Encrypted Image Classification with Low Memory Footprint using Fully Homomorphic Encryption

Lorenzo Rovida* Alberto Leporati*

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

4th Workshop on Artificial Intelligence and Cryptography (AICrypt 2024)
Eurocrypt 2024 @ ETH Zurich, Switzerland
May 26, 2024

Agenda

1 Introduction
- Motivations
- Fully Homomorphic Encryption (FHE)
- Related works

2 Methodology
- The CKKS scheme
- Different convolution approaches
- Our convolutions appraoch
- Chebyshev polynomials

3 Experiments
- Results

4 Conclusion

Introduction

Motivations

- Classifying images has become a standard and straightforward task, thanks to the advent of Deep Neural Networks (DNNs), and, in particular, of Convolutional Neural Networks (CNNs).
- In Europe, when online services deal with private user data, usually it is **protected** by the GDPR law [16].
- There is no effective protection of data during computations (e.g., cryptography?)

Motivations

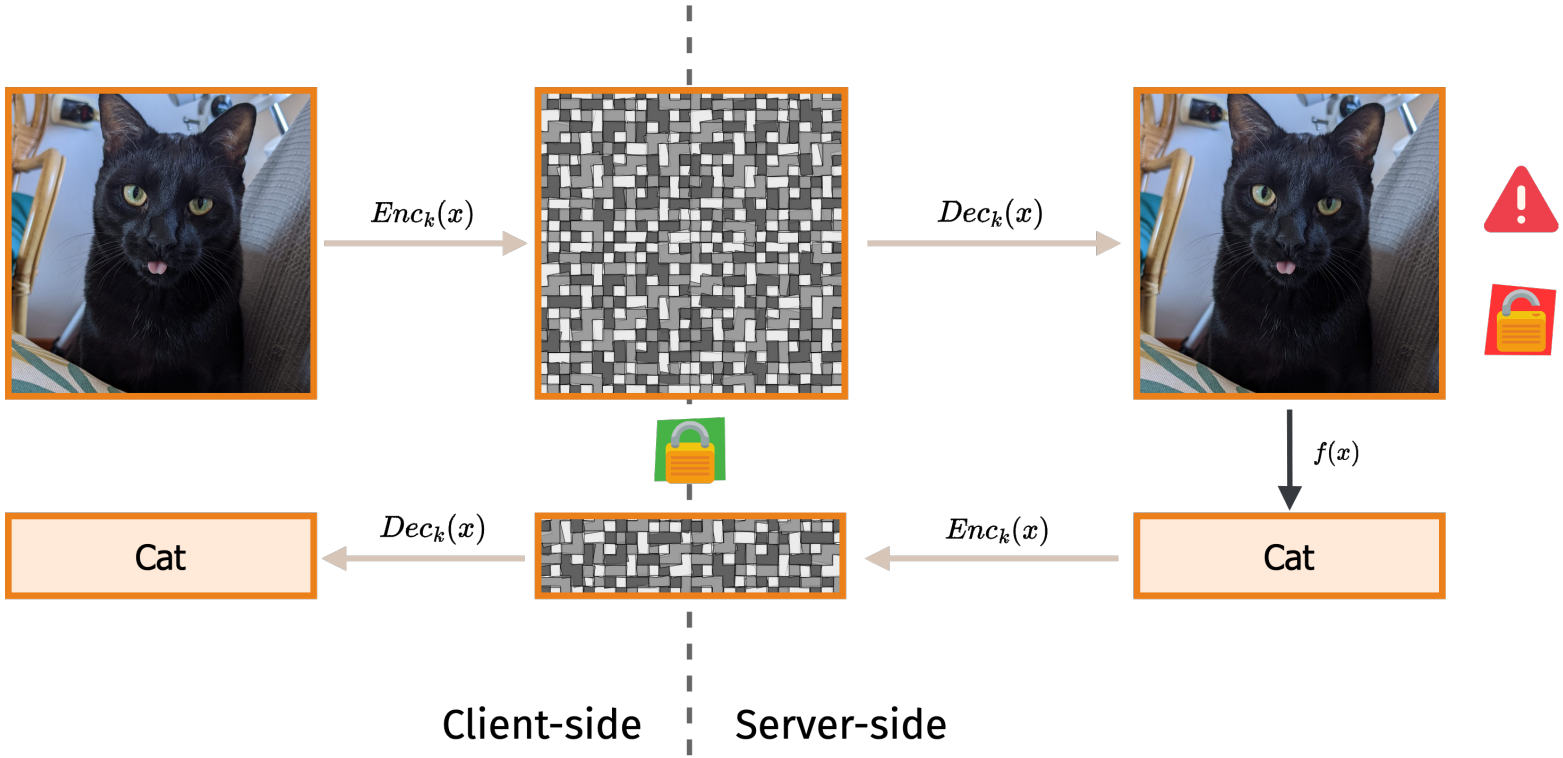
- Classifying images has become a standard and straightforward task, thanks to the advent of Deep Neural Networks (DNNs), and, in particular, of Convolutional Neural Networks (CNNs).
- In Europe, when online services deal with private user data, usually it is **protected** by the GDPR law [16].
- There is no effective protection of data during computations (e.g., cryptography?)

Motivations

- Classifying images has become a standard and straightforward task, thanks to the advent of Deep Neural Networks (DNNs), and, in particular, of Convolutional Neural Networks (CNNs).
- In Europe, when online services deal with private user data, usually it is **protected** by the GDPR law [16].
- There is no effective protection of data during computations (e.g., cryptography?)

Motivations

Let us think about a cloud service that classifies images:



Motivations

- Any service provider requires **plain data** in order to offer services.
- How can it offer the service without having access to the plain data?

Motivations

- Any service provider requires **plain data** in order to offer services.
- How can it offer the service without having access to the plain data?

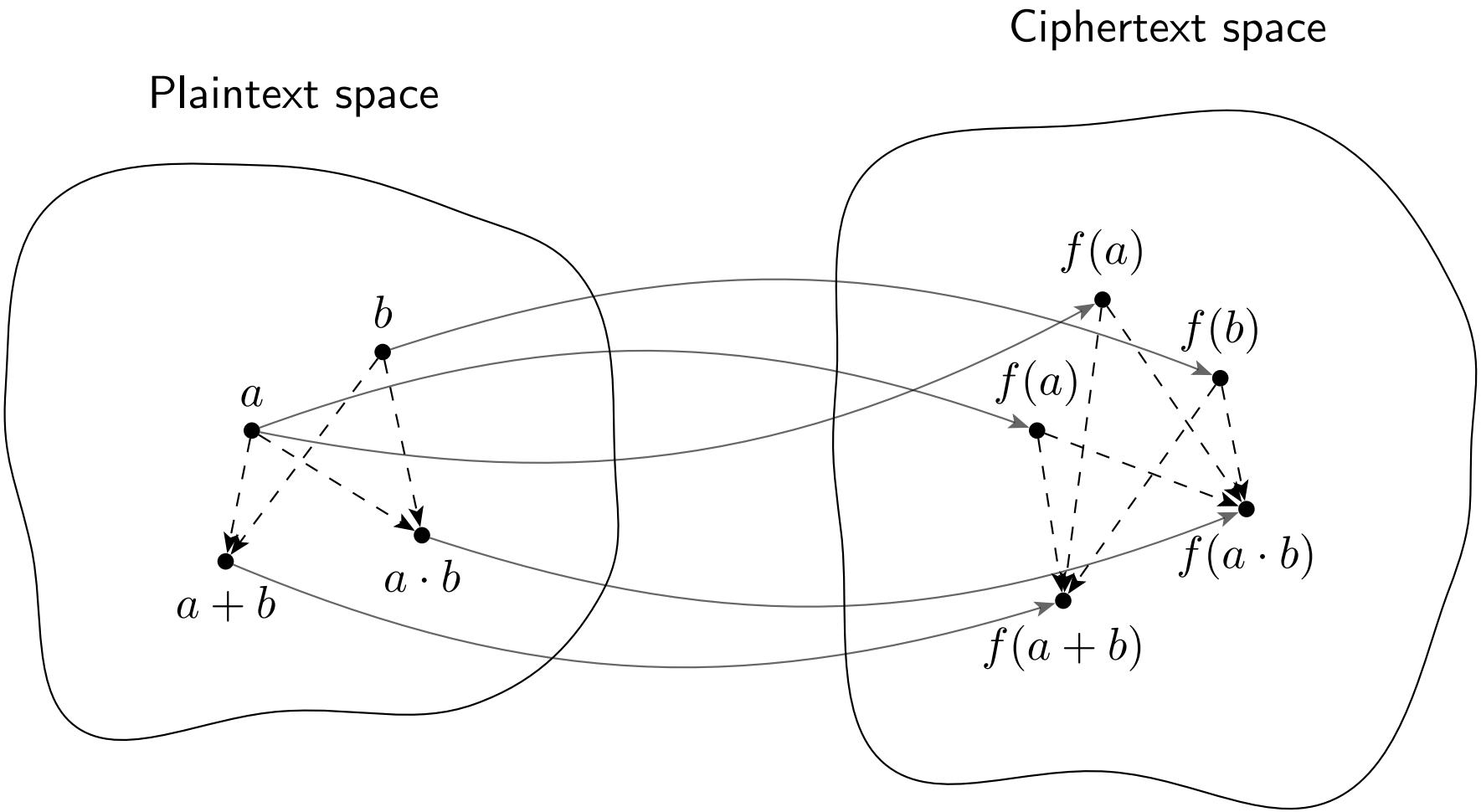
Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that enables computations on encrypted data.

