26.56	22.59	16.07	14.34	12.71	12.06	7.036	6.520
C4 Perplexity (quant)	26.65	22.66	16.12	14.37	12.73	12.07	7.049	6.528

**Figure 4: Comparison between zkLLM and zkML. Results of test cases with OOM errors are estimated and marked in red.**

unable to glean any additional information about the model or the inference result, ensuring the correctness of the inference while maintaining the confidentiality of the model parameters.

Moreover, the numerical error due to the unavoidable discretization of the entire process for the application of the cryptographic tools does not cause significant accuracy drops: on the C4 dataset [45], the increase of perplexity is less than 0.1, and the impact diminishes to less than 0.01 as the sizes of the models scale to 13B.

In Figure 4, we further compare zkLLM with zkML [33], the first zero-knowledge proof to have achieved verifiable inference for GPT-2 under identical hardware conditions. Beyond the size of GPT-2 (1.5B parameters), where zkML results in an out-of-memory (OOM) error, we provide an estimation of the required proving time. Thanks to the design tailored for non-arithmetic tensor operations and the attention mechanism prevalent in LLMs, as well as its CUDA implementation, zkLLM extends the zero-knowledge verifiability to LLMs with 10x larger sizes, achieving an approximate 50x speedup.

8.1 Additional experimental results on tlookup and zkAttn

To further demonstrate the efficiency of tlookup in addressing non-arithmetic tensor operations in deep learning, as well as zkAttn, which is pivotal for verifiable computations in LLMs, we isolated an instance of zkAttn from the first layer of variously sized OPT models on input sequences of different lengths. We then measured

the overhead incurred by zkAttn, including the multiple tlookups it encompasses. The results are presented in Figure 5.

It is evident that, in contrast to the overall overhead, the overhead specific to zkAttn is less influenced by the size of the model, particularly regarding proving time. However, the length of the input sequence significantly impacts various aspects of the overhead. This observation can be attributed to the fact that the Attention mechanism, while involving LLM parameters, is more significantly affected by the interactions between intermediate values (e.g., Q, K, V), where their dimensions play a crucial role in determining the overhead.

Notably, the largest tested sequence length, 4096, exceeds the original design specifications of OPT models and was primarily included as a reference to assess the impact of sequence length on overhead. In contrast, in Table 1, which documents overall results for zkLLM, we set the sequence length to 2048—the maximum feasible value—to maintain experimental consistency and fairness.

9 RELATED WORKS

Starting with Zhang et al. in 2020 [65], the field of zero-knowledge machine learning inference has seen active development. Initial research, parallel to the surge in computer vision studies, primarily concentrated on authenticating inference results for computer vision tasks over convolutional neural networks (CNNs). Key contributions include zkCNN [38], ZEN [18], vCNN [35], pvCNN [57], zkML [33], Mystique [56], and ezDPS [55]. These works aimed to optimize the adaptation of the entire training process to zero-knowledge proof (ZKP) backends, such as zkSNARKS [3–5, 7, 10, 24, 28, 36, 42], by leveraging the special structures of computations within CNNs. Notably, zkCNN [38] introduced a specialized interactive proof protocol for convolutional layers based on the GKR protocol [27] and its refinements [60, 61, 66]. This protocol achieved efficient proofs (less than 2 minutes) on VGG-scale CNNs, highlighting the necessity of specialized protocols for realistic zero-knowledge machine learning inference. However, as of our current knowledge, there exists a gap in zero-knowledge inferences over LLMs. The intricate structures and enormous sizes of LLMs present challenges not addressed by previous studies focused on CNNs, necessitating novel theoretical and experimental developments.

