

Transformer-based Language Models and Homomorphic Encryption: an intersection with BERT-tiny

Lorenzo Rovida¹ Alberto Leporati¹

¹ University of Milano-Bicocca
Department of Informatics, Systems and Communication
Milan, Italy

10th ACM International Workshop on Security and Privacy Analytics (IWSPA '24)
June 21, 2024
Porto, Portugal

Agenda

1. Introduction

- Motivations
- Fully Homomorphic Encryption
- BERT
- Related works

2. Preliminaries

- Vector–matrix multiplication
- Vectors wrapping
- Non–linear functions approximation

3. Methodology

- Encrypted circuit

4. Experiments

5. Conclusion and future work

Introduction

Motivations

- After their introduction, Large Language Models (LLMs) have significantly impacted various fields.
- They enable advanced natural language understanding and generation, facilitating more intuitive and effective communication with machines.
- What about the privacy of these models?

Motivations

- According to the terms of use of a very famous LLM service:

We retain certain data from your interactions with us, but we take steps to reduce the amount of personal information in our training datasets before they are used to improve and train our models.

Motivations

- According to the terms of use of a very famous LLM service:

We retain certain data from your interactions with us, but we take steps to reduce the amount of personal information in our training datasets before they are used to improve and train our models.



Motivations

- A possible solution would be to send user data in an encrypted state, without giving the service provider the key to decrypt
- How can the service provider still be able to offer the service?

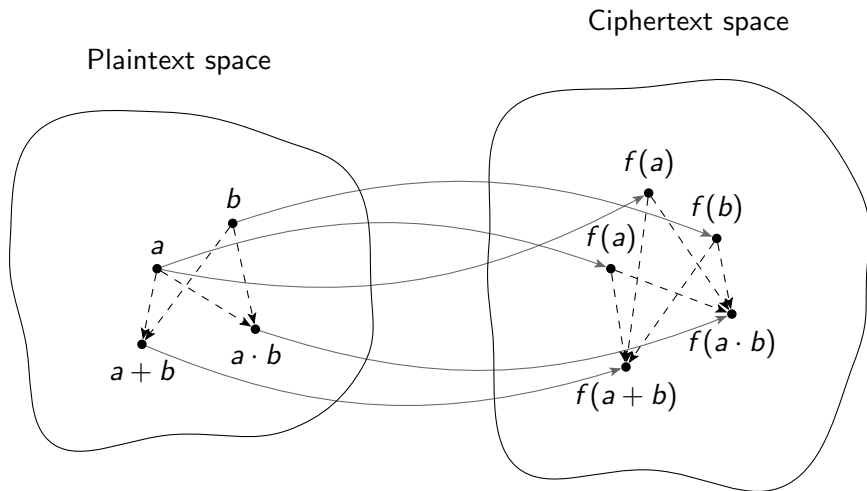
Fully Homomorphic Encryption

- Fully Homomorphic Encryption (FHE) is a cryptographic primitive that allows to encrypt data in such a way that operations performed on ciphertexts are reflected as the same operations on corresponding plaintexts.

$$a + b = \text{dec}(\text{enc}(a) + \text{enc}(b))$$

$$a \times b = \text{dec}(\text{enc}(a) \times \text{enc}(b))$$

Fully Homomorphic Encryption



Fully Homomorphic Encryption

- Almost all FHE schemes are based on the hardness of lattice problems and on variations of the Learning with Errors (LWE) problem
- A ciphertext is typically a (R)LWE instance which represents a lattice point with an error in it.
- Each operation between ciphertexts increases the amount of error
- A *bootstrapping* procedure is evaluated to decrease the error (i.e., an homomorphic evaluation of the decryption circuit)

