Perseus: A Full-Stack System for Homomorphic Model Inference

Alessandro Zirilli*[1] Davide Marincione*[1,3]
Evgenios M. Kornaropoulos[2] Giuseppe Ateniese[2] Emanuele Rodolà[1,3]
[1]Sapienza University of Rome [2]George Mason University [3]Paradigma

Large Language Models (LLMs) have made it exceptionally convenient for users to send their private data to model providers, which process them in plaintext. Fully Homomorphic Encryption (FHE) promises to secure remote inference by computing directly on encrypted data, yet it has remained prohibitively slow. The effort to make it practical remains fragmented, with most existing frameworks being *closed-source*, which keeps progress from compounding. We present `Perseus`, a free and *open*[1] Python software library that supports the full stack of encrypted model inference, from the cryptographic backend and data packing, to encrypted nonlinearities and end-to-end model execution. `Perseus` ships with built-in novel optimizations, one at each level of the stack: in-kernel regeneration of key-switching keys at the (i) backend level, a complex variant of `Cachemir` packing [1] at the (ii) data level, and a state-of-the-art bootstrap placer that is both packing- and magnitude-aware at the (iii) model level. `Perseus` achieves state-of-the-art latency across multiple models and modalities, within 3% of the plaintext models' accuracy and perplexity. Remarkably, it is 3.96× faster at single-token decoding than any other implementation on `Llama-3-8B` (Table 1). We initiate, for the first time, a public speedrun on encrypted `GPT-2`, so that progress can compound.

Table 1: Encrypted `Llama-3-8B` decode of one token; †composed from per-block or per-operator timings, not run end to end. Throughout the paper we mark competing frameworks as: ✓ available, ∼ temporarily or partially released, and ✗ closed-source.

① `Llama-3-8B`	context	s/tok	GPU	availability
`AR-HE` (Kanpak and Unat, 2026)	32	1569.0†	H100	✗
`ROSETTA` (Yu et al., 2026b)	512	311.0†	A100	✗
`Cachemir` (Yu et al., 2026a)	512	97.0†	A100	∼
`Sylph` (Park et al., 2026)	128	31.0†	RTX PRO 6000	✗
`Perseus` (ours)	**512**	**7.8**	**RTX PRO 6000**	✓

Code Speedrun

1 Introduction

Fully Homomorphic Encryption (FHE) promises a future of trustless outsourced computation. One of its most appealing applications is Machine Learning, where successful task execution often depends on disclosing sensitive data. In particular, the recent expansion of capabilities in Large Language Models (LLMs) has made them extremely convenient for many users, who accept to share their private, plaintext data with the models' inference providers. While encrypted models exist, the extreme latency FHE introduces through bootstrapping limits the deployment of these models in practice.

The landscape of encrypted execution is evolving rapidly, with various approaches aiming to reduce the high latency imposed by FHE (and in particular CKKS [5]). New techniques are being released for all the different abstraction levels of the encrypted models' inference stack, from bootstrap speed-ups, GPU implementations [6–9], and 32-bit arithmetic [7] at the cryptographic backend level; to task-specific packing systems [1, 10–16] at the data level; and to new techniques for approximating machine learning layers [4, 17–24], bootstrap placers [25–27], and fine-tuning techniques [23, 28] at the model level. The unfortunate reality of these optimizations is that most

*Equal contribution. Contact: {zirilli, marincione}@di.uniroma1.it.
[1]Released under the BUSL 1.1: free for research and non-commercial use, commercial use available upon request.

are implemented into different, sometimes *closed-source*, frameworks, making integration between one another difficult and time-consuming.

We present `Perseus`, a *publicly available Python library* for encrypted machine learning model inference that enables practitioners to implement models and optimization techniques in an easy-to-use environment. `Perseus`' contributions are:

- a free and open Python library (code) that supports full-stack development of models, built on `Perseus-core`, a Rust and CUDA cryptographic backend for CKKS `32-bit` arithmetic and bootstrapping.
- Built-in optimizations at all levels of the stack, from in-kernel regeneration of key-switching keys using `ChaCha12` (halving key storage), a complex variant of `Cachemir` packing (roughly halving memory), to packing- and magnitude-aware bootstrap planning with an iterated minimum cut ($\approx 30\%$ fewer bootstraps), which result in state-of-the-art latency on all already-implemented models (`GPT-2`, `Llama-3-8B`, `ViT`, `BERT`).
- A public leaderboard to cooperatively optimize encrypted language models' inference, a speedrun (leaderboard) that will condense the community's effort to reduce latency.