Conversely, while pioneering studies have addressed ZKPs for machine learning training, with works such as VeriML [68], proof of unlearning [17, 58], and zkPoT [25] focusing on elementary algorithms like Support Vector Machines (SVM), logistic regression, and small neural networks (up to several thousand parameters),

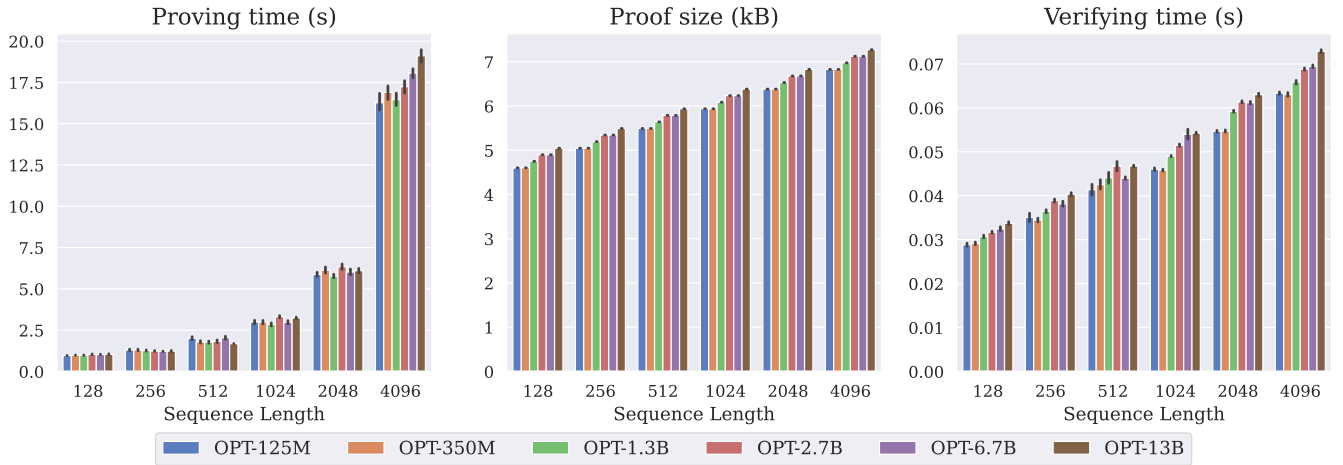


Figure 5: Overhead of zkAttn.

extending zero-knowledge proofs to training LLMs may pose insurmountable challenges. The vast complexity inherent in training LLMs could render the zero-knowledge proofs for their training impractical.

10 CONCLUSION

This paper introduces zkLLM, marking the inaugural specialized zero-knowledge proof tailored for large language models, as far as our current knowledge extends. zkLLM pioneers zero-knowledge verifiable computations for general non-arithmetic operations within neural networks, ensuring full parallelizability and zero additional overhead through the implementation of tlookup. Building on this foundation, we further present zkAttn, a novel zero-knowledge proof specifically designed for the attention mechanism—a pivotal component underpinning the exceptional performance of modern LLMs. With a CUDA implementation optimized for parallel computing resources in deep learning, zkLLM achieves a groundbreaking milestone as the first study to provide zero-knowledge verifiability for LLMs with 13 billion parameters. This endeavor stands as a significant contribution towards fortifying the legitimacy of LLMs in light of their transformative impact on various domains.

REFERENCES

- [1] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR* abs/2312.11805 (2023). <https://doi.org/10.48550/ARXIV.2312.11805> arXiv:2312.11805
- [2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016). <http://arxiv.org/abs/1607.06450>
- [3] Rishabh Bhaduria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkatasubramanian, Tiancheng Xie, and Yupeng Zhang. 2020. Liger++: A New Optimized Sublinear IOP. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM, 2025–2038. <https://doi.org/10.1145/3372297.3417893>
- [4] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. 2022. Succinct Non-Interactive Arguments via Linear Interactive Proofs. *J. Cryptol.* 35, 3 (2022), 15. <https://doi.org/10.1007/Ss00145-022-09424-4>
- [5] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. 2020. Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme. *IACR Cryptol. ePrint Arch.* (2020), 1536. <https://eprint.iacr.org/2020/1536>
- [6] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2248)*, Colin Boyd (Ed.). Springer, 514–532. https://doi.org/10.1007/3-540-45682-1_30
- [7] Sean Bowe, Jack Grigg, and Daira Hopwood. 2019. Halo: Recursive Proof Composition without a Trusted Setup. *IACR Cryptol. ePrint Arch.* (2019), 1021. <https://eprint.iacr.org/2019/1021>
- [8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>
- [9] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. *CoRR* abs/2307.13304 (2023). <https://doi.org/10.48550/ARXIV.2307.13304> arXiv:2307.13304
- [10] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. 2023. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 14005)*, Carmit Hazay and Martijn Stam (Eds.). Springer, 499–530. https://doi.org/10.1007/978-3-031-30617-4_17
- [11] Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. 2017. A Zero Knowledge Sumcheck and its Applications. *Electron. Colloquium Comput. Complex.* TR17-057 (2017). ECCC:TR17-057 <https://eccc.weizmann.ac.il/report/2017/057>
- [12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin,

- Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shrivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling Language Modeling with Pathways. *J. Mach. Learn. Res.* 24 (2023), 240:1–240:113. <http://jmlr.org/papers/v24/22-1144.html>
- [13] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. 2012. Practical verified computation with streaming interactive proofs. In *Innovations in Theoretical Computer Science 2012*. Cambridge, MA, USA, January 8-10, 2012, Shafi Goldwasser (Ed.). ACM, 90–112. <https://doi.org/10.1145/2090236.2090245>
- [14] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *CoRR abs/2208.07339* (2022). <https://doi.org/10.48550/ARXIV.2208.07339> arXiv:2208.07339
- [15] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *CoRR abs/2305.14314* (2023). <https://doi.org/10.48550/ARXIV.2305.14314> arXiv:2305.14314
- [16] Liam Eagen, Dario Fiore, and Ariel Gabizon. 2022. cq: Cached quotients for fast lookups. *IACR Cryptol. ePrint Arch.* (2022), 1763. <https://eprint.iacr.org/2022/1763>
- [17] Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. 2022. Verifiable and Provably Secure Machine Unlearning. *CoRR abs/2210.09126* (2022). <https://doi.org/10.48550/arXiv.2210.09126> arXiv:2210.09126
- [18] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. 2021. ZEN: Efficient Zero-Knowledge Proofs for Neural Networks. *IACR Cryptol. ePrint Arch.* (2021), 87. <https://eprint.iacr.org/2021/087>
- [19] Filecoin. 2023. ec-gpu. <https://github.com/filecoin-project/ec-gpu>. Accessed: 2024-01-22.
- [20] Elias Frantar and Dan Alistarh. 2023. QMoE: Practical Sub-1-Bit Compression of Trillion-Parameter Models. *CoRR abs/2310.16795* (2023). <https://doi.org/10.48550/ARXIV.2310.16795>
- [21] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2022. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *CoRR abs/2210.17323* (2022). <https://doi.org/10.48550/ARXIV.2210.17323> arXiv:2210.17323
- [22] Ariel Gabizon and Dmitry Khovratovich. 2022. flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. *IACR Cryptol. ePrint Arch.* (2022), 1447. <https://eprint.iacr.org/2022/1447>
- [23] Ariel Gabizon and Zachary J. Williamson. 2020. plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptol. ePrint Arch.* (2020), 315. <https://eprint.iacr.org/2020/315>
- [24] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. 2019. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. *IACR Cryptol. ePrint Arch.* (2019), 953. <https://eprint.iacr.org/2019/953>
- [25] Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoudy, Guru-Vamsi Policharla, and Mingyuan Wang. 2023. Experimenting with Zero-Knowledge Proofs of Training. *IACR Cryptol. ePrint Arch.* (2023), 1345. <https://eprint.iacr.org/2023/1345>
- [26] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. 2017. 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*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 4672–4681. <https://proceedings.neurips.cc/paper/2017/hash/6048ff4e8cb07aa60b677b6f7384d52-Abstract.html>
- [27] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. 2008. 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*, Cynthia Dwork (Ed.). ACM, 113–122. <https://doi.org/10.1145/1374376.1374396>
- [28] Jens Groth. 2016. 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 (Lecture Notes in Computer Science, Vol. 9666)*, Marc Fischlin and Jean-Sébastien Coron (Eds.). Springer, 305–326. https://doi.org/10.1007/978-3-662-49896-5_11
- [29] Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Chou-Jui Hsieh. 2022. Watermarking Pre-trained Language Models with Backdoor-ing. *CoRR abs/2210.07543* (2022). <https://doi.org/10.48550/ARXIV.2210.07543> arXiv:2210.07543
- [30] Ulrich Haböck. 2022. Multivariate lookups based on logarithmic derivatives. *IACR Cryptol. ePrint Arch.* (2022), 1530. <https://eprint.iacr.org/2022/1530>