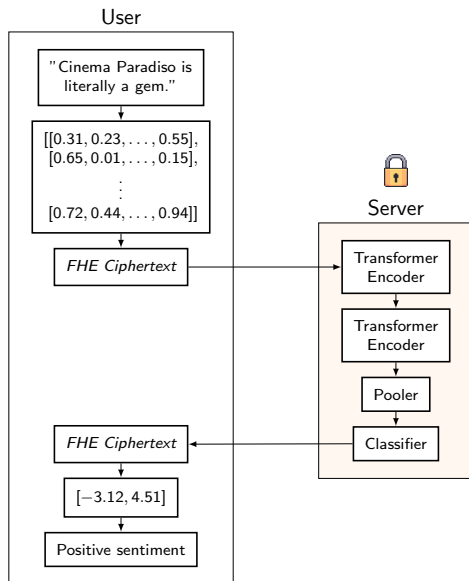
- The Bidirectional Encoder Representations from Transformers (BERT) model was introduced in 2018 by Google.
- It is usually used to extract features from sentences and to use these features with a classifier
- Google Search uses BERT to understand search queries better, but it is heavily used in many applications (Microsoft Office Suite, Facebook, Alexa, Grammarly...)

Related works

Some works tried to find an intersection between Language Models and FHE:

- [BHM+20] proposed *PrivFT*, a text classifier based on FHE that implemented the *fasttext* network.
- [LKP+22] proposed a method to classify BERT embeddings using an logistic regression model based on FHE, although the client themselves had to generate BERT embeddings.
- [ZLJ23] proposed *Primer*: a Transformer-based architecture based on FHE, MPC and GC. Nevertheless, it relies on many different layers and different kind of computations (both online and offline).

Proposed architecture



Preliminaries

It is possible to find different types of FHE schemes, according to the required application:

- DM/CGGI-like cryptosystems : blazing fast bootstrapping, the plaintext space is binary \mathbb{Z}_2 [DM15; CGGI16].
- BFV/BGV-like cryptosystems: slow bootstrapping, multiple values in a single ciphertext, the plaintext space is $\mathbb{Z}_q^{N/2}$ [FV12; BGV14].
- CKKS-like cryptosystems: slow bootstrapping, multiple values in a single ciphertext, the plaintext space is $\mathbb{C}^{N/2}$ [CKKS17].

- Since the BERT architecture is very large and based on a many non-linear functions, we implemented the circuit using the (RNS and error-reduced version [KPP22]) CKKS scheme.
- Additionally, we used the smallest possible BERT model, namely BERT-tiny [BDR21], which consists of $L = 2$ Transformer Encoders with $H = 128$ hidden units.
 - Pros: smaller model, faster inference
 - Cons: poorer performance

Before introducing the circuit, we highlight the main challenges in implementing such a model in CKKS:

- Matrix multiplications: this is a basic operation in Transformers, finding a way to quickly evaluate it is essential.
- Non-linear functions: the model relies on different activation functions (Softmax, GELU, tanh), evaluating them is not trivial since it is only possible to evaluate additions and multiplications $(+, \cdot)$.

Vector–matrix multiplication

Remark

In CKKS, a ciphertext represents vector in $C^{N/2}$, and operations are performed slot-wise between pairs of elements.

Vector-matrix multiplication

Remark

In CKKS, a ciphertext represents vector in $C^{N/2}$, and operations are performed slot-wise between pairs of elements.

Remark

Elements can be rotated by i positions using the corresponding automorphism key.

Vector-matrix multiplication

Remark

In CKKS, a ciphertext represents vector in $C^{N/2}$, and operations are performed slot-wise between pairs of elements.

Remark

Elements can be rotated by i positions using the corresponding automorphism key.

We propose two simple algorithms for matrix multiplication:

- Expanded vector \times Row-wise matrix \rightarrow Repeated vector
- Repeated vector \times Column-wise matrix \rightarrow Expanded vector

To be used sequentially!

