Exploiting Unstructured Sparsity in Fully Homomorphic Encrypted DNNs

Aidan Ferguson[1], Perry Gibson[1], Lara D'Agata[1], Parker McLeod[2], Ferhat Yaman[2], Amitabh Das[2], Ian Colbert[2], José Cano[1]

[1]University of Glasgow, UK [2]AMD

EuroMLSys workshop @ EuroSys 2025

Rotterdam, The Netherlands, 31/03/2025

Key concept: Deep Learning Acceleration Stack (DLAS)

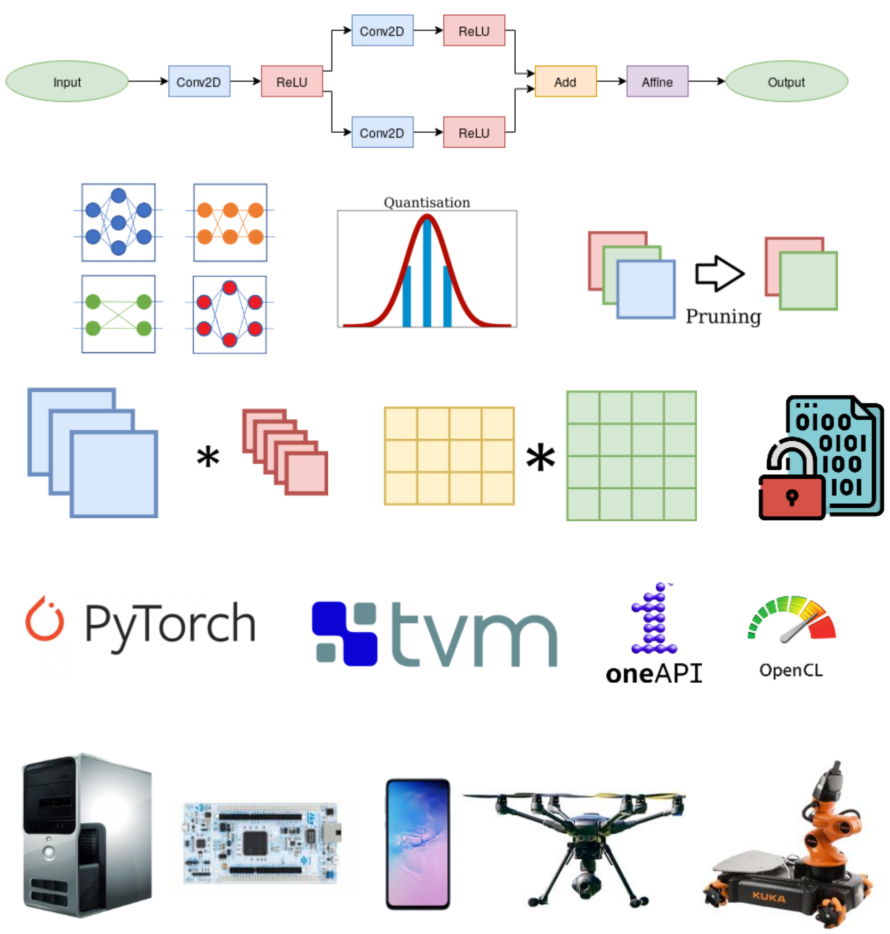
Neural Network Models & Datasets
(Image, video, voice, text, etc)

Optimization Techniques
(Pruning, quantization, NAS/HPO, etc)

Algorithmic Primitives & Data Formats
(GEMM, Winograd, CSR, Encryption, etc)

Systems Software
(Libraries, frameworks, compilers, etc)

Hardware
(Server class, Edge/IoT/Tiny devices)



Across-stack optimizations are required to provide efficient solutions!

