TAIPEI TECH 國立臺北科技大學 National Taipei University of Technology

zkLLM: Zero-Knowledge Proofs for Large Language Models

Group 3

Member

110590005 蕭耕宏, 112C53035 王煥昇, 113C53006 吳仲霖,

113598043 張育丞, 113598088 李以謙

Introduction

AI Legitimacy and Concerns

- Surge in AI, especially LLMs, has led to legal and ethical challenges (e.g., lawsuits, executive orders).
- Verifying LLM-generated outputs is crucial for addressing concerns like copyrighted material misuse and harmful content.

Challenges in LLM Auditing

- LLM parameters are protected as intellectual property, making direct examination impossible.
- Existing ZKP frameworks are not optimized for LLMs, which require:
  - Handling non-arithmetic operations (e.g., GELU, SwiGLU activation functions).
  - Managing memory and computational overhead.

Necessity of Zero-Knowledge Proofs

- ZKPs ensure verifiable computations without exposing model details.
- Traditional ZKP methods struggle with LLM-specific challenges like the attention mechanism and frequent use of Softmax functions.

BY 張育丞

Proposed Solution: zkLLM

zkLLM is a Zero-Knowledge Proof scheme tailored specifically for LLMs, addressing the challenges of scalability, efficiency, and security.

Core Innovations

1. tlookup:

- A unique ZKP protocol for universal non-arithmetic operations in deep learning.
- No asymptotic overhead in memory or runtime.
- High parallelization capability, leveraging GPUs.

Proposed Solution: zkLLM

2. zkAttn:
   - A ZKP protocol for attention mechanisms in LLMs.
   - Enhances tlookup to address attention mechanism challenges.
   - Avoids bit-decompositions and polynomial approximations.
   - Mitigates proof overhead while maintaining accuracy.

Proposed Solution: zkLLM

Implementation Highlights

1. Efficient CUDA implementation.
2. Suitable for LLMs with up to 13 billion parameters.
3. Performance:
   - Proving times: 1-15 minutes.
   - Proof size: <200 kB.
   - Verification time: 1-3 seconds.

Key Technical Contributions

Handling Non-Arithmetic Operations

1. Addresses challenges with activation functions like GELU and SwiGLU.
2. Utilizes lookup arguments for efficient verification.

Refined Attention Mechanism Verification

1. Tackles challenges of multivariate Softmax functions in LLMs.
2. Tailored approach for layers where Softmax is extensively used.

Parallel Computing Optimization:

1. High-level parallelization using GPU resources to minimize runtime and memory usage.

Significance

1. zkLLM is the first ZKP protocol specifically designed for LLMs.
2. It balances accuracy, efficiency, and security, addressing the unique demands of modern LLMs.
3. Facilitates trust and transparency in AI applications, especially in legally sensitive scenarios.

Technical Overview - Key Requirements

Parallelization

1. Full utilization of CUDA for GPU acceleration.
2. Enables efficient handling of large-scale proofs.

Non-Arithmetic Operations

1. Addressing operations like GELU and SwiGLU.
2. Adapting existing ZKP methods for modern LLM structures.

Core Innovations

Tlookup

1. Verifiable non-arithmetic tensor operations.
2. Ensures efficient memory usage and supports high parallelization.

zkAttn

1. Tailored for attention mechanisms.
2. Reduces overhead while maintaining accuracy.
3. Handles multivariate Softmax operations efficiently.

Implementation Features

CUDA Optimization

1. Fully parallelized implementation.
2. Scalable to LLMs with up to 13 billion parameters.

Compact Proofs

1. Proof size: <200 kB.
2. Proving time: 1-15 minutes.
3. Verification time: 1-3 seconds.

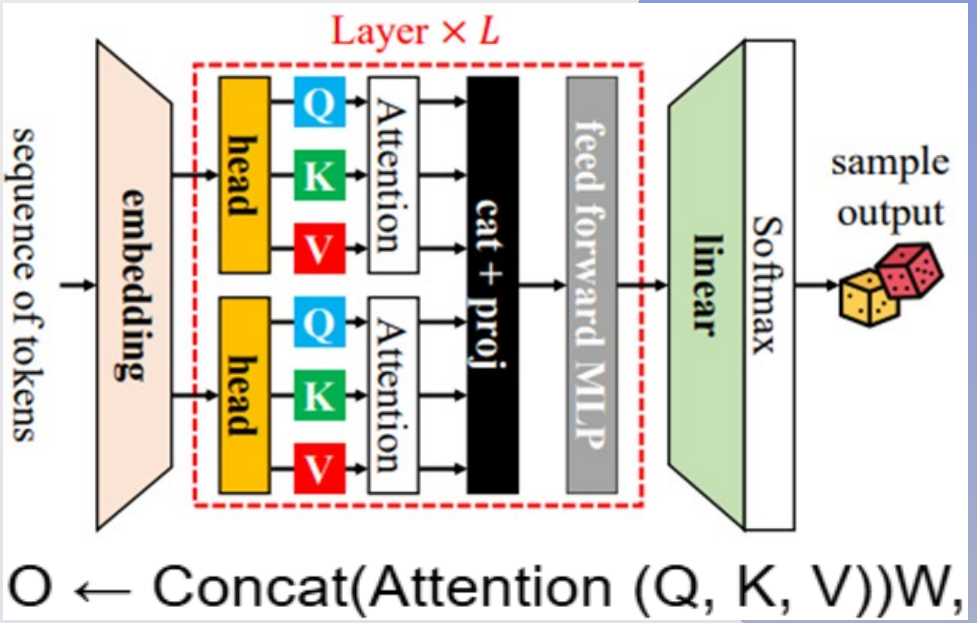
Applications

Key Benefits

1. Verifies LLM inference without exposing model details.
2. Supports legal and compliance use cases, such as verifying AI-generated outputs for regulatory requirements.

Large language models, transformers, and the attention mechanism

Exceptional performance in general-purpose language understanding and generation tasks.



BY 吳仲霖

What is the Sum Check Protocol?

An interactive proof protocol used to verify the correctness of summation for multivariate polynomials.

$$S = \sum_{x \in \{0,1\}^n} f(x)$$

$$f(x_1, \ldots, x_n) \xrightarrow{\sum x_i \in \{0,1\}} S$$

$$g_1(x_1) = \sum_{a_2, \ldots, a_n} f(x_1, a_2, \ldots, a_n)$$

The verifier wants to know:
Whether S equals the value S* claimed by the prover, without having to traverse all input combinations personally.

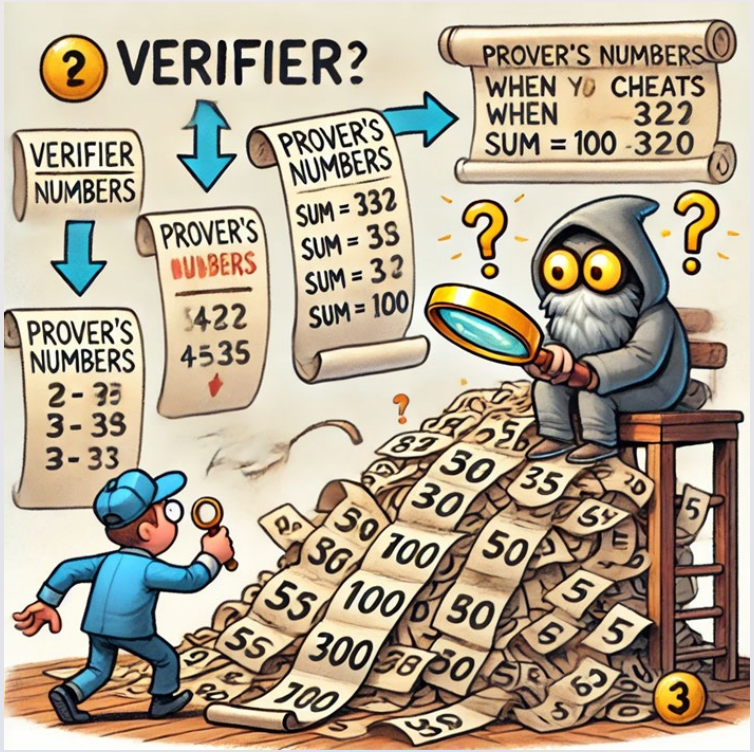
The prover claims:

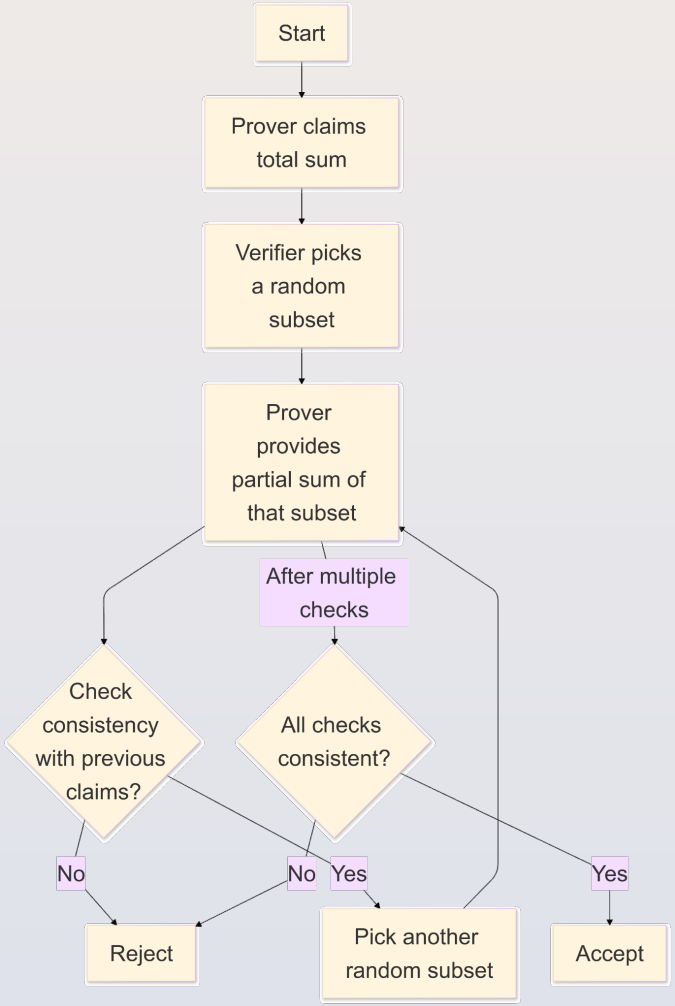
$$g_1(0) + g_1(1) = S$$

The verifier randomly selects r1 and requests the prover to provide the correct value of g1(r1). If correct, the problem is transformed into verifying the summation properties of g1(r1).

BY 吳仲霖

SumCheck flowchart





BY 吳仲霖

SumCheck Features

Interactive Process

Reduces the success rate of the prover's deception through the verifier's random queries.

Efficient verification

Eliminates the need to compute the total sum of all inputs.

Low error probability

It is difficult for the prover to pass multiple random checks if they are deceitful.

BY 吳仲霖

What is the Polynomial Commitment?

Definition: A cryptographic commitment mechanism for polynomials.

Committee ‘s Role:

- Commits to a specific polynomial f(x).
- Opens the commitment later to prove its validity.

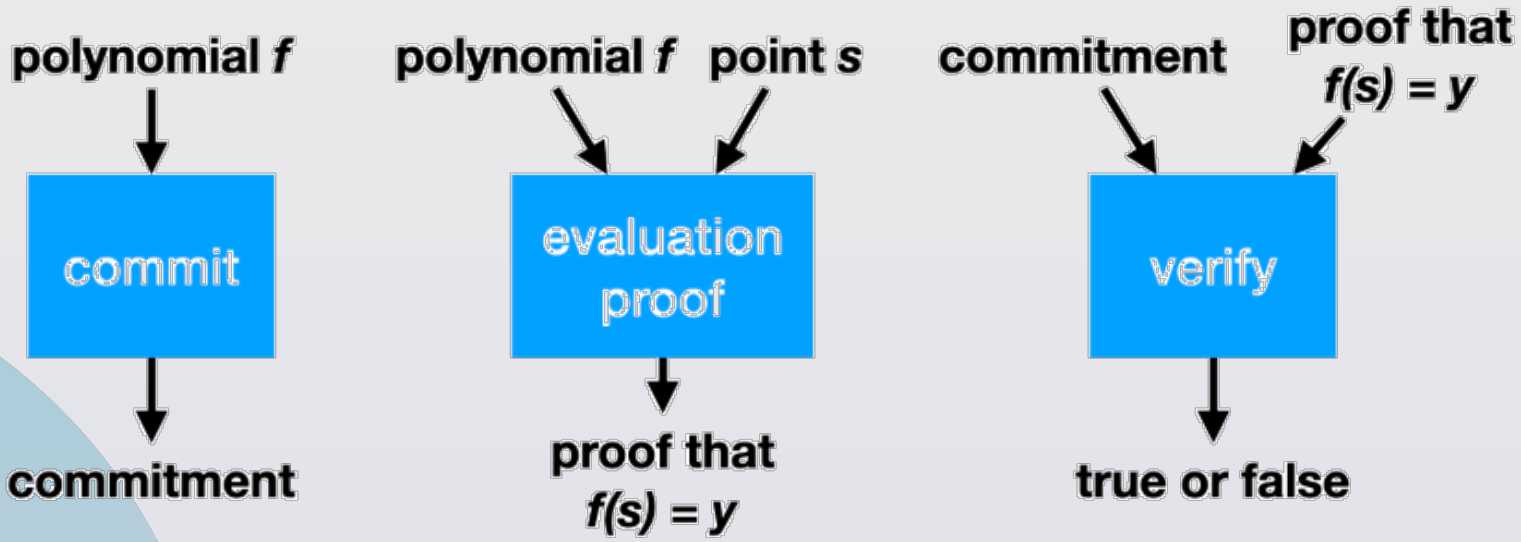
Verifier's Role:

- Confirms the committed polynomial without seeing the full polynomial.
- Queries any point xxx and verifies the correctness of f(x) at that point with a proof.

Key Feature: Efficient and secure verification of polynomial values.

BY 吳仲霖

Polynomial Commitment flowchart



BY 吳仲霖

Polynomial Commitment Features

Hiding & Binding

Once the polynomial is committed, it cannot be easily altered; the full details of the polynomial are not exposed.

Compact Proof

The proof is small in size, with short verification times.