Vector-matrix multiplication

Expanded vector \times Row-wise matrix \rightarrow Repeated vector:

$$\begin{pmatrix} \boxed{w_{1,1} \ w_{1,2} \ \dots \ w_{1,n}} \\ \boxed{w_{2,1} \ w_{2,2} \ \dots \ w_{2,n}} \\ \vdots \\ \boxed{w_{n,1} \ w_{n,2} \ \dots \ w_{n,n}} \end{pmatrix} \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$$

$$\begin{pmatrix} \begin{matrix} a_1 & a_1 & \dots & a_1 & a_2 & a_2 & \dots & a_2 & \dots \end{matrix} \\ \begin{matrix} \boxed{w_{1,1} \ w_{1,2} \ \dots \ w_{1,n}} & \boxed{w_{2,1} \ w_{2,2} \ \dots \ w_{2,n}} & \dots \end{matrix} \end{pmatrix} \times \dots$$

$$\begin{pmatrix} \boxed{\Sigma} \ \boxed{\Sigma} \ \dots \ \boxed{\Sigma} \ \dots \ \dots \end{pmatrix}$$

Vector-matrix multiplication

Repeated vector \times Column-wise matrix \rightarrow Expanded vector:

$$\begin{pmatrix} \begin{matrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{matrix} \end{pmatrix} \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$$

$$\rightarrow \left(\begin{matrix} a_1 & a_2 & \dots & a_n & a_1 & a_2 & \dots & a_n & \dots \\ w_{1,1} & w_{2,1} & \dots & w_{n,1} & w_{1,2} & w_{2,2} & \dots & w_{n,2} & \dots \end{matrix} \right) \times \dots$$

$$= \left(\begin{matrix} \Sigma & \dots & \dots & \dots & \Sigma & \dots & \dots & \dots & \dots \end{matrix} \right)$$

Vectors wrapping

- These two algorithms, used sequentially, allows to quickly evaluate a vector–matrix multiplication with a single ciphertexts multiplication.
- Although in practice, many of these operations are performed (namely, one for each input token), so we require an algorithm to merge all the results in a single ciphertext.

Vectors wrapping

In case there are many Expanded vectors, wrap them as shown:

$$\begin{array}{l} a : \left(\boxed{a_1} \text{ } \boxed{a_1} \text{ } \boxed{a_1} \text{ } \boxed{a_2} \text{ } \boxed{a_2} \text{ } \boxed{a_2} \text{ } \boxed{a_3} \text{ } \boxed{a_3} \text{ } \boxed{a_3} \right) \\ b : \left(\boxed{b_1} \text{ } \boxed{b_1} \text{ } \boxed{b_1} \text{ } \boxed{b_2} \text{ } \boxed{b_2} \text{ } \boxed{b_2} \text{ } \boxed{b_3} \text{ } \boxed{b_3} \text{ } \boxed{b_3} \right) \\ c : \left(\boxed{c_1} \text{ } \boxed{c_1} \text{ } \boxed{c_1} \text{ } \boxed{c_2} \text{ } \boxed{c_2} \text{ } \boxed{c_2} \text{ } \boxed{c_3} \text{ } \boxed{c_3} \text{ } \boxed{c_3} \right) \\ \hline M : \left(\boxed{a_1} \text{ } \boxed{b_1} \text{ } \boxed{c_1} \text{ } \boxed{a_2} \text{ } \boxed{b_2} \text{ } \boxed{c_2} \text{ } \boxed{a_3} \text{ } \boxed{b_3} \text{ } \boxed{c_3} \right) \end{array}$$

Vectors wrapping

In case there are many Repeated vectors, wrap them as shown:

$$\begin{array}{l} a : \left(\boxed{a_1} \boxed{a_2} \boxed{a_3} \quad \boxed{a_1} \boxed{a_2} \boxed{a_3} \quad \boxed{a_1} \boxed{a_2} \boxed{a_3} \right) \\ b : \left(\boxed{b_1} \boxed{b_2} \boxed{b_3} \quad \boxed{b_1} \boxed{b_2} \boxed{b_3} \quad \boxed{b_1} \boxed{b_2} \boxed{b_3} \right) \\ c : \left(\boxed{c_1} \boxed{c_2} \boxed{c_3} \quad \boxed{c_1} \boxed{c_2} \boxed{c_3} \quad \boxed{c_1} \boxed{c_2} \boxed{c_3} \right) \\ \hline M : \left(\boxed{a_1} \boxed{a_2} \boxed{a_3} \quad \boxed{b_1} \boxed{b_2} \boxed{b_3} \quad \boxed{c_1} \boxed{c_2} \boxed{c_3} \right) \end{array}$$

