Encrypted neural networks without overflows

P. Kern L. Rovida S. Teuber E. Manino C. Sinz A. Leporati

KIT Polytechnic University of Turin University of Manchester Karlsruhe UAS University of Milano-Bicocca



Oct. 26, 2026 — WPEC 2026: NIST Workshop on Privacy-Enhancing Cryptography

Agenda

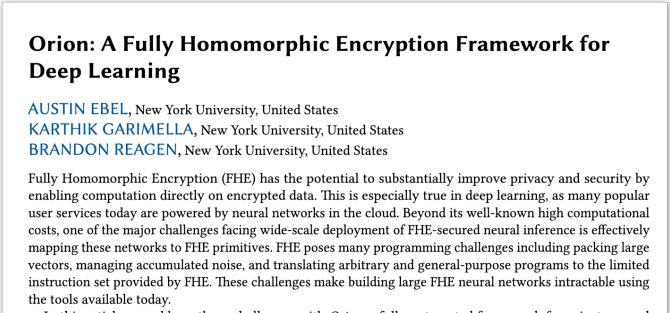
1. Motivation
2. The problem: overflow events
3. The solution: certified design
4. Experiments
5. Conclusions

Private inference

- Fully homomorphic encryption (FHE) is becoming more and more practical
- For private inference via neural networks, the CKKS scheme seems like the most promising one as its plaintext space is $\mathbb{C}^{N/2}$ for some power of two N.
- Computations happen slot-wise (SIMD), for massive parallel operations.

Private inference

- Nowadays we can find state-of-the-art tools to evaluate (relatively) small neural networks.
- You don't even have to be a cryptographer! Current solutions extend PyTorch to FHE, e.g., Orion [EGR25].



Orion: A Fully Homomorphic Encryption Framework for Deep Learning

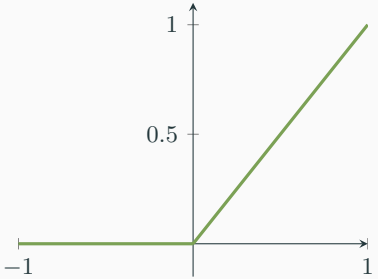
AUSTIN EBEL, New York University, United States
KARTHIK GARIMELLA, New York University, United States
BRANDON REAGEN, New York University, United States

Fully Homomorphic Encryption (FHE) has the potential to substantially improve privacy and security by enabling computation directly on encrypted data. This is especially true in deep learning, as many popular user services today are powered by neural networks in the cloud. Beyond its well-known high computational costs, one of the major challenges facing wide-scale deployment of FHE-secured neural inference is effectively mapping these networks to FHE primitives. FHE poses many programming challenges including packing large vectors, managing accumulated noise, and translating arbitrary and general-purpose programs to the limited instruction set provided by FHE. These challenges make building large FHE neural networks intractable using the tools available today.

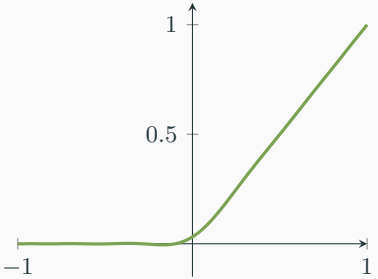
Private inference

- FHE allows us to compute additions and multiplications over ciphertexts, but neural networks require activation functions!
- Luckily, we can approximate them via polynomials:

True ReLU



Poly ReLU (degree 15)



Range estimation

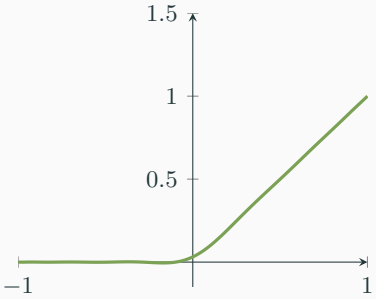
- **However**,

Range estimation

- **However**, polynomial approx. work well in a **fixed** interval.

