Secure Transformer Inference Made Non-interactive

Jiawen Zhang*, Xinpeng Yang*, Lipeng He‡, Kejia Chen*, Wen-jie Lu*,
Yinghao Wang*, Xiaoyang Hou*, Jian Liu*, Kui Ren*, Xiaohu Yang*

*Zhejiang University, ‡University of Waterloo

Secure ML Inference

Client

Input (Prompts/Images)

Sensitive Information

- Healthcare Data
- Financial Records
- Proprietary Business Data

Server

Model

- **Convolutional neural network (CNN)**
- **Transformer-based (BERT, GPT), etc.**

GPUs, FPGAs, etc.

$[X]_{\text{🔒}}$

$\not\Leftarrow \quad X \quad \Rightarrow \quad M \quad \Rightarrow$

$M(X)$

Transformer vs CNN Inference

In the context of non-interactive secure inference, there are two key differences between transformers and CNN models:

Larger Scale Matrix-Matrix Multiplications

Gazalle [USENIX Security '18], Cheetah [USENIX Security '22] and Iron compute matrix multiplications via **inner dot products**, and employ **sparse packing** for the resulting ciphertexts. The majority of the data slots in the ciphertexts are wasted.

> Input dimensions of matrices in transformer-based models are much higher, leading to **increased computational cost**.

Higher Dimensional Inputs to Argmax

Input to Argmax is a probability vector of size m. In CNN classification tasks, m is usually not large (m = 1,000 in ImageNet-1k). However, in transformer-based NLP tasks, m = vocabulary size, with m reaching 30,522 in BERT and 128,256 in Llama-3-8B.

> Previous SOTA Phoenix [CCS '22] can only achieves a **computational complexity of $O(m)$**

Prior Work: Interactive Protocols

Interactive **Secure Transformer Inference Protocols**

- Iron [NeurIPS '22]
- BOLT [IEEE S&P '24]
- BumbleBee [NDSS '25]

Based on Multi-party Secure Computation (MPC)

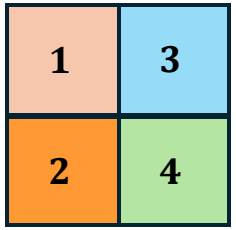
- **Substantial communication overhead**
- Unable to support hardware acceleration via GPU, FPGA, etc.

e.g. BOLT requires:

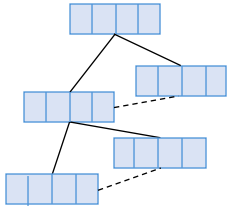
- **59.61 GB** of bandwidth
- **10,509** interaction rounds
- **$5.44** per response token

Our Work: NEXUS

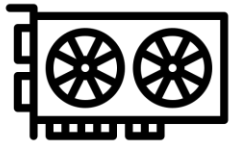
Based on RNS-CKKS **Fully Homomorphic Encryption (FHE)**, we achieve:



Efficient and Communication-optimized **Matrix Multiplication**



Efficient **Argmax** and other **non-linear function** evaluation

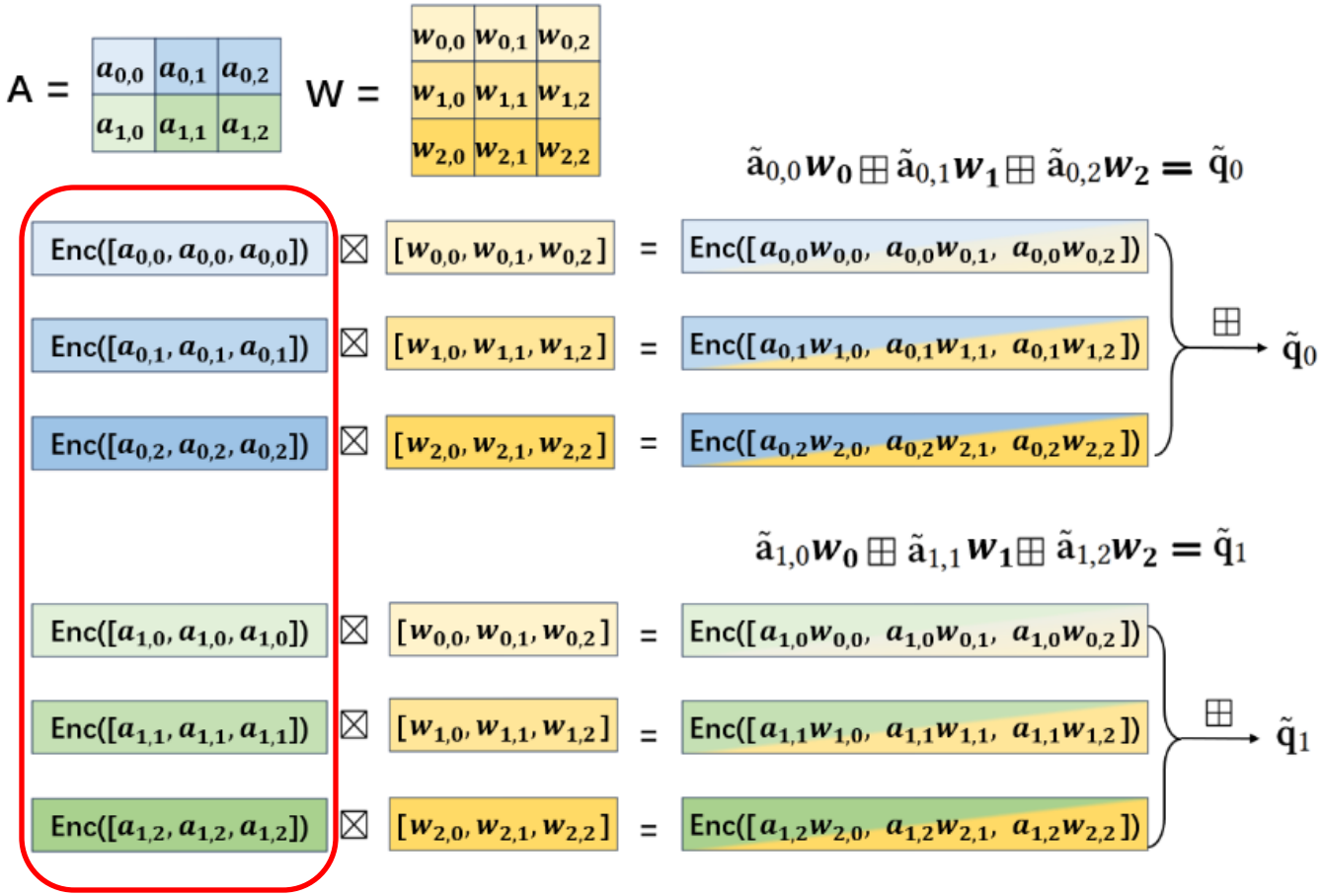


CPU/GPU Implementation

Efficient **Matrix Multiplication**

Toy Example

$\boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{W}_Q \in \mathbb{R}^{3\times 3},$

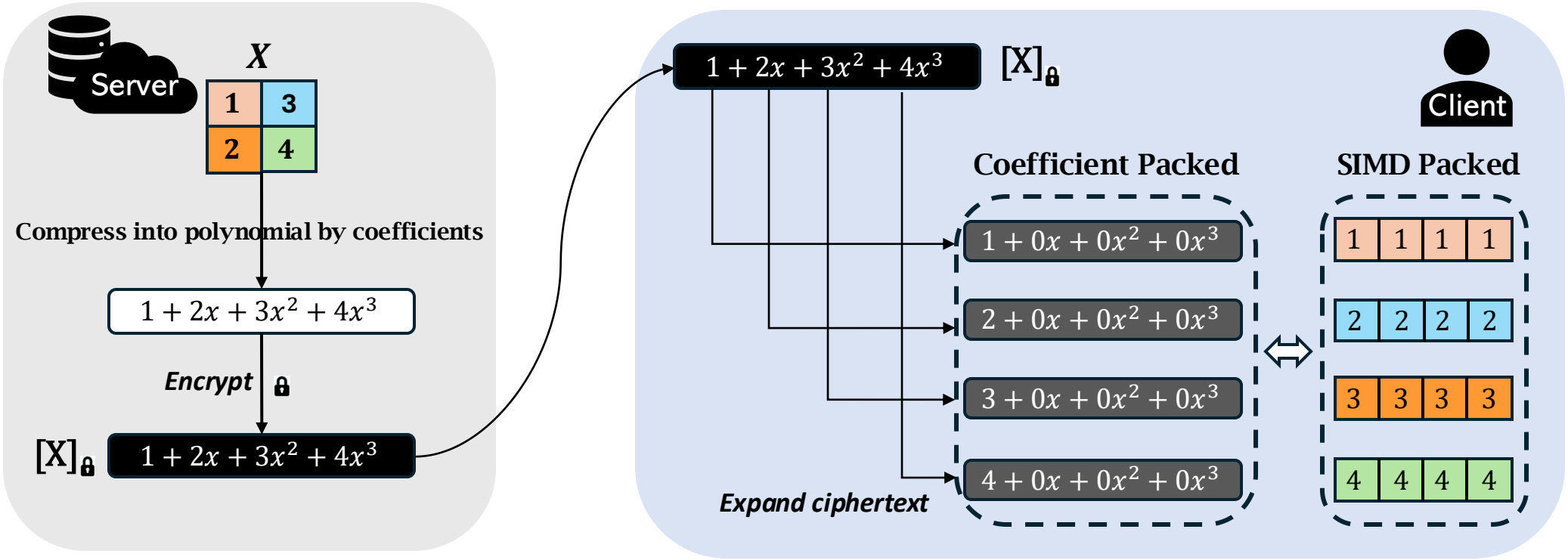


$\boldsymbol{Q} = \boldsymbol{A} \times \boldsymbol{W}_Q \in \mathbb{R}^{2\times 3}$