Non-linear functions approximation

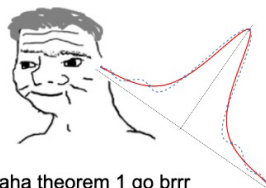
- The last challenge is to find a way to evaluate non-linear functions.
- In a Transformer Encoder there are many: $\exp(x)$, $\frac{1}{x}$, $\text{GELU}(x)$, ...

Non-linear functions approximation

- The last challenge is to find a way to evaluate non-linear functions.
- In a Transformer Encoder there are many: $\exp(x)$, $\frac{1}{x}$, $\text{GELU}(x)$, ...



Finding coefficients for a polynomial such that it well approximates a non-linear function is a complex task



haha theorem 1 go brrr

Non-linear functions approximation

Theorem (Chebyshev roots are “near-best” [Tre13])

Let f be a continuous function on $[-1, 1]$, let p denote its interpolant in n Chebyshev points and let p^ be its best degree n approximation with respect to the infinity norm. It holds that:*

$$\|f - p\|_{\infty} \leq \|f - p^*\|_{\infty} \cdot \left(2 + \frac{2}{\pi} \log(n)\right)$$

Non-linear function approximations

- Chebyshev polynomials are flexible and give a very good approximation.
- We already have algorithms (such as the one described in [CCS19], adapted from [PS73]) to evaluate them quickly.
- Remark that these approximations work in a fixed interval $[a, b]$, thus an empirical study on the input values must be performed.

Methodology

- The BERT-tiny circuit is composed of 2 encoders with different weights but identical structure.
- An encoder is composed of four blocks:
 - 1 Self-attention
 - 2 Self-output
 - 3 Intermediate
 - 4 Output

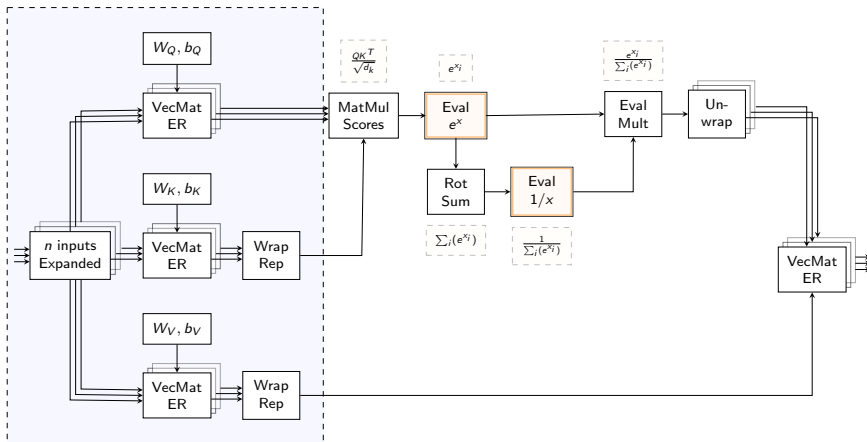
- The first block (Self-Attention) computes the following function:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

- where $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- and $Q, K, V \in \mathbb{R}^{128}$ are obtained as the result of a linear transformation, and $\sqrt{d_k} = 8$ can be precomputed.

