Finding the Sweet Spot: Speed vs. Fidelity in Encrypted LLMs via Proxy Simulation

Rajan Savani
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
rsavani@hawk.illinoistech.edu

Anwar Benhnini
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
abenhnini@hawk.illinoistech.edu

André Bauer
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
abauer7@illinoistech.edu

2025 IEEE International Conference on eScience (eScience) | 979-8-3315-9145-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/eScience65000.2025.00060

***Abstract*—Fully homomorphic encryption (FHE) enables arithmetic operations to be performed directly on ciphertext, but this encrypted arithmetic is orders of magnitude slower than plaintext. When securing a large language model (LLM) using the CKKS encryption scheme, the slowdown is compounded by the need to replace each non-linear layer with a polynomial, which also injects approximation noise that can erode accuracy. Higher-degree polynomials reduce this error yet increase multiplicative depth and latency. Therefore, designers need to pinpoint where additional depth no longer pays off. To this end, we introduce a lightweight noise-injection proxy that measures CKKS residuals once, stores them, and replays the noise inside the plaintext model while sweeping polynomial degrees. Our proxy produces an accuracy-versus-latency curve and reveals the "sweet spot": The polynomial degree beyond which extra depth yields only marginal accuracy gains for a steep time cost. This curve guides developers in choosing an appropriate polynomial degree, eliminating the need for time-consuming encryption benchmarks.**

***Index Terms*—homomorphic encryption, large language models, approximation, privacy-preserving inference, simulation**

I. Introduction

Large language models (LLMs) are finding their way into privacy-sensitive domains such as healthcare, finance, and law. Existing projects such as PETALS distribute inference across peers to cut hardware costs [1], but they do so without encryption, leaving user data exposed. Fully homomorphic encryption (FHE) offers a promising solution by enabling servers to perform computations directly on encrypted data, ensuring user privacy throughout the process. Unfortunately, current FHE deployments remain slow: Running GPT-2 entirely on the CPU can take hours per encrypted query [2].

As the CKKS encryption scheme supports only addition and multiplication, every non-linear layer in an must be approximated by a polynomial. Deeper polynomials improve fidelity but demand larger multiplicative depth, which lengthens runtime and amplifies noise. Exhaustively benchmarking this tradeoff is prohibitively expensive during model design.

We present a lightweight *noise-injection proxy* that helps LLM developers explore this design space rapidly. We measure once how CKKS noise perturbs each layer, store those residuals, and then replay them inside the plaintext model while sweeping polynomial degrees. Based on the accuracy-vs-latency curve, designers can identify candidate "sweet spots", and decide whether additional depth is worth the cost.

Our contributions are three-fold: (i) A reusable proxy that estimates the loss of accuracy induced by CKKS without running full encryption; (ii) An empirical study of the accuracy-vs-latency trade-off on eight NLP tasks, showing how designers can identify their own operating point or push deeper when fidelity matters most; and (iii) A detailed workflow that others can replicate to accelerate encrypted LLM development.

II. Methodology

We present the three steps of the workflow, enabling any encrypted-LLM designer to replicate the proxy simulation.

Step 1: Swap Non-linear Layers for Polynomials. CKKS can evaluate only additions and multiplications, so every activation must be turned into a plain polynomial. We follow the implementation used in [2] and reuse their approximations for GeLU and LayerNorm. For the Softmax function we approximate $\exp(x)$ on $[-4, 4]$ with a Chebyshev polynomial, sweeping degrees $d = 2 \dots 9$ to see how latency scales. These choices serve as examples; developers are free to choose their own polynomials and degrees without modifying the rest of the workflow.

Step 2: Measure CKKS Noise Once. To capture the noise introduced by CKKS, forward hooks on GPT-2 are utilized. These hooks allows capturing about 10^5 pre-activation values per operator. For each value we compute the polynomial approximation $\varepsilon_{\text{poly}}$ residuals and the $\varepsilon_{\text{ckks}}$ residuals as follows

$$\varepsilon_{\text{poly}} = P_d(x) - f(x) \text{ and } \varepsilon_{\text{ckks}} = \tilde{P}_d(x) - P_d(x),$$

where $f(x)$ is the non-linear layer, $P_d(x)$ the plaintext polynomial, and $\tilde{P}_d(x)$ the polynomial encrypted with the OpenFHE library. The residuals are stored in memory-mapped files.

Step 3: Inject Noise and Benchmark. In the last step, the user has to choose a benchmarking dataset and run it (see Section III as an example). During plaintext inference i.i.d. samples are drawn from the stored residual pools and added as tensor to activations. This way, the compounded effect throughout the whole network can be assessed. As the model weights stay untouched, the proxy runs almost as fast as the original network.

2325-3703/25/$31.00 ©2025 IEEE
DOI 10.1109/eScience65000.2025.00060

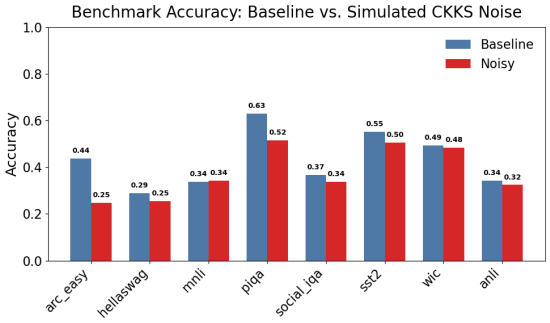

Fig. 1. Accuracy on eight NLP tasks. Blue bars: plaintext baseline. Red bars: proxy model with degree 7 Softmax polynomial approximation.