Definition (Homomorphism)

A mapping between two similar algebraic structures which preserves the relational properties of elements in the two structures.
If f is a homomorphism, and $*, \circ$ are two corresponding operations in the two structures, $f(x * y) = f(x) \circ f(y)$.

Fully Homomorphic Encryption (FHE)



Fully Homomorphic Encryption (FHE)

- Informally, when the plaintext and ciphertext spaces are homomorphic, an operation on the ciphertext is "reflected" in the same operation applied to the plaintext.
- An encryption function f, homomorphic with respect to the addition operation, enables the following:

$$\underbrace{f(a) + f(b)}_{\text{Encrypted sum}} = f(\underbrace{a + b}_{\text{Plain sum}})$$

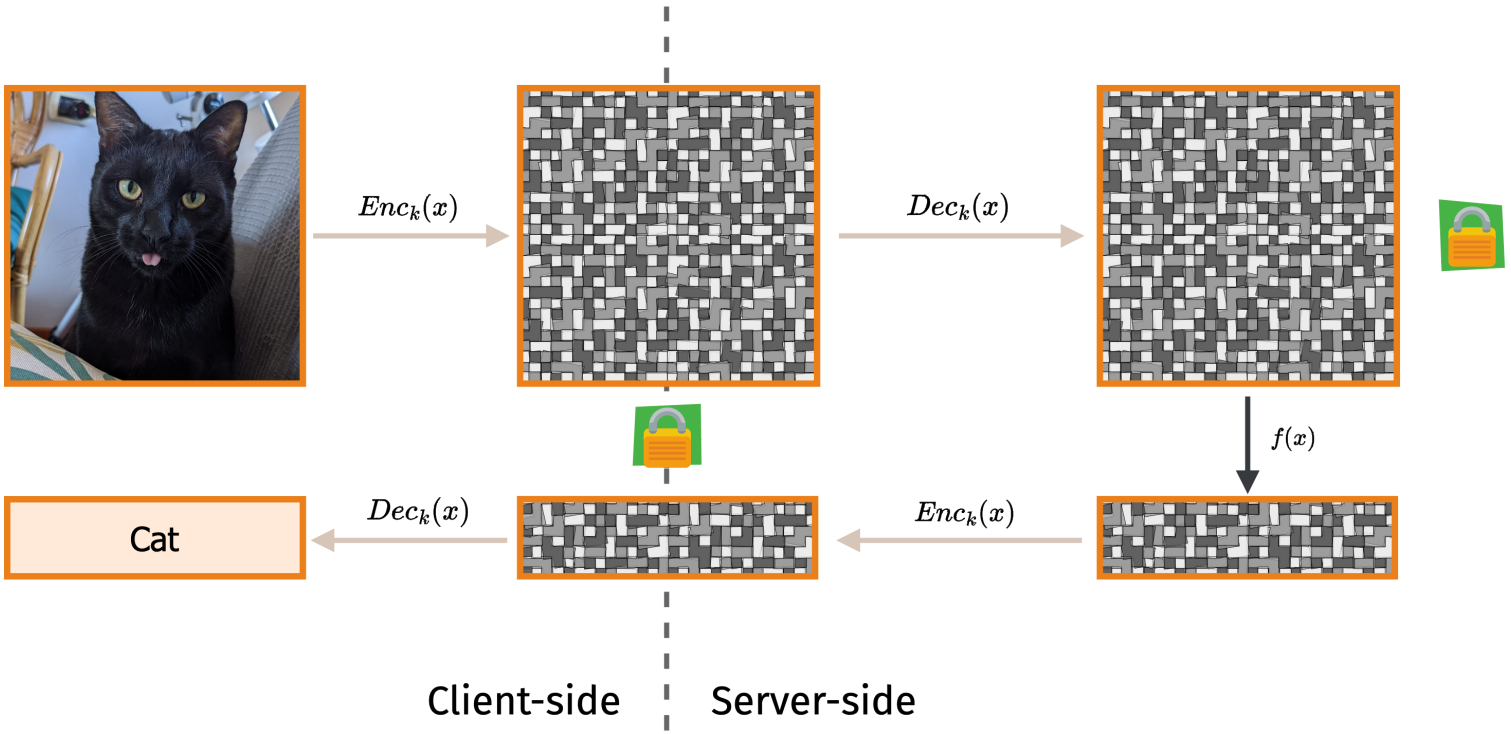
Fully Homomorphic Encryption (FHE)

- Informally, when the plaintext and ciphertext spaces are homomorphic, an operation on the ciphertext is "reflected" in the same operation applied to the plaintext.
- An encryption function f, homomorphic with respect to the addition operation, enables the following:

$$\underbrace{f(a) + f(b)}_{\text{Encrypted sum}} = f(\underbrace{a + b}_{\text{Plain sum}})$$

Fully Homomorphic Encryption (FHE)

Theoretically, by using FHE, the following architecture could be implemented:



- RQ: Is it possible to build a CNN using FHE operations?