- [31] Dan Hendrycks and Kevin Gimpel. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR abs/1606.08415* (2016). arXiv:1606.08415 <http://arxiv.org/abs/1606.08415>
- [32] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased Watermark for Large Language Models. *CoRR abs/2310.10669* (2023). <https://doi.org/10.48550/ARXIV.2310.10669> arXiv:2310.10669
- [33] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. 2022. Scaling up Trustless DNN Inference with Zero-Knowledge Proofs. *CoRR abs/2210.08674* (2022). <https://doi.org/10.48550/arXiv.2210.08674> arXiv:2210.08674
- [34] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A Watermark for Large Language Models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 17061–17084. <https://proceedings.mlr.press/v202/kirchenbauer23a.html>
- [35] Seunghwa Lee, Hankyung Ko, Jiye Kim, and Hyunok Oh. 2020. vCNN: Verifiable Convolutional Neural Network. *IACR Cryptol. ePrint Arch.* (2020), 584. <https://eprint.iacr.org/2020/584>
- [36] Marta Ligeró, Guillermo Torres, Carles Sánchez, Katerine Diaz-Chito, Raquel Perez, and Debora Gil. 2019. Selection of Radiomics Features based on their Reproducibility. In *41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2019, Berlin, Germany, July 23-27, 2019*, IEEE, 403–408. <https://doi.org/10.1109/EMBC.2019.8857879>
- [37] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *CoRR abs/2306.00978* (2023). <https://doi.org/10.48550/ARXIV.2306.00978> arXiv:2306.00978
- [38] Tianyi Liu, Xiang Xie, and Yupeng Zhang. 2021. 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*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 2968–2985. <https://doi.org/10.1145/3460120.3485379>
- [39] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. 1992. Algebraic Methods for Interactive Proof Systems. *J. ACM* 39, 4 (1992), 859–868. <https://doi.org/10.1145/146585.146605>
- [40] Shigeo Mitsunari. 2023. MCL: A portable and fast pairing-based cryptography library. <https://github.com/herumi/mcl>. Accessed: 2024-01-22.
- [41] OpenAI. 2023. GPT-4 Technical Report. *CoRR abs/2303.08774* (2023). <https://doi.org/10.48550/ARXIV.2303.08774> arXiv:2303.08774
- [42] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. 2013. 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, 238–252. <https://doi.org/10.1109/SP.2013.47>
- [43] Torben P. Pedersen. 1991. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings (Lecture Notes in Computer Science, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, 129–140. https://doi.org/10.1007/3-540-46766-1_9
- [44] Jim Posen and Assimakis A. Kattis. 2022. Caulk+: Table-independent lookup arguments. *IACR Cryptol. ePrint Arch.* (2022), 957. <https://eprint.iacr.org/2022/957>
- [45] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. <http://jmlr.org/papers/v21/20-074.html>
- [46] Jacob T. Schwartz. 1980. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM* 27, 4 (1980), 701–717. <https://doi.org/10.1145/322217.322225>
- [47] Noam Shazeer. 2020. GLU Variants Improve Transformer. *CoRR abs/2002.05202* (2020). arXiv:2002.05202 <https://arxiv.org/abs/2002.05202>
- [48] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yufeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing* 568 (2024), 127063. <https://doi.org/10.1016/J.NEUCOM.2023.127063>
- [49] Justin Thaler. 2013. 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 (Lecture Notes in Computer Science, Vol. 8043)*, Ran Canetti and Juan A. Garay (Eds.). Springer, 71–89. https://doi.org/10.1007/978-3-642-40084-1_5
- [50] Justin Thaler. 2022. Proofs, Arguments, and Zero-Knowledge. *Found. Trends Priv. Secur.* 4, 2-4 (2022), 117–660. <https://doi.org/10.1561/33000000030>
- [51] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR abs/2302.13971* (2023). <https://doi.org/10.48550/ARXIV.2302.13971> arXiv:2302.13971