Throughout the paper, we report published results for competing methods when their code is unavailable and reproduce their results when the code is publicly available.

2 Perseus

We conceptualize an encrypted models' inference stack into three abstraction levels: (i) the *cryptographic backend* that executes CKKS primitives (*e.g.* encryption, additions, multiplications, bootstrapping, etc.); (ii) the *data level*, which controls the layout of ciphertexts' slots (the packing) and thus defines encrypted algorithms; and (iii) the *bootstrap placement* that decides where and how to refresh the multiplicative depth of the model's circuit.

State-of-the-art CKKS execution of LLMs is still far from practical, despite frequent optimizations across all levels of the stack. Although many optimizations are orthogonal and could be combined to compound their benefits, doing so remains difficult in practice, as the closed-source nature of most such systems fragments the stack and limits opportunities for integration.

`Perseus` offers an alternative that supports the entire encrypted inference stack. Adopters can implement a new encrypted machine learning model, a packing layout, a new approximation, and/or a bootstrap placement algorithm in pure Python on top of `Perseus-core`, *i.e.*, our Rust and CUDA CKKS runtime. In `Perseus`, the ease of development comes with state-of-the-art end-to-end performance, driven not only by an efficient implementation designed with machine learning models in mind, but also by three contributions, one for each level of the stack.

(i) Cryptographic backend: in-kernel key regeneration. Key-switching keys dominate GPU memory and bandwidth during bootstrapping. Half of each key is uniformly random, and thus can be regenerated from a seed. Instead of storing this half, `Perseus` recomputes it with a counter-mode `ChaCha12` expander inside the two kernels that make use of the key. The remaining, non-random, part is stored as a dense `28-bit` stream, trading extra arithmetic for less memory traffic and halving key storage.

(ii) Data: complex-`Cachemir`. `Cachemir` [1] packs activations and key/value caches for autoregressive decoding, but exploits only the real slots of a ciphertext. `Perseus` extends it to the complex slots of CKKS, carrying two real vectors in each ciphertext, which roughly doubles the throughput of many operations and layers, and roughly halves memory requirements.

(iii) Model: bootstrap placer. Since bootstraps are two to three orders of magnitude slower than any other primitive, minimizing their usage makes a concrete difference in reducing latency. `Perseus` first captures a model's computation graph, typed by each ciphertext's coefficient magnitude and packing periodicity, then optimizes the count by placing bootstraps via an iterated minimum cut over this typed graph, unlike prior min-cut placers [29] that decide from depth and scale alone. The annotations about period and magnitude are used by selecting the most appropriate bootstrap for both, imposing a cheaper and more accurate sparse bootstrap when possible, and choosing the right scaling factor to keep coefficients in the `EvalMod` approximation band. On average, the placer executes $\approx 30\%$ fewer bootstraps than comparable placers [25–27].

Table 2: Encrypted `Llama-3-8B` prefill; †composed from per-block or per-operator timings, *amortized over a batch, §replicated from their released code. →linearly scaled timings to adjust prompt size. ✓ available, ✗ closed-source, ∼ temporarily or partially released.

① `Llama-3-8B`	prompt	prefill (s)	GPU	availability
`MOAI` (Zhang et al., 2026)	8→128	3592.0†*	NVIDIA H200 GPU	∼
`NEXUS` (Zhang et al., 2025)	8→128	829.0†*	4×A100	∼
`Euston` (Gao et al., 2026)	128	484.0†*	RTX 6000 ADA	∼
`STIP` (Wang et al., 2026)	128	406.6†*	A100	✗
`Odin` (Fan et al., 2026)	128	366.4	H100	∼
`Cerium` (Jayashankar et al., 2025)	128	347.5§	RTX PRO 6000	✓
`Sylph` (Park et al., 2026)	128	116.0†	RTX PRO 6000	✗
`Perseus` (ours)	128	**54.8**	RTX PRO 6000	✓

3 Results

All `Perseus`'s measurements use one NVIDIA RTX PRO 6000 and a 32-bit RNS CKKS chain with composite scaling of degree 2 (each level a pair of 27-bit primes, q_0 a pair of 28-bit primes), $\log N = 16$, 128-bit security and $q_0 = 2^{56}$, giving 27 levels over 51 primes.