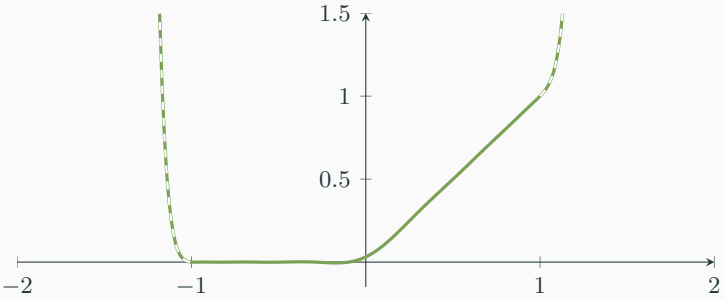
Range estimation

- **However**, polynomial approx. work well in a **fixed** interval.



Range estimation

- **However**, polynomial approx. work well in a **fixed** interval. Outside they diverge to $\pm\infty$!



Range estimation

- **However**, polynomial approx. work well in a **fixed** interval.
- In other words, we can not evaluate a polynomial approximation of ReLU over any input $\boldsymbol{x} \in \mathbb{R}^m$, but we can only hope to do that in some $[\boldsymbol{l}, \boldsymbol{u}]$ for $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^m$.
- The polynomial approximations tend to strongly diverge outside of the intervals!

Range estimation

- **However**, polynomial approx. work well in a **fixed** interval.
- In other words, we can not evaluate a polynomial approximation of ReLU over any input $\boldsymbol{x} \in \mathbb{R}^m$, but we can only hope to do that in some $[\boldsymbol{l}, \boldsymbol{u}]$ for $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^m$.
- The polynomial approximations tend to strongly diverge outside of the intervals!

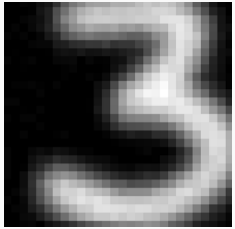
RQ: How can we be sure that the input of each approximation is in $[\boldsymbol{l}, \boldsymbol{u}]$?

Range estimation

- In real-world use cases, it is hard to predict what the user will query...
- Moreover, neural networks behave in weird ways...!

Range estimation

- In real-world use cases, it is hard to predict what the user will query...
- Moreover, neural networks behave in weird ways...!



→ Input to first ReLU: 0.74

Range estimation

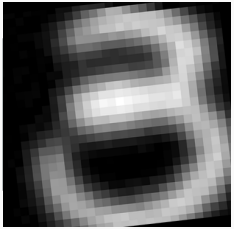
- In real-world use cases, it is hard to predict what the user will query...
- Moreover, neural networks behave in weird ways...



→ Input to first ReLU: 0.54

Range estimation

- In real-world use cases, it is hard to predict what the user will query...
- Moreover, neural networks behave in **weird** ways...



→ Input to first ReLU: **8.74**

Polynomial networks and range estimation

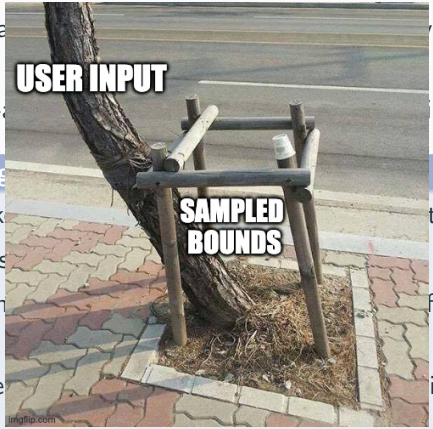
- A polynomial approximation of σ is only accurate on an interval $[\boldsymbol{l}^{(\ell)}, \boldsymbol{u}^{(\ell)}]$; outside it diverges to $\pm\infty$
- Need to bound the pre-activations $f(x) \in [\boldsymbol{l}, \boldsymbol{u}]$ *for all inputs* in some set $\mathcal{I}$

Current practice: sampling

- For each activation, take min/max of pre-activation over a dataset $\mathcal{D}$
- (Optional) widen ranges e.g., by $\times 2$.
- (Optional) use orthogonal transformation to reduce the size of these intervals [Ash+24; Par+26]
- **No guarantees** outside $\mathcal{D}$. If one neuron goes outside of the interval the result gets corrupted