- [52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shriti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucu-rull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). <https://doi.org/10.48550/ARXIV.2307.09288> arXiv:2307.09288
- [53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [54] Riad S. Wahby, Ioanna Tziaila, Abhi Shelat, Justin Thaler, and Michael Walfish. 2018. 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, 926–943. <https://doi.org/10.1109/SP.2018.00060>
- [55] Haodi Wang and Thang Hoang. 2023. ezDPS: An Efficient and Zero-Knowledge Machine Learning Inference Pipeline. *Proc. Priv. Enhancing Technol.* 2023, 2 (2023), 430–448. <https://doi.org/10.56553/popets-2023-0061>
- [56] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. 2021. Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, Michael Bailey and Rachel Greenstadt (Eds.)*. USENIX Association, 501–518. <https://www.usenix.org/conference/usenixsecurity21/presentation/weng>
- [57] Jia-Si Weng, Jian Weng, Gui Tang, Anjia Yang, Ming Li, and Jia-Nan Liu. 2023. pvcNN: Privacy-Preserving and Verifiable Convolutional Neural Network Testing. *IEEE Trans. Inf. Forensics Secur.* 18 (2023), 2218–2233. <https://doi.org/10.1109/TIFS.2023.3262932>
- [58] Jiasi Weng, Shenglong Yao, Yuefeng Du, Junjie Huang, Jian Weng, and Cong Wang. 2022. Proof of Unlearning: Definitions and Instantiation. *CoRR* abs/2210.11334 (2022). <https://doi.org/10.48550/ARXIV.2210.11334> arXiv:2210.11334
- [59] Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 38087–38099. <https://proceedings.mlr.press/v202/xiao23c.html>
- [60] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. 2019. 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 (Lecture Notes in Computer Science, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 733–764. https://doi.org/10.1007/978-3-030-26954-8_24
- [61] Tiancheng Xie, Yupeng Zhang, and Dawn Song. 2022. 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 (Lecture Notes in Computer Science, Vol. 13510)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, 299–328. https://doi.org/10.1007/978-3-031-15985-5_11
- [62] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. 2022. Caulk: Lookup Arguments in Sublinear Time. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, 3121–3134. <https://doi.org/10.1145/3548606.3560646>
- [63] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. 2022. Baloo: Nearly Optimal Lookup Arguments. *IACR Cryptol. ePrint Arch.* (2022), 1565. <https://eprint.iacr.org/2022/1565>
- [64] Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 12360–12371. <https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html>
- [65] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. 2020. Zero Knowledge Proofs for Decision Tree Predictions and Accuracy. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM, 2039–2053. <https://doi.org/10.1145/3372297.3417278>
- [66] Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. 2021. 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*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 159–177. <https://doi.org/10.1145/3460120.3484767>
- [67] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Mylène Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *CoRR* abs/2205.01068 (2022). <https://doi.org/10.48550/ARXIV.2205.01068> arXiv:2205.01068
- [68] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, and Bo Feng. 2021. VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service. *IEEE Trans. Parallel Distributed Syst.* 32, 10 (2021), 2524–2540. <https://doi.org/10.1109/TPDS.2021.3068195>
- [69] Richard Zippel. 1979. Probabilistic algorithms for sparse polynomials. In *Sym-bolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings (Lecture Notes in Computer Science, Vol. 72)*, Edward W. Ng (Ed.). Springer, 216–226. https://doi.org/10.1007/3-540-09519-5_73