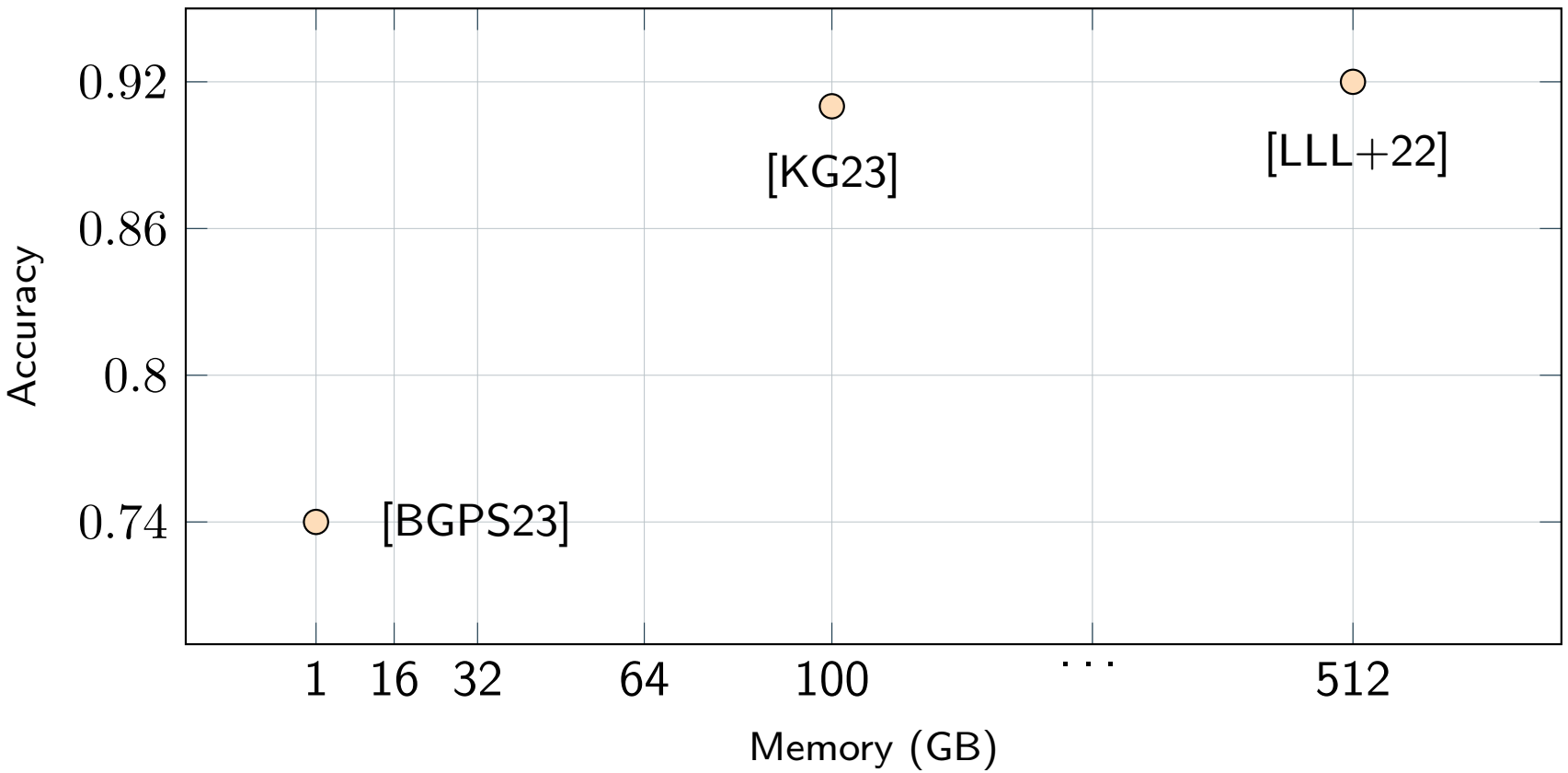
Related works

- There are works that implement a CNN using FHE [BGPS23; KG23; LLL+22]
- Nevertheless, they either have **large memory requirements** (512GB for [LLL+22], 100GB for [KG23]) or **poor performance** (1GB for [BGPS23] but 74% of accuracy on CIFAR-10)

Related works

- There are works that implement a CNN using FHE [BGPS23; KG23; LLL+22]
- Nevertheless, they either have **large memory requirements** (512GB for [LLL+22], 100GB for [KG23]) or **poor performance** (1GB for [BGPS23] but 74% of accuracy on CIFAR-10)

Related works



- RQ: Is it possible to build a CNN using FHE operations?

- RQ: Is it possible to build a CNN using FHE operations, *with a good tradeoff between memory requirements and accuracy?*

Methodology

Methodology

The main challenges are the following:

- Building a low-complexity circuit that performs convolutions.
- Finding an approach to approximate the ReLU(x) and the tanh(x) function using basic arithmetic operators $(+, \cdot)$.

The CKKS scheme

We propose to use the CKKS scheme [CKKS17], which is an *approximate* FHE scheme that encrypts vectors of complex numbers.

$$v \in \mathbb{C}^{N/2} \xrightarrow{\text{Encoding}} p \in \mathbb{Z}[X]/(X^N + 1)$$
$$\xrightarrow{\text{Encryption}} c \in \left(\mathbb{Z}_Q[X]/(X^N + 1)\right)^2$$

where N is a power of two.

The CKKS scheme

The scheme is said to be *approximate* because it encodes complex values in a fixed-point representation:

$$\text{Encode}(v, \Delta) = \sigma^{-1}\left(\lfloor \Delta \pi^{-1}(v) \rceil\right)$$

It holds that $\text{Decode}(\text{Decrypt}(\text{Encrypt}(\text{Encode}(v)))) \approx v$. The precision can be controlled with Δ.

The CKKS scheme

Operations are performed between ciphertexts and can be described with a Single Instruction, Multiple Results (SIMD) computational paradigm:

4	3	6	9	$\cdots$	8	3.1	0	$\circ$
1	2	4	1	$\cdots$	-3	2	1	$\approx$
$\downarrow$								
$4 \circ 1$	$3 \circ 2$	$6 \circ 4$	$9 \circ 1$	$\cdots$	$8 \circ -3$	$3.1 \circ 2$	$0 \circ 1$	

The CKKS scheme

- Add(x, y): given two ciphertexts x and y, performs the slot-wise addition between each pair of elements (in general, y can also be a plaintext).
- Mul(x, y): given two ciphertexts x and y, performs the slot-wise multiplication between each pair of elements.
- Rot$_i(x)$: given a ciphertext x, rotate its elements by i positions. Positive values of i indicate a rotation to the right, negative values to the left.

The CKKS scheme

- Each rotation by i positions requires a key which defines an automorphism that transforms the ciphertext in another ciphertext s.t. the encoded values are rotated by i.
- These keys are heavy! (hundreds of MBs per rotation index i when considering "practical" parameters).

Recall the main challenges:

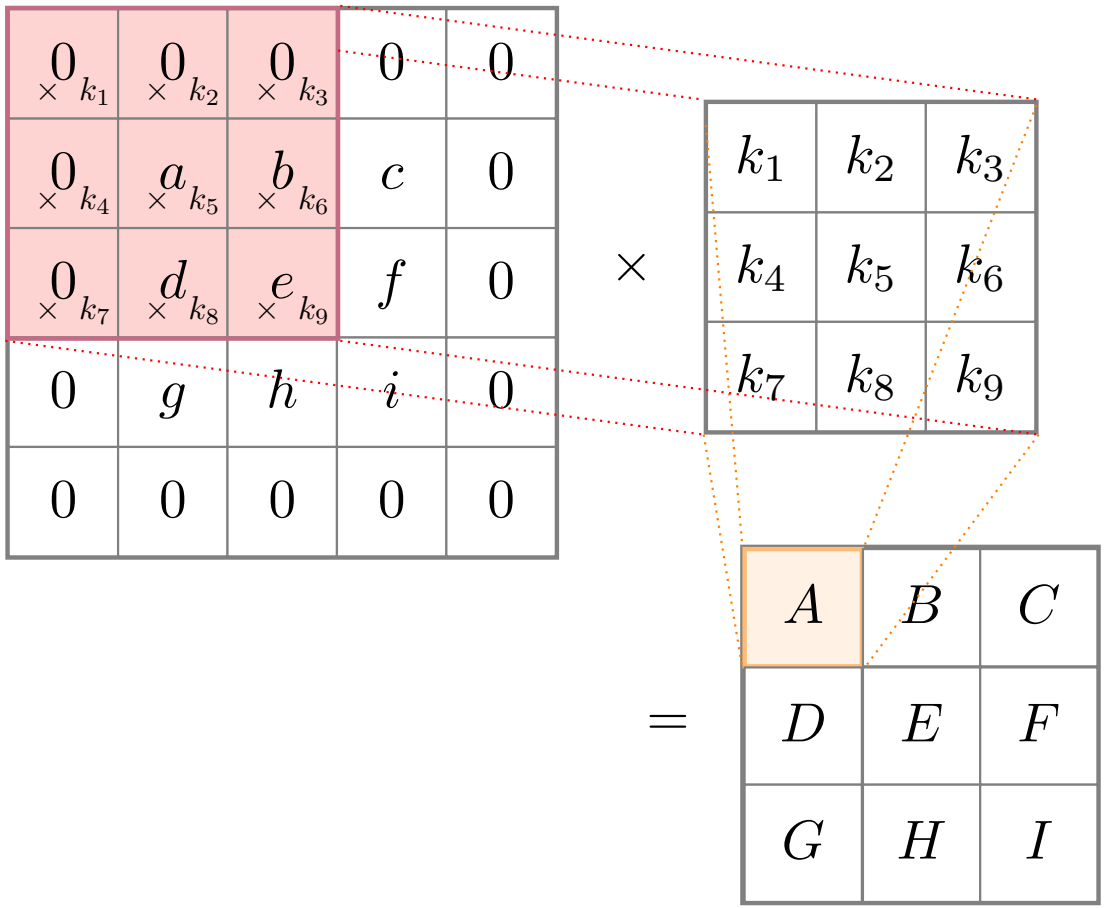
- Building a low-complexity circuit that performs convolutions.
- Finding an approach to approximate the ReLU(x) and the tanh(x) function using basic arithmetic operators $(+, \cdot)$.