We implement in `Perseus` four deep-learning models, ① `Llama-3-8B`, ② `GPT-2-small`, ③ `ViT-B/16`, and ④ `BERT-base`, and report their latency end-to-end, *i.e.* accounting for weights movement and management during the actual inference. We address two execution modalities, one being *prefilling* (or analogously *classification*) and the other being single-token *decoding*. They fundamentally differ as the first involves multiple tokens being processed at the same time, while the latter autoregressively generates tokens considering the current KV-caches' state.

For prefill and classification we also report the normalized accuracy, *i.e.,* the accuracy of the encrypted model over the plaintext one on the same inputs (next-token accuracy for prefill, label accuracy for classification), as a proxy for performance degradation due to inherent CKKS approximated arithmetic, bootstraps, and required nonlinearities' approximations. For decoding, we report the perplexity of the encrypted model next to the reference plaintext one. All quality metrics are computed on held-out data from the approximations' calibration set.

① `Llama-3-8B` is the most challenging model we test, due to the sheer number of parameters and operations it requires. It is one of the most popular models transferred to FHE in the literature, and, when it becomes practical to run, it will mark a milestone, as it is in the same size class as state-of-the-art models. We test it in both execution modalities, decode (Table 1) and prefill (Table 2).

Decode uses in-kernel key regeneration, our complex-`Cachemir` packing and complex-KV-cache, and `Cutmax` [20] argmax. Under a fixed bootstrap schedule, decided with our planner, `Perseus` reduces latency by 3.96× with respect to the fastest known implementation (*i.e.* `Sylph` [4], running on roughly the same hardware, but on the closed-source `HEaaN` [9]). The encrypted perplexity settles on 7.77 against a plaintext value of 7.76 (at 512 context). Prefill uses PCMM [4, 17, 23, 33,

Table 3: Encrypted `GPT-2`: prefill of a prompt and decode of one token; †composed from per-block timings. ✓ available, ✗ closed-source, ∼ temporarily or partially released.

	② `GPT-2` small	prompt/context	latency	GPU	availability
prefill (s)	`AR-HE` (Kanpak and Unat, 2026)	128	18520.0	H100	✗
	`Perseus` (ours)	128	**6.8**	RTX PRO 6000	✓
decode (s/tok)	`AR-HE` (Kanpak and Unat, 2026)	33	544.0	H100	✗
	`ROSETTA` (Yu et al., 2026b)	512	78.6†	A100	✗
	`EncryptedLLM` (Castro et al., 2025)	128	68.0	A100	∼
	`Perseus` (ours)	512	**1.9**	RTX PRO 6000	✓

34] for the MLP execution algorithm, and CCMM [4, 34–36] for the Attention mechanism, which are two FHE implementations of regular Generalized Matrix Multiplication (unlike decode, which uses a Baby-Step-Giant-Step algorithm). Analogously to decode, `Perseus` establishes itself as the fastest implementation. In prefill, normalized accuracy is 100% over 64 held-out prompts.

② `GPT-2-small` is a toy model, but at $1.9\,^{s}/_{tok}$ ($41\times$ faster than the closest baseline, Table 3) we believe it can provide a fast-iteration development platform for further ML-dedicated optimizations in the encrypted domain. We therefore publish and commit to maintaining a public leaderboard that invites the community to improve encrypted `GPT-2`. Our objective is to trigger a snowball effect similar to what the plaintext community is experiencing [37, 38]. Our implementation maintains a perplexity of 26.46, compared with 26.47 in plaintext. Its prefill reaches a normalized accuracy of 100% over 64 held-out prompts.

Table 4: Encrypted `ViT` inference of one image; *amortized over a batch of 32, on a model retrained with polynomial layers. ✓ available, ✗ closed-source, ∼ temporarily or partially released.