A CODE

Our open-source implementation is available at <https://github.com/jvhs0706/zkllm-ccs2024>.

B ADDITIONAL RELATED WORKS

In this appendix, we overview of two classes of related studies, and explain their relatedness and distinction with this study.

LLM quantization. To facilitate the deployment of cryptographic tools, this study employs quantization to map the entire computation over LLMs into finite fields. Despite state-of-the-art quantization methods (e.g., LLM.int8()[14], SmoothQuant [59], GPTQ[21], AWQ[37], QuIP [9], QMoE[20], QLoRA[15]) already achieving low-bit compression of model parameters and activations in LLMs, intermediate computations between these values still involve floating-point numbers to maintain accuracy under low-bit settings. Consequently, these quantization methods are not directly applicable to our study, where computations are strictly confined to finite fields. In our experiments, we utilize a larger bit size (16 bits) to ensure the preservation of accuracy within the finite field constraints.

LLM watermark. Watermarks in LLMs [29, 32, 34] serve as an additional mechanism to indicate that an output is generated by a specific model. However, it is crucial to note that watermarks in LLMs have vastly different application domains compared to zkLLM. The primary distinction lies in the nature of the assurances they provide. Watermarks offer a form of model identification, while zkLLM, in contrast, asserts a much stronger statement by proving the correctness of the output concerning the committed parameters and the prescribed computational process within the LLM. Unlike watermarks, zkLLM provides a guarantee that the computational process has not been tampered with, offering a more robust and verifiable assurance of the authenticity of the output.

C APPENDIX OF SECTION 7

C.1 Appendix on Error Analysis

In this appendix, we analyze the numerical of zkAttn, by giving a detailed proof of Theorem 7.1.

PROOF OF THEOREM 7.1. We first consider the errors introduced in the exponentiation operation, with the input of $-B < z \leq 0$, such that $-z$ is decomposed as $(x^{(0)}, x^{(1)}, \dots, x^{(K-1)})$ following (20).

The error incurred in the estimation of $\exp\left(\frac{z}{\gamma\sqrt{d}}\right)$ can be expressed as

$$\left| \exp\left(\frac{z}{\gamma\sqrt{d}}\right) - \frac{1}{\theta} \prod_{k=0}^{K-1} y^{(k)} \right|, \quad (36)$$

where each $(x^{(k)}, y^{(k)}) \in \mathbf{T}^{(k)}$ is encoded in the lookup tables $\mathbf{T}^{(k)}$ s after the optimization for the most and least significant segments in Section 5.1.1.

For $z \leq -B_{K-M}$, (36) becomes $\exp\left(\frac{z}{\gamma\sqrt{d}}\right)$ as at least one of the $y^{(k)}$ s become 0. Therefore, we focus on the case that $-B_{K-M} < z \leq 0$, such that $-z$ is decomposed as $(x_0, x_1, \dots, x_{K-M-1}, 0, 0, \dots, 0)$, (36) becomes

$$\begin{aligned} & \frac{1}{\theta} \left| \prod_{k=0}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right| \\ & \leq \frac{1}{\theta} \left(\left| \prod_{k=0}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right| + \right. \\ & \quad \left. \left| \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right| \right) \\ & \leq \left| \prod_{k=0}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right| \\ & \quad + \frac{1}{\theta} \left| \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right|, \end{aligned} \quad (37)$$

$$\begin{aligned} & \leq \left| \prod_{k=0}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right| \\ & \quad + \frac{1}{\theta} \left| \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right|, \end{aligned} \quad (38)$$

$$\begin{aligned} & \leq \left| \prod_{k=0}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right| \\ & \quad + \frac{1}{\theta} \left| \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right|, \end{aligned} \quad (39)$$

where the first term of (39) can be upper bounded by

$$\exp\left(\frac{z}{\gamma\sqrt{d}}\right) \left(\exp\left(\frac{B_L}{\gamma\sqrt{d}}\right) - 1 \right). \quad (40)$$

Therefore, we focus our effort on bounding the second term, where

$$\frac{1}{\theta} \left| \prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) - \prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right] \right| \quad (41)$$

$$\leq \prod_{k=L}^{K-M-1} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \left| \frac{\prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right]}{\prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right)} - 1 \right| \quad (42)$$

$$\leq \exp\left(\frac{z+B_L}{\gamma\sqrt{d}}\right) \left| \frac{\prod_{k=L}^{K-M-1} \left[\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) \right]}{\prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right)} - 1 \right| \quad (43)$$

$$\leq \exp\left(\frac{z+B_L}{\gamma\sqrt{d}}\right) \left(\frac{\prod_{k=L}^{K-M-1} \left(\theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right) + \frac{1}{2} \right)}{\prod_{k=L}^{K-M-1} \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x^{(k)}\right)} - 1 \right) \quad (44)$$