BY 吳仲霖

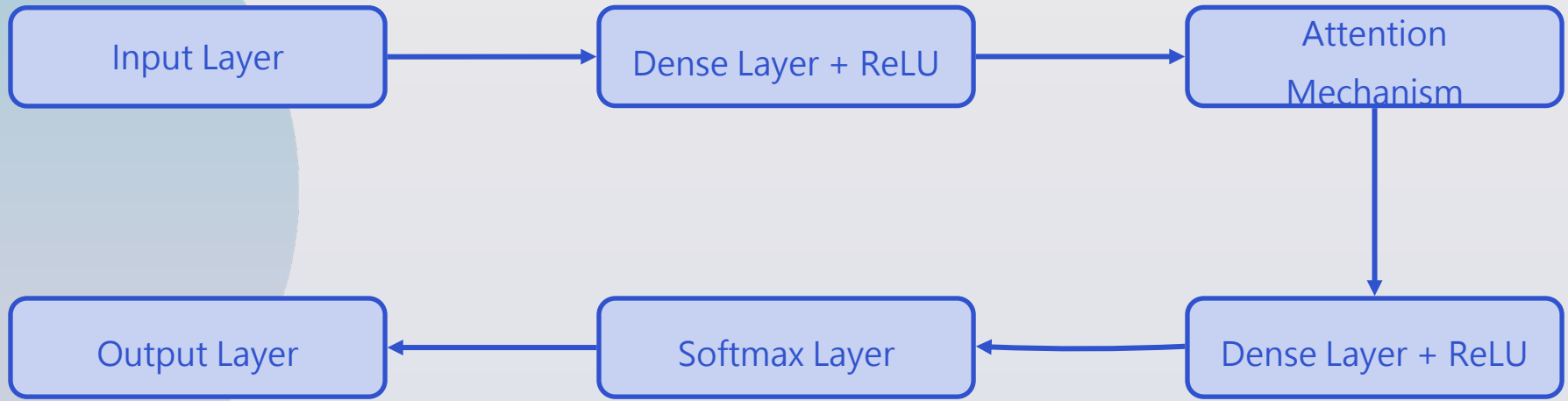
Comparison Table

Aspect	Polynomial Commitment	SumCheck Protocol
Core Function	Commits to a specific polynomial and efficiently verifies its value at a given point.	Verifies the correctness of the sum of a multivariate polynomial over its entire domain through an interactive process.
Interactivity	Can be non-interactive or require minimal interaction.	Fundamentally a multi-round interactive proof.
Verification	Verifies the correctness of f(x) at any point x=a, ensuring f(a) is accurate.	Verifies whether equals the claimed value.
Technical Focus	Compresses polynomial information into small commitments and point-value proofs.	Uses randomness and phased checks to reduce the success rate of fraudulent proofs.

BY 吳仲霖

Tlookup Background Introduction

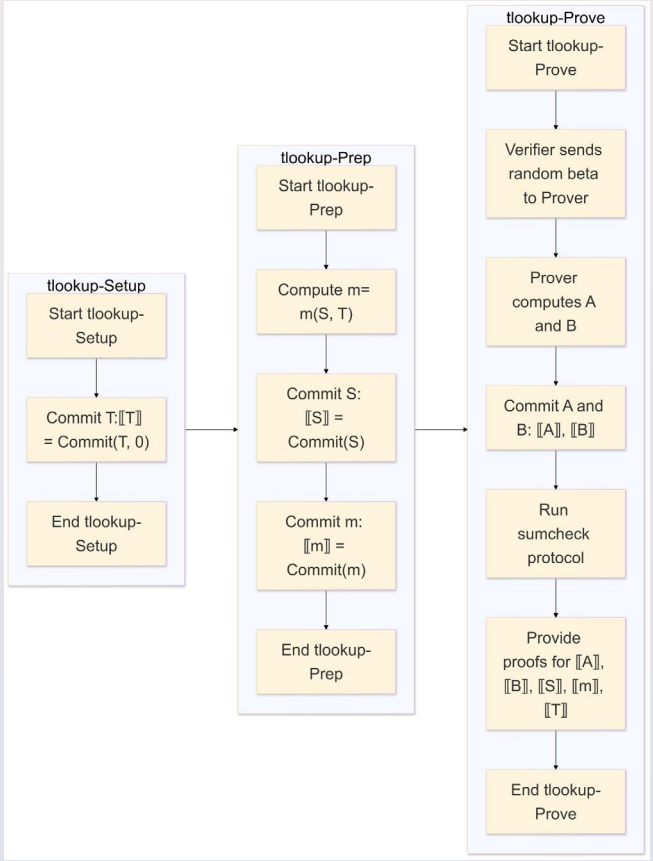
- Preserves the characteristics and advantages of tensor structures.
- Is compatible with existing deep learning computation frameworks, making it easier to implement Zero-Knowledge Proofs in mainstream frameworks.



BY 吴仲霖

The flowchart of Tlookup

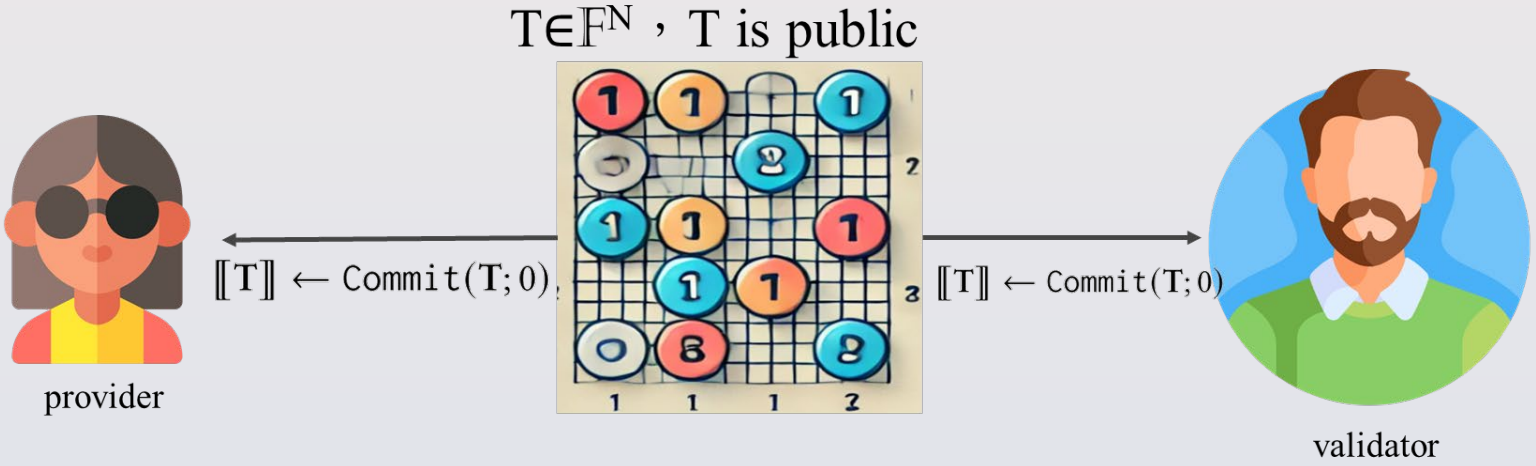
- Create a table T containing all tensors.
- Ensure that the tensor S owned by the provider must exist in the table.
- Transform the computation of non-arithmetic operations into table lookups.



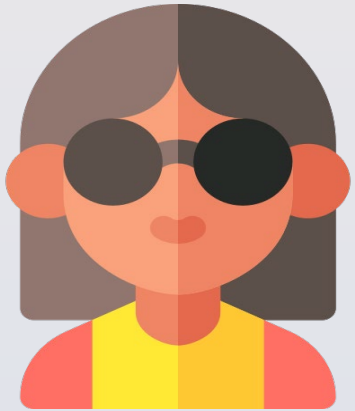
BY 吳仲霖

1st tlookup-Setup

N, D are both powers of 2 such that N divides D.



