Adaption of Cheon-Kim-Kim-Song (CKKS) for Practical Use in Computer Systems

BOSTON UNIVERSITY

Smaran Manchala[1,2], Seyda Nur Guzelhan[2], Ajay Joshi[2]

Centennial High School, 6901 Coit Rd, Frisco, TX 75035[1],

Boston University Integrated Circuits and Systems Group (ICSG), 8 St Mary's St, Boston, MA 02215[2]

Introduction

- Traditionally encrypted data cannot be altered without decryption, limiting privacy and cloud computing, where third parties benefit from processing data without viewing it.
- Homomorphic Encryption (HE) allows mathematical operations on ciphertext, overcoming this limitation. Refer to the image for details. However, HE is impractical due to slow run time.

Objective

- This study aims to improve the efficiency of Cheon-Kim-Kim-Song (CKKS), a HE algorithm, on a microarchitecture level

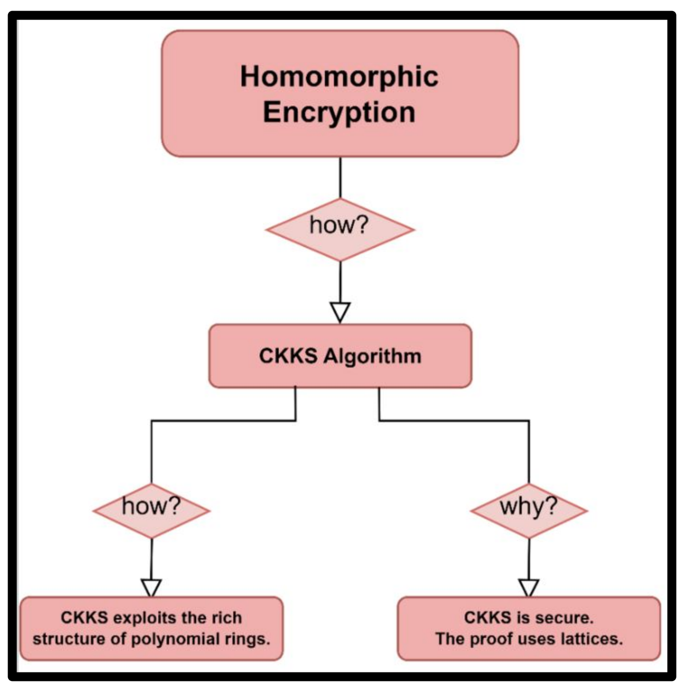

Figure 1. CKKS Algorithm Walkthrough

CKKS

- CKKS is a fully homomorphic encryption (FHE) system

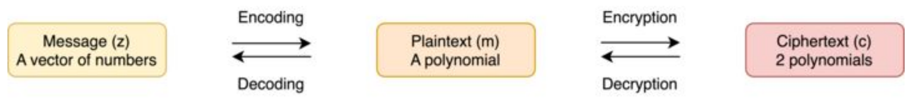

Figure 2. CKKS Algorithm Encrpytion and Decrpytion

Methods

Overview

- To test variable factors, the training runtime of various Neural Networks were compared

Neural Network

- Neural Network is trained on MNIST Database, database of hand drawn numbers
- The Neural Network structure is shown