$$\leq \exp\left(\frac{z+B_L}{\gamma\sqrt{d}}\right) \left(\prod_{k=L}^{K-M-1} \left(1 + \frac{\exp\left(\frac{B^{(k)}(b^{(k)}-1)}{\gamma\sqrt{d}}\right)}{2\theta^{(k)}} \right) - 1 \right), \quad (45)$$

such that with the constraint that $\prod_{k=L}^{K-M-1} \theta^{(k)} = \theta$, setting $\theta^{(k)} \leftarrow \theta^{(k)*} := \exp\left(\frac{B^{(k)}}{\gamma\sqrt{d}}(b^{(k)}-1)\right) \left(\theta \exp\left(-\frac{B_{K-M}-B_L}{\gamma\sqrt{d}}\right) \right)^{\frac{1}{K-M-L}}$ for each i minimizes (45) as

$$\exp\left(\frac{z+B_L}{\gamma\sqrt{d}}\right) \left(\left(1 + \frac{\exp\left(\frac{B_{K-M}-B_L}{(K-M-L)\gamma\sqrt{d}}\right)}{2\theta^{\frac{1}{K-M-L}}}\right)^{K-M-L} - 1 \right). \quad (46)$$

Summarizing above, the approximation error of $\exp\left(\frac{z}{\gamma\sqrt{d}}\right)$, namely

$$\varepsilon_{\exp}(z) = \begin{cases} C \exp(z), & -B_{K-M} < z \leq 0; \\ \exp(z), & z \leq -B_{K-M} \end{cases}, \text{ for some asymptotically small coefficient } C$$

$$C := \left(\exp\left(\frac{B_L}{(K-M-L)\gamma\sqrt{d}}\right) + \frac{\exp\left(\frac{B_{K-M}}{(K-M-L)\gamma\sqrt{d}}\right)}{2\theta^{\frac{1}{K-M-L}}} \right)^{K-M-L} - 1.$$

Therefore, consider a row $\mathbf{z} = (z_0, z_1, \dots, z_{n-1})$ where the Softmax is applied, due to the additional error incurred by the rounding of $\hat{\mathbf{z}}$ defined by (17), the L_1 -error of approximation is upper-bounded

by

$$\begin{aligned}
& \sum_{i=0}^{n-1} \left(\left| \exp\left(\frac{z_i - \hat{z}}{\gamma\sqrt{d}}\right) - \exp\left(\frac{z_i - \lfloor \hat{z} \rfloor}{\gamma\sqrt{d}}\right) \right| + \varepsilon_{\exp}(z_i - \lfloor \hat{z} \rfloor) \right) \quad (47) \\
& \leq \exp\left(\frac{1}{2\gamma\sqrt{d}}\right) + \sum_{i=0}^{n-1} \varepsilon_{\exp}(z_i - \lfloor \hat{z} \rfloor) \\
& \leq \left(\exp\left(\frac{1}{2\gamma\sqrt{d}}\right) - 1 \right) \sum_{i=0}^{n-1} \exp\left(\frac{z_i - \hat{z}}{\gamma\sqrt{d}}\right) \\
& \quad + \sum_{z_i - \lfloor \hat{z} \rfloor \leq -B_{K-M}} \varepsilon_{\exp}(z_i - \lfloor \hat{z} \rfloor) + \sum_{z_i - \lfloor \hat{z} \rfloor > -B_{K-M}} \varepsilon_{\exp}(z_i - \lfloor \hat{z} \rfloor) \\
& \leq C \exp\left(\frac{1}{2\gamma\sqrt{d}}\right) + (n-1) \exp\left(-\frac{B_{K-M}}{\gamma\sqrt{d}}\right) =: \varepsilon_{\text{attn}}, \quad (48)
\end{aligned}$$

such that with other variables fixed, $\varepsilon_{\text{attn}}$ should be minimized as a function B_{K-M} which gives an optimal B_{K-M}^* .