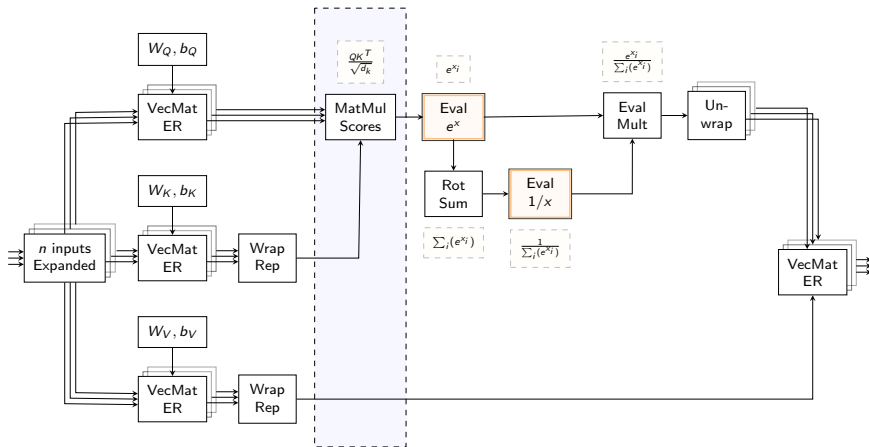
Self-attention

- The first part computes the Q, K, V matrices.



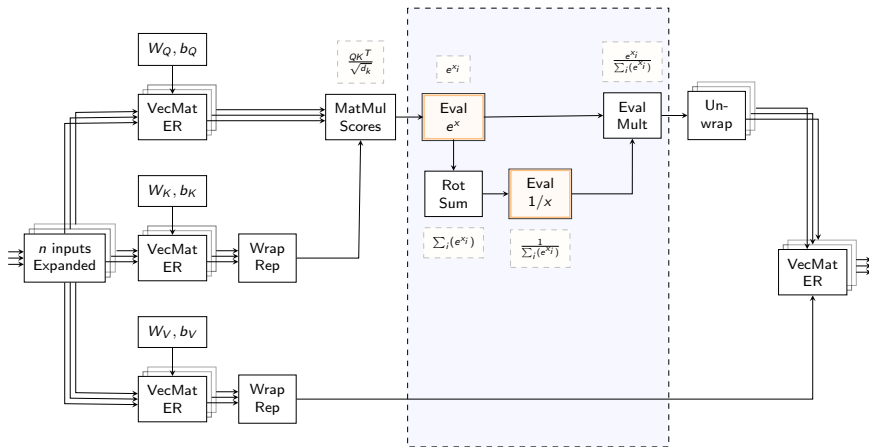
Self-attention

- The second part computes the Dot-product Attention (*scores*).



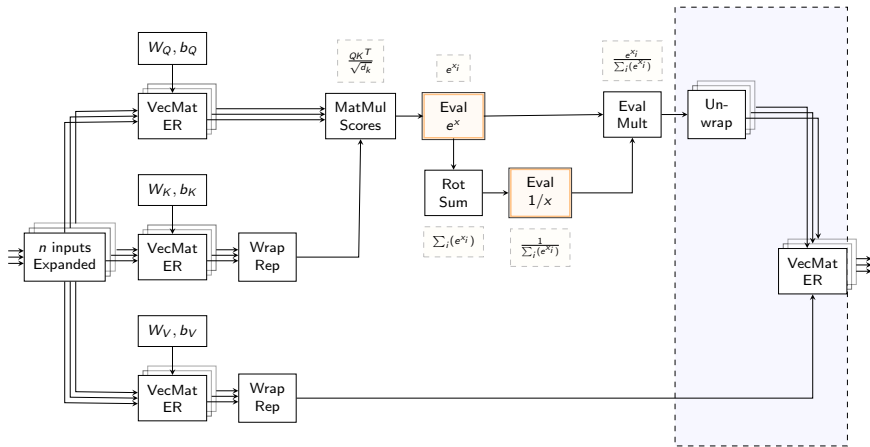
Self-attention

- The softmax function is computed by approximating e^x and $1/x$.



Self-attention

- The product with V concludes the evaluation of the Self-Attention.