③ `ViT`	model	s/img	**Hardware**	availability
`ATLAS` (Xie et al., 2026)	`ViT-B/16`	2259.0	RTX 4090 48GB	✓
`FIONA` (Peng et al., 2026)	`ViT-S/16`	374.4*	CPU, 48 cores	✗
`Perseus` (ours)	`ViT-B/16`	**21.1**	RTX PRO 6000	✓

Table 5: Encrypted `BERT` inference of one input. §replicated from their released code on our hardware. ✓ available, ✗ closed-source, ∼ temporarily or partially released.

④ `BERT`	model	tokens	s/sentence	**GPU**	availability
`THOR` (Moon et al., 2025)	`BERT-base`	128	602.0	A100	∼
`POLARIS` (Badawi et al., 2026)	`BERT-Mini`	–	83.0	A100	✗
`Cerium` (Jayashankar et al., 2025)	`BERT-base`	128	43.6§	RTX PRO 6000	✓
`Perseus` (ours)	`BERT-base`	128	**8.6**	RTX PRO 6000	✓

③ `ViT` and ④ `BERT`, follow the prefill implementation, with the only difference in Attention patterns, and we test respectively encrypted sentence and image classification (Tables 4 and 5). We achieve state-of-the-art latency in both cases, proving `Perseus`' efficiency and coverage of modern deep learning tasks. Their normalized accuracies are 100% on 64 SST-2 validation sentences for ④ `BERT` and 96.9% on 64 `Tiny-ImageNet` validation images for ③ `ViT`.

4 Related Works

CKKS naturally supports SIMD operations, a parallelism for which modern GPUs are built and keep getting better at. That's why, even in the encrypted landscape, classical CPU frameworks such as `OpenFHE` [41], `SEAL` [42], and `Lattigo` [43] are losing ground in favor of backends with GPU support [6, 44–46], including the closed-source `HEaaN` [9] on which current state-of-the-art encrypted latencies are reported [4]. For this reason newest approaches [7, 8] moved encrypted computation to 32-bit arithmetic, aligning to GPUs' cores design. `Medha` [47] and `REED` [48] halve key storage by regenerating the random part of each key-switching key from a seed, but only in custom hardware, while `Perseus` regenerates it in-kernel on the GPU.

Encrypted models have scaled too. Thanks to more mature systems and faster hardware, the field is shifting from smaller models such as `BERT` [10, 11, 17] towards bigger LLMs [1, 4, 31, 32], yet most of these systems usually improve only one level of the stack and release no code or only part of it. At the (ii) data level, `STIP` [12] and `THEMIS` [15] already use the imaginary half of the slots, but only within the prefill stage, while our complex-`Cachemir` brings it to decoding. At the (iii) model level, existing placers [2, 25–27, 29, 49] decide from depth, latency, and scale, none from the values themselves, while ours also reads the coefficient magnitude and packing period of every ciphertext.

5 Conclusions

We presented `Perseus`, a publicly available Python library for encrypted model inference. Its main contribution is state-of-the-art latency across multiple models and modalities, up to practically useful model sizes. It introduces optimizations at all levels of the stack, namely key regeneration, complex-`Cachemir`, and magnitude-aware placement. We also proposed a new workbench, a public speedrun on encrypted `GPT-2`, where the community's optimizations can compound.

Acknowledgments

This work was supported through the MUR FIS2 grant n. FIS-2023-00942 "NEXUS" (CUP B53C25001030001), and by NSF award #2439951. We thank Cuiyi Zhang for her valuable advice on the choice of software license.

References

[1] Ye Yu, Yifan Zhou, Yi Chen, Pedro Soto, Wenjie Xiong, and Meng Li. "Cachemir: Fully Homomorphic Encrypted Inference of Generative Large Language Model with KV Cache". In: *CoRR* abs/2602.11470 (2026). DOI: 10.48550/ARXIV.2602.11470. arXiv: 2602.11470. URL: https://doi.org/10.48550/arXiv.2602.11470.

[2] Halil Ibrahim Kanpak and Didem Unat. *Static Bootstrap Placement for Encrypted Language Model Decoding*. 2026. arXiv: 2610.04912 [cs.CR]. URL: https://arxiv.org/abs/2610.04912.