Recall the main challenges:

- Building a low-complexity circuit that performs convolutions.
- Finding an approach to approximate the ReLU(x) and the tanh(x) function using basic arithmetic operators $(+, \cdot)$.

Different convolution approaches

Recall how a convolution works:



Different convolution approaches

- The CKKS cryptosystem encrypts vectors and performs slot-wise operations
- Classical ways to perform convolutions (based on matrices and, in general, on tensors) **can not be applied**!

Different convolution approaches

- The CKKS cryptosystem encrypts vectors and performs slot-wise operations
- Classical ways to perform convolutions (based on matrices and, in general, on tensors) **can not be applied**!

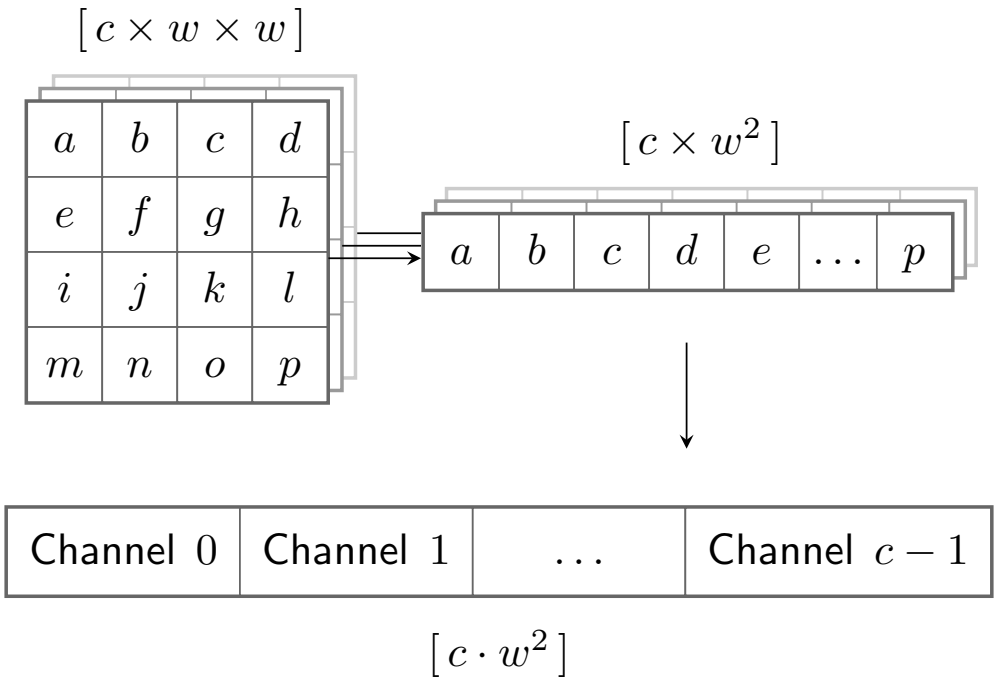
Different convolution approaches

In literature, we can find three different approaches:

- *Vector Encoding* [JVC18]: the channels are encoded in row-major order vectors and convolutions are performed by creating $k-1$ rotations of the image
- *Coefficients Encoding* [KG23]: the channels are encoded in polynomials and convolutions are performed as polynomial multiplications between the channels and the filters.
- *Spectral Encoding* [LLHJ20]: convolutions are evaluated in the frequency domain to reduce the number of rotations.

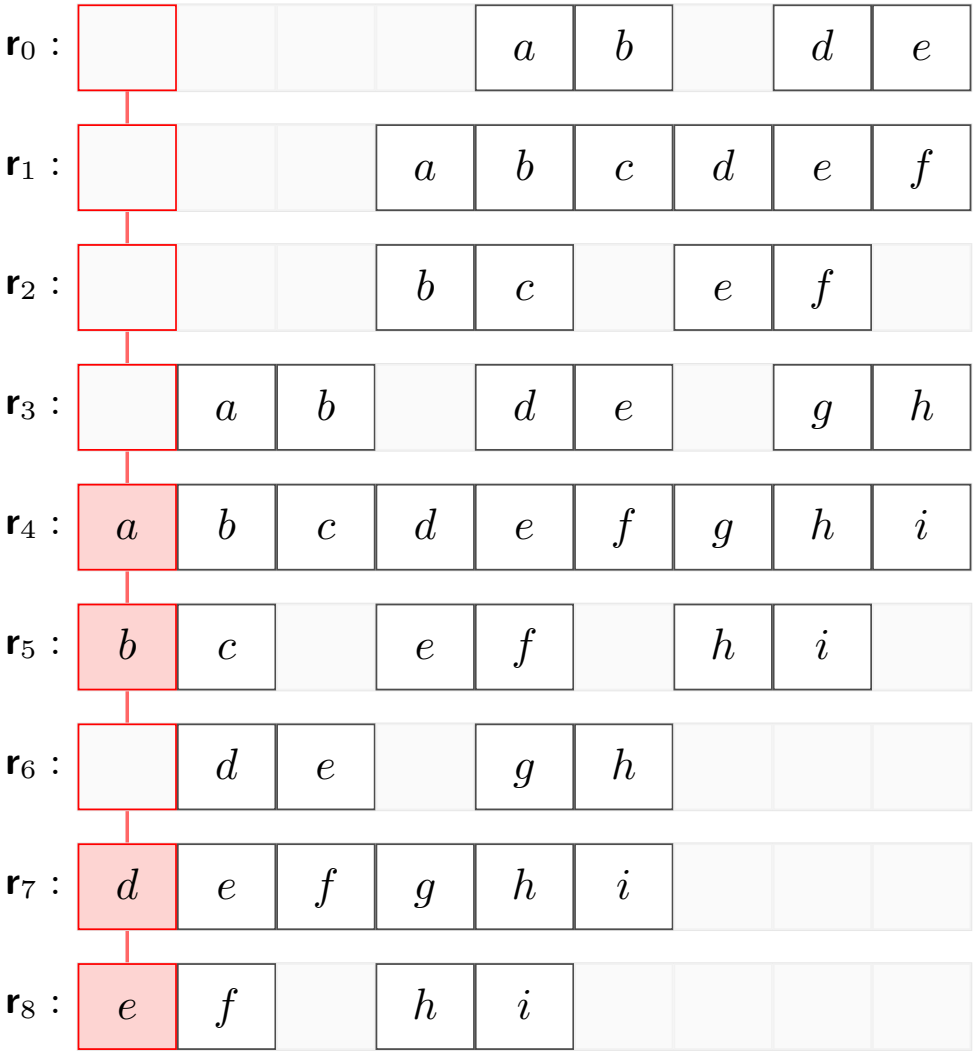
The proposed approach

We now introduce our proposal, starting from Vector Encoding. First of all, channels are encoded and encrypted as follows.