$\boldsymbol{q}_i \in \mathbb{R}^k$: i-th row of $\boldsymbol{Q}$

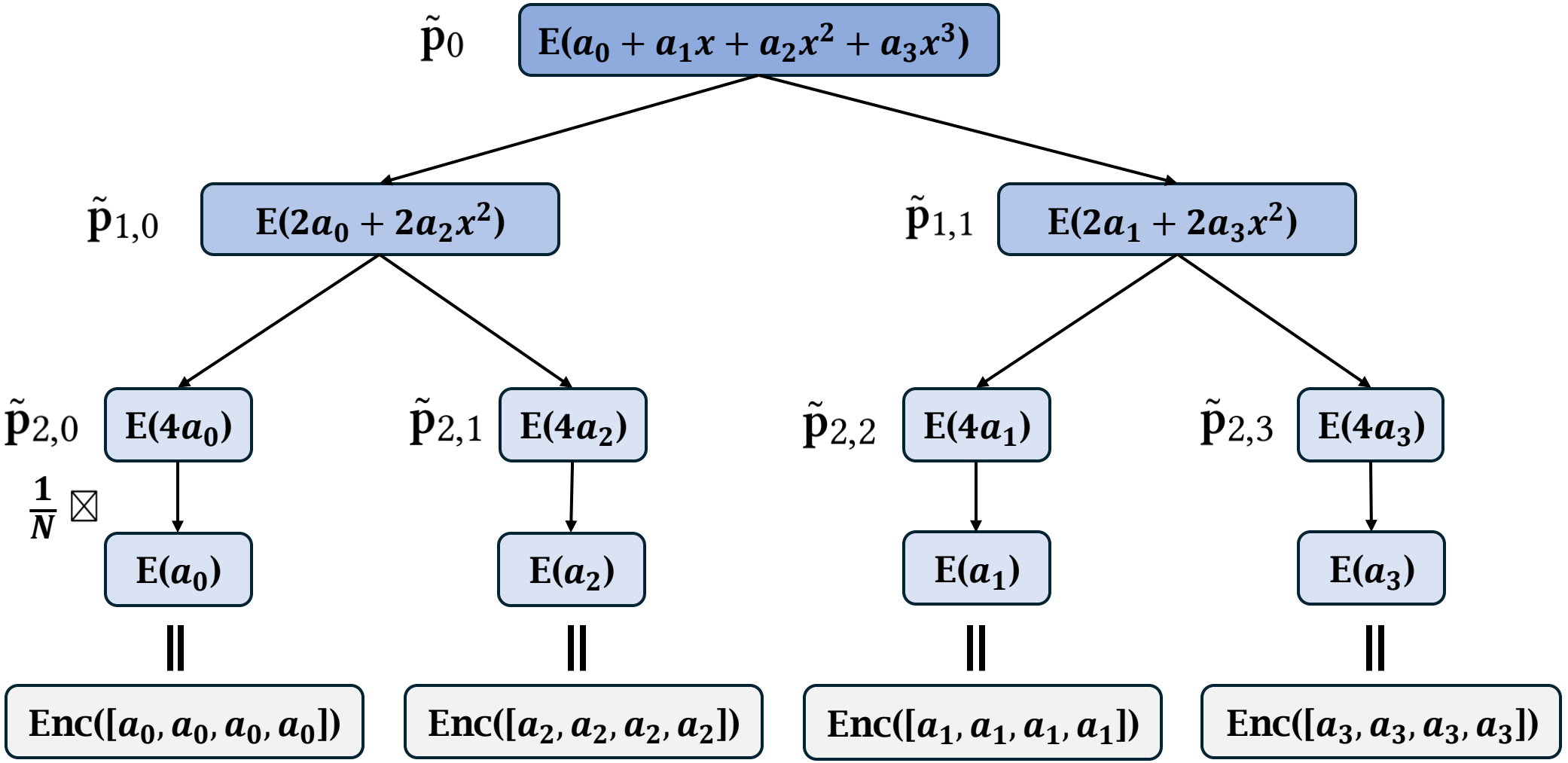
Efficient **Matrix Multiplication**

Secure ciphertext compression



Efficient **Matrix Multiplication**

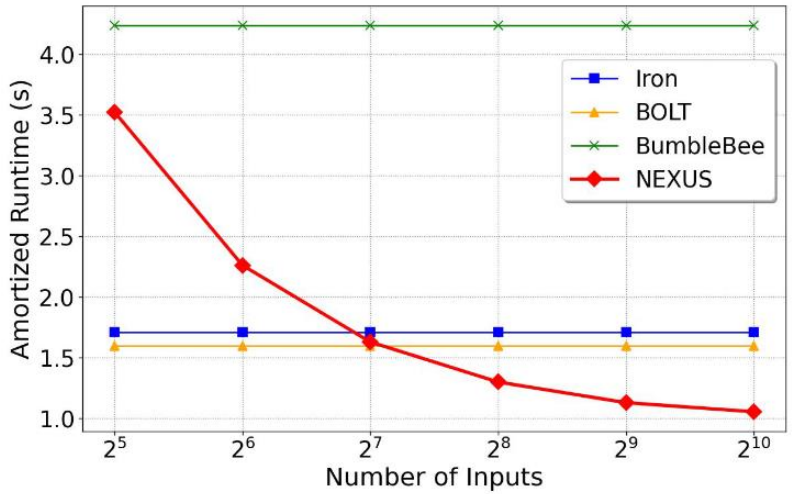
Secure ciphertext decompression [SealPIR S&P'18]



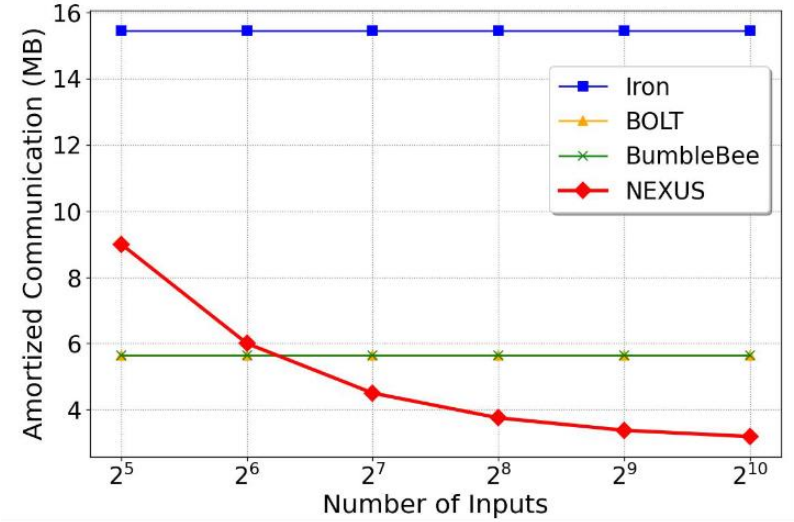
Efficient **Matrix Multiplication**
Results Comparison

BERT-base: $m = 256, n = 768, k = 64$
Amortized cost of $t = 256$ matrix multiplications

Methods	# Ciphertexts		# Key switching	
Iron [NeurIPS '22]	$\frac{2\sqrt{mnk}}{\sqrt{N}}$	111	-	0
Bumblebee [NDSS '25]	$\frac{m(n+k)}{N}$	52	$\frac{mk\log(N)}{2\sqrt{N}}$	1536
BOLT [IEEE S&P '24]	$\frac{m(n+k)}{N}$	52	$\sqrt{\frac{m^2n^2k}{N^2}}$	384
Ours	$\frac{mk}{N} + \frac{nk}{Nt}$	5	$\frac{2nk-2}{t}$	384



(a) Amortized Runtime vs. #inputs.