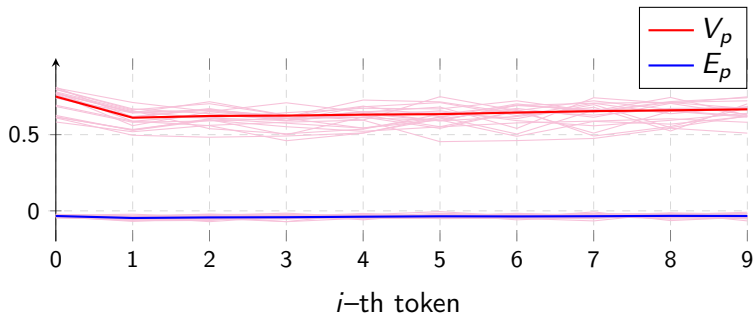
- The second block (Self-output) computes a linear transformation followed by a Layer normalization, defined as:

$$\text{LayerNorm}(X) = \frac{X_i - E[X]}{\sqrt{\text{Var}[X] + \varepsilon}} \cdot \gamma + \beta$$

- where $E[X]$, $\text{Var}[X]$ are the mean and the variance, respectively.
- This would require another approximation ($\frac{1}{\sqrt{x}}$).

Self-output

- We propose to skip this approximation, and to use constant values for mean and variance.
- This makes the layer equivalent to a linear transformation. As a cons, results will have some error.



Intermediate

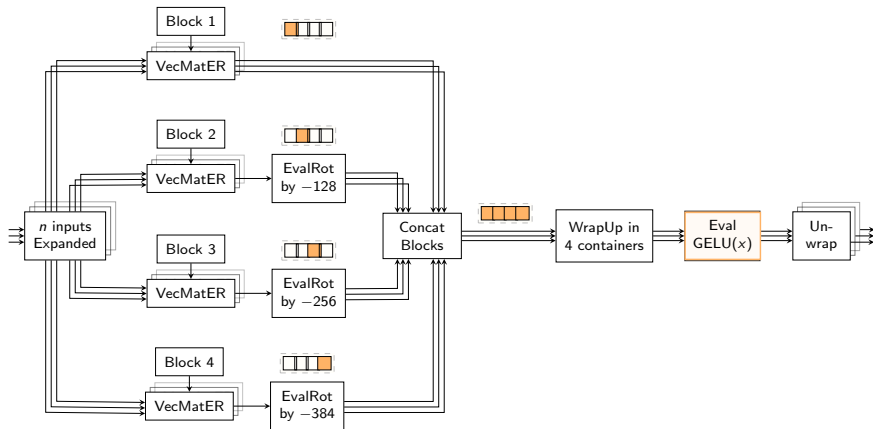
- The third block, called Intermediate, increases the size of the features from 128 to 512.
- After that, a $\text{GELU}(x)$ activation function is evaluated:

$$\text{GELU}(x) = x \cdot \frac{1}{2} \left(1 + \text{erf}(x/\sqrt{2}) \right)$$

- where $\text{erf}(x)$ is the error function.

Intermediate

- The linear transformation is executed four times ($4 \cdot 128 = 512$), wrapped up in (at most) four containers and $\text{GELU}(x)$ is evaluated.

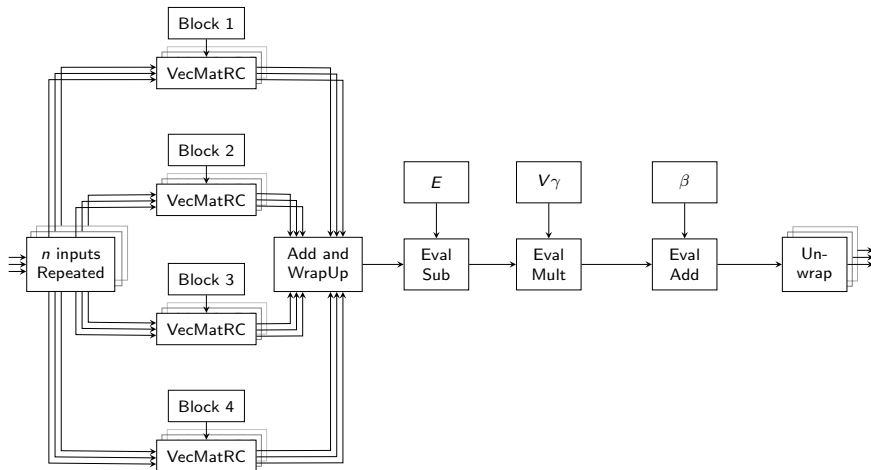


Output

- The last block, called Output, decreases the size of the features back to 128.
- After that, a Layer normalization is performed.
- As before, we use precomputed values for mean and variance.