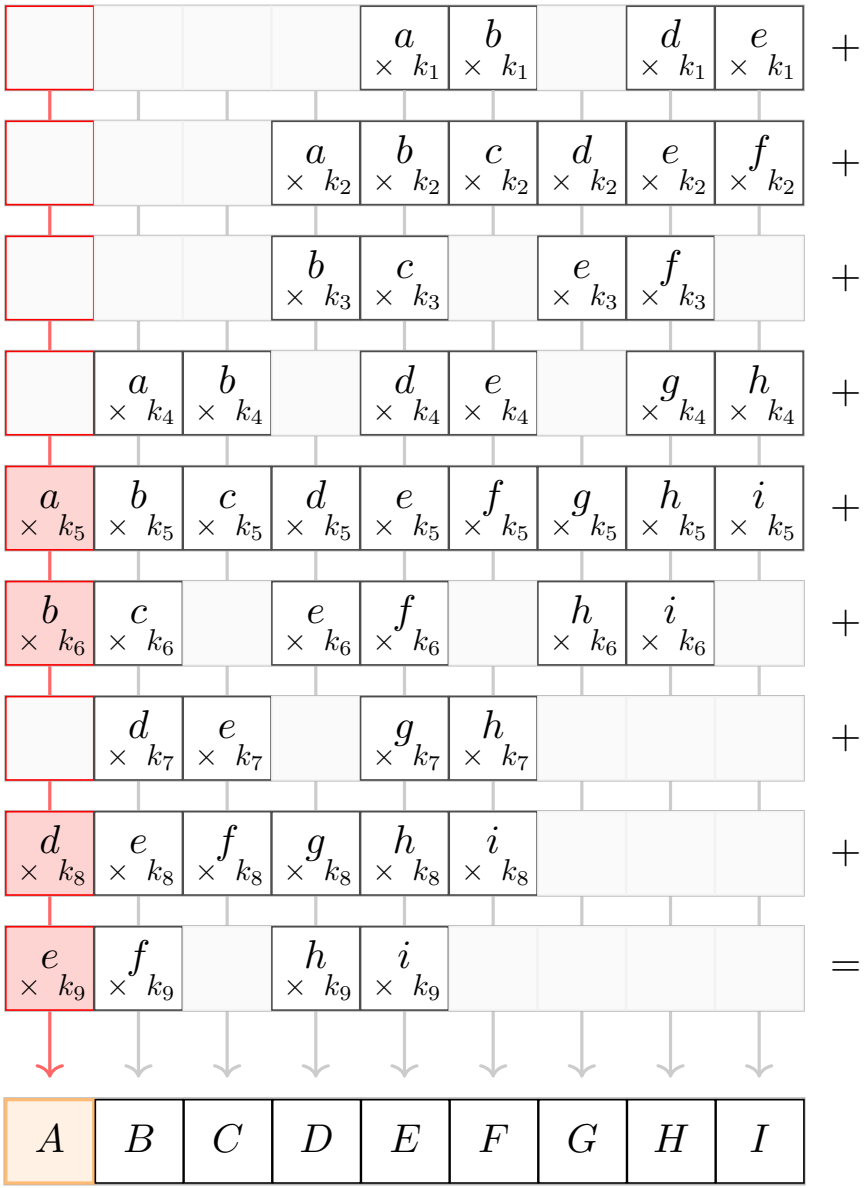
The proposed approach

- Then, $k-1$ rotations are computed, just like in the *Vector Encoding* approach.
- Each rotation is aligned one on top of the other, each column will contain the pixels required by the application of a filter.


$\mathbf{r}_0$: a b d e
$\mathbf{r}_1$: a b c d e f
$\mathbf{r}_2$: b c e f
$\mathbf{r}_3$: a b d e g h
$\mathbf{r}_4$: a b c d e f g h i
$\mathbf{r}_5$: b c e f h i
$\mathbf{r}_6$: d e g h
$\mathbf{r}_7$: d e f g h i
$\mathbf{r}_8$: e f h i

The proposed approach

- This set of rotations gives the possibility to apply a filter to a channel by multiplying each row by the corresponding filter value.
- Performing a convolution becomes easy!


a × k1 b × k1 d × k1 e × k1 +
a × k2 b × k2 c × k2 d × k2 e × k2 f × k2 +
b × k3 c × k3 e × k3 f × k3 +
a × k4 b × k4 d × k4 e × k4 g × k4 h × k4 +
a × k5 b × k5 c × k5 d × k5 e × k5 f × k5 g × k5 h × k5 i × k5 +
b × k6 c × k6 e × k6 f × k6 h × k6 i × k6 +
d × k7 e × k7 g × k7 h × k7 +
d × k8 e × k8 f × k8 g × k8 h × k8 i × k8 +
e × k9 f × k9 h × k9 i × k9 =
A B C D E F G H I

The proposed approach

This procedure can be extended to apply c filters to all the c channels at each iteration:

$\mathbf{c}_1 \cdot \mathbf{K}_{1,1}$	$\mathbf{c}_2 \cdot \mathbf{K}_{2,2}$	$\dots$	$\mathbf{c}_{16} \cdot \mathbf{K}_{16,16}$
$\mathbf{c}_2 \cdot \mathbf{K}_{1,2}$	$\mathbf{c}_3 \cdot \mathbf{K}_{2,3}$	$\dots$	$\mathbf{c}_1 \cdot \mathbf{K}_{16,1}$
$\mathbf{c}_3 \cdot \mathbf{K}_{1,2}$	$\mathbf{c}_4 \cdot \mathbf{K}_{2,4}$	$\dots$	$\mathbf{c}_2 \cdot \mathbf{K}_{16,2}$
$\vdots$			
$\mathbf{c}_{16} \cdot \mathbf{K}_{1,16}$	$\mathbf{c}_1 \cdot \mathbf{K}_{2,1}$	$\dots$	$\mathbf{c}_{15} \cdot \mathbf{K}_{16,15}$

The proposed approach

- Since computations are performed SIMD, each convolution applies a filter to the whole channel ($k^2 - 1$ rotation keys required)
- In order to apply the whole kernel to all the channels, this procedure is repeated c times, where c is the number of channels. At the end of each iteration, the output ciphertext is rotated by $i \cdot w^2$, with $0 < i \leq c$
- Required rotation keys: $(k^2 - 1) + c$

Recycling rotations

Rotations required for the previous example ($w = 3$):

- $\mathbf{r}_0 \leftarrow \text{ROT}_{-w-1}(\mathbf{c})$
- $\mathbf{r}_1 \leftarrow \text{ROT}_{-w}(\mathbf{c})$
- $\mathbf{r}_2 \leftarrow \text{ROT}_{-w+1}(\mathbf{c})$
- $\mathbf{r}_3 \leftarrow \text{ROT}_{-1}(\mathbf{c})$
- $\mathbf{r}_4 \leftarrow \mathbf{c}$
- $\mathbf{r}_5 \leftarrow \text{ROT}_{+1}(\mathbf{c})$
- $\mathbf{r}_6 \leftarrow \text{ROT}_{w-1}(\mathbf{c})$
- $\mathbf{r}_7 \leftarrow \text{ROT}_{w}(\mathbf{c})$
- $\mathbf{r}_8 \leftarrow \text{ROT}_{w+1}(\mathbf{c})$

Recycling rotations

Instead of performing $k^2 - 1$ rotations, it is possible to *recycle* rotations that have already been performed:

- $\mathbf{r}_0 \leftarrow \text{ROT}_{-1}(\mathbf{r}_1)$
- $\mathbf{r}_1 \leftarrow \text{ROT}_{-w}(\mathbf{r}_4)$
- $\mathbf{r}_2 \leftarrow \text{ROT}_{+1}(\mathbf{r}_1)$
- $\mathbf{r}_3 \leftarrow \text{ROT}_{-1}(\mathbf{r}_4)$
- $\mathbf{r}_4 \leftarrow \mathbf{c}$
- $\mathbf{r}_5 \leftarrow \text{ROT}_{+1}(\mathbf{r}_4)$
- $\mathbf{r}_6 \leftarrow \text{ROT}_{-1}(\mathbf{r}_7)$
- $\mathbf{r}_7 \leftarrow \text{ROT}_{+w}(\mathbf{r}_4)$
- $\mathbf{r}_8 \leftarrow \text{ROT}_{+1}(\mathbf{r}_7)$

Recycling rotations

- This approach, without loss of generality, allows to reduce the number of rotations from $k^2 - 1$ to a constant value 4.
- The idea is to use rotations keys for indexes $+1$ and -1 to rotate horizontally, and the indexes $+w$ and $-w$ to rotate the channel vertically.