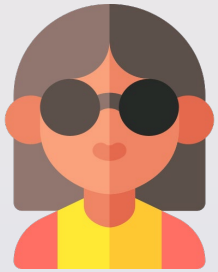
I have S · S∈F^D and S⊆T, S can’ t let validator know value is or any information



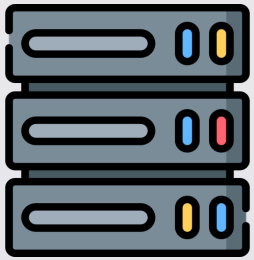
I want to verify the value S you say you have is correct

BY 吳仲霖

2nd tlookup-Setup



input S∈ F^D , T∈ F^N



Commit(S),

Commit(m)

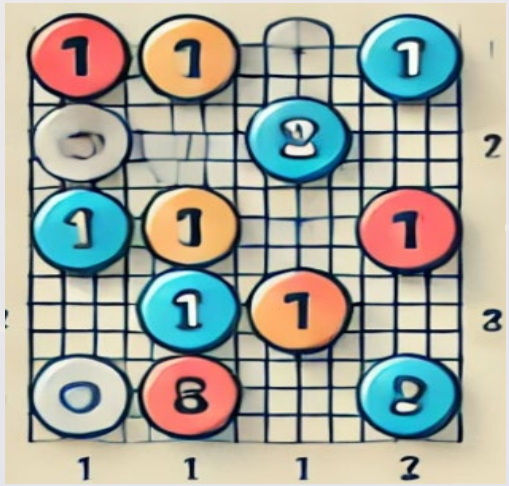
Calculate m:m←m(S,T)

$$\mathbf{m}_i \leftarrow \left|\left\{j : \mathbf{S}_j = \mathbf{T}_i\right\}\right|, \text{ for } 0 \le i \le N-1.$$

Gen Commit(S), Commit(m)

3rd tlookup-Setup

⟦S⟧, ⟦m⟧ , ⟦T⟧



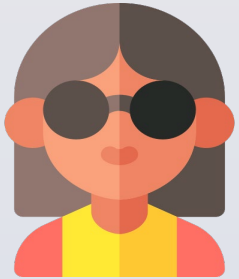
Random choose β

β~F

β

Commit(A),

Commit(B)



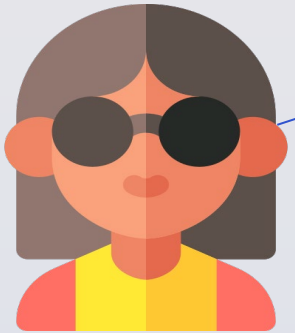
Calculate A, B

$$\mathbf{A} := \left(\frac{1}{\beta + \mathbf{S}_i}\right)_{i=0}^{D-1}, \mathbf{B} := \left(\frac{1}{\beta + \mathbf{T}_i}\right)_{i=0}^{N-1}$$

BY 吳仲霖

Verification Procedure

⟦A⟧ , ⟦B⟧ , ⟦S⟧ , ⟦m⟧ · ⟦T⟧



RUN

SUM-CHECK

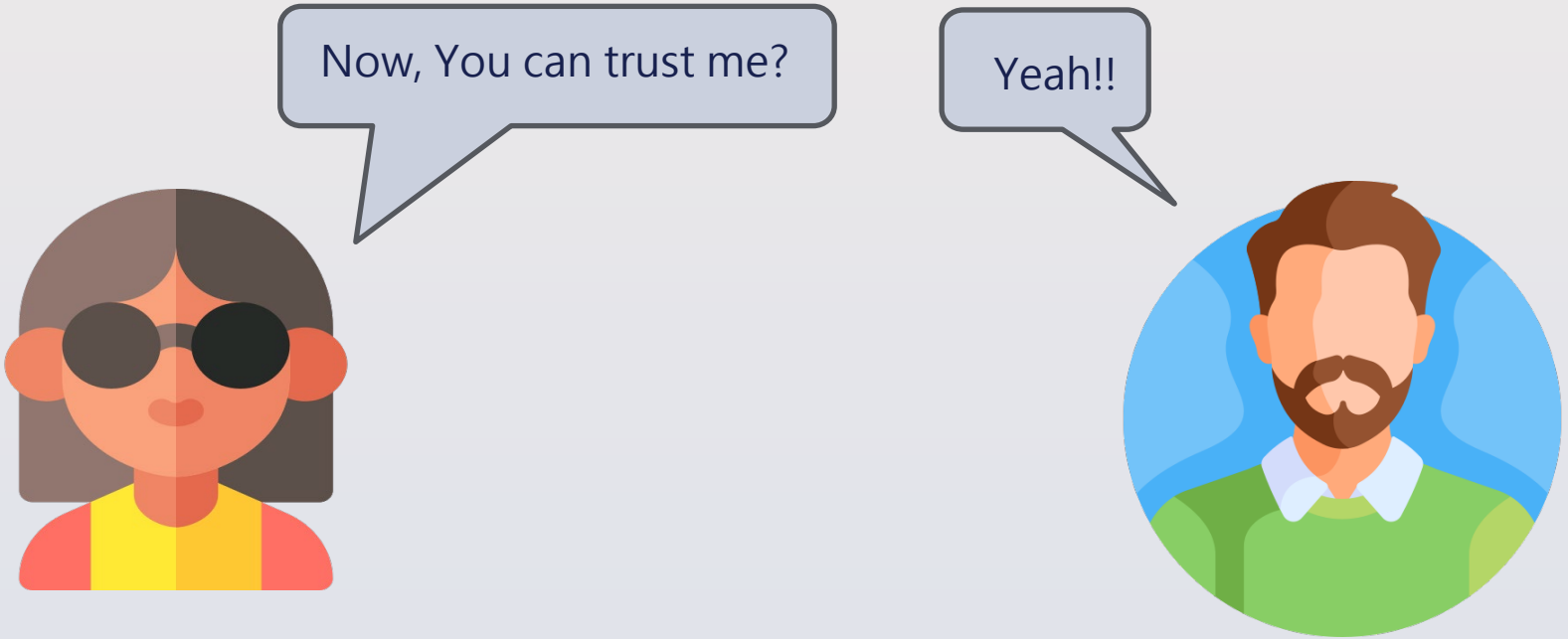
If S⊆T

$$\sum_{i=0}^{D-1} A_i = \sum_{j=0}^{N-1} m_j B_j$$

$$\alpha + \alpha^2 = \sum_{i \in [\frac{D}{N}]} \sum_{j \in [N]} \left(\widetilde{\mathbf{A}}(\mathbf{i} \oplus \mathbf{j}) \left(\alpha \widetilde{\mathbf{e}}(\mathbf{u}, \mathbf{i} \oplus \mathbf{j}) \left(\widetilde{\mathbf{S}}(\mathbf{i} \oplus \mathbf{j}) + \beta \right) + 1 \right) \right.$$
$$\left. + ND^{-1} \widetilde{\mathbf{B}}(\mathbf{j}) \left(\alpha^2 \widetilde{\mathbf{e}} \left(\mathbf{u}_{[\log_2 \frac{D}{N}:]}, \mathbf{j} \right) \left(\widetilde{\mathbf{T}}(\mathbf{j}) + \beta \right) - \widetilde{\mathbf{m}}(\mathbf{j}) \right) \right).$$

BY 吳仲霖

Final result (Repeat several times)



How tlookup used on Example: ReLU

- γ represents the scaling factor used in the system
- assume that $-\frac{B}{2} \leq \left\lfloor \frac{Z}{\gamma} \right\rceil < \frac{B}{2}$
- B is a positive even integer
- Prover demonstrate to the verifier
  α is random

$$\mathbf{A} \leftarrow ReLU(\mathbf{Z}) = \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rceil \odot \mathbb{1}\left\{ \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rceil \geq 0 \right\}$$

decomposed

$$\mathbf{Z}' := \left\lfloor \frac{\mathbf{Z}}{\gamma} \right\rceil \qquad \mathbf{R} = \mathbf{Z} - \gamma \mathbf{Z}'$$

input-output lookup tables

additional lookup table

$$\mathbf{T}_{\mathcal{X}} := \left[-\tfrac{B}{2}, \tfrac{B}{2} - 1\right] \qquad \mathbf{T}_{\mathcal{R}} = \left[-\tfrac{\gamma}{2}, \tfrac{\gamma}{2} - 1\right]$$

$$\mathbf{T}_{\mathcal{Y}} := \mathbf{T}_{\mathcal{X}}^{+}$$

$$\mathbf{Z}' + \alpha\mathbf{A} \subset \mathbf{T}_{\mathcal{X}} + \alpha\mathbf{T}_{\mathcal{Y}} \qquad \mathbf{R} \subset \mathbf{T}_{\mathcal{R}}$$

Proving $\mathbf{Z} = \gamma\mathbf{Z}' + \mathbf{R}$ correctness

Advantage

Method	Memory Overhead	Time Overhead
Traditional Method (single lookup table)	$O(B\gamma)$	$O(B\gamma)$
Using Two Lookup Tables	$O(B+\gamma)$	$O(B+\gamma)$

Verifiable Tensor Operations

- Matrix multiplication
- Activation function
- Normalization

Matrix multiplications

Sumcheck Protocol

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

Efficiency

Avoids recalculating all elements in AB. Suitable for LLM-scale computations.