Polynomial networks and range estimation

- A polynomial approxima… …val $[\boldsymbol{l}^{(\ell)}, \boldsymbol{u}^{(\ell)}]$; outside it diverges to $\pm\infty$
- Need to bound the pre-… …in some set $\mathcal{I}$



Current practice: sampling

- For each activation, tak… …taset $\mathcal{D}$
- (Optional) widen ranges
- (Optional) use orthogon… …f these intervals [Ash+24; Par+26]
- **No guarantees** outside… …interval the result gets corrupted

Overflow events

- Given a polynomial network f_π:

Definition (Polynomial overflow event)

$\boldsymbol{x} \in \mathcal{I}$ causes an overflow event on f_π iff $\exists$ a layer ℓ such that $f_\pi^{(\ell)}(\boldsymbol{x}) \notin [\boldsymbol{l}^{(\ell)}, \boldsymbol{u}^{(\ell)}]$.

- Notice that $[\boldsymbol{l}^{(\ell)}, \boldsymbol{u}^{(\ell)}]$ are bounds on the *preactivation* values (i.e., before entering the polynomial approximation of the activation)

Definition (Overflow robustness)

A polynomial network f_π is *overflow robust* on $\mathcal{I}$ if *no* $\boldsymbol{x} \in \mathcal{I}$ causes an overflow event.

Finding overflows: an auditing attack

Threat model: an honest user submits a legitimate input that *accidentally* overflows. We propose a technique that crafts such "fine-looking" inputs.

Finding overflows: an auditing attack

Threat model: an honest user submits a legitimate input that *accidentally* overflows. We propose a technique that crafts such "fine-looking" inputs.

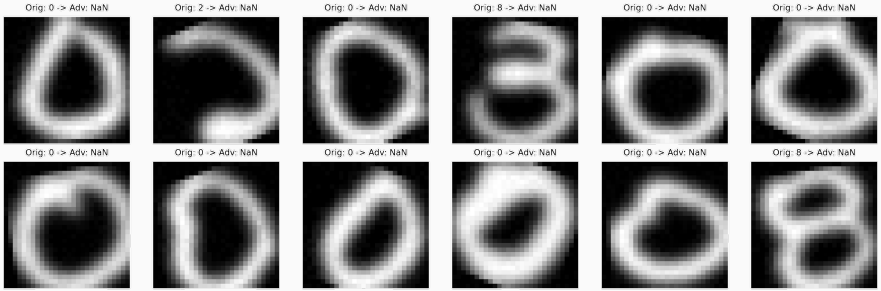
- Perturb a benign $\boldsymbol{x}$ as $\boldsymbol{x}' = \gamma(\boldsymbol{x}, \boldsymbol{\delta})$, with $\boldsymbol{\delta} = (\theta, t, \eta)$: rotation, translation, noise
- Keep inputs realistic by constraining $\boldsymbol{\delta} \in \Delta = \{\, |\theta| \leq \theta_{\max},\ \|t\|_\infty \leq t_{\max},\ \|\eta\|_\infty \leq \varepsilon \,\}$ (e.g. MNIST: $23°$, 20%, $\varepsilon = 5/255$)

We optimize $\boldsymbol{\delta} \in \Delta$ with projected gradient descent until a bound is violated:

$$\max_{\boldsymbol{\delta}} \sum_j \left\| \left(f^{(l)}(\gamma(\boldsymbol{x}, \boldsymbol{\delta}))\right)_j - \tfrac{1}{2}\left(\boldsymbol{u}_j^{(l)} + \mathbf{l}_j^{(l)}\right) \right\|_\infty$$

Overflows in sampling-based networks

- Sample MNIST images that cause overflow in a sampling-based network (with doubled observed ranges)



Overflows in sampling-based networks

- Sample CIFAR-10 images that cause overflow in a sampling-based network (with doubled observed ranges) in a Orion sample network [EGR25]