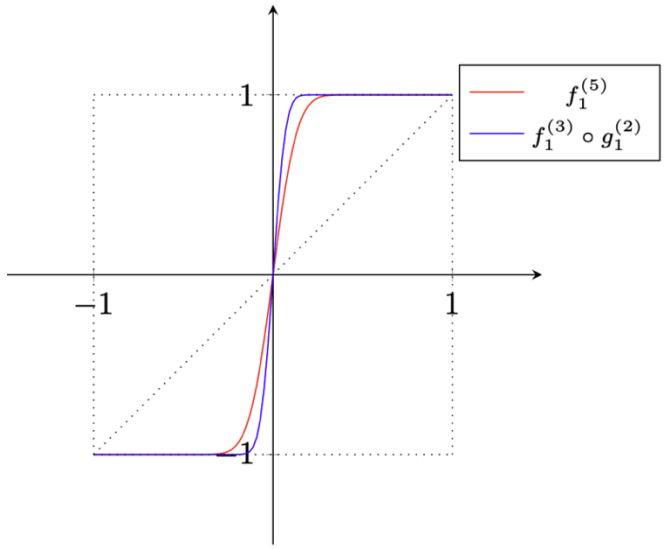


(b) Amortized Communication vs. #inputs.

Efficient **Argmax** evaluation
Preliminaries

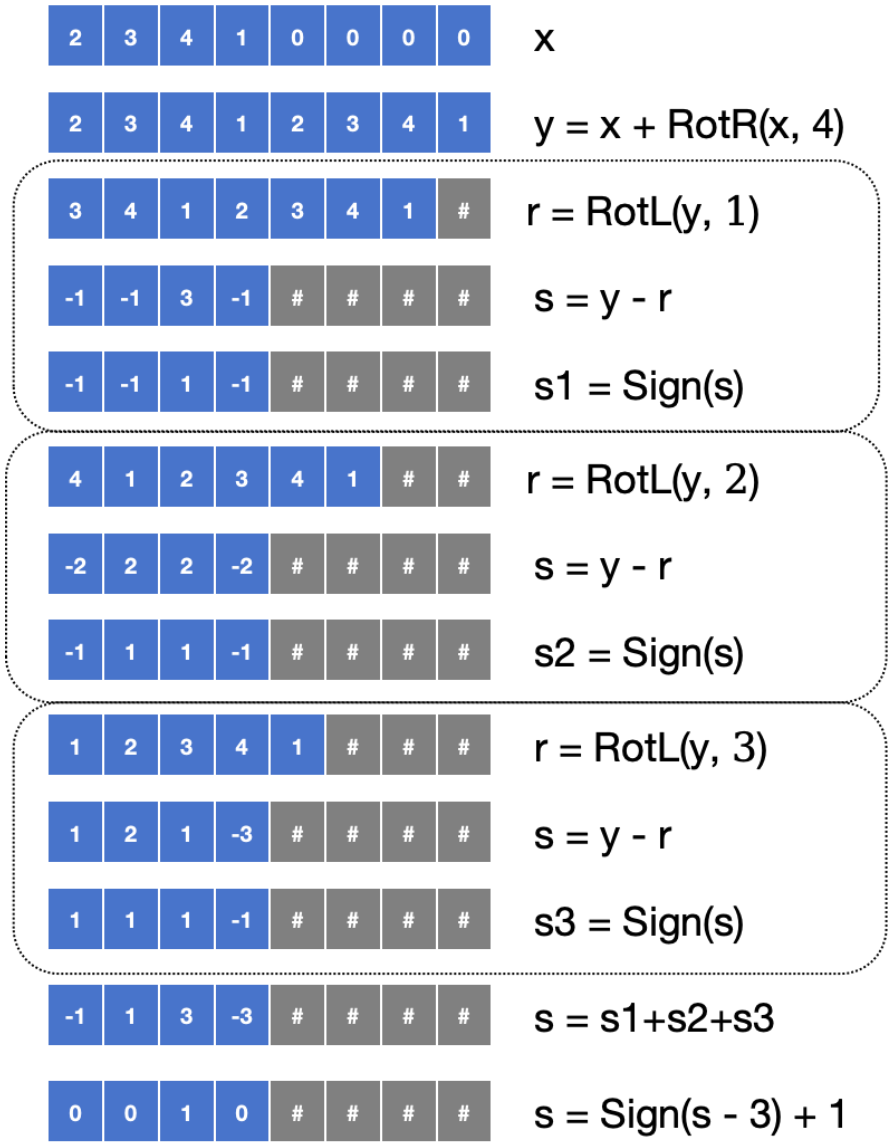
Argmax: Returns the indices of the maximum values along an vector.
The Argmax of [2, 3, 4, 1] is [0, 0, 1, 0]

Sign Function in FHE: $$Sign(x) = \begin{cases} -1 & -1 \leq x \leq -2^{-\alpha} \\ 0 & x = 0 \\ 1 & 2^{-\alpha} \leq x \leq 1 \end{cases}$$