Recycling rotations

- Eventually, this approach is extended also for the rotations applied at the end of the i-th iteration ($\text{Rot}_{i \cdot w^2}$).
- The previously rotated ciphertext is recycled and rotated again by w^2, only use one rotation key is used.
- In general the proposed approach requires only 5 rotation keys, for any convolution with odd-sized kernel and stride equal to $\{1, 1\}$.

Recall the main challenges:

- Building a low-complexity circuit that performs convolutions.
- Finding an approach to approximate the ReLU(x) and the tanh(x) function using basic arithmetic operators $(+, \cdot)$.

Recall the main challenges:

- Building a low-complexity circuit that performs convolutions.
- Finding an approach to approximate the ReLU(x) and the tanh(x) function using basic arithmetic operators $(+, \cdot)$.

Chebyshev polynomials

Chebyshev polynomials [Tre13] refers to a family of orthogonal polynomials defined by the following recurrence relation:

$$
\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_{2n}(x) &= 2T_n(x)^2 - 1 \\
T_{2n+1}(x) &= 2T_n(x) \cdot T_{n+1}(x) - x
\end{aligned}
$$

Chebyshev polynomials

Given this set of polynomials, it is also possible to define the so-called *Chebyshev roots* (or *Chebyshev interpolants*), a set of n points defined as:

$$p_n(x) = \sum_{k=0}^{n-1} c_k T_k(x)$$

where each coefficient c_k is distributed according to a certain continuous function f to be approximated over $[-1, 1]$.

Chebyshev polynomials

Given the following set:

$$x_i = \cos\left(\frac{i\pi}{n}\right) \text{ with } 0 \leq i < n$$

the coefficients c_k are determined such that $p_n(x_i) = f(x_i)$.

Performing a regression over these points is not far from the *best* approximation, with respect to the infinity norm.

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:

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

Putting everything together, it is possible to define polynomial approximations of any continuous function f by performing a regression over the *Chebyshev roots*.

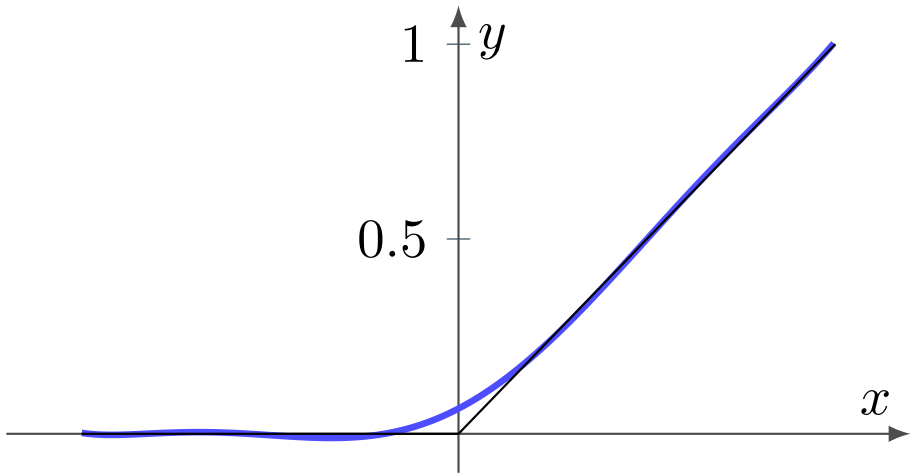


Figure: Chebyshev approximation of ReLU(x) with a polynomial of degree 7 in $[-1, 1]$

Experiments

Experiments

- Putting everything together, it is possible to build a FHE-based circuit to evaluate a CNN.
- We chose to implement the ResNet20 [HZRS15] circuit, which has a good tradeoff between depth and accuracy.
- In order to compare to related works, we evaluate the circuit on CIFAR-10 [KH+09].

Experiments

Four experiments have been performed, each with a different set of parameters. All of them respect the level of $\lambda = 128$ security bits, using ≤ 1772 bits for the moduli chain $\log(qp)$.

Exp	Scaling factor (Δ)	Bit length of q_i's	HKS digits (d_{num})	Moduli chain $\log(qp)$	Degree ReLU (d)	`CtoS` and `StoC` levels
1	2^{52}	2^{48}	2	1756 bits	59	$\{3, 3\}$
2	2^{50}	2^{46}	3	1772 bits	200	$\{4, 4\}$
3	2^{50}	2^{46}	3	1772 bits	119	$\{5, 4\}$
4	2^{48}	2^{44}	2	1748 bits	59	$\{4, 4\}$

Experiments

Each experiment will be evaluated from three perspectives:

- Computational runtime
- Required memory
- Precision of computations, as:

$$p(v, v') = 1 - \left(\sum_{i=1}^{n} \left(\frac{|v_i - v'_i|}{|max(v)|} \right) \cdot ||v||^{-1} \right)$$

where v is the expected vector, v' the encrypted one.

Runtime

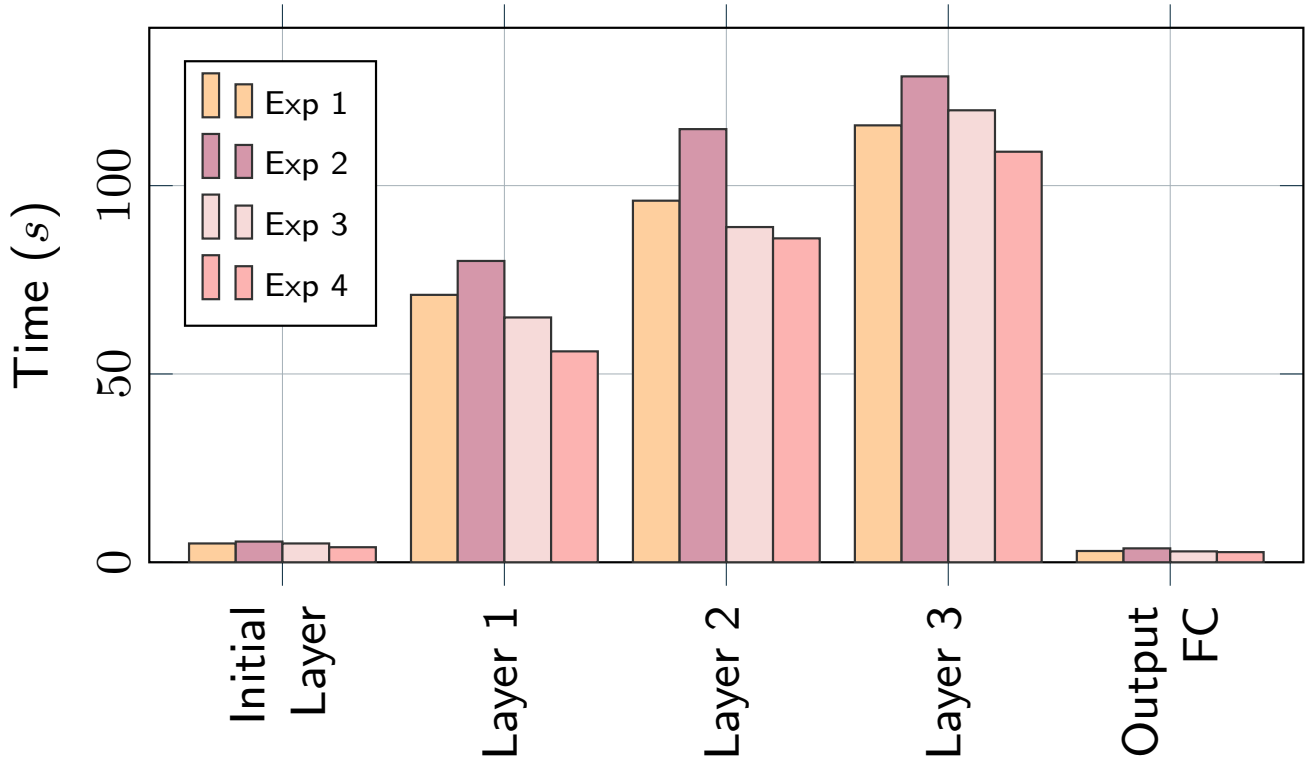


Figure: Visual representation of computational runtime of the encrypted circuits, computed at the end of each layer