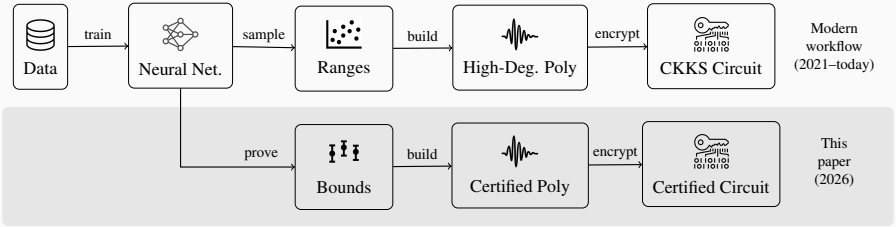
Overflows in sampling-based networks

Dataset	Setting	Sampled bounds	Certified (ours)
CIFAR10 (AlexNet)	Gradient descent (Orion [EGR25])	39.3%	-
CIFAR10 (Resnet)	Gradient descent (Orion [EGR25])	15.2%	-
MNIST $\rightarrow$ USPS	Natural	5.18%	0.0%
MNIST $\rightarrow$ USPS	Gradient descent	12.30%	0.0%
CIFAR10 (small NN)	Gradient descent	0.10%	0.0%
CIFAR10 (deep NN)	Gradient descent	46.70%	0.0%

- Note: sometimes even changing the test set naturally causes overflows! E.g., a MNIST-trained network evaluated on the USPS dataset (Natural setting)
- N.b.: perturbations on images are, e.g., rotation $\leq 23°$, translation $\leq 20\%$

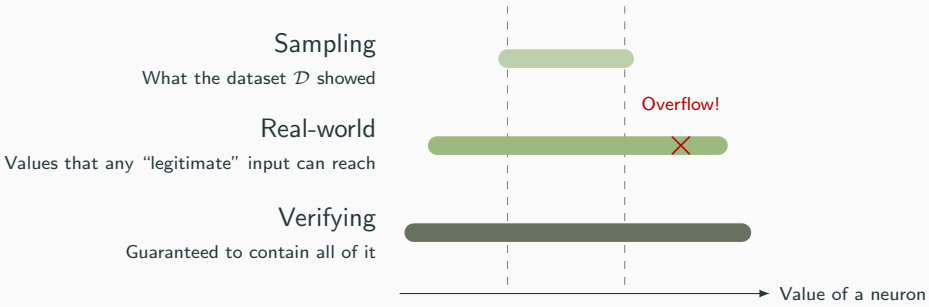
Our approach: certified design

- Our solution: building upon [KMS26], we explore the intersection between FHE/ML/Formal methods!



- Overflow-free *by construction*, with certified error bound $\|f_\pi(x) - f(x)\|_\infty$

Testing vs. proving the ranges



- Sampling is testing: It only tells you about what has been tried (only $\mathcal{D} \subset \mathcal{I}$)
- Verification is proving! Computes the worst case by design; it is mathematically guaranteed for the whole $\mathcal{I}$

How a verifier bounds a neuron

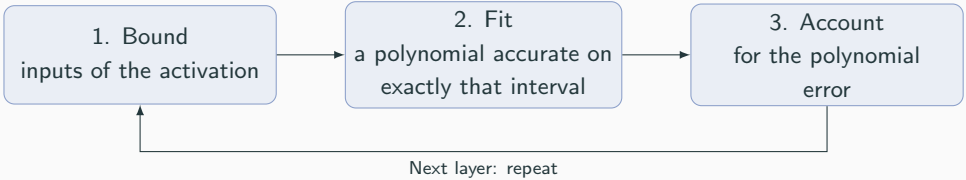
A tiny example

Inputs $x_1, x_2 \in [0,1] \subset \mathbb{R}$; neuron $z = 2x_1 - x_2 - 1$

- Linear part: best case $(x_1, x_2) = (1, 0)$ gives 1, worst case $(0, 1)$ gives -2, hence $z \in [-2,\ 1]$
- ReLU is monotone, so apply it to the endpoints: $\mathrm{ReLU}(z) \in [\mathrm{ReLU}(-2), \mathrm{ReLU}(1)]$
- In practice, we use a polynomial p instead of ReLU, with $\|p - \mathrm{ReLU}\|_\infty \leq \varepsilon$ over $[-2, 1]$. Therefore, $y = p(z) = \mathrm{ReLU}(z) + e,\ e \in [-\varepsilon, \varepsilon]$. Finally,

$$y \in [-\varepsilon,\ 1 + \varepsilon]$$

Building the network layer by layer



- The earlier polynomial errors are included in the bounds of later layers, so the guarantee holds through the whole network