[P. Gibson, J. Cano, E. J. Crowley, A. Storkey, M. O'Boyle, "*DLAS: A Conceptual Model for Across-Stack Deep Learning Acceleration*", **ACM TACO'25**]

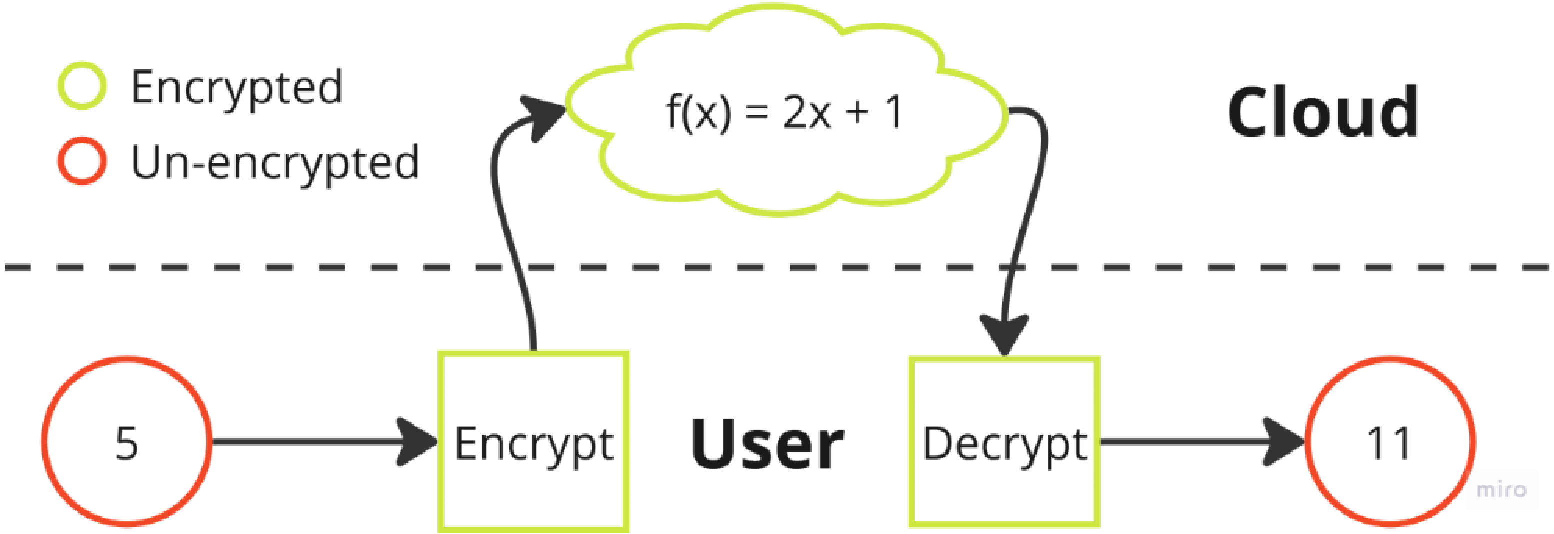
Outline

- Introduction
- Methodology
- Evaluation
- Conclusions and Future Work

Introduction

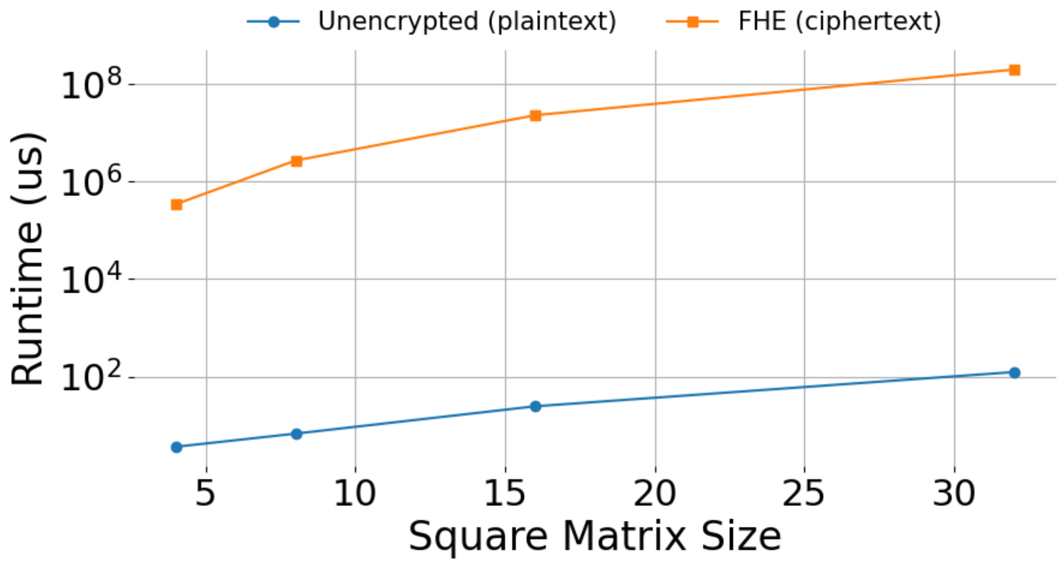
- Deployment of DNNs has raised privacy concerns, particularly where sensitive user data is involved

