Encrypted Neural Networks

SP21 Capstone Design: ***Final Design Report***

Team Number SP22-49

Team members: Andrew Park, August Seiple, Noah Choe, William Basanaga
Advisor: Prof. Sheng Wei

Submitted in partial fulfillment of the requirements for senior design project
Electrical and Computer Engineering Department
Rutgers University, Piscataway, NJ 08854

***Abstract*—Homomorphic encryption is an advancing field within cryptography with many implications regarding data privacy. Machine learning is one field where data privacy is of high importance. We aim to provide a method for which neural networks are able to perform inference on encrypted data securely without exposing any sensitive information. To accomplish this, we implement a standard neural network with PALISADE, a state-of-the-art encryption library which is able to accomplish homomorphic encryption with arithmetic on approximate numbers. We implement a neural network model which is encrypted and is able to perform accurate inferences on encrypted input data. Using these techniques, we achieve an accuracy which matches the standard network.**

***Index Terms*—Machine Learning, Cryptography, Homomorphic Encryption, Data Privacy**

I. INTRODUCTION

A. BACKGROUND

Homomorphic encryption is a form of encryption which allows one to perform computations on encrypted data without having to decrypt the data beforehand. The result of these computations remain encrypted which will yield the resulting computation when decrypted. This allows for computation to be out-sourced without jeopardizing the confidentiality of any information which is exchanged between parties. Some classical cryptosystems are only partially homomorphic, such as RSA, which supports an unbounded number of modular multiplications. In order for a cryptosystem to be considered fully homomorphic, it must allow for the evaluation of arbitrary circuits composed of multiple gates of unbounded depth. This requirement is considered the strongest notion of homomorphic encryption and is difficult to achieve practically. First-generation FHE is most notably introduced by Craig Gentry. Gentry's scheme uses lattice-based cryptography which introduced the first plausible construction for a FHE scheme. It supported addition and multiplication and was limited to evaluating low degree polynomials. It was heavily limited by the noise generated from addition and multiplication, which would ultimately make decryption impossible. This cryptosystem is considered somewhat homomorphic due to these limitations. The introduction of a process known as bootstrapping allowed for the system to evaluate its own decryption circuit and then at least one more operation would allow for somewhat homomorphic encryption to become fully homomorphic. Current fourth-generation FHE(fully homomorphic encryption) is based on the CKKS scheme, named after its creators Cheon, Kim, Kim, and Song. This proposed cryptosystem supported approximate homomorphic encryption that supports a kind of fixed point arithmetic known as block floating point arithmetic. The CKKS scheme allows for an arbitrary number of computations over approximate real numbers with an efficient rescale operation which scaled down encrypted messages after a multiplication. This makes the CKKS scheme the most efficient method for evaluating polynomial approximations.

B. PROBLEM

Machine learning software is popular among companies because it can provide insight into customer behavior, and business trends. However, machine learning often involves the use of massive data-sets in order to train models for accurate performance. Additionally, the deployment of these networks often can not ensure the security of the data it is performed on. Often, these networks are deployed on sensitive or private information. This creates a possible attack vector, where attackers may potentially leak sensitive data such as medical records or financial history. This becomes a bigger challenge with larger models, which may require more computation power than an individual company may possess. In order to alleviate this issue, third-party servers are often used for this task. If the data is too sensitive or private, this poses a challenge for companies: how does one utilize a model which is too computationally heavy on data which is too sensitive to reveal to any third party.

C. OBJECTIVE

Our objective is to create a framework that would allow for machine learning models to be developed and implemented in a way that maintains data security and privacy. Our intention is to create a example using a multilayer perceptron to classify digits using the MNIST dataset. In doing so we hope to be able to demonstrate and evaluate the efficacy of such a solution to data security in machine learning models.

D. APPROACH

In order to implement the machine learning library, we will use the PALISADE library for encryption, specifically

utilizing the CKKS scheme for encryption. The model weights will use the CKKS scheme which allows for arithmetic over approximate numbers. This enables accurate real number representations in the encrypted domain. By encrypting the model weights, the framework ensures that private or sensitive data can not be revealed to any third party as well as ensure that the result of any computation is not revealed. We aim to use fully homomorphic encryption to enable the encryption and computation of data. The client will receive the encrypted output and decrypt it locally to view the result. By implementing these functions we aim to test and evaluate the viability of using fully homomorphic encryption in machine learning models. If successful this framework could be used to create fully secure networks for use in other industries and applications.