Output

- Features are decreased back to 128 and a linear transformation is evaluated.



Experiments

Experiments

- The circuit has been implemented in the OpenFHE implementation [ABB+22] of the RNS-CKKS scheme [KPP22].
- The scheme is instantiated with the following parameters:
 - Ring of dimension $N = 2^{16}$
 - Ciphertext slots $s = N/4$
 - Precision factor $\Delta = 55$ bits
 - Moduli chain values $q_i = 52$ bits
 - Circuit depth $d = 24$
 - Ciphertext modulus $Q = 1767$ bits
 - Security bits $\lambda > 128$ bits (classical)

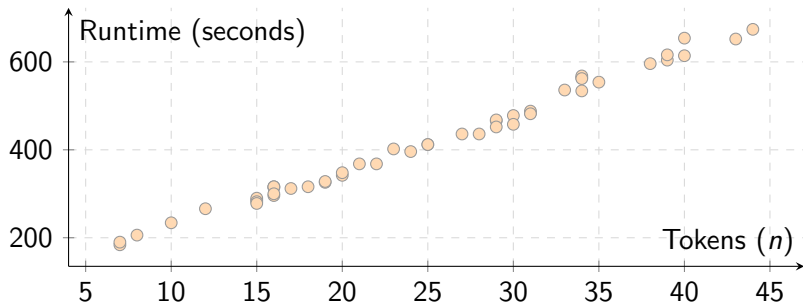
Evaluation – accuracy

- The circuit has been implemented using the weights of SST-2 fine-tuned plain model. Results refer to the SST-2 test set.

Model	Accuracy	Loss
BERT _{tiny}	0.837	(<i>reference</i>)
BERT _{tiny} with precomputed LN	0.815	0.022
FHE-BERT _{tiny}	0.790	0.047

Evaluation – timing

- The runtime has been evaluated on a ARM M1 Pro CPU.



Conclusion and future work

Conclusion and future work

- We presented a circuit that evaluates a Transformer-based language model.
- Results show that the inference is somewhat practical (hundreds of seconds on a laptop).
- Nevertheless, this can be considered as a starting point from where many optimizations can be performed
- It would be interesting to explore softmax-free implementations of Transformers

Conclusion and future work

- The circuit is freely accessible as an open-source repository.

The screenshot shows the GitHub repository for "FHE-BERT-Tiny". At the top, it indicates the repository is "Public" and shows interaction buttons: Pin, Unwatch (1), Fork (0), and Starred (4). Below this, the repository is set to the "main" branch with 1 branch and 0 tags. A search bar and "Add file" / "Code" buttons are present. The commit history table lists recent updates:

Commit Message	Author	Time
Update README.md	narger-ef	4 days ago
Update .gitignore	narger-ef	3 months ago
First commit	narger-ef	5 months ago
Refactor code	narger-ef	3 months ago
Test completato	narger-ef	5 months ago
Update main.cpp	narger-ef	2 days ago
Remove hidden folder	narger-ef	3 months ago
Create architecture.png	narger-ef	3 months ago


On the right, the "About" section describes the source code for the paper "Transformer-based Language Models and Homomorphic Encryption: an intersection with BERT-tiny", providing a DOI link and tags like "encrypted", "homomorphic-encryption", "privacy-preserving", "fhe", "fully-homomorphic-encryption", and "lim". It also shows repository statistics: 46 commits, MIT license, 4 stars, 1 watching, and 0 forks.