[3] Jiangrui Yu, Baosheng Zhang, Liang Kong, Lin Ding, Yi Chen, Ye Yu, Mingzhe Zhang, and Meng Li. "ROSETTA: Efficient and Accurate Privacy-Preserving LLM Decoding via Hybrid CKKS/TFHE Evaluation". In: *CoRR* abs/2609.16915 (2026). DOI: 10.48550/ARXIV.2609.16915. arXiv: 2609.16915. URL: https://doi.org/10.48550/arXiv.2609.16915.

[4] Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. *Scaling up FHE-based Privacy-Preserving ML: Higher Throughput, Longer Inputs for LLama-3-8B*. 2026. arXiv: 2601.18511 [cs.CR]. URL: https://arxiv.org/abs/2601.18511.

[5] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. Lecture Notes in Computer Science. Springer, 2017, pp. 409–437. DOI: 10.1007/978-3-319-70694-8_15. URL: https://doi.org/10.1007/978-3-319-70694-8%5C_15.

[6] Carlos Agulló-Domingo, Óscar Vera-López, Seyda Nur Güzelhan, Lohit Daksha, Aymane El Jerari, Kaustubh Shivdikar, Rashmi S. Agrawal, David R. Kaeli, Ajay Joshi, and José L. Abellán. "FIDESlib: A Fully-Fledged Open-Source FHE Library for Efficient CKKS on GPUs". In: *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2025, Ghent, Belgium, May 11-13, 2025*. IEEE, 2025, pp. 1–3. DOI: 10.1109/ISPASS64960.2025.00045. URL: https://doi.org/10.1109/ISPASS64960.2025.00045.

[7] Wonseok Choi, Jongmin Kim, and Jung Ho Ahn. "Cheddar: A Swift Fully Homomorphic Encryption Library Designed for GPU Architectures". In: *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. ASPLOS '26. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 35–49. ISBN: 9798400721656. DOI: 10.1145/3760250.3762223. URL: https://doi.org/10.1145/3760250.3762223.

[8] Wonseok Choi, Hyunah Yu, Jongmin Kim, Hyesung Ji, Jaiyoung Park, and Jung Ho Ahn. "Theodosian: A Deep Dive into Memory-Hierarchy-Centric FHE Acceleration". In: *2026 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2026, pp. 299–310. DOI: 10.1109/ISPASS69572.2026.00037.

[9] Wonkyung Jung, Eojin Lee, Sangpyo Kim, Keewoo Lee, Namhoon Kim, Chohong Min, Jung Hee Cheon, and Jung Ho Ahn. "HEAAN Demystified: Accelerating Fully Homomorphic Encryption Through Architecture-centric Analysis and Optimization". In: *CoRR* abs/2003.04510 (2020). DOI: 10.48550/ARXIV.2003.04510. arXiv: 2003.04510. URL: https://doi.org/10.48550/arXiv.2003.04510.

[10] Linru Zhang, Xiangning Wang, Sim Jun Jie, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. "MOAI: Module-Optimizing Architecture for Non-Interactive Secure Transformer Inference". In: *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. Ed. by Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust. proceedings.iclr.cc, 2026. URL: https://proceedings.iclr.cc/paper%5C_files/paper/2026/hash/2f2b1d6bbd50865eca40e2774a057eef-Abstract-Conference.html.

[11] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. "Secure Transformer Inference Made Non-interactive". In: *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. URL: https://www.ndss-symposium.org/ndss-paper/secure-transformer-inference-made-non-interactive/.

[12] Zihao Wang, Rongmao Chen, Xinwen Gao, Yi Wang, Lin Liu, Zixin Lan, Zhaoyu Wang, Shaojing Fu, Qiong Wang, and Xinyi Huang. "STIP: Efficient and Secure Non-Interactive Transformer Inference via Compact Packing". In: *IACR Cryptol. ePrint Arch.* 2026 (2026), p. 174. URL: https://eprint.iacr.org/2026/174.

[13] Lawrence Lim, Vikas Kalagi, Divyakant Agrawal, and Amr El Abbadi. "Tricycle: Private Transformer Inference with Tricyclic Encodings". In: *IACR Cryptol. ePrint Arch.* 2025 (2025), p. 1200. URL: https://eprint.iacr.org/2025/1200.

