FastFFN: Low-Latency Feedforward Block Evaluation for Secure Transformer Inference

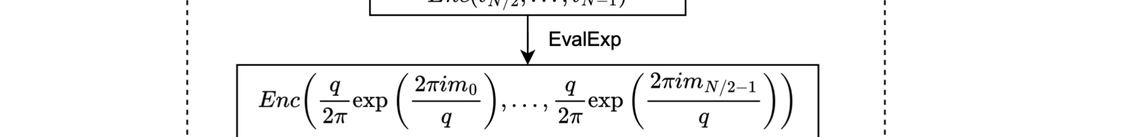
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

INFOCOMM MEDIA DEVELOPMENT AUTHORITY

Linru Zhang, Xiangning Wang, Xianhui Lu, Huaxiong Wang, Adhiraj Saxena and Kwok Yan Lam

Background: FHE-powered models can be slow & expensive

- **Problem**: The feedforward block is one of the most computationally expensive components in privacy-preserving transformer inference.



- **Root Cause:** Current solutions [1,2,3,4] use **slot-encoded CKKS scheme**, but performance is significantly constrained by use of extremely large ciphertext modulus.
- **Opportunity:** Low-latency capabilities in FHE-based inference can enable organizations in **Finance** and **Healthcare** to build applications where data protection and low computation are crucial.

Figure 1: Feedforward evaluation in NEXUS

Our solution: A Novel Framework

- Using **coefficient encoding** and adopting the efficient plaintext-ciphertext matrix multiplication algorithm from [5].
- Combining the refreshing process and the functional evaluation into one process by adapting, optimizing, and leveraging **functional bootstrapping** technique.
- Requiring only 13 multiplicative layers, corresponding to a ciphertext modulus **of less than 700 bits** which significantly reducing computational overhead.

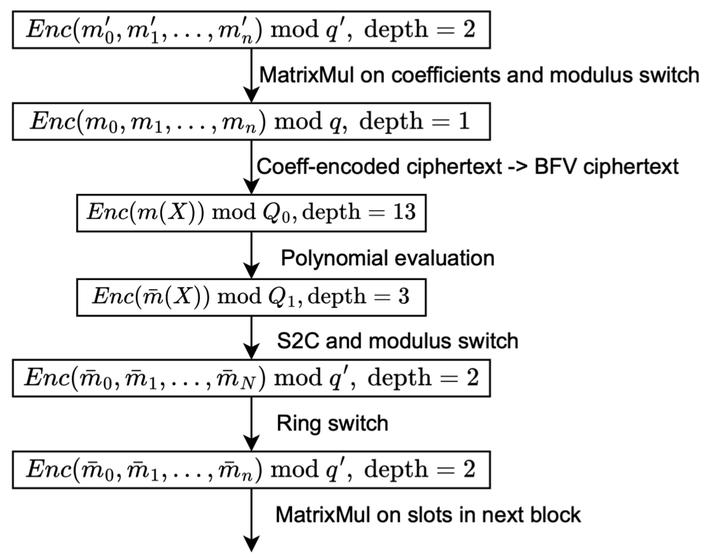


Figure 2: Feedforward evaluation in Our scheme

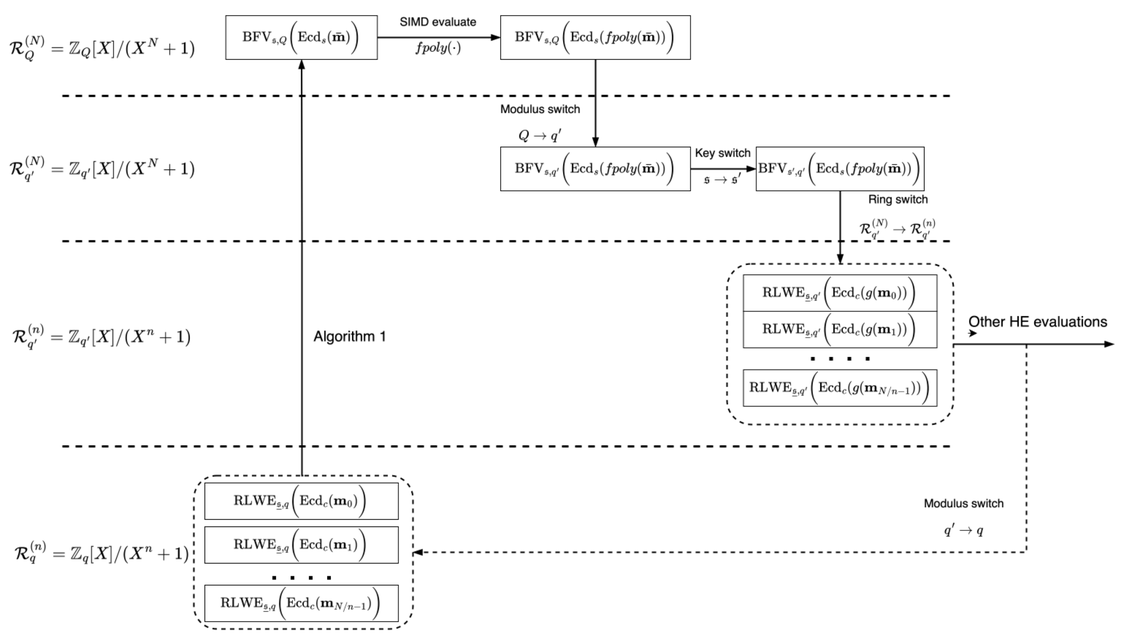
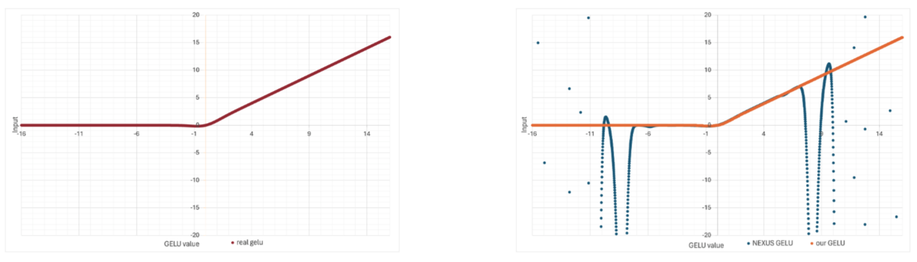