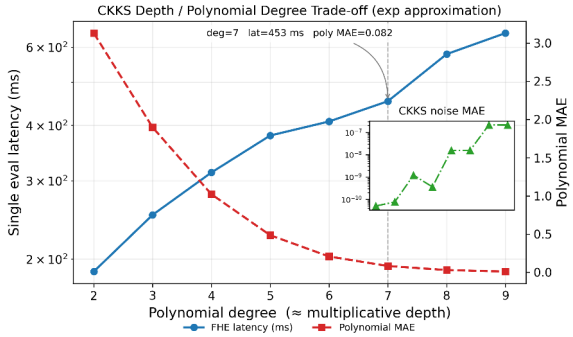

Fig. 2. Latency (left) and MAE (right) versus polynomial degree for the Softmax exponential. CKKS noise (green) is orders of magnitude smaller than the polynomial error.

III. Results

We evaluate baseline and noisy models with `lm-eval-harness` [3] on eight NLP tasks: ARC-Easy [4], HellaSwag [5], PIQA [6], Social IQa [7], MNLI [8], SST-2 [9], ANLI [10], and WIC [11]. The entire run took $\sim$3 hours on a single A6000 Ada. Running the benchmark with fully encrypted CKKS inference would take on the order of a week.

Degrees $d = 2, \ldots, 9$ are tested. For each d we log task-level accuracy and the encrypted polynomial evaluation latency, producing the error–runtime curves shown in Figure 2.

Figure 1 shows that injecting realistic CKKS noise drops average accuracy by 5.4%. The largest loss is 19% on ARC-Easy. MNLI even gains a small margin, implying that noise is not uniformly harmful.

Figure 2 plots the mean absolute error (MAE) and CKKS polynomial evaluation runtime as degree grows. Error falls quickly and improves by under 0.01 beyond degree 7, while latency rises about 1.3× per additional degree. Designers can decide whether that extra depth and the week-scale encryption cost are worthwhile for their use case.

IV. Discussion

The proxy shows that encrypted LLM trade-offs can be explored in the magnitude of hours instead of weeks. For the Softmax exponential, a degree-7 cuts encrypted latency by roughly 30 % compared with degree 8, yet the polynomial MAE improves by less than 0.01, signaling clear diminishing returns. Latency-critical services, such as conversational agents, can select this operating point. Precision-focused settings, such as medical analysis, may accept the slower circuit to gain the last fraction of a percent in accuracy.

Design flexibility. Since residuals are measured once and then replayed at negligible cost, researchers can test alternative activation ranges, activation functions, or quantization schemes without re-running CKKS. The same noise files can be shared across projects, reducing duplicated effort and lowering the barrier to encrypted inference research.

Bootstrapping and I/O. The proxy models per-layer noise only. It does not account for bootstrapping or network transfer, which may dominate end-to-end latency in large deployments.

Input distribution shift. Residuals were collected from random valid token IDs, not from task-specific inputs. A very different token distribution could change the activation ranges and therefore the noise pattern.

Ground-truth calibration. Latency estimates for full CKKS inference are extrapolated from small pilot runs. A complete encrypted benchmark is still needed to confirm absolute timings and further calibrate the proxy.

V. Conclusion

We introduced a noise-injection proxy that predicts CKKS accuracy loss in minutes instead of days. Applied to GPT-2, the workflow maps the full speed–accuracy trade-off and highlights usable operating regions such as the degree 7 example discussed earlier. The proxy reduces the evaluation time from roughly a week of end-to-end encryption to three hours on a single RTX A6000 Ada GPU. Beyond this case study, the same procedure can guide depth selection in larger models, different activation ranges, or hybrid precision schemes. Future work will extend the proxy to account for bootstrapping overhead and will validate its predictions by benchmarking against full, end-to-end CKKS inference.

Acknowledgments

This work is supported in part by the National Science Foundation OAC-2150500 award.

References

[1] A. Borzunov, D. Baranchuk, T. Dettmers, M. Ryabinin, Y. Belkada, A. Chumachenko, P. Samygin, and C. Raffel, "Petals: Collaborative inference and fine-tuning of large models," *arXiv preprint arXiv:2209.01188*, 2023.
[2] L. de Castro, D. Escudero, A. Agrawal, A. Polychroniadou, and M. Veloso, "EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption," in *Forty-second International Conference on Machine Learning*, 2025.
[3] "lm-eval-harnes," https://github.com/EleutherAI/lm-evaluation-harness.
[4] "ARC-Easy," https://huggingface.co/datasets/allenai/ai2_arc.
[5] "HellaSwag," https://huggingface.co/datasets/Rowan/hellaswag.
[6] "PIQA," https://huggingface.co/datasets/ybisk/piqa.
[7] "Social IQa," https://huggingface.co/datasets/allenai/social_i_qa.
[8] "MNLI," https://huggingface.co/datasets/nyu-mll/multi_nli.
[9] "SST-2," https://huggingface.co/datasets/stanfordnlp/sst2.
[10] "ANLI," https://huggingface.co/datasets/facebook/anli.
[11] "WiC," https://huggingface.co/datasets/sapienzanlp/wic.