Scalability

Handles the large matrix sizes typical in LLMs.

Correctness Assurance

Ensures high confidence in AB correctness without full re-computation.

Activation function

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

$$\mathrm{Swish}_\beta(z) := z \cdot \mathrm{Sigmoid}(\beta z)$$

$$\mathrm{Softmax}\begin{pmatrix} z \\ 0 \end{pmatrix} = \begin{pmatrix} \mathrm{Sigmoid}(z) \\ \mathrm{Sigmoid}(-z) \end{pmatrix}$$

Lower Overhead

Memory Efficiency

Better Scalability

Normalization

Formula for LayerNorm

(non-linear)

$$y \leftarrow \frac{x - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

tlookup steps:

Step 1. Input Scaling:

Prepares the input(x) by reducing numerical instability and optimizing memory usage.

Step 2. Quantized Compound Operation:

Computes the LayerNorm output efficiently and accurately using preprocessed inputs.

BY 李以謙

Assembly of the proofs

To improve readability, zkLLM can be simplified into three main components (omitting additional commitments caused by tLookup):

Step 1. Commitment

$$[[W]] \leftarrow \text{zkLLM-Commit}(W, pp, r)$$

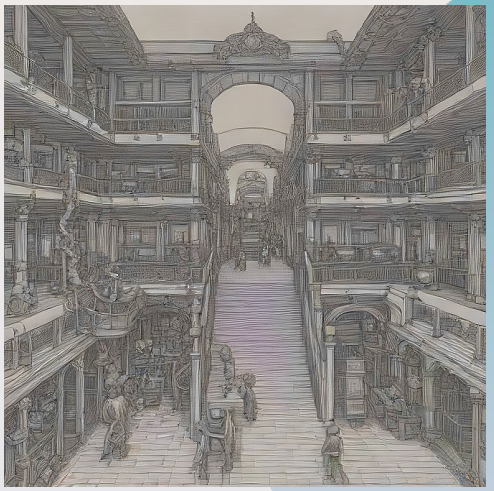
Step 2. Proof Generation

$$(y, \pi) \leftarrow \text{zkLLM-Prove}(W, X, pp, r)$$

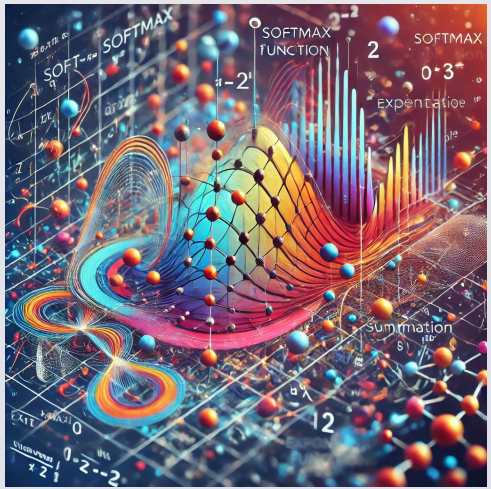
Step 3. Verification

$$b \leftarrow \text{zkLLM-Verify}(X, y, [[W]])$$

zkAttn



Dedicated ZKP for the attention mechanism in LLMS.



Issues is on **Softmax function**

- non-arithmetic operations
- Multivariate nature

BY 蕭耕宏

The attention mechanism

$$\mathbf{V} \in \mathbb{F}^{n \times d}, \quad \mathbf{K} \in \mathbb{F}^{n \times d}, \quad \mathbf{Q} \in \mathbb{F}^{m \times d}.$$

$$\Rightarrow \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} \right) \mathbf{V}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Z}}{\sqrt{d}} \right)$$

$$\text{where the input matrix } \mathbf{Z} = \mathbf{Q}\mathbf{K}^\top$$

zkAttn Formulation

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Z}}{\sqrt{d}}\right)$$

$$\text{where the input matrix } \mathbf{Z} = \mathbf{Q}\mathbf{K}^\top$$

$$z = (Z_0, Z_1, \ldots, Z_{n-1})$$

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

Devise an algorithm that computes s(z) in the real domain.

zkAttn Formulation

How to have manageable proof generation?

Use the **shift-invariance** property of Softmax

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

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

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

zkAttn Formulation

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

$$\text{How to approximate } \theta \exp\left(\frac{\cdot}{\gamma\sqrt{d}}\right)?$$

Observation

$$\Rightarrow \quad z_i - \hat{z} \leq 0$$

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

logarithm of a sum of exponentials.

Assumption 1

$$z_i - \hat{z} \text{ are contained within } (-B, 0]$$

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

Assumption 2

$$\gamma \ll B \ll |\mathbb{F}|$$

zkAttn Formulation

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

A bijection can be established

$$\mathbf{b} : \prod_{k=0}^{K-1} \left[b^{(k)} \right] \to [B]$$

zkAttn Formulation

$$\mathbf{b}\left(x^{(0)}, x^{(1)}, \ldots, x^{(K-1)}\right) = \sum_{k=0}^{K-1} x^{(k)} B^{(k)}$$

$$\text{for each } \left(x^{(0)}, x^{(1)}, \ldots, x^{(K-1)}\right) = \mathbf{b}^{-1}(x)$$

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

zkAttn Formulation

Taking into account the scaling factor gamma $\exp\left(-\frac{x}{\gamma\sqrt{d}}\right)$

$$\text{further decompose } \gamma \text{ as } \theta = \prod_{k=0}^{K-1} \theta^{(k)}$$

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

zkAttn Formulation

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

$$\mathcal{T}_{\chi}^{(k)} := \left[b^{(k)}\right], \quad \mathcal{T}_{y}^{(k)} := \left(\left\lfloor \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x\right) \right\rfloor\right)_{x \in \left[b^{(k)}\right]}$$

zkAttn Formulation

$$\mathbf{b}\left(x^{(0)}, x^{(1)}, \ldots, x^{(K-1)}\right) = \sum_{k=0}^{K-1} x^{(k)} B^{(k)}$$

$$\mathcal{T}_{\chi}^{(k)} := \left[b^{(k)}\right], \quad \mathcal{T}_{y}^{(k)} := \left(\left\lfloor \theta^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}} x\right)\right\rfloor\right)_{x \in \left[b^{(k)}\right]}$$

$$y \leftarrow \prod_{k=0}^{K-1} y^{(k)}$$

zkAttn Formulation

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

$$\left(x^{(k)}, y^{(k)}\right) \in \mathcal{T}^{(k)}, \quad \forall\, 0 \le k \le K-1, \prod_{k=0}^{K-1} y^{(k)} = y.$$

Experiments

Development

- zkLLM was developed using CUDA, leveraging the BLS12-381 curve from the EC-GPU package.
- Sequential verifier tasks were adapted from zkCNN, utilizing the mcl package.

Model and Dataset

- Tested on two open-source LLMs: OPT and LLaMa-2.
- Supported up to 13 billion parameters.
- Used samples from the C4 dataset, with a default sequence length of 2048.

Hardware Resources

- Memory: 124.5 GB.
- CPU: 12 cores of AMD EPYC 7413 (2.65 GHz, 128M L3 cache).
- GPU: NVIDIA A100 SMX4 with 40 GB memory.

Experiments

Scaling Factor: Set to 2^16 for data and parameters.

tLookups

- 5 segments (K=5), each size 2^16.
- Precision maintained at 2^-16.

Error Control

- Tolerable L1 error: 10^-2.
- Comparable to rounding errors in modern LLMs.