[14] Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. *ARION: Attention-Optimized Transformer Inference on Encrypted Data*. Cryptology ePrint Archive, Paper 2025/2271. 2025. URL: https://eprint.iacr.org/2025/2271.

[15] Shihao Li, Wenhao Wang, Lu Li, Xiaomei Tang, Jian Liu, Rongmao Chen, Cheng Hong, and Guangfu Sun. *THEMIS: A Co-Designed System for Encrypted Transformer Inference*. Cryptology ePrint Archive, Paper 2026/2264. 2026. URL: https://eprint.iacr.org/2026/2264.

[16] Xinwen Gao, Shaojing Fu, Lin Liu, Zhuotao Liu, Yuchuan Luo, and Yongjun Wang. "Euston: Efficient and User-Friendly Secure Transformer Inference with Non-Interactivity". In: *IEEE Symposium on Security and Privacy, SP 2026, San Francisco, CA, USA, May 18-21, 2026*. IEEE, 2026, pp. 1443–1462. DOI: 10.1109/SP63933.2026.00048. URL: https://doi.org/10.1109/SP63933.2026.00048.

[17] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. "THOR: Secure Transformer Inference with Homomorphic Encryption". In: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. Ed. by Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier. ACM, 2025, pp. 3765–3779. DOI: 10.1145/3719027.3765150. URL: https://doi.org/10.1145/3719027.3765150.

[18] Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K. Ryu, and Jung Hee Cheon. "Encryption-Friendly LLM Architecture". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=pbre0HKsfE.

[19] Itamar Zimerman, Allon Adir, Ehud Aharoni, Matan Avitan, Moran Baruch, Nir Drucker, Jenny Lerner, Ramy Masalha, Reut Meiri, and Omri Soceanu. "Power-Softmax: Towards Secure LLM Inference over Encrypted Data". In: *CoRR* abs/2410.09457 (2024). DOI: 10.48550/ARXIV.2410.09457. arXiv: 2410.09457. URL: https://doi.org/10.48550/arXiv.2410.09457.

[20] Matan Avitan, Moran Baruch, Nir Drucker, Itamar Zimerman, and Yoav Goldberg. "Efficient Decoding Methods for Language Models on Encrypted Data". In: *CoRR* abs/2509.08383 (2025). DOI: 10.48550/ARXIV.2509.08383. arXiv: 2509.08383. URL: https://doi.org/10.48550/arXiv.2509.08383.

[21] Hongyuan Qu and Guangwu Xu. "Improvements of Homomorphic Secure Evaluation of Inverse Square Root". In: *Information and Communications Security - 25th International Conference, ICICS 2023, Tianjin, China, November 18-20, 2023, Proceedings*. Ed. by Ding Wang, Moti Yung, Zheli Liu, and Xiaofeng Chen. Vol. 14252. Lecture Notes in Computer Science. Springer, 2023, pp. 110–127. DOI: 10.1007/978-981-99-7356-9_7. URL: https://doi.org/10.1007/978-981-99-7356-9%5C_7.

[22] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun-Hee Lee, and Keewoo Lee. "Numerical Method for Comparison on Homomorphically Encrypted Numbers". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11922. Lecture Notes in Computer Science. Springer, 2019, pp. 415–445. DOI: 10.1007/978-3-030-34621-8_15. URL: https://doi.org/10.1007/978-3-030-34621-8%5C_15.

[23] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. "Powerformer: Efficient and High-Accuracy Privacy-Preserving Language Model with Homomorphic Encryption". In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*. Ed. by Wanxiang Che, Joyce

Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar. Association for Computational Linguistics, 2025, pp. 11090–11111. DOI: 10.18653/V1/2025.ACL-LONG.543. URL: https://doi.org/10.18653/v1/2025.acl-long.543.

[24] Yiteng Peng, Zhibo Liu, Dongwei Xiao, and Shuai Wang. "Fiona: Accelerating FHE Inference with Packing-Aware Ternary Weights". In: *CoRR* abs/2609.35352 (2026). DOI: 10.48550/ARXIV.2609.35352. arXiv: 2609.35352. URL: https://doi.org/10.48550/arXiv.2609.35352.