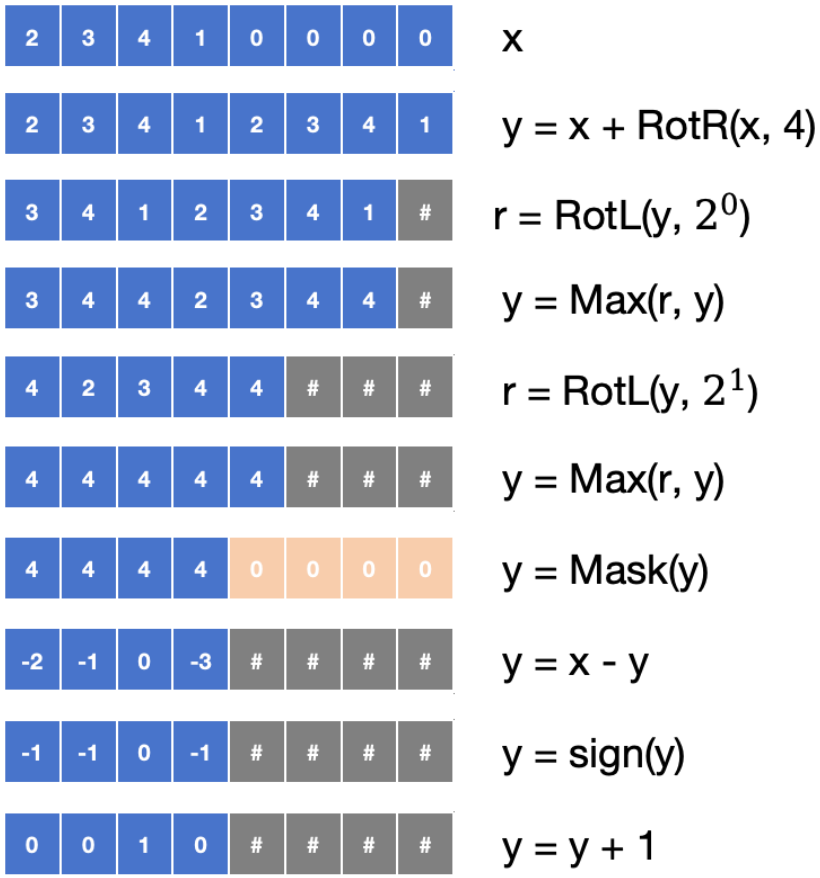
Max Function in FHE: $$Max(a, b) = \frac{a+b}{2} + \frac{a-b}{2} \cdot Sign(a-b)$$

Efficient **Argmax** evaluation



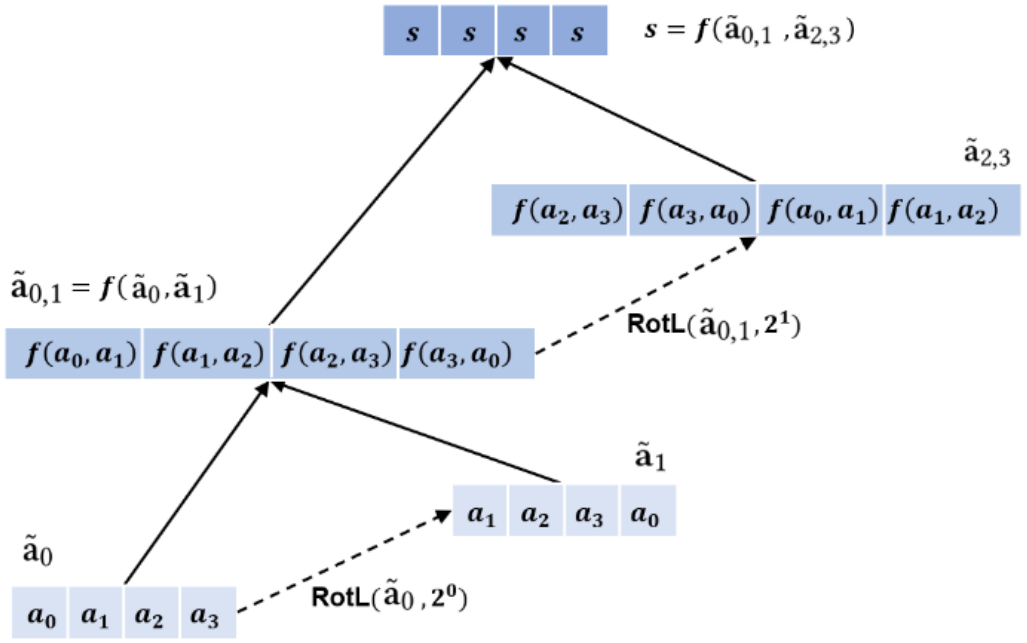
Phoneix [ACM CCS '22]: $O(N)$

$y = Sign(x - x_{max}) + 1$



Ours: $O(log\ N)$

Efficient **Argmax** evaluation



Computing the root of a *binary tree*: $O(\log N)$

$f()$ can be any associative, such as sum, max, min...

Other **Non-linear Function** evaluation
GELU, Softmax, LayerNorm

Following similar methods presented in BumbleBee [NDSS '25] and PUMA [DLZ+23]:

GELU: approximate using a piecewise polynomial:

$$\mathrm{GELU}(x) = \begin{cases} 0 & x \le -4 \\ P(x) = \sum_{i=0}^{i=3} c_i x^i & -4 < x \le -1.95 \\ Q(x) = \sum_{i=0}^{i=6} d_i x^i & -1.95 < x \le 3 \\ x & x > 3 \end{cases}$$

Softmax: taking a_{max} as a constant as its value does not affect the result of Softmax:

$$y_i = \frac{\mathrm{EXP}(a_i - a_{max})}{\sum_{j=0}^{m-1} \mathrm{EXP}(a_j - a_{max})}$$

→ Goldschmidt division algorithm

→ QuickSum

$$\mathrm{EXP}(x) \approx (1 + \frac{x}{2^r})^{2^r}, \quad x \le 0$$

→ Approximated using this Taylor sereis

LayerNorm:

$$\begin{aligned} y_i &= \gamma \cdot \frac{a_i - \mu}{\sigma} + \beta \\ &= \gamma \cdot \frac{n(a_i - \mu)}{n\sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(a_i - \mu)^2}} + \beta \\ &= \sqrt{n}\gamma \cdot \frac{na_i - n\mu}{\sqrt{\sum_{i=0}^{n-1}(na_i - n\mu)^2}} + \beta. \end{aligned}$$

Let $z_i = na_i - n\mu = na_i - \sum_{i=0}^{n-1} a_i$, then

$$y_i = \gamma\sqrt{n} \cdot \frac{z_i}{\sqrt{\sum_{i=0}^{n-1} z_i^2}} + \beta.$$