Theorem (Informal)

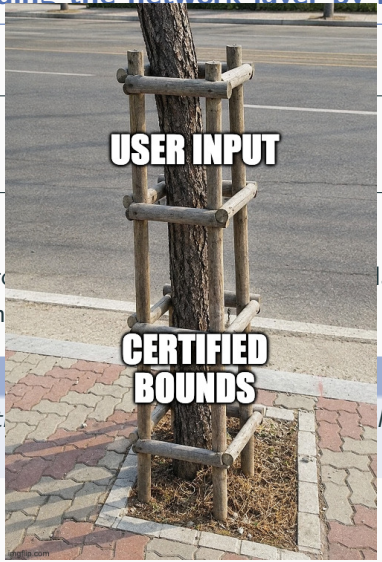
For some neural network f with input space $\mathcal{I}$, our approach yields an overflow robust polynomial network f_π on $\mathcal{I}$.

Building the network layer by layer

1. Bound inputs of the activation

2. Bound the activation

3. Account for the polynomial error



- The earlier polynomial err… later layers, so the guarantee holds through the whole n…

Theorem (Informal)

For some neural network f wit… lds an overflow robust polynomial network f_π on $\mathcal{I}$.

∗AI-generated img by ChatGPT

GELU converges faster

- Approximate σ on $[-1, 1]$ (after rescaling) by a degree-n Chebyshev polynomial π_n. How fast does $\|\sigma - \pi_n\|_\infty \to 0$ as n grows? The smoother σ is, the faster [Tre19]:

σ is ν-times differentiable

$$|\sigma - \pi_n| \leq \frac{4V}{\pi\nu(n-\nu)^\nu}$$

Decays as $O(n^{-\nu})$ (ReLU)

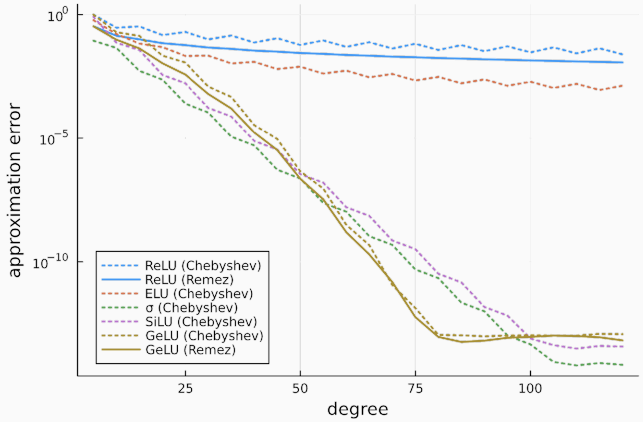
σ is analytic

$$|\sigma - \pi_n| \leq \frac{4M\rho^{-n}}{\rho - 1}$$

Decays as $O(\rho^{-n})$ (GELU, SiLU, ...)

- ReLU: σ is the activation, π_n its degree-n polynomial approximation, ν the number of derivatives σ has, and V the total variation (how much it wiggles) of the ν-th derivative.
- GELU: $\rho > 1$ the size of the ellipse E_ρ (foci ± 1, semi-axes $\frac{\rho \pm \rho^{-1}}{2}$) inside which σ stays analytic, and $M = \max_{E_\rho} |\sigma|$ its largest absolute value there.