[25] Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. "DaCapo: Automatic Bootstrapping Management for Efficient Fully Homomorphic Encryption". In: *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/cheon.

[26] Aleksandar Krastev, Nikola Samardzic, Simon Langowski, Srinivas Devadas, and Daniel Sanchez. "A Tensor Compiler with Automatic Data Packing for Simple and Efficient Fully Homomorphic Encryption". In: *Proc. ACM Program. Lang.* 8.PLDI (June 2024). DOI: 10.1145/3656382. URL: https://doi.org/10.1145/3656382.

[27] Austin Ebel, Karthik Garimella, and Brandon Reagen. "Orion: A Fully Homomorphic Encryption Framework for Deep Learning". In: *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2025, Rotterdam, Netherlands, 30 March 2025 - 3 April 2025*. Ed. by Lieven Eeckhout, Georgios Smaragdakis, Katai Liang, Adrian Sampson, Martha A. Kim, and Christopher J. Rossbach. ACM, 2025, pp. 734–749. DOI: 10.1145/3676641.3716008. URL: https://doi.org/10.1145/3676641.3716008.

[28] Alessandro Zirilli, Davide Marincione, Evgenios M. Kornaropoulos, Giuseppe Ateniese, and Emanuele Rodolà. "HEAT: Faster Fully Homomorphic Inference via Approximations-Weights Co-Adaptation". In: *CoRR* abs/2609.01730 (2026). DOI: 10.48550/ARXIV.2609.01730. arXiv: 2609.01730. URL: https://doi.org/10.48550/arXiv.2609.01730.

[29] Yan Liu, Jianxin Lai, Long Li, Tianxiang Sui, Linjie Xiao, Peng Yuan, Xiaojing Zhang, Qing Zhu, Wenguang Chen, and Jingling Xue. "ReSBM: Region-based Scale and Minimal-Level Bootstrapping Management for FHE via Min-Cut". In: *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1, ASPLOS 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*. Ed. by Lieven Eeckhout, Georgios Smaragdakis, Kaitai Liang, Adrian Sampson, Martha A. Kim, and Christopher J. Rossbach. ACM, 2025, pp. 924–939. DOI: 10.1145/3669940.3707276. URL: https://doi.org/10.1145/3669940.3707276.

[30] Yuhang Fan, Yusi Chen, Kanyu Ye, and Zhuoran Ji. "An Open-Source End-to-End FHE Implementation for Privacy-Preserving Llama 3 8B Inference". In: *CoRR* abs/2609.12378 (2026). DOI: 10.48550/ARXIV.2609.12378. arXiv: 2609.12378. URL: https://doi.org/10.48550/arXiv.2609.12378.

[31] Siddharth Jayashankar, Joshua Kim, Michael B. Sullivan, Wenting Zheng, and Dimitrios Skarlatos. "A Scalable Multi-GPU Framework for Encrypted Large-Model Inference". In: *CoRR* abs/2512.11269 (2025). DOI: 10.48550/ARXIV.2512.11269. arXiv: 2512.11269. URL: https://doi.org/10.48550/arXiv.2512.11269.

[32] Leo de Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. "EncryptedLLM: Privacy-Preserving Large Language Model Inference via GPU-Accelerated Fully Homomorphic Encryption". In: *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. Vol. 267. Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL: https://proceedings.mlr.press/v267/de-castro25a.html.

[33] Xiaoqian Jiang, Miran Kim, Kristin E. Lauter, and Yongsoo Song. "Secure Outsourced Matrix Computation and Application to Neural Networks". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 1209–1222.

[34] Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. "Plaintext-Ciphertext Matrix Multiplication and FHE Bootstrapping: Fast and Fused". In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference,*

Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part III. Vol. 14922. Springer, 2024, pp. 387–421. DOI: 10.1007/978-3-031-68382-4_12.

[35] Jung Hee Cheon, Minsik Kang, and Junho Lee. "Fast Batch Matrix Multiplication in Ciphertexts". In: *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part II*. Ed. by Nadia Heninger and Mike Rosulek. Vol. 16801. Springer, 2026, pp. 558–590. DOI: 10.1007/978-3-032-35374-0_18.