The "Releases" section states "No releases published" with a link to "Create a new release". The "Packages" section also states "No packages published" with a link to "Publish your first package".




The main content area displays the repository's README, titled "Transformer-based Language Models and Homomorphic Encryption: an intersection with BERT-tiny". A terminal window at the bottom shows the command `build - -zsh - 82x7`.




Conclusion and future work

Thank you!

 **FHE-BERT-Tiny** Public

  Unwatch 1  Fork 0  Starred 4

 main  1 Branch  0 Tags







 Go to file  Add file  Code

About

Source code for the paper
"Transformer-based Language Models and Homomorphic Encryption: an intersection with BERT-tiny"

dl.acm.org/doi/10.1145/3643651.365...











[encrypted](#) [homomorphic-encryption](#)
[privacy-preserving](#) [fhe](#)
[fully-homomorphic-encryption](#) [lim](#)



 Readme
 MIT license
 Activity
 4 stars
 1 watching
 0 forks

Releases

No releases published
[Create a new release](#)

narger-ef [Update main.cpp](#) s1f494a · 2 days ago 46 Commits

	imgs	Create archite		3 months ago
	notebooks	Remove hidden		3 months ago
	src	Update main.c		2 days ago
	weights-sst2	Test complet		5 months ago
	.gitattributes	First commit		5 months ago
	.gitignore	Update .gitign		3 months ago
	CMakeLists.txt	Refactor code		3 months ago
	LICENSE	First commit		5 months ago
	README.md	Update README.md		4 days ago

 **README**  MIT license

References I

- [ABB+22] Ahmad Al Badawi et al. “OpenFHE: Open-Source Fully Homomorphic Encryption Library”. In: *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. WAHC'22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 53–63.
- [BDR21] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. *Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics*. 2021. arXiv: 2110.01518 [cs.CL].
- [BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), pp. 1–36.

References II

- [BHM+20] Ahmad Al Badawi et al. “PrivFT: Private and Fast Text Classification With Homomorphic Encryption”. In: *IEEE Access* 8 (2020), pp. 226544–226556.
- [CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. “Improved Bootstrapping for Approximate Homomorphic Encryption”. In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 34–54. ISBN: 978-3-030-17656-3.
- [CGGI16] Ilaria Chillotti et al. “Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds”. In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3–33. ISBN: 978-3-662-53887-6.

References III

- [CKKS17] Jung Hee Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437. ISBN: 978-3-319-70694-8.
- [DM15] Léo Ducas and Daniele Micciancio. “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second”. In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 617–640. ISBN: 978-3-662-46800-5.
- [FV12] Junfeng Fan and Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. <https://eprint.iacr.org/2012/144>. 2012. URL: <https://eprint.iacr.org/2012/144>.

References IV

- [KPP22] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. “Approximate Homomorphic Encryption with Reduced Approximation Error”. In: *Topics in Cryptology – CT-RSA 2022*. Ed. by Steven D. Galbraith. Cham: Springer International Publishing, 2022, pp. 120–144. ISBN: 978-3-030-95312-6.
- [LKP+22] Garam Lee et al. “Privacy-Preserving Text Classification on BERT Embeddings with Homomorphic Encryption”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, USA, July 2022, pp. 3169–3175.

References V

- [PS73] Michael S. Paterson and Larry J. Stockmeyer. “On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials”. In: *SIAM Journal on Computing* 2.1 (1973), pp. 60–66. DOI: 10.1137/0202007. eprint: <https://doi.org/10.1137/0202007>. URL: <https://doi.org/10.1137/0202007>.
- [Tre13] L.N. Trefethen. *Approximation Theory and Approximation Practice*. Other Titles in Applied Mathematics. Siam, 2013.
- [ZLJ23] Mengxin Zheng, Qian Lou, and Lei Jiang. *Primer: Fast Private Transformer Inference on Encrypted Data*. 2023. arXiv: 2303.13679 [cs.CR].