GELU converges faster



*from degree 75 on we have floating point imprecision

Extra: heterogeneous polynomials per layer

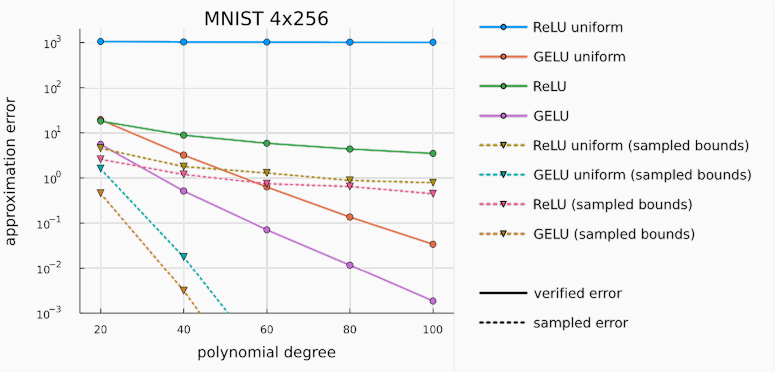
- Per-neuron ranges $[\boldsymbol{l}_i^{(\ell)}, \boldsymbol{u}_i^{(l)}]$ differ a lot, but existing CKKS NNs use a single polynomial per layer
- We fit a different polynomial for each neuron, which needs a different polynomial in each slot of a ciphertext
- Solution: generalize Paterson–Stockmeyer from (ct $\times$ constant) to (ct $\times$ plaintext) — coefficients packed in plaintexts. The cost with NTT polynomials is roughly the same (depends on the implementation).

Experimental setup

- We tested our method on different datasets/networks:
  - Fully connected: HELOC, MNIST, EC (energy), NN4Sys
  - Convolutional NN: CIFAR10, Collins RUL
- From 64 up to 10,300 neurons
- All NNs trained with L_1 regularization to obtain tighter bounds
- The main tools are: (i) VeryDiff [Teu+25] and α-CROWN [Xu+21] for polynomial networks, (ii) OpenFHE [Al +22] for the CKKS implementation.

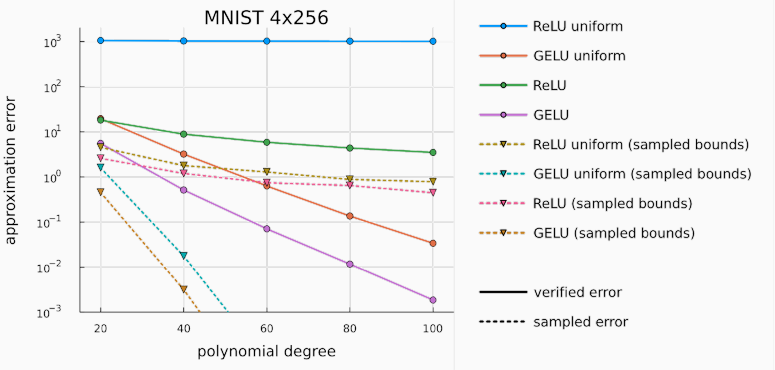
Ablation: what reduces the verified error?

GELU vs. ReLU: error up to three orders of magnitude smaller (MNIST, $d{=}100$)



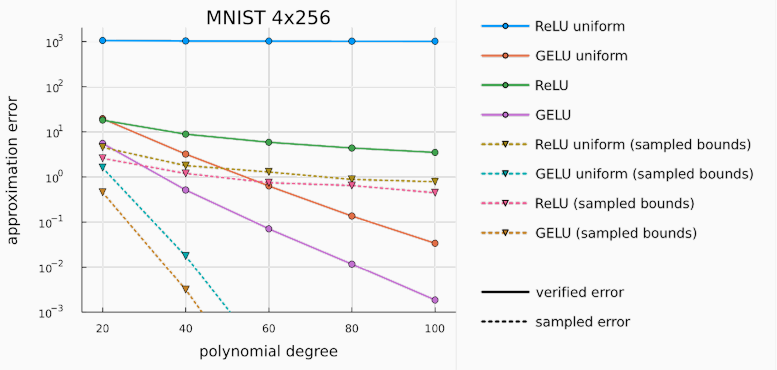
Ablation: what reduces the verified error?

Heterogeneous vs. uniform: $\geq 18\times$ on MNIST at degree $d{=}100$



Ablation: what reduces the verified error?