- **Fully Homomorphic Encryption** (FHE) allows for computation on encrypted data



Introduction

- While FHE is promising, currently the computational cost is prohibitive to its adoption
  - We observed ~10^6 slower execution time than plaintext matrix multiplication (matmul)

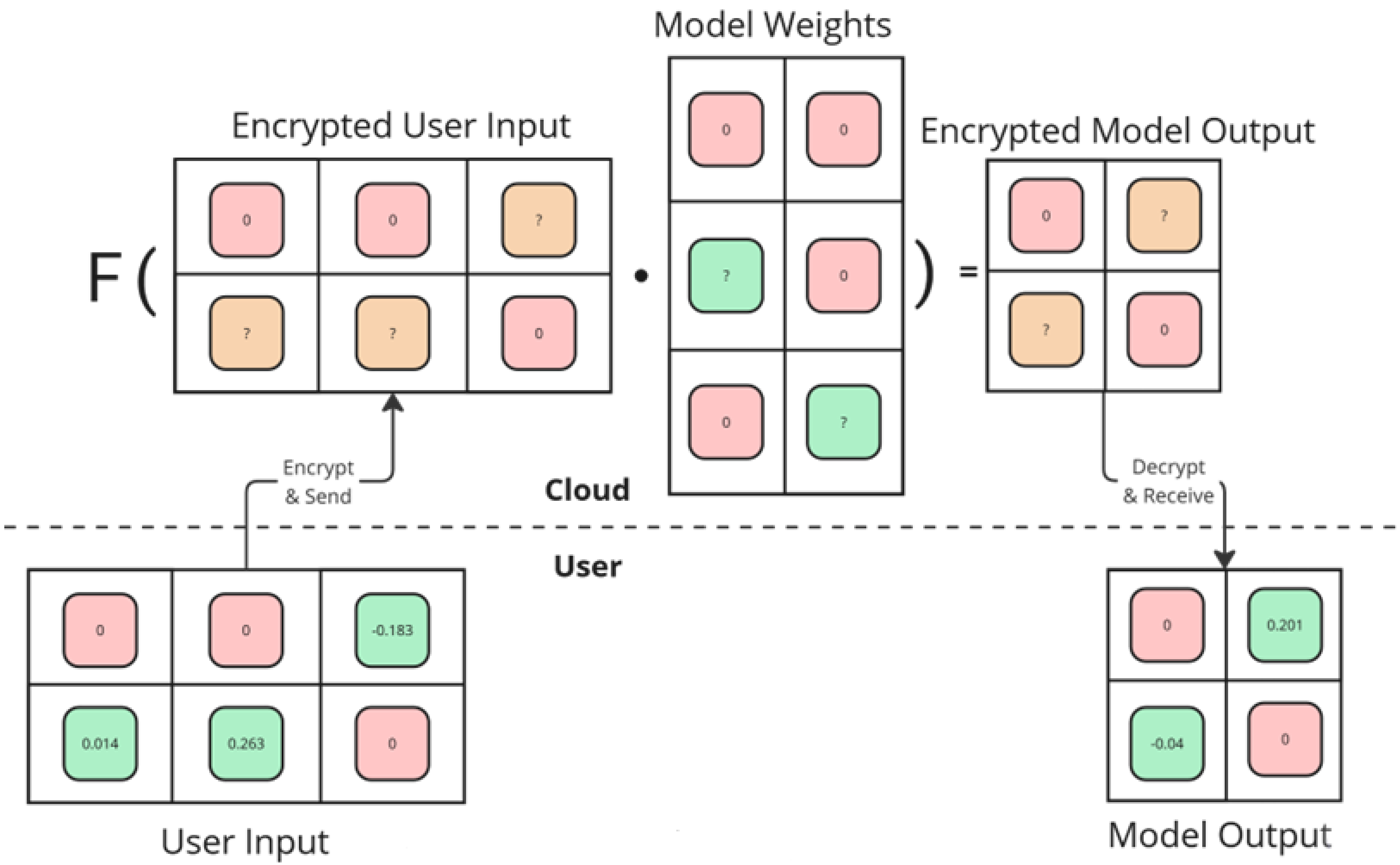


- As matmul is the majority of computation during DNN inference, we target it for optimization
  - Specifically unstructured sparsity