Precision

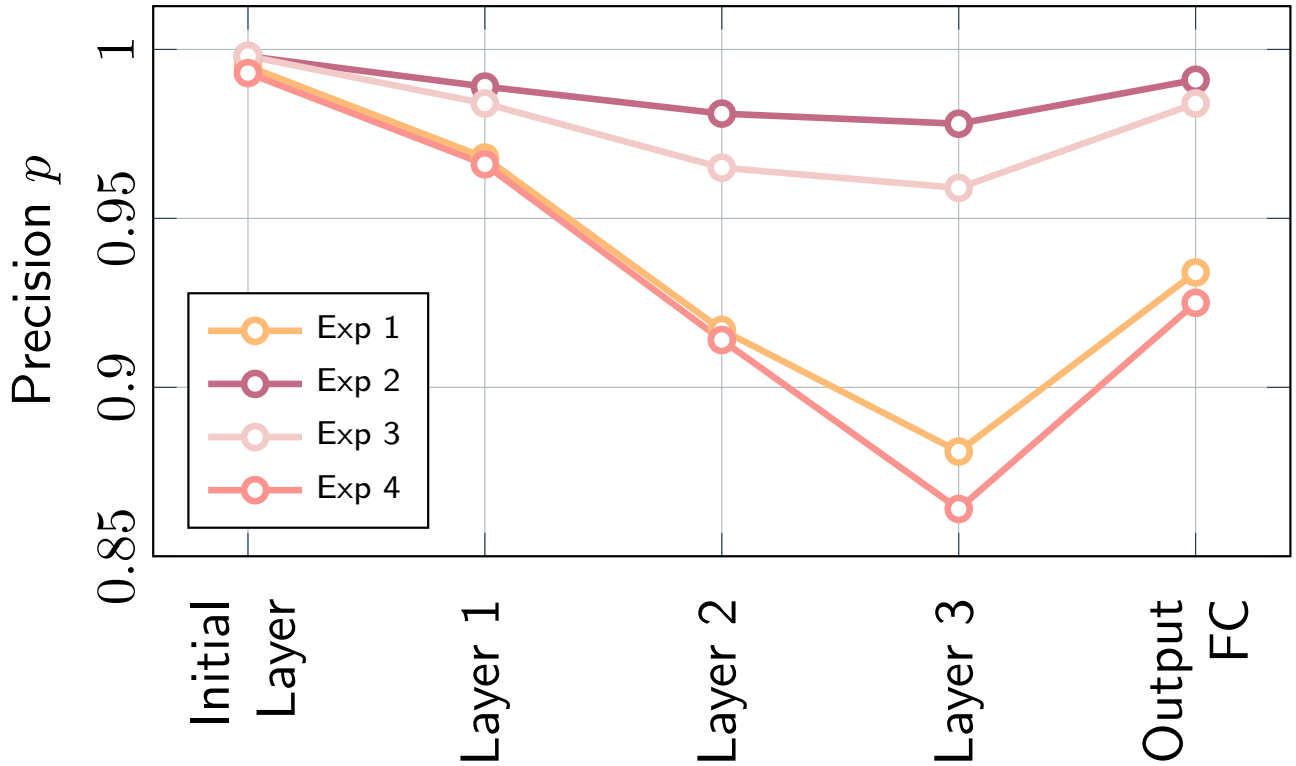


Figure: Visual representation of the precision of the encrypted circuits, computed at the end of each layer

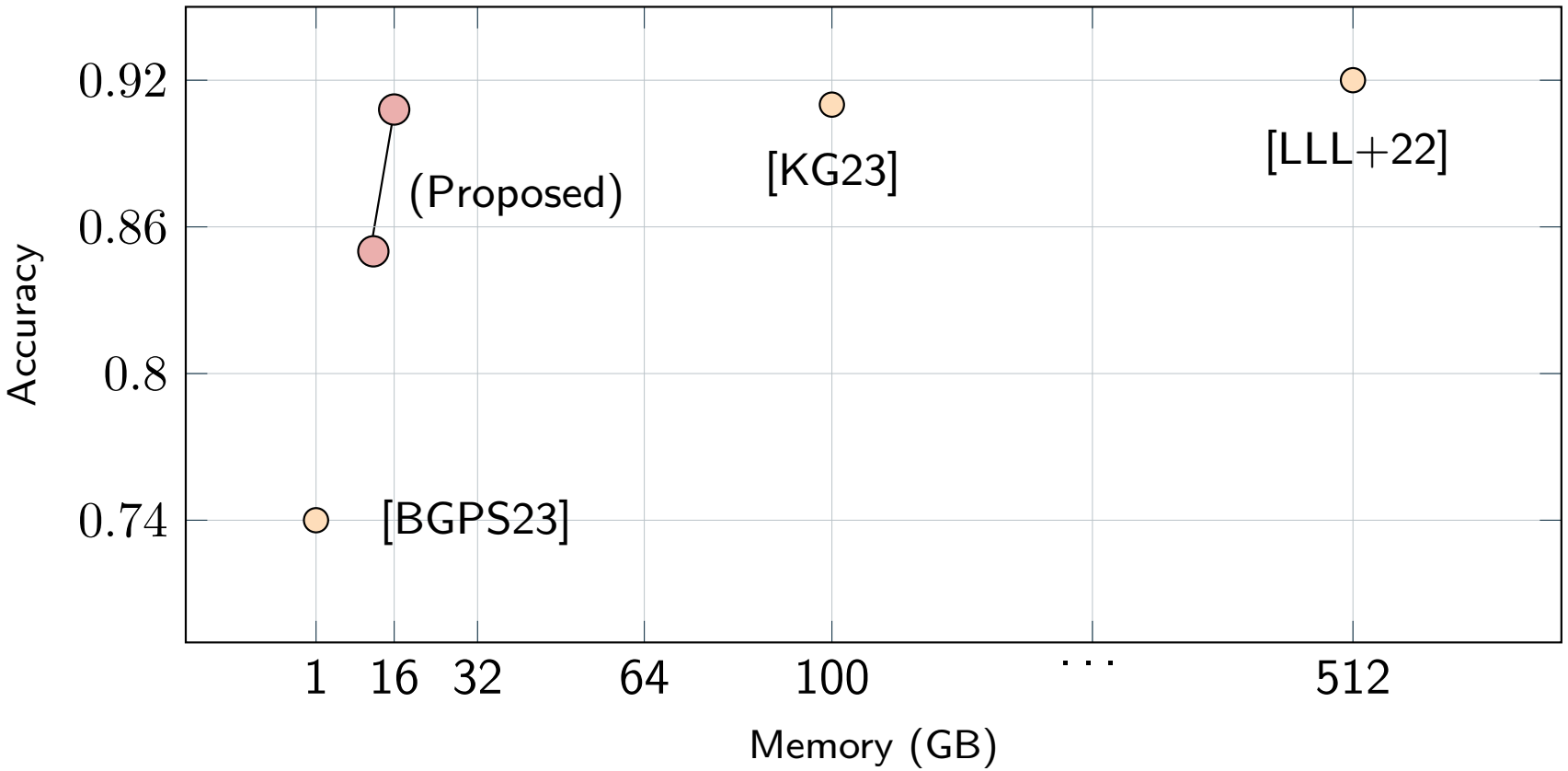
Experiments results

Table: Comparison of results obtained in the experiments. Relative accuracy is computed with respect to the output of the plain circuit on 1000 images from the test set