Figure 3: Full flow of non-polynomial activation evaluation in Our scheme

Experiment result: GELU evaluation

- Accuracy:
  - Outputs of our GELU evaluation are close to real GELU values for all inputs , while the output of NEXUS's GELU evaluation introduces a huge error when $x \notin [-8,8]$.
  - Our solution can flexibly expand the input. This adjustment results in only a slight increase in error, avoiding the significant inaccuracies observed with the approximation method used by NEXUS.



(a) Real GELU value (b) NEXUS and our GELU values

Figure 4: Output of GELU values and error analysis

- Efficiency:
  - Besides the improvement on accuracy and the flexibility of expand the input range, our algorithm is also > 3 times faster than CKKS-based evaluation algorithms.
  - PEGASUS is able to achieve the same accuracy and input range with ours, but it is more than 300 times slower than ours.

Algorithm	CKKS-based solution [1,2,3,4]	LUT-based solution PEGASUS [6]	Our scheme
Amortized time per input	7ms – 8.5ms	749ms	2.3ms
Total time	228s – 277.8s	191.74s	76.6s
Number of inputs	32768	256	32768
Breakdown time cost	Bootstrapping: 220s – 262s PolyEval: 8s – 15.8s	S2C:0.54s Extract: 6.15s LUT: 162.7s LT: 22.3s	LT: 7.6s PolyEval: 54s S2C:15s Switches: 0.01s

Experiment result: Feedforward block in BERT-base

Procedure	Ciphertext	BERT base
Input	$\mathrm{RLWE}_{\underline{s},q'}(\mathrm{Ecd}_c(\boldsymbol{m}_i))$	-
Pt-ct matrix multiplication [13]	$\mathrm{RLWE}_{\underline{s},q'}(\mathrm{Ecd}_s((WM)_i))$	0.18s
Mod Switch $q' \to q$	$\mathrm{RLWE}_{\underline{s},q}(\mathrm{Ecd}_s((WM)_i))$	0.0025s
Coeff-encoded RLWE to slot-encoded BFV	$\mathrm{BFV}_{\underline{s},Q}(\mathrm{Ecd}_s(\bar{\boldsymbol{m}}_i))$ 12 BFV cts in BERT base, or 4 BFV cts in Llama-3-8B	6.4s
Polynomial evaluation	$\mathrm{BFV}_{\underline{s},Q'}(\mathrm{Ecd}_s(f(\bar{\boldsymbol{m}}_i)))$	42.8s
Slot-encoded BFV to coeff-encoded BFV	$\mathrm{BFV}_{\underline{s},Q'}(\mathrm{Ecd}_c(f(\bar{\boldsymbol{m}}_i)))$	4.02s
Switches	$\mathrm{RLWE}_{\underline{s},q}(\mathrm{Ecd}_c(f((WM)_i)))$	0.0038s
Total latency	-	53.4s (128 tokens)

- NEXUS and MOAI support batching of up to 32 and 256 inputs, respectively. Their total feedforward-layer latency remains constant at 5,378 seconds and 3,138 seconds, independent of the batch size, which is approximately $\mathbf{58} \times - \mathbf{100} \times$ slower than our approach.
- Similar with our idea, THOR and Powerformer do not apply batching technique and focus on reducing the end-to-end latency. However, they still requires 614.8 seconds and 588.9 seconds, which are about $\mathbf{11} \times$ slower than our approach.

Reference

[1] J. Zhang, X. Yang, L. He, K. Chen, W.-j. Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang, "Secure transformer inference made non-interactive," in NDSS, 2025.
[2] L. Zhang, X. Wang, J. J. Sim, Z. Huang, J. Zhong, H. Wang, P. Duan, and K. Y. Lam, "MOAI: Module-optimizing architecture for non-interactive secure transformer inference," ICLR 2026.
[3] J. Moon, D. Yoo, X. Jiang, and M. Kim, "THOR: Secure transformer inference with homomorphic encryption," in CCS 2025.
[4] D. Park, E. Lee, and J.-W. Lee, "Powerformer: Efficient and high accuracy privacy-preserving language model with homomorphic encryption," in ACL 2025.
[5] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, and D. Stehle, "Plaintext-ciphertext matrix multiplication and fhe bootstrapping: Fast and fused," in Crypto 2024.
[6] W.-j. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption," in S&P 2021.

Full version

Our code