- To our knowledge, no public implementations of sparsity utilization in FHE matrix multiplication

Methodology

- We implement **sparse FHE inference** using server/client with a simple **MNIST model** in the repository



– **Model weights** are known only to server, but encrypted during inference

– **F** represents applicable activation functions in FHE

Methodology

- Three predominant schemes in FHE
  - **BFV/BGF**: Exact integer arithmetic, often overflows in the context of quantised DNN inference
  - **CKKS**: Approximate floating point computation, faster bootstrapping algorithms

- As we perform computations in FHE, the ciphertexts accumulate 'noise’
  - Some noise is permissible in the context of DNNs

- If this noise exceeds a threshold, we cannot reliably decrypt it
  - **Bootstrapping** is a technique for refreshing the noise 'budget' without decrypting, however it is a computationally expensive operation

Methodology

- We adapt plaintext sparse encoding schemes: **CSR** and **ELLPACK**

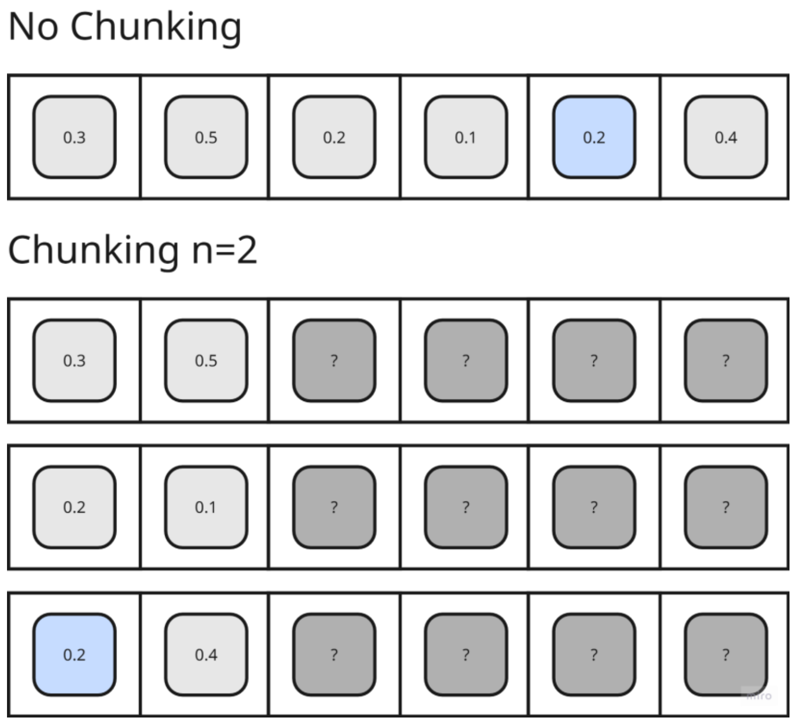
- CSR composed of: an array of values **V**, an array of row index pointers **R,** an array of column indices **C**
  - Our scheme encrypts **V** while leaving **R** and **C** unencrypted; similarly the ELLPACK format exposes metadata about the structure of the sparsity
  - Exposing this metadata is necessary for accelerating computation as we cannot determine if an encrypted value is zero at multiplication time
  - For some applications this may not be acceptable still (i.e. One-Hot encoding), in this case we can encrypt user input as if it were a dense matrix and multiply with the sparse server matrices, trading some runtime for efficiency

Methodology

- In order to reduce noise and runtime, we aim to reduce the amount of rotations we have to perform on a ciphertext

- We achieve this by chunking matrix values into encrypted vectors
  - i.e. for a chunk size of 2, we encode a maximum of 2 values in an encrypted vector

- We also utilize multi-threading by allocating one thread per resultant value
  - Allows our multithreading scheme to scale arbitrarily with thread count and matrix size