MNIST Example

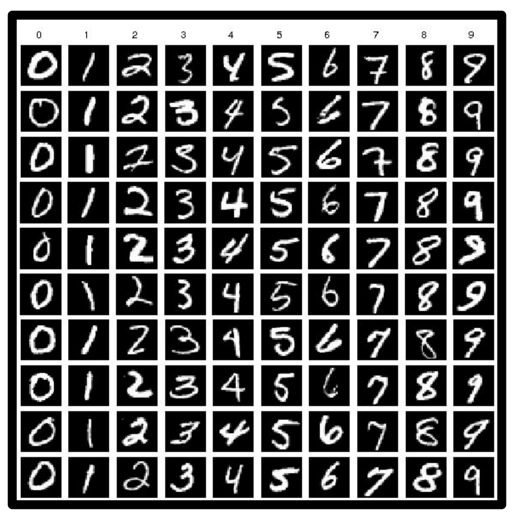

Figure 4. Example images in MNIST Database

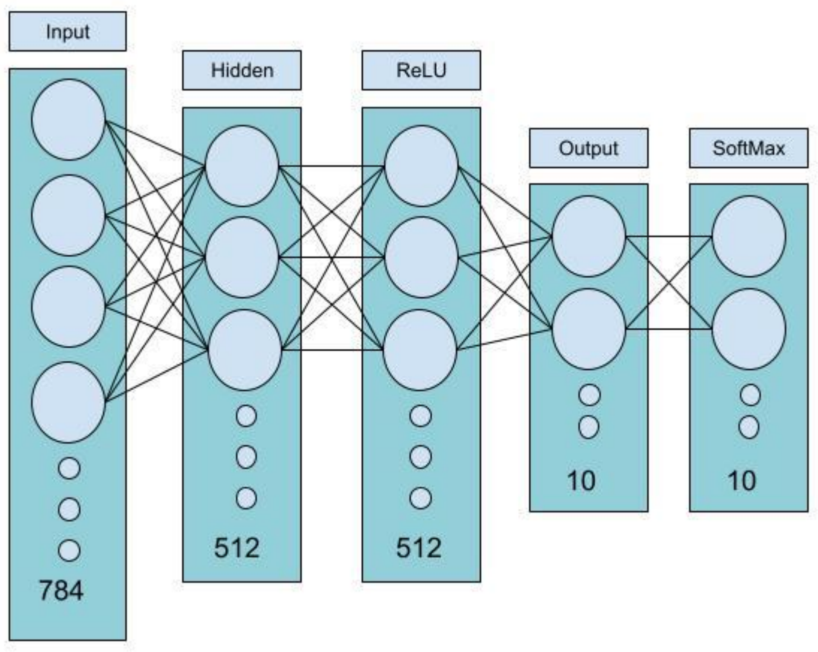

Figure 3. Neural Network Base Architecture

Approximations

- CKKS supports limited operations, mainly addition and multiplication, enabling polynomial evaluations.
- Functions like ReLU, SoftMax, and Loss cannot be directly performed and must be approximated with Chebyshev polynomials.

Variable Factors

- **Computer Architecture:** The two most common computer architectures, ARM and x86 each have layouts that either that could benefit or detriment the runtime.
- **Vector Parallel Processing:** Parallel Processing would cause multiple vectors to processed at once, possibly increasing speed. However, overhead cost cache coherence, and false sharing could cause an algorithm to take more time when run in parallel.

Experiment

- **The experiment with consist of 6 Neural Network Variations**
- **Each of these networks were run 10 times and were recorded and compared**

	ARM	x86
Traditional	Model 1	Model 2
CKKS	Model 3	Model 4
CKKS + Vector Optimization	Model 5	Model 6

Figure 5. Neural Network Variation Classification

Discussion/ Conclusions

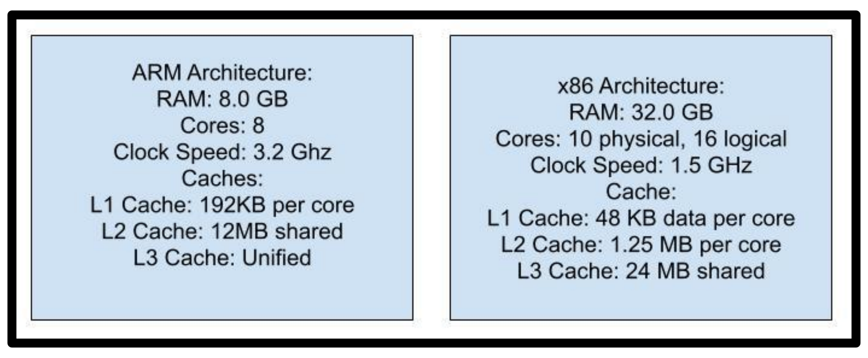

Figure 12. Architecture Scale Factor Application

Results

- Figure above shows design specifications for the Computer used in each Architecture
- **Scale Factor**: 1.84

Runtime Improvements

- For traditional encryption, Model 2 significantly outperformed Model 1 with a 2.63X performance, scaled to 1.43X performance boost for x86 compared to ARM
- Applying encryption to network apprised approximately a 6417X slowdown from performance overhead
- Applying Vector Parallelization with x86 gave approximately a 5.67X performance increase, as well as an approximately 2.45X performance increase with ARM

Conclusion: x86 architecture is more effective with the application of Cryptography. Vector Parallelization is also effective with runtime decrease, the most effective being with x86.