Subsequently, with the choice of

$$B_{K-M}^* \leftarrow \frac{\gamma\sqrt{d}}{K-M-L+1} ((K-M-L) \ln(2n) + \ln \theta), \quad (49)$$

the error bound in (48) can be minimized as

$$\varepsilon_{\text{attn}} = O\left((K-M-L) \left(\frac{n}{\theta}\right)^{\frac{1}{K-M-L+1}}\right). \quad (50)$$

□

C.2 Appendix on Security and Privacy Analysis

In this appendix, we present the details of the security and privacy analysis omitted in Section. 7.2, giving the proofs of Theorems 7.2 and 7.3.

PROOF OF THEOREM 7.2. If $S \subset T$ as sets, where $T \in \mathbb{F}^N$, then in Line 10 of Protocol 1, the random challenge \mathcal{V} sent to \mathcal{P} , namely β , is excluded from the set $E := \{x \in \mathbb{F} : -x \in S \vee -x \in T\}$ with probability $1 - \frac{N}{|\mathbb{F}|}$ since $|E| \leq N$. Therefore, with probability at least $1 - \frac{N}{|\mathbb{F}|}$, all $\beta + S_i$ ($i \in [D]$) and $\beta + T_i$ ($i \in [N]$) are invertible, and (12) holds with the definitions of \mathbf{m} and \mathbf{A}, \mathbf{B} as in (10) and (11). Therefore, by denoting the RHS of (13) as a quadratic formula of α , i.e., $c_0 + c_1\alpha + c_2\alpha^2$, we have $c_0 = 0$. Moreover, the definitions of \mathbf{A} and \mathbf{B} automatically guarantee that $c_1 = c_2 = 0$. Therefore, with probability $1 - \frac{N}{|\mathbb{F}|}$, the equality of (13) (and therefore (14)) holds, such that the semi-honest verifier accepts the proof due to the perfect completeness of the sumcheck protocol. □

PROOF OF THEOREM 7.3. Assume the acceptance of the proof by the semi-honest verifier \mathcal{V} . Therefore:

- In Line 13 of Protocol 1, by the soundness of the Pedersen commitment scheme [43] (instantiated by Hyrax [54]), with probability $1 - \text{negl}(\lambda)$, the prover \mathcal{P} has computed the transmitted $\llbracket \mathbf{A} \rrbracket$, $\llbracket \mathbf{B} \rrbracket$, $\llbracket \mathbf{S} \rrbracket$, and $\llbracket \mathbf{m} \rrbracket$ with correct multilinear extension values $\tilde{\mathbf{A}}(\mathbf{v})$, $\tilde{\mathbf{B}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$, $\tilde{\mathbf{S}}(\mathbf{v})$, $\tilde{\mathbf{m}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$ subject to the random challenges \mathbf{v} chosen by the verifier \mathcal{V} during the execution of the sumcheck protocol. Also, with probability $1 - \text{negl}(\lambda)$, the claimed value of $\tilde{\mathbf{T}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$

is correct with respect to the committed \mathbf{T} which \mathcal{P} and \mathcal{V} agree upon.

- By the soundness of the sumcheck protocol, with probability $1 - O\left(\frac{D}{|\mathbb{F}|}\right)$, the correctness of $\tilde{\mathbf{A}}(\mathbf{v})$, $\tilde{\mathbf{B}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$, $\tilde{\mathbf{S}}(\mathbf{v})$, $\tilde{\mathbf{m}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$, and $\tilde{\mathbf{T}}(\mathbf{v}_{\lfloor \log_2 \frac{D}{N} \rfloor})$ implies the equality of (14), and therefore the equality of (13).
- By the Schwartz-Zippel Lemma [46, 69], the equality of (13) implies that each term on the RHS of (13) is 0 with $1 - \frac{2}{|\mathbb{F}|}$ probability. Therefore, the equality of (9) holds for $X \leftarrow \beta$.
- Upon applying the Schwartz-Zippel Lemma once more, given the randomness of β , if the equality in (9) is valid for $X \leftarrow \beta$, then the equality as rational functions is also valid with a probability of $1 - \frac{1}{D^N}$. This, in turn, implies the inclusion relation $S \subset T$.

Combining the arguments above, under our assumption that both $\frac{D}{|\mathbb{F}|}$ and $\frac{N}{|\mathbb{F}|}$ are both negligible in λ , the acceptance of the proof implies $S \subset T$ with probability $1 - \text{negl}(\lambda)$. □