Evaluation

- We evaluate our sparse schemes against **two baselines**
  - Our implementation of naïve dense-dense multiplication in FHE
  - A SOTA implementation, HEMat, restricted to n^2 x m^2 matrices (we only evaluate these sizes)

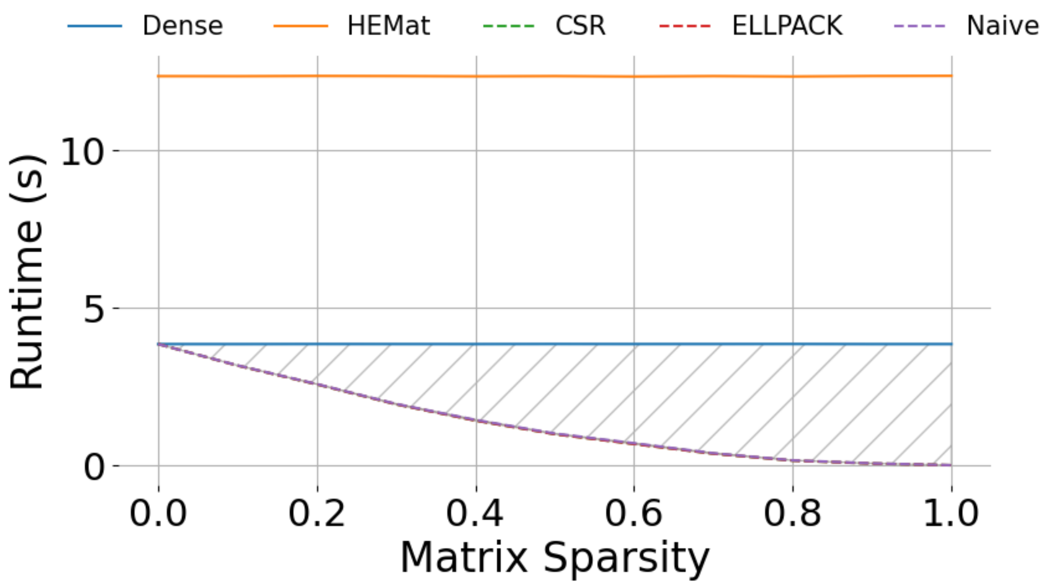
- We sample
  - Matrix values from a normal distribution that emulates neural network initialisation
  - Then zero values until we reach a desired sparsity threshold

- We perform matrix multiplication in plaintext with the Eigen3 library to verify correctness (within $\varepsilon=10^{-3}$)

- All evaluations conducted on an AMD EPYC 7V13 64-core CPU on the AMD HPC cluster

Results: 1 thread

- With **1 thread** at **8x8 matrix sizes**, we observe
  - Our sparse schemes perform better than the naïve baseline at all sparsity levels
  - A departure from plaintext, where we typically see a 'break-even' point
  - There is very little variation between the different sparse schemes