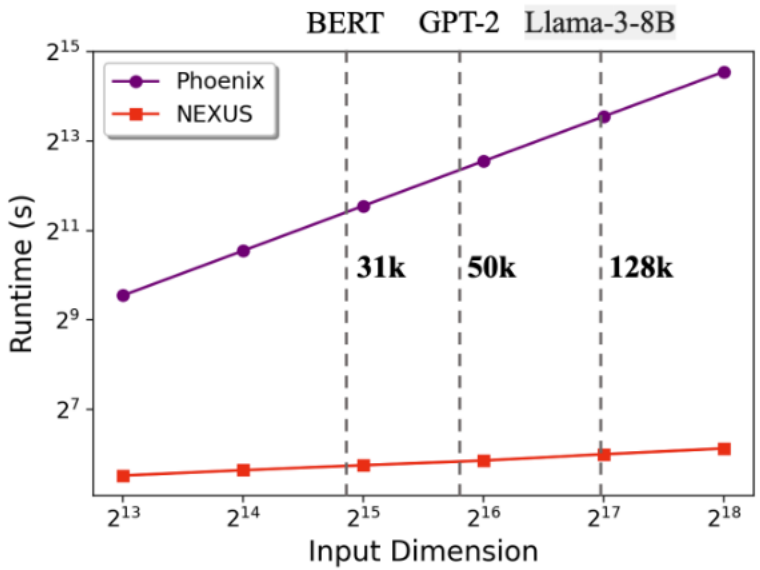
- **QuickSum**: $f(\widetilde{\boldsymbol{a}}, \widetilde{\boldsymbol{b}})$ is given by $\widetilde{\boldsymbol{a}} \boxplus \widetilde{\boldsymbol{b}}$
- **Inverse Square Root**: Newton's iteration