Verified vs. sampled: sampled errors are smaller, but they are not overflow robust



Certified design at scale

Does certified design keep accuracy, in the clear and under encryption?

Dataset + Network layers	σ	Deg.	Accuracy f / f_π	$\|f(\boldsymbol{x}) - f_\pi(\boldsymbol{x})\|_\infty$	CKKS error
MNIST [256, 256, 256, 256]	ReLU	119	98.64 / 98.64%	2.93	10^{-7}
	GELU	119	98.93 / 98.93%	0.0003	10^{-7}
CIFAR10 [900, 392, 144, 256]	ReLU	247	67.24 / 67.57%	49.03	10^{-4}
	GELU	247	69.56 / 69.50%	0.43	10^{-4}
CIFAR10 [1.8k, 784, 288, 256]	ReLU	247	76.30 / 76.31%	111.40	10^{-4}
	GELU	247	77.28 / 75.13%	18.64	10^{-4}

Limitations and future work

- Bounds do not natively include the stochastic CKKS noise e_{CKKS}. Combining them via, e.g., [Bos+25; Noh+26] would yield precise noise models
- Arbitrary-depth networks need bootstrapping and co-designed convolutions; otherwise expect slack bounds on deeper architectures
- Certified polynomials may have ill-scaled coefficients, which reduces CKKS precision
- Verified bounds might get very large for deeper/wider networks/networks trained w/o L_1 regularization

Take-home messages

- Overflow events are real: sampled ranges let up to 47% of benign CIFAR10 inputs corrupt the CKKS output
- Certified design removes them by construction and gives a provable bound on $\|f_\pi - f\|_\infty$
- GELU + heterogeneous polynomials recover the precision lost to conservative bounds
- Open-source: OpenFHE-based framework[1]
- Unity is strength: private ML is much stronger when the cryptographic and the formal methods communities unite their forces!

[1] `github.com/lorenzorovida/encrypted-neural-networks-without-overflows`

Thank you! ^^

Questions?



arxiv.org/abs/2605.23096

References

[Al +22] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. **"OpenFHE: Open-Source Fully Homomorphic Encryption Library".** In: *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography.* WAHC'22. 2022, pp. 53–63.

[Ash+24] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. **"QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs".** In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems.* 2024. URL: https://openreview.net/forum?id=dfqsW38v1X.

References

[Bos+25] Jean-Philippe Bossuat, Anamaria Costache, Christian Mouchet, Lea Nürnberger, and Juan Ramón Troncoso-Pastoriza. **"Accurate and Composable Noise Estimates for CKKS with Application to Exact HE Computation".** In: *IACR Communications in Cryptology* 2.2 (2025).

[EGR25] 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*. ASPLOS '25. 2025, pp. 734–749. DOI: 10.1145/3676641.3716008.

[KMS26] Philipp Kern, Edoardo Manino, and Carsten Sinz. **"Certified Error Analysis of Homomorphically Encrypted Neural Networks".** In: *AI Verification*. Springer Nature Switzerland, 2026, pp. 156–179.

References

[Noh+26] Taeyeong Noh, Aimad Chabounia, Guillaume Hanrot, and Alain Passelègue. **RDK-CKKS: A Framework for Rigorous Error Analysis for CKKS Computation.** Cryptology ePrint Archive, Paper 2026/2575. 2026. URL: https://eprint.iacr.org/2026/2575.

[Par+26] 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.

[Teu+25] Samuel Teuber, Philipp Kern, Marvin Janzen, and Bernhard Beckert. **"Revisiting Differential Verification: Equivalence Verification with Confidence".** In: *Tools and Algorithms for the Construction and Analysis of Systems - 31st International Conference, TACAS*. 2025, pp. 257–278. DOI: 10.1007/978-3-031-90653-4_13.

References

[Tre19] Lloyd N. Trefethen. **Approximation Theory and Approximation Practice.** SIAM, 2019. URL: https://epubs.siam.org/doi/book/10.1137/1.9781611975949.

[Xu+21] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. **"Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers".** In: *Proc. of the 9th International Conference on Learning Representations(ICLR'21), Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.