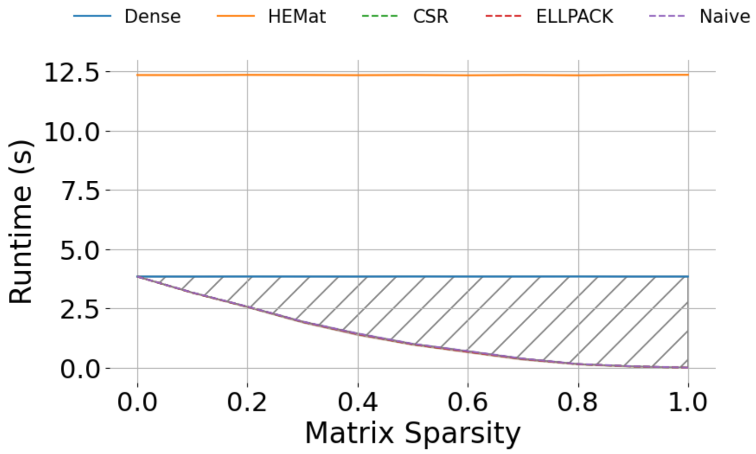


Results: multi-threading

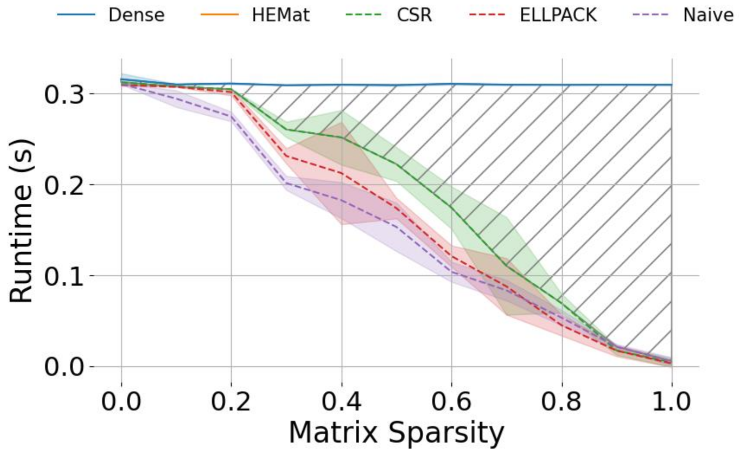
- Our multi-threading scheme appears to scale better than the HEMat baseline
  - – The sparse schemes still maintain a performance advantage at all sparsity levels