Experiments

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

Experiments

Commit Efficiency

The model commit is completed in 20 minutes, generating approximately 10MB of commitment data.

Inference Proof

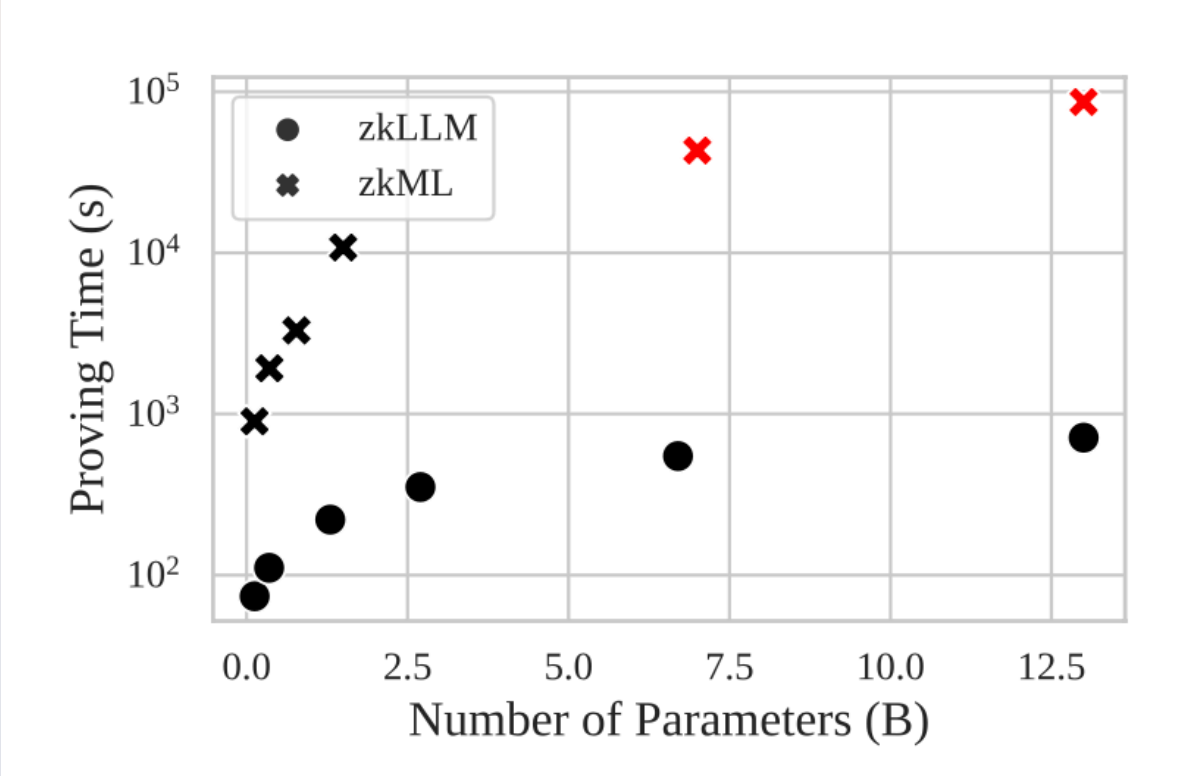
The correctness proof for the inference process is generated in less than 15 minutes.

Memory Optimization: Memory consumption is controlled under 23.1GB, making it compatible with commonly used GPUs such as V100 and A100.

Perplexity Impact

On the C4 dataset, quantization increases perplexity by under 0.01 for 13B models, ensuring high accuracy.

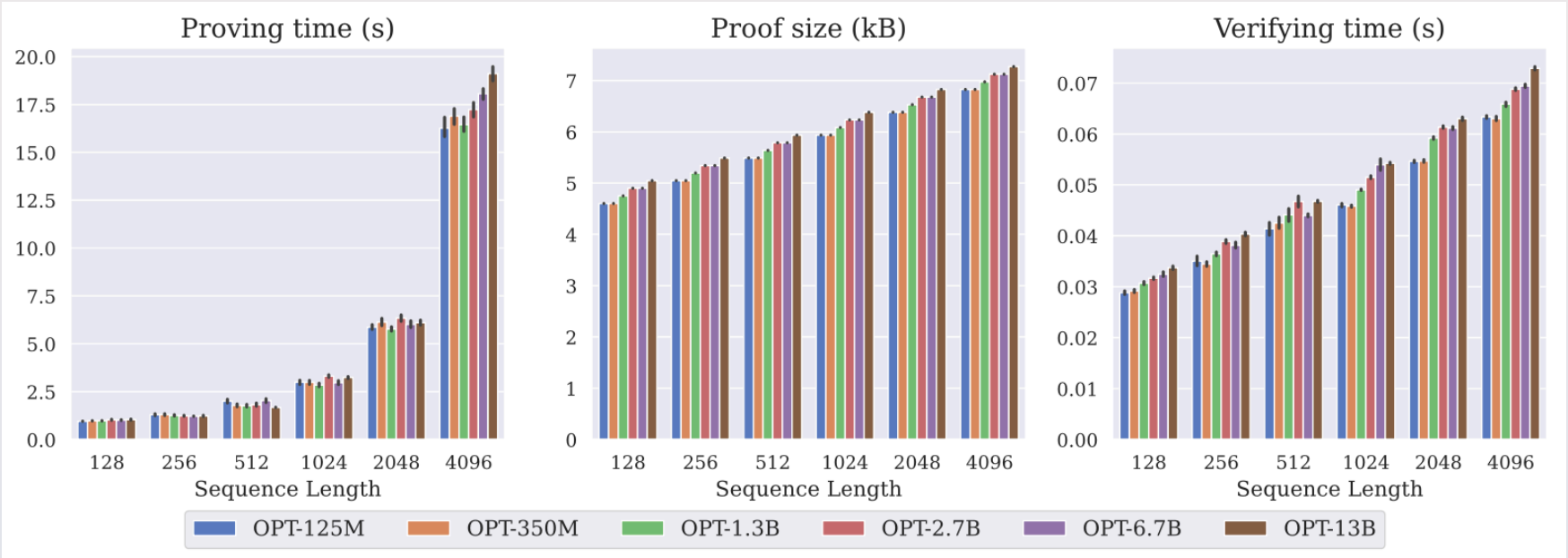
Experiments



10x larger sizes

50x speedup

Experiments



Analysis

1. Error Analysis on zkAttn

   - Examine the error by zkAttn.

2. Security & Privacy Analysis

   - Facilitates the verifiable computation of lookup protocol across all non-arithmetic components.

3. Overhead Analysis

   - Analyze the running times, memory, and communication costs of the prover and verifier.

7.1. Error Analysis on zkAttn

The errors of zkAttn are from 2 sources:

1. The rounding of the shifting fact $\hat{z} := \gamma\sqrt{d}\ln\left(\sum_{j=0}^{n-1}\exp\left(\frac{z_i}{\gamma\sqrt{d}}\right)\right),$

2. The rounding of each segment encoded in lookup tables $\mathbf{T}_X^{(k)} := \left[b^{(k)}\right], \quad \mathbf{T}_y^{(k)} := \left(\left\lfloor \theta^{(k)}\exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}}x\right)\right\rfloor\right)_{x\in\left[b^{(k)}\right]}$

7.1. Error Analysis on zkAttn

$$\textbf{procedure } \textsc{zkAttn-Setup}\left(B, K, M, L, \left(b^{(k)}\right)_{k=0}^{K-1}, \gamma, \theta, (\theta_K)_{k=L}^{K-M-1}, m, n, d, E\right)$$