[36] Jai Hyun Park. "Ciphertext-Ciphertext Matrix Multiplication: Fast for Large Matrices". In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques Madrid, Spain, May 4-8, 2025, Proceedings, Part VIII*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15608. Springer, 2025, pp. 153–180. DOI: 10.1007/978-3-031-91101-9_6.

[37] Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. *modded-nanogpt: Speedrunning the NanoGPT baseline*. 2024. URL: https://github.com/KellerJordan/modded-nanogpt.

[38] Serdar Gülbahar, Bishwas Mandal, Akshay Vegesna, and Samip Dahal. *Noise: Speedrun for pretraining without backpropagation*. 2026. URL: https://github.com/qlabs-eng/noise.

[39] Jianhang Xie, Sicheng Tan, Vishnu Naresh Boddeti, and Zhichao Lu. *ATLAS: Automated Approximation of Transformers for Efficient Homomorphic Inference in One Hour*. 2026. arXiv: 2607.23478 [cs.CR]. URL: https://arxiv.org/abs/2607.23478.

[40] Ahmad Al Badawi, Andreea Alexandru, Yuriy Polyakov, and Vinod Vaikuntanathan. "SoK: Private LLM Inference using Approximate Homomorphic Encryption". In: *IACR Cryptol. ePrint Arch.* 2026 (2026), p. 935. URL: https://eprint.iacr.org/2026/935.

[41] Ahmad Al Badawi et al. "OpenFHE: Open-Source Fully Homomorphic Encryption Library". In: *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022*. Ed. by Michael Brenner, Anamaria Costache, and Kurt Rohloff. ACM, 2022, pp. 53–63. DOI: 10.1145/3560827.3563379. URL: https://doi.org/10.1145/3560827.3563379.

[42] *Microsoft SEAL (release 4.7)*. https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA. Oct. 2026.

[43] *Lattigo v6*. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. Aug. 2024.

[44] Hao Yang, Shiyu Shen, Wangchen Dai, Lu Zhou, Zhe Liu, and Yunlei Zhao. "Phantom: A CUDA-Accelerated Word-Wise Homomorphic Encryption Library". In: *IEEE Trans. Dependable Secur. Comput.* 21.5 (2024), pp. 4895–4906. DOI: 10.1109/TDSC.2024.3363900. URL: https://doi.org/10.1109/TDSC.2024.3363900.

[45] Ali Sah Özcan and Erkay Savas. "HEonGPU: a GPU-based Fully Homomorphic Encryption Library 1.0". In: *IACR Cryptol. ePrint Arch.* 2024 (2024), p. 1543. URL: https://eprint.iacr.org/2024/1543.

[46] Shengyu Fan, Zhiwei Wang, Weizhi Xu, Rui Hou, Dan Meng, and Mingzhe Zhang. "TensorFHE: Achieving Practical Computation on Encrypted Data Using GPGPU". In: *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2023, Montreal, QC, Canada, February 25 - March 1, 2023*. IEEE, 2023, pp. 922–934. DOI: 10.1109/HPCA56546.2023.10071017. URL: https://doi.org/10.1109/HPCA56546.2023.10071017.

[47] Ahmet Can Mert, Aikata, Sunmin Kwon, Youngsam Shin, Donghoon Yoo, Yongwoo Lee, and Sujoy Sinha Roy. "Medha: Microcoded Hardware Accelerator for computing on Encrypted Data". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 463–500. DOI: 10.46586/TCHES.V2023.I1.463-500. URL: https://doi.org/10.46586/tches.v2023.i1.463-500.

[48] Aikata, Ahmet Can Mert, Sunmin Kwon, Maxim Deryabin, and Sujoy Sinha Roy. "REED: Chiplet-based Accelerator for Fully Homomorphic Encryption". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.2 (2025), pp. 163–208. DOI: 10.46586/TCHES.V2025.I2.163-208. URL: https://doi.org/10.46586/tches.v2025.i2.163-208.

[49] Zikai Zhou, William Seo, Edward Chen, Alex Ozdemir, Fraser Brown, and Wenting Zheng. "Orbit: Optimizing Rescale and Bootstrap Placement with Integer Linear Programming Techniques for Secure Inference". In: *IACR Cryptol. ePrint Arch.* 2026 (2026), p. 213. URL: https://eprint.iacr.org/2026/213.