Efficient **Non-linear Function Eval**

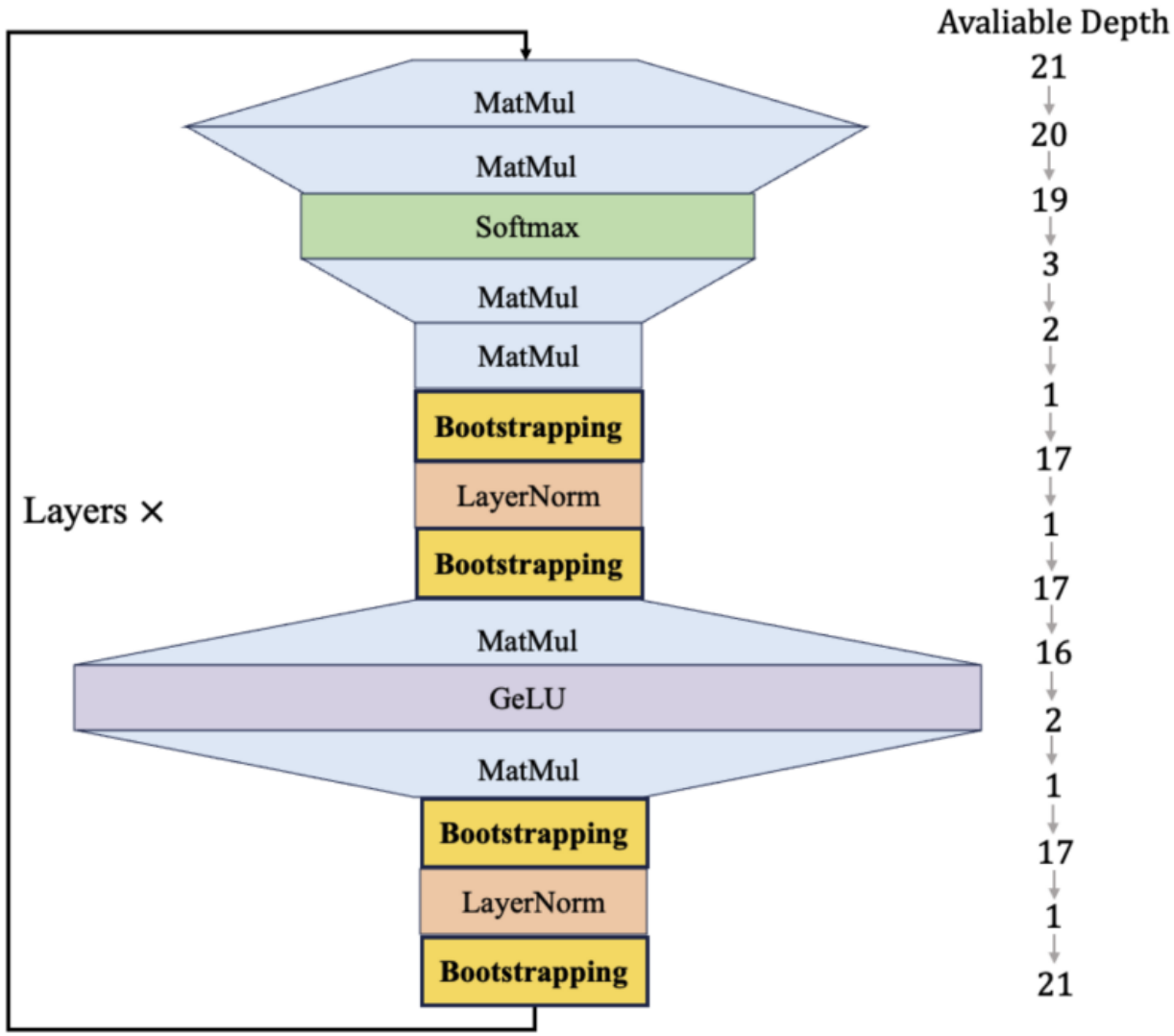
Results Comparison

Setting	Protocol	Comm	LAN(s)	WAN(s)	Error	Price($)
GELU	Iron	93.3	126	4118	5.8e-4	8.453
	BOLT	17.2	14	774	9.8e-4	1.554
	BumbleBee	3.3	24	338	1.1e-3	0.308
	NEXUS	0	44	44	7.7e-4	0.020
	NEXUS*	0	2.1	2.1	7.7e-4	0.003
SOFTMAX	Iron	42.1	60	1900	3.2e-5	3.816
	BOLT	16.9	16	775	1.4e-6	1.528
	BumbleBee	1.7	23	241	7.2e-6	0.170
	NEXUS	0	47	47	3.1e-5	0.019
	NEXUS*	0	1.2	1.2	3.1e-5	0.002
LAYERNORM	Iron	20.4	16	1158	1.7e-3	1.843
	BOLT	14.0	14	914	-	1.266
	NEXUS	0	32	32	4.5e-4	0.013
	NEXUS*	0	2.0	2.0	4.5e-4	0.003
ARGMAX	Phoenix	0	3004	3004	1.9e-2	1.252
	NEXUS	0	54	54	7.6e-4	0.023
	NEXUS*	0	2.5	2.5	7.6e-4	0.004