The Lower Bound of Input and the Factorizatio $B = \prod_{k=0}^{K-1} b^{(k)}$

Total Numbers of Segments K

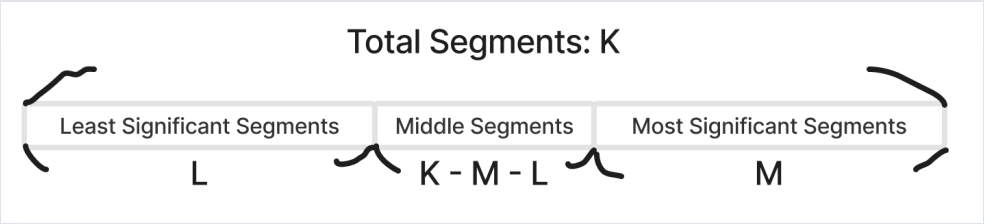
Most Significant Segments: M

Lease Significant Segments: L

Scaling Factor of Input, Output, Each Segment $\gamma, \theta, (\theta_K)_{k=L}^{K-M-1}$

Dimension Parameter of the Input: $m, n, d,$

Tolerable Error in Row-wise Normalization: E



7.1. Error Analysis on zkAttn

procedure zkAttn-Setup$(B, K, M, L, \left(b^{(k)}\right)_{k=0}^{\cdot}, \gamma, \theta, (\theta_K)_{k=L}^{K-M-1}, m, n, d, E)$

  for $0 \le k \le K-1$ **do**

    $\left[\!\left[\mathrm{T}_{\mathcal{X}}^{(k)}\right]\!\right] \leftarrow$ tlookup-Setup$(\mathrm{T}_{\mathcal{X}}^{(k)})$ ▷ $\mathrm{T}_{\mathcal{X}}^{(k)} = \left[b^{(k)}\right]$, i.e., the input of the k-th segment

  end for

  for $L \le k \le K-M-1$ **do**

    $\left[\!\left[\mathrm{T}_{\mathcal{Y}}^{(k)}\right]\!\right] \leftarrow$ tlookup-Setup$(\mathrm{T}_{\mathcal{Y}}^{(k)})$ ▷ $\mathrm{T}_{\mathcal{Y}}^{(k)}$ as defined in (22), i.e., the output of the k-th segment

  end for

  for $K-M \le k \le K-1$ **do**

    $\left[\!\left[\mathrm{T}_{\mathcal{Y}}^{(k)}\right]\!\right] \leftarrow$ tlookup-Setup$(\mathrm{T}_{\mathcal{Y}}^{(k)})$ ▷ $\mathrm{T}_{\mathcal{Y}}^{(k)} = \mathbb{1}\left\{\left[b^{(k)}\right] = 0\right\}$, i.e., the optimized output of the k-th segment

  end for

  $[\![\mathrm{T}_{\mathcal{R}}]\!] \leftarrow$ tlookup-Setup$(\mathrm{T}_{\mathcal{R}})$ ▷ $\mathrm{T}_{\mathcal{R}} = [\theta - E, \theta + E]$, i.e., all tolerable values of row-wise sum of the output

  return $\left(\left[\!\left[\mathrm{T}_{\mathcal{X}}^{(k)}\right]\!\right]\right)_{k=0}^{K-1}, \left(\left[\!\left[\mathrm{T}_{\mathcal{Y}}^{(k)}\right]\!\right]\right)_{k=L}^{K-1}, [\![\mathrm{T}_{\mathcal{R}}]\!]$

end procedure

	Least Significant Segments	Middle Segments	Most Significant Segments
tlookup-Setup$(\mathrm{T}_{\mathcal{X}}^{(k)})$	$\mathrm{T}_{\mathcal{X}}^{(k)} = \left[b^{(k)}\right]$	$\mathrm{T}_{\mathcal{X}}^{(k)} = \left[b^{(k)}\right]$	$\mathrm{T}_{\mathcal{X}}^{(k)} = \left[b^{(k)}\right]$
tlookup-Setup$(\mathrm{T}_{\mathcal{Y}}^{(k)})$	$\mathrm{T}_{\mathcal{Y}}^{(k)} = 0$	$\mathrm{T}_{\mathcal{Y}}^{(k)} := \left(\left\lfloor b^{(k)} \exp\left(-\frac{B^{(k)}}{\gamma\sqrt{d}}x\right)\right\rfloor\right)_{x \in [b^{(k)}]}$	$\mathrm{T}_{\mathcal{Y}}^{(k)} = \mathbb{1}\left\{\left[b^{(k)}\right] = 0\right\}$

BY 王煥昇

7.1. Error Analysis on zkAttn

14: **procedure** $\mathcal{P}$.zkAttn-Compute($\mathbf{Z} \in \mathbb{F}^{m \times n}$, $\left(\mathbf{T}_{\mathcal{X}}^{(k)}\right)_{k=0}^{K-1}$, $\left(\mathbf{T}_{\mathcal{Y}}^{(k)}\right)_{k=L}^{K-1}$) ▷ Some implicit parameters included in Setup(·) omitted

15: $\mathbf{Z}' \leftarrow \mathbf{Z} - \lfloor \hat{\mathbf{z}} \rceil \mathbf{1}^\top$, where $\hat{\mathbf{z}} \in \mathbb{R}^m$ is computed row-wise as (17)

16: $\left(\mathbf{X}^{(k)}\right)_{k=0}^{K-1} \leftarrow \mathfrak{b}^{-1}(-\mathbf{Z}')$ ▷ $-\mathbf{Z}$ is decomposed elementwisely

17: **for** $k \leftarrow L, L+1, \ldots, K-1$ **do**

18: $\mathbf{Y}^{(k)} \leftarrow f^{(k)}(\mathbf{X}^{(k)})$ elementwisely, where $f^{(k)}$ is defined by $\mathbf{T}_{\mathcal{X}}^{(k)}, \mathbf{T}_{\mathcal{Y}}^{(k)}$

19: **end for**

20: $\mathbf{Y} \leftarrow \bigodot_{k=L}^{K-1} \mathbf{Y}^{(k)}$ ▷ Compute the final output

21: $\hat{\mathbf{y}} \leftarrow \mathbf{Y}.\mathrm{sum}(\mathrm{axis} = 1)$ ▷ For checking the normalization of each row

22: $\mathcal{P} \to \mathcal{V} : [\![\mathbf{Z}]\!], [\![\lfloor \hat{\mathbf{z}} \rceil]\!], \left([\![\mathbf{X}^{(k)}]\!]\right)_{k=0}^{K-1}, [\![\mathbf{Y}]\!], \left([\![\mathbf{Y}^{(k)}]\!]\right)_{k=L}^{K-1}, [\![\hat{\mathbf{y}}]\!]$

23: **end procedure**

7.1. Error Analysis on zkAttn

procedure $\langle \mathcal{P}, \mathcal{V} \rangle$.zkAttn-Prove($[\![\mathbf{Z}]\!], [\![\lfloor \hat{\mathbf{z}} \rceil]\!], \left([\![\mathbf{X}^{(k)}]\!]\right)_{k=0}^{K-1}, [\![\mathbf{Y}]\!], \left([\![\mathbf{Y}^{(k)}]\!]\right)_{k=L}^{K-1}, [\![\hat{\mathbf{y}}]\!]$)

  for $k \leftarrow 0, 1, \ldots, K-1$ **do**

    $\mathcal{P}$.tlookup-Prep($\mathbf{X}^{(k)}, \mathbf{T}_{\mathcal{X}}^{(k)}$) ▷ $[\![\mathbf{m}^{(k)} := \mathbf{m}\left(\mathbf{X}^{(k)}, \mathbf{T}_{\mathcal{X}}^{(k)}\right)]\!]$ transmitted to $\mathcal{V}$