Exp	Runtime	Memory	Output layer precision p	Relative accuracy	Absolute accuracy
1	$291s \pm 4s$	≈ 13.6GB	0.93 ± 0.02	$945/1000 \approx 0.95\%$	87.97%
2	$336s \pm 6s$	≈ 15.2GB	0.98 ± 0.01	$986/1000 \approx 0.99\%$	**91.67**%
3	$285s \pm 6s$	≈ 12.5GB	0.97 ± 0.01	$978/1000 \approx 0.98\%$	90.75%
4	$260s \pm 3s$	$\approx$ **11.6GB**	0.92 ± 0.02	$931/1000 \approx 0.93\%$	86.12%

Comparison with related works

The objective was to fill the high accuracy/low memory area in the current literature frame



Results showed that the proposed circuits have a good tradeoff between accuracy, memory requirements and runtime.

Table: Comparison of the latest solutions for CIFAR-10 FHE-based CNN inference. * denotes that the authors did not provide information about the memory usage, we thus assume the RAM capacity of their machine

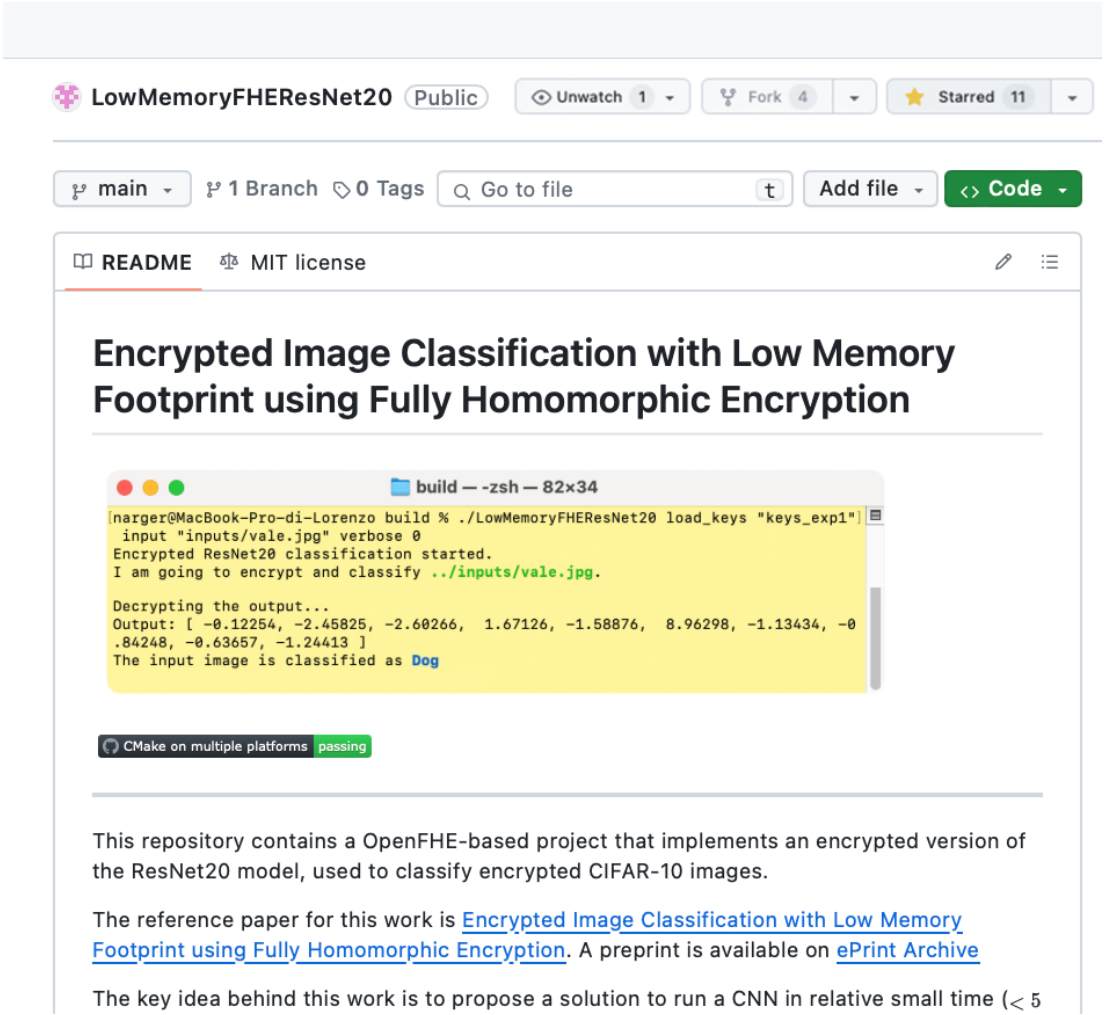
Proposal	Year	Scheme	Accuracy	Runtime	Cores	Memory
[LKL+21]	2022	CKKS	90.67%	10602s	64	512 GB*
[LLL+22]	2022	CKKS	91.31%	2271s	1	512 GB*
[BGPS23]	2023	TFHE	74.10%	570s	4	0.8 GB
[KG23]	2023	CKKS	92.04%	255s	1	100 GB
(proposed)	2024	CKKS	91.67%	260s	1	**15.2 GB**

Conclusion

Conclusion

- We filled the empty space in the literature by proposing a set of circuits with reduced memory consumption and a level of accuracy at the same level to that of current work.
- Nevertheless, new ideas can be tested and implemented:
  - Different circuits to perform convolutions
  - Other approaches to ReLU and tanh approximations (e.g. Remez Algorithm)

All the experiments are reproducible with our open repository.[1]



[1] https://github.com/narger-ef/LowMemoryFHEResNet20

Thank you!

- Talk based on a published paper[2], a preprint is available at `ia.cr/2024/460`
- Source-code available at `github.com/narger-ef/LowMemoryFHEResNet20`

GitHub Repository

ePrint Archive preprint

[2] Lorenzo Rovida and Alberto Leporati. "Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption". In: *International Journal of Neural Systems* 34.05 (2024), p. 2450025.

References I

[16] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)*. May 2016.

[BGPS23] Adrien Benamira et al. *TT-TFHE: a Torus Fully Homomorphic Encryption-Friendly Neural Network Architecture*. 2023. arXiv: 2302.01584 [cs.CR].

References II

[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.

[HZRS15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

References III

[JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. "GAZELLE: A Low Latency Framework for Secure Neural Network Inference". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1651–1669. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

[KG23] Dongwoo Kim and Cyril Guyot. "Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption". In: *Trans. Info. For. Sec.* 18 (Mar. 2023).

References IV

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[LKL+21] Joon-Woo Lee et al. *Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network*. 2021. arXiv: 2106.07229 [cs.LG].

[LLHJ20] Qian Lou et al. "Falcon: Fast Spectral Inference on Encrypted Data". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 2364–2374. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/18fc72d8b8aba03a4d84f66efabce82e-Paper.pdf.

References V

[LLL+22] Eunsang Lee et al. "Low-Complexity Deep Convolutional Neural Networks on Fully Homomorphic Encryption Using Multiplexed Parallel Convolutions". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 12403–12422.

[RL24] Lorenzo Rovida and Alberto Leporati. "Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption". In: *International Journal of Neural Systems* 34.05 (2024), p. 2450025.

References VI

[Tre13] L.N. Trefethen. *Approximation Theory and Approximation Practice*. Other Titles in Applied Mathematics. Siam, 2013.