- In the future we will investigate how this transfers to GPUs

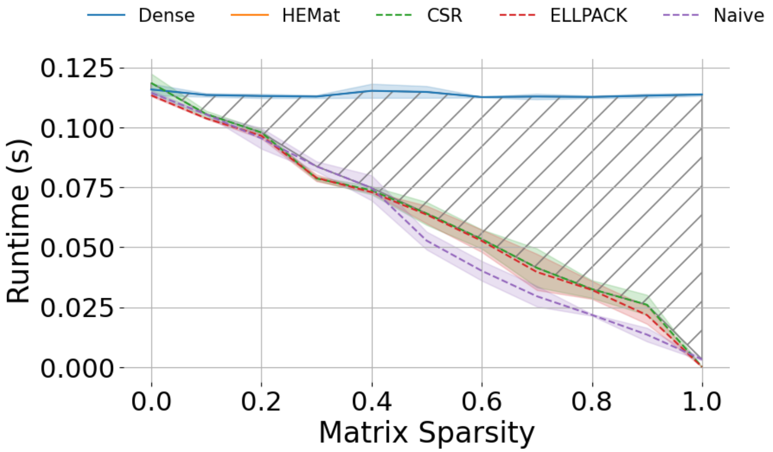
1 Thread



16 Threads - 13x



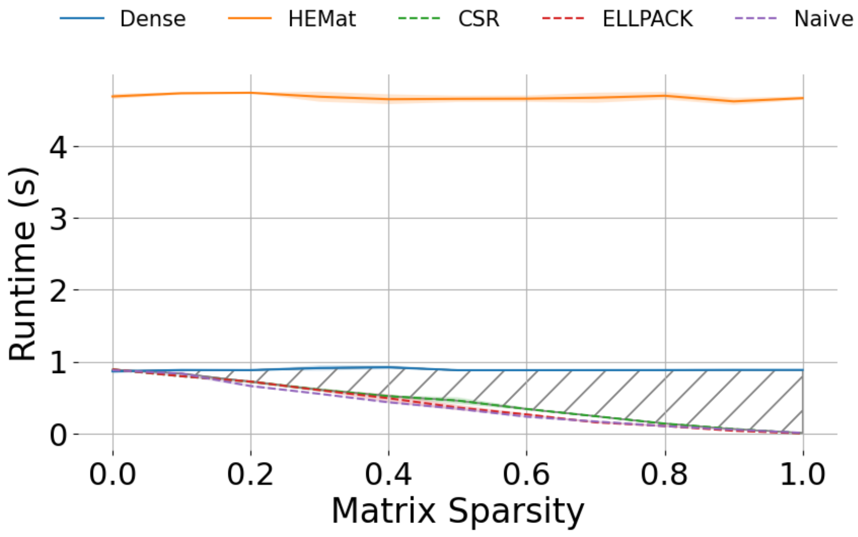
64 Threads – 32.5x



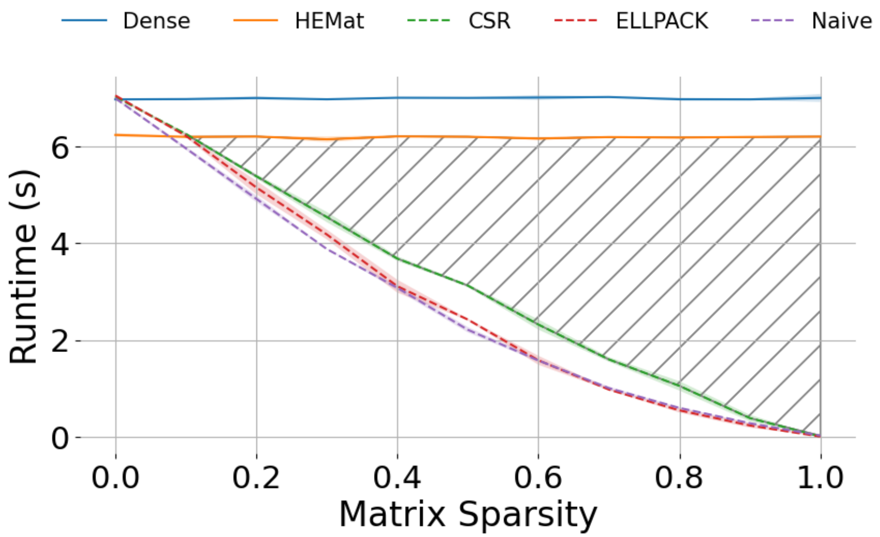
Results: matrix size

- Previous evaluations were conducted with matrix sizes of 8x8, the smallest size that can saturate our CPU's thread count and conform to HEMat's requirements
  - DNN layers are often much larger, so we test how our algorithms scale with respect to matrix size
- Relative to HEMat, they scale poorly, due to the underlying algorithmic complexity advantage of HEMat
  - For many real applications our schemes still provide an increased performance

16x16



32x32



Results: corretness

- We verify all the algorithms for **correctness** at different sparsity levels

- Our proposed schemes are inherently more accurate as sparsity increases
  - Since zero values can be computed outwith the FHE domain, eliminating noise

- Our implementations are, on average, more accurate than the HEMat implementation
  - Average increase in accuracy of 7.6 x 10^{-4}

Sparsity	Dense	HEMat	CSR	ELLPACK	Naïve
0.0	**5.98E-09**	1.34E-03	6.03E-09	6.97E-09	6.66E-09
0.1	5.63E-09	1.44E-03	5.74E-09	6.01E-09	**5.14E-09**
0.2	5.47E-09	1.35E-03	**5.15E-09**	5.77E-09	6.44E-09
0.3	6.03E-09	1.27E-03	6.29E-09	**4.59E-09**	5.11E-09
0.4	4.84E-09	6.81E-04	3.45E-09	**3.21E-09**	3.38E-09
0.5	5.75E-09	7.64E-04	**3.24E-09**	4.36E-09	3.55E-09
0.6	5.06E-09	5.02E-04	2.72E-09	**2.24E-09**	2.80E-09
0.7	4.73E-09	3.96E-04	1.38E-09	**1.32E-09**	1.77E-09
0.8	4.84E-09	1.96E-04	**5.44E-10**	5.92E-10	1.31E-09
0.9	4.61E-09	8.80E-05	**2.85E-10**	2.91E-10	1.28E-09
1.0	4.40E-09	3.15E-06	**0.00E+00**	0.00E+00	8.90E-10

Bold values indicate the most accurate scheme for a given sparsity level

Conclusions

- We have proposed **matrix multiplication schemes** in FHE that exploit **unstructured sparsity** in the context of DNN inference
  - With a 2.5x performance increase at 50% sparsity

- We provide a method for **multithreading** these sparse computations which exhibits strong scaling behavior

- In the **future** we will be exploring
  - Sparsity utilization on GPUs
  - Extending usage of sparsity to better algorithmic complexity
  - Demonstrating sparsity on high dimensional matrices (in SLMs, …)

Exploiting Unstructured Sparsity in Fully Homomorphic Encrypted DNNs

Aidan Ferguson[1], Perry Gibson[1], Lara D'Agata[1], Parker McLeod[2], Ferhat Yaman[2], Amitabh Das[2], Ian Colbert[2], José Cano[1]

[1]University of Glasgow, UK [2]AMD

Thank you! Questions?

Code

Paper