    $\langle \mathcal{P}, \mathcal{V} \rangle$.tlookup-Prove($[\![\mathbf{X}^{(k)}]\!], [\![\mathbf{m}^{(k)}]\!], [\![\mathbf{T}_{\mathcal{X}}^{(k)}]\!]$) ▷ Prove the correctness on the k-th segment

  end for

  $\mathcal{V} \rightarrow \mathcal{P} : \alpha \sim \mathbb{F}$

  for $k \leftarrow L, L+1, \ldots K-1$ **do**

    $\mathcal{P}$.tlookup-Prep($\mathbf{X}^{(k)} + \alpha \mathbf{Y}^{(k)}, \mathbf{T}_{\mathcal{X}}^{(k)} + \alpha \mathbf{T}_{\mathcal{Y}}^{(k)}$) ▷ $[\![\mathbf{m}^{(k)} := \mathbf{m}\left(\mathbf{X}^{(k)} + \alpha \mathbf{Y}^{(k)}, \mathbf{T}_{\mathcal{X}}^{(k)} + \alpha \mathbf{T}_{\mathcal{Y}}^{(k)}\right)]\!]$ transmitted to $\mathcal{V}$

    $\langle \mathcal{P}, \mathcal{V} \rangle$.tlookup-Prove($[\![\mathbf{X}^{(k)}]\!] + \alpha [\![\mathbf{Y}^{(k)}]\!], [\![\mathbf{m}^{(k)}]\!], [\![\mathbf{T}_{\mathcal{X}}^{(k)}]\!] + \alpha [\![\mathbf{T}_{\mathcal{Y}}^{(k)}]\!]$) ▷ Prove the correctness on the k-th segment

  end for

  $\mathcal{P}$.tlookup-Prep($\hat{\mathbf{y}}, \mathbf{T}_{\mathcal{R}}$) ▷ $[\![\mathbf{m}_{\mathcal{R}} := \mathbf{m}(\mathbf{Y}, \mathbf{T}_{\mathcal{R}})]\!]$ transmitted to $\mathcal{V}$

  $\langle \mathcal{P}, \mathcal{V} \rangle$.tlookup-Prove($[\![\hat{\mathbf{y}}]\!], [\![\mathbf{m}_{\mathcal{R}}]\!], [\![\mathbf{T}_{\mathcal{R}}]\!]$) ▷ Prove the correctness on the k-th segment

  $\mathcal{P}$ and $\mathcal{V}$ run the sumcheck for $\mathfrak{b}(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{K-1}) + \mathbf{Z}' = \mathbf{0} \wedge \mathbf{Z}' = \mathbf{Z} - \lfloor \hat{\mathbf{z}} \rceil \mathbf{1}^\top \wedge \mathbf{Y} = \bigodot_{k=L}^{K-1} \mathbf{Y}^{(k)} \wedge \hat{\mathbf{y}} = \mathbf{Y}.\mathrm{sum}(\mathrm{axis} = 1)$, followed by the proof of evaluations on $[\![\mathbf{Z}]\!], [\![\lfloor \hat{\mathbf{z}} \rceil]\!], \left([\![\mathbf{X}^{(k)}]\!]\right)_{k=0}^{K-1}, [\![\mathbf{Y}]\!], \left([\![\mathbf{Y}^{(k)}]\!]\right)_{k=L}^{K-1}, [\![\hat{\mathbf{y}}]\!]$

end procedure

7.1. Error Analysis on zkAttn

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

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

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

Conclusion

1. Minimizing $K-M-L$ and B_L can reduce error.
2. ε_{attn} has no dependence on the segmentation except B_L and B_{K-M}
3. ε_{attn} defines the tolerable error row-wise sum upon checking the normalization and it must be in $[(1-\varepsilon_{\text{attn}})\theta, (1+\varepsilon_{\text{attn}})\theta]$

BY 王煥昇

7.2 Security & Privacy Analysis

pp: Public Parameter

W: Private Parameter

com: Commitment of Encryption, [W]

X: Query

A: Adversary

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

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

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

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

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

7.3 Overhead Analysis

1. tlookup overhead
   - N = O(D),
     N: total size of tensor
     D: dimensions of the tensor
   Committing + Proving Commitment + SumCheck Protocols: $O(D)$
   Memory Requirement: $O(D)$
   Committing: $O(\sqrt{D})$
   Proving: $O(\log D)$
   SumCheck: $O(\log D)$

Component	Prover Overhead	Verifier Overhead	Communication Overhead
tlookup	$O(D)$	$O(\log D)$	$O(\sqrt{D})$
zkAttn	$O(K \cdot m \cdot n)$	$O(K\sqrt{m \cdot n})$	$O(K\sqrt{m \cdot n})$
zkLLM (Overall)	$O(L \cdot t_P)$	$O(\sqrt{L \cdot t_V})$	$O(\sqrt{L \cdot C})$

7.3 Overhead Analysis

Component	Prover Overhead	Verifier Overhead	Communication Overhead
tlookup	$O(D)$	$O(\log D)$	$O(\sqrt{D})$
zkAttn	$O(K \cdot m \cdot n)$	$O(K\sqrt{m \cdot n})$	$O(K\sqrt{m \cdot n})$
zkLLM (Overall)	$O(L \cdot t_P)$	$O(\sqrt{L \cdot t_V})$	$O(\sqrt{L \cdot C})$

2. zkAttn overhead

- K: the numbers of input segment
- m * n: the rows * columns of input segment

Prover Overhead: $O(Kmn)$

Verifier & Communication Overhead: $O(K\sqrt{mn})$

Which is better than bit-decomposition method: $\Omega(mn \log_2 B)$

The Time Reduction is: $O\left(\frac{K}{\log_2 B}\right)$

7.3 Overhead Analysis

Component	Prover Overhead	Verifier Overhead	Communication Overhead
tlookup	$O(D)$	$O(\log D)$	$O(\sqrt{D})$
zkAttn	$O(K \cdot m \cdot n)$	$O(K\sqrt{m \cdot n})$	$O(K\sqrt{m \cdot n})$
zkLLM (Overall)	$O(L \cdot t_P)$	$O(\sqrt{L \cdot t_V})$	$O(\sqrt{L \cdot C})$

3. Overall overhead (zkLLM overhead)

Prover Overhead: $O(Lt_{\mathcal{P}})$

Verifier Overhead: $O(\sqrt{L}t_{\mathcal{V}})$

Communication Overhead: $O(\sqrt{LC})$

Unlike univariate polynomial-based ZKP system that must be serialized, the use of sum check protocols over multilinear extensions allows highly parallelized proof generation, thereby enabling efficient proof generation.

7.3 Overhead Analysis

Component	Prover Overhead	Verifier Overhead	Communication Overhead
tlookup	$O(D)$	$O(\log D)$	$O(\sqrt{D})$
zkAttn	$O(K \cdot m \cdot n)$	$O(K\sqrt{m \cdot n})$	$O(K\sqrt{m \cdot n})$
zkLLM (Overall)	$O(L \cdot t_P)$	$O(\sqrt{L \cdot t_V})$	$O(\sqrt{L \cdot C})$

Conclusion

1. zkLLM achieves significant efficiency improvements for both **prover** and **verifier** compared to traditional methods.
2. zkAttn reduces overhead compared to bit-decomposition, particularly in large-scale inputs, making it more practical for real-world applications like LLM inference.
3. The overall system is designed to exploit parallelism, repetitive structures, and batching to minimize costs, especially for large L-layer models.

Conclusion

Innovative Solution

- zkLLM introduces the first Zero-Knowledge Proof (ZKP) system designed specifically for Large Language Models (LLMs).

Key Contributions

- tlookup: Efficiently handles non-arithmetic operations in deep learning.
- zkAttn: Tailored protocol for the attention mechanism, balancing accuracy, runtime, and memory usage.

Conclusion

Scalability and Efficiency

- Supports LLMs up to 13 billion parameters with compact proofs and fast verification.

Real-World Applications

- Ensures inference authenticity while protecting intellectual property.
- Facilitates compliance with AI regulations and ethical guidelines.

Takeaway: zkLLM is a groundbreaking step towards secure and transparent AI verifiability, combining computational rigor with practical efficiency.

Thank You

Member

110590005 蕭耕宏, 112C53035 王煥昇, 113C53006 吳仲霖,

113598043 張育丞, 113598088 李以謙