* GPU accelerated



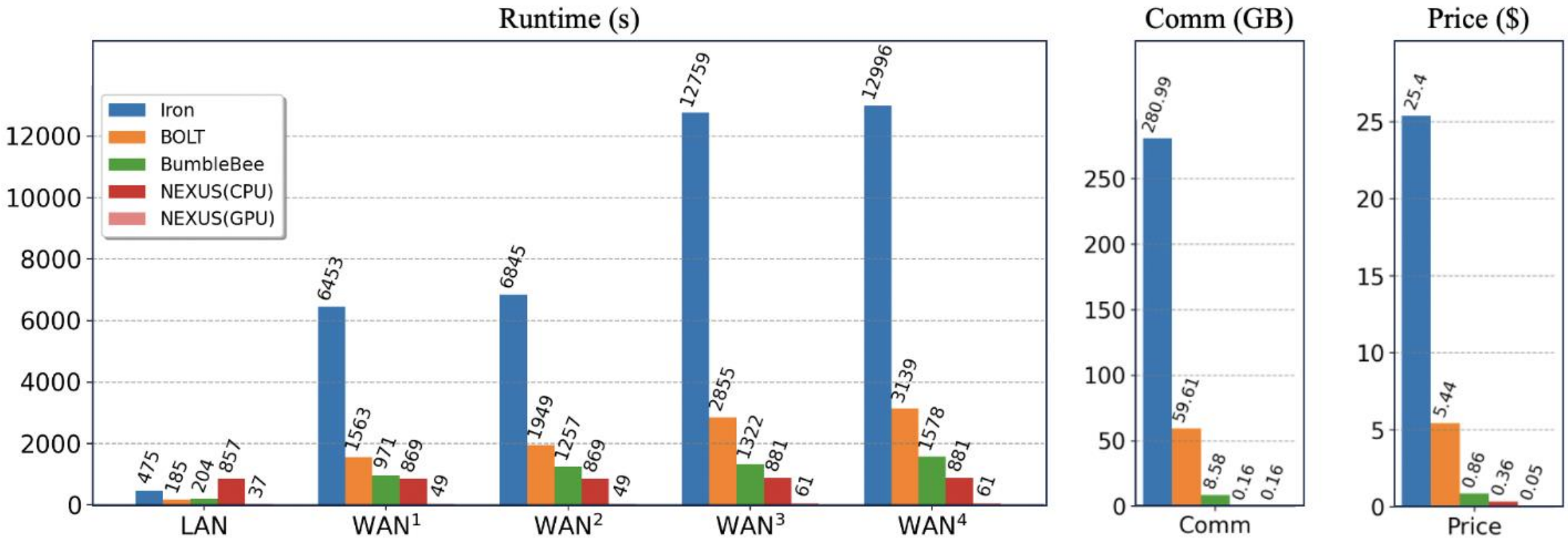
Placement of Bootstrapping



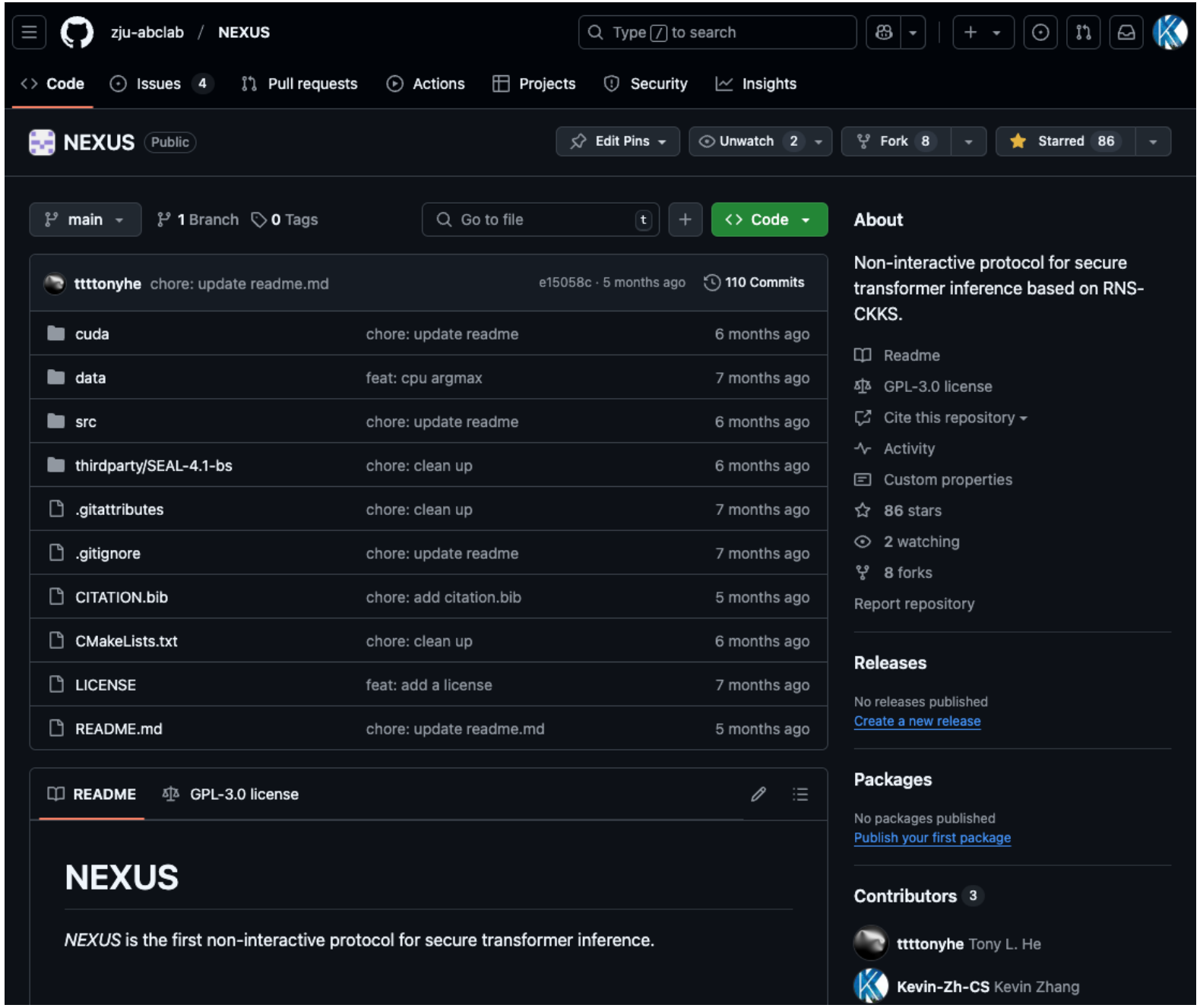
End-to-end Results

Operation	Depth	BERT-base (12 layers)		
		Input	CPU(s)	GPU(s)
MatrixMul	$21 \rightarrow 20$	$(\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times768}) \times 3$	65	2.68
MatrixMul	$20 \rightarrow 21$	$(\mathbb{R}^{128\times64} \times \mathbb{R}^{64\times128}) \times 12$	14	0.54
Softmax	$19 \rightarrow 3$	$(\mathbb{R}^{128\times128}) \times 12$	47	1.15
MatrixMul	$3 \rightarrow 2$	$(\mathbb{R}^{128\times128} \times \mathbb{R}^{128\times64}) \times 12$	9	0.36
MatrixMul	$2 \rightarrow 1$	$\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times768}$	2	0.06
Bootstrapping	$1 \rightarrow 17$	$\mathbb{R}^{128\times768}$	127	5.63
LayerNorm	$17 \rightarrow 1$	$\mathbb{R}^{128\times768}$	16	1.01
Bootstrapping	$1 \rightarrow 17$	$\mathbb{R}^{128\times768}$	127	5.63
MatrixMul	$17 \rightarrow 16$	$\mathbb{R}^{128\times768} \times \mathbb{R}^{768\times3072}$	48	1.71
GELU	$16 \rightarrow 2$	$\mathbb{R}^{128\times3072}$	44	3.35
MatrixMul	$2 \rightarrow 1$	$\mathbb{R}^{128\times3072} \times \mathbb{R}^{3072\times768}$	8	0.20
Bootstrapping	$1 \rightarrow 17$	$\mathbb{R}^{128\times768}$	127	5.63
LayerNorm	$17 \rightarrow 1$	$\mathbb{R}^{128\times768}$	16	1.01
Bootstrapping	$1 \rightarrow 21$	$\mathbb{R}^{128\times768}$	153	5.90
Argmax	*	$\mathbb{R}^{30522}$	54	2.48
Total	-	-	**857**	**37.34**

End-to-end Results



Code


zju-abclab / NEXUS
Type / to search
Code
Issues 4
Pull requests
Actions
Projects
Security
Insights
NEXUS Public
Edit Pins
Unwatch 2
Fork 8
Starred 86
main
1 Branch
0 Tags
Go to file
t
Code
ttttonyhe chore: update readme.md
e15058c · 5 months ago
110 Commits
cuda chore: update readme 6 months ago
data feat: cpu argmax 7 months ago
src chore: update readme 6 months ago
thirdparty/SEAL-4.1-bs chore: clean up 6 months ago
.gitattributes chore: clean up 7 months ago
.gitignore chore: update readme 7 months ago
CITATION.bib chore: add citation.bib 5 months ago
CMakeLists.txt chore: clean up 6 months ago
LICENSE feat: add a license 7 months ago
README.md chore: update readme.md 5 months ago
README
GPL-3.0 license
NEXUS
NEXUS is the first non-interactive protocol for secure transformer inference.
About
Non-interactive protocol for secure transformer inference based on RNS-CKKS.
Readme
GPL-3.0 license
Cite this repository
Activity
Custom properties
86 stars
2 watching
8 forks
Report repository
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 3
ttttonyhe Tony L. He
Kevin-Zh-CS Kevin Zhang

Summary

- We propose NEXUS, the first non-interactive secure transformer inference protocol.
- Based on RNS-CKKS Fully Homomorphic Encryption scheme, we design a series of efficient and communication-optimized matrix multiplication and non-linear function evaluation protocols.
- NEXUS **reduces bandwidth consumption by 372.5 ×** compared to BOLT [Oakland '24] and 53.6 × compared to BumbleBee [NDSS '25]. Its non-interactive property allows for better hardware acceleration, with the GPU version achieving a **42.3 × speedup** in runtime.

PAPER

CODE