Results

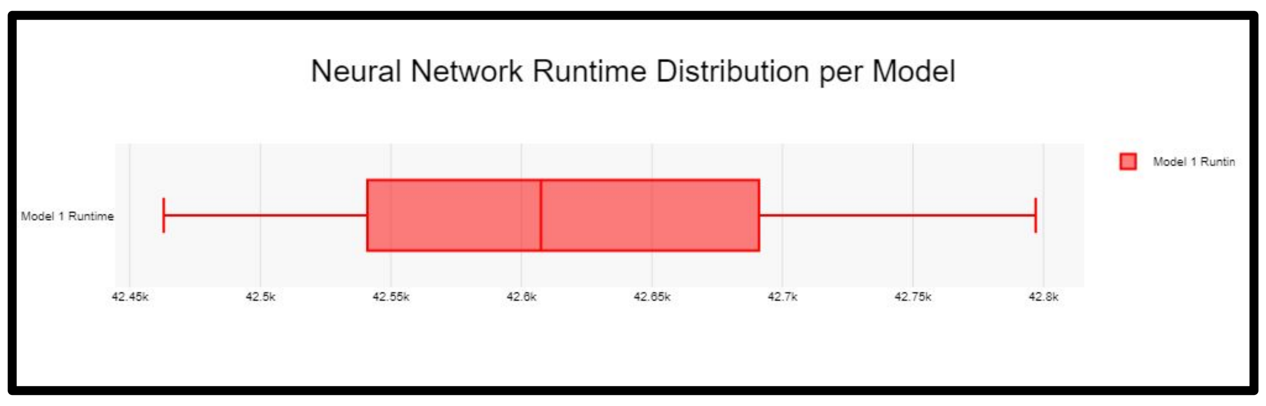

Figure 6. Model 1 Runtime Distribution

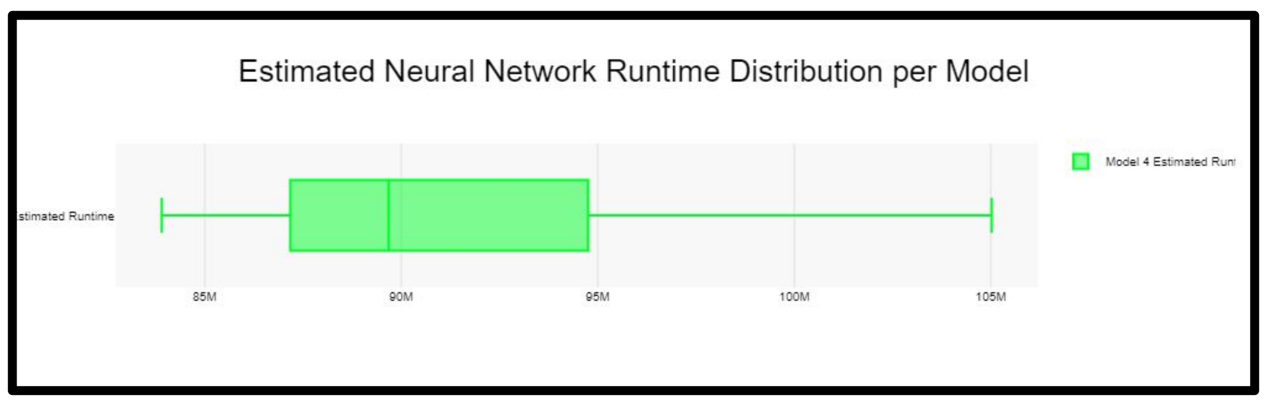

Figure 9. Model 4 Runtime Distribution

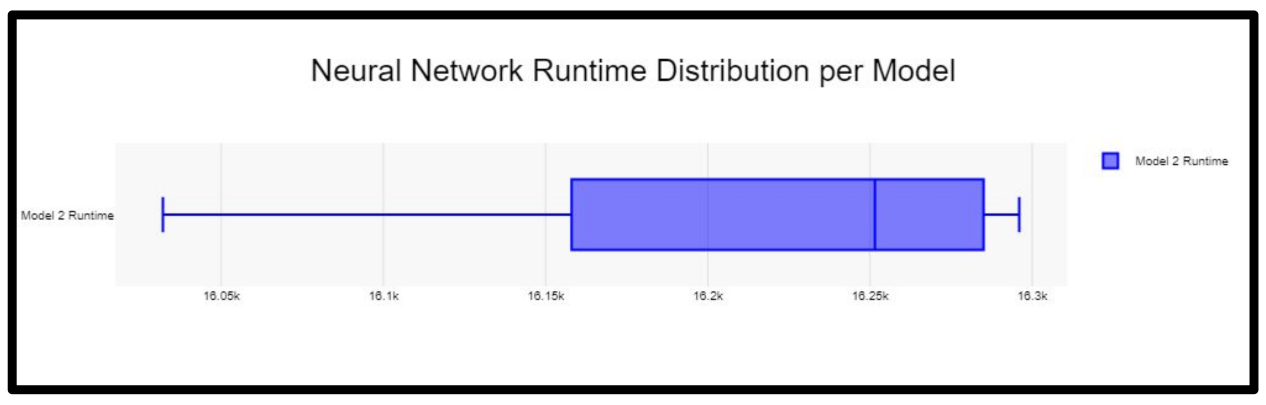

Figure 7. Model 2 Runtime Distribution

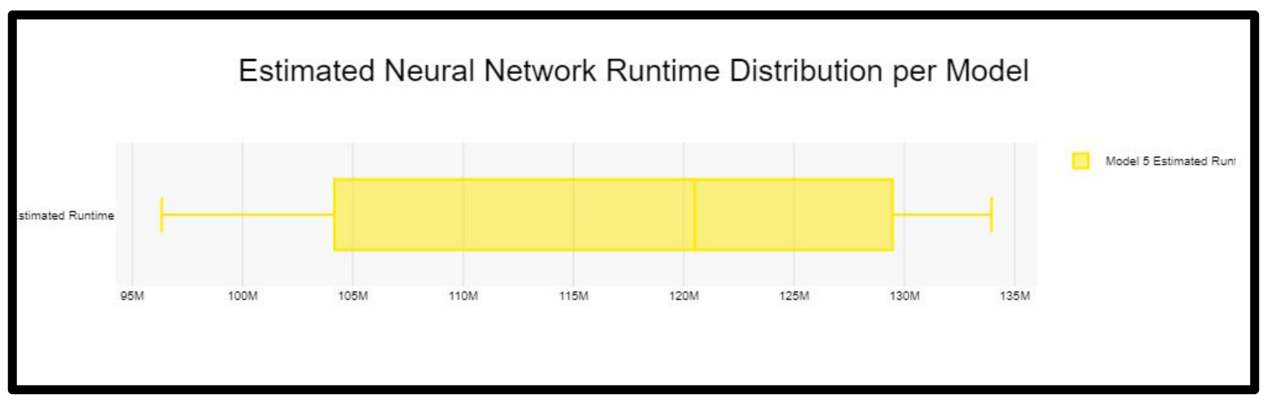

Figure 10. Model 5 Runtime Distribution

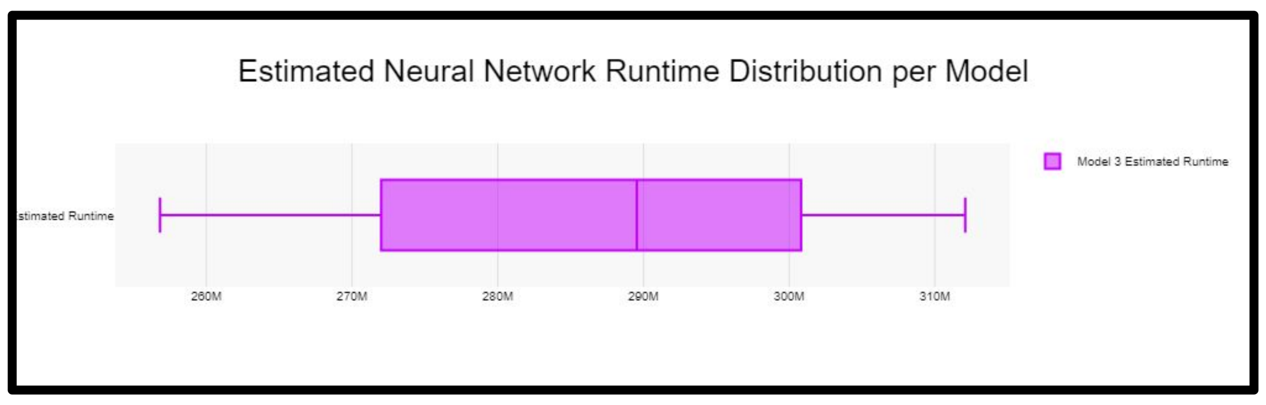

Figure 8. Model 3 Runtime Distribution

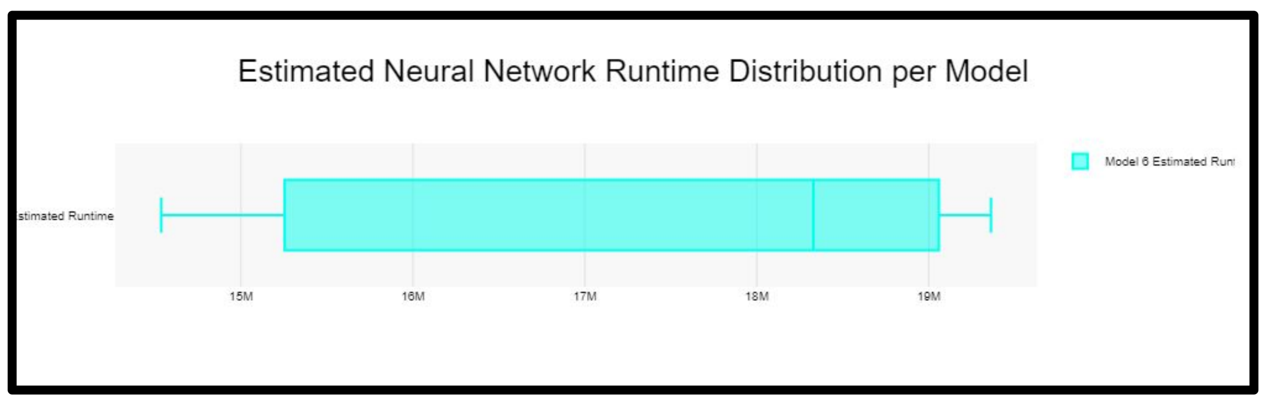

Figure 11. Model 6 Runtime Distribution

References

- Mihara, Kentaro, et al. “Neural Network Training with Homomorphic Encryption.” ArXiv.org, 2020, arxiv.org/abs/2012.13552. Accessed 1 Aug. 2024.
- K. Nandakumar, N. Ratha, S. Pankanti and S. Halevi, "Towards Deep Neural Network Training on Encrypted Data," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 40-48, doi: 10.1109/CVPRW.2019.00011. keywords: {Training;Data models;Computational modeling;Neural networks;Encryption;Machine learning},
- Tsu-Tian Lee and Jin-Tsong Jeng, "The Chebyshev-polynomials-based unified model neural networks for function approximation," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 28, no. 6, pp. 925-935, Dec. 1998, doi: 10.1109/3477.735405.
- keywords: {Chebyshev approximation;Neural networks;Least squares approximation;Recurrent neural networks;Polynomials;Feedforward neural networks;Multi-layer neural network;Multilayer perceptrons;Least squares methods;Computer errors},

Acknowledgements

I would like to acknowledge my mentor, Sedya Nur Guzelhan, in taking time out of her busy schedule to teacher me and guide me as I conducted my own research. I would also like to thank Professor Ajay Joshi for giving me this opportunity as well as guiding me on my project development journey.