II. METHODS

A. CONCEPTUAL DESIGN

To implement this system we aim to create a machine learning network based on a multilayer perceptron to classify handwritten digits from the MNIST dataset. We will use a fully homomorphic encryption library to allow for the encryption and computation. The framework will encompass two main components a matrix library for enabling the encryption and computation of the matrices used for the network and a network library that will implement the multilayer perceptron and handle the higher level network operations. By encrypting the input to the network and the network weights we can ensure that the network remains fully encrypted on the server side using the matrix library to preform the necessary operations. This would allow for the data to be fully secure throughout the entire infrence process.

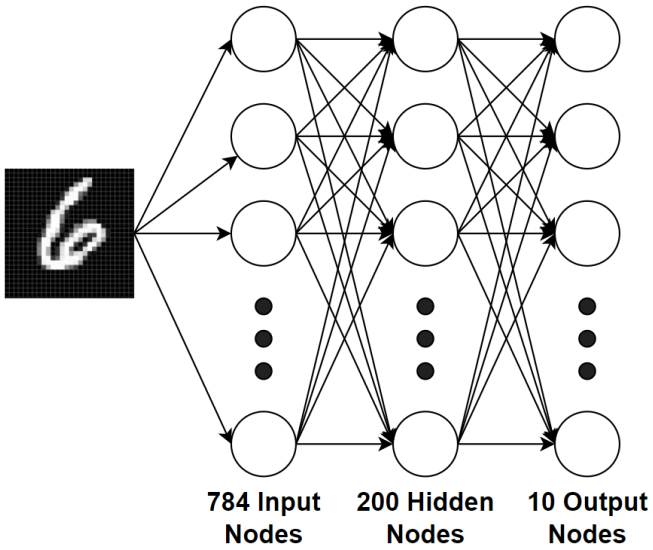
B. DETAILED DESIGN

In order to encrypt the network, we used the PALISADE library for encryption, specifically the CKKS scheme to perform computations over approximate real numbers. While the 128-bit library would allow for higher precision, the memory usage and time needed to perform encrypted computations made it impractical for our use. We opted to use a batch size of 784 with 50 bits of scaling factor. While these parameters were not the most optimal, we found that it provided enough precision for our network to perform an inference while not causing a performance overhead. In order to store the weights in our network, we opted to store each weights matrix as a set of row vectors. These row vectors would be treated as a single ciphertext and the whole matrix as a vector of ciphertexts. We found that we do not have to compute any matrix multiplications. There were only two operations which we needed to implement, matrix-vector multiplication and vector addition. With each matrix being stored as row vectors, we perform the multiplication as follows:

$$W = \begin{bmatrix} w_{11} & \dots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{n1} & \dots & w_{nm} \end{bmatrix} \qquad V = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}$$

$$WV = \begin{bmatrix} w_{11}v_1 + & \dots & +w_{1m}v_m \\ \vdots & \ddots & \vdots \\ w_{n1}v_m + & \dots & +w_{nm}v_m \end{bmatrix}$$

The network we used is a multilayer perceptron with 784 input nodes, 200 hidden nodes, and 10 output nodes. As stated before, weights are stored as a vector of ciphertexts, each ciphertext representing a row vector. The bias for each layer is also stored as a single column vector, which is then stored as a single ciphertext. Each ciphertext is serialized into individual files and stored in a directory. This is then implemented into a client and server. The server initially loads the network from given files and listens on a given port for an incoming connection, from which it will perform commands sent by the client. The client is responsible for connecting to the server and sending a ciphertext which contains the input image. We designed our own simple message passing protocol for our network in order to make communication simpler. Once the server receives the input, it performs an inference. The inference is divided into two phases. The first phase will multiply the input with the first set of weights and add the bias, which results in a ciphertext of intermediary values. This result is sent back to the client, which decrypts the result and then performs the sigmoid activation function over the intermediary values. This is then encrypted and sent to the server for a second phase. The second phase will multiply by another set of weights and add the second set of bias values and sends the result to the client. The client can then decrypt the result and interpret the output data. For our case, we used MNIST data for digit recognition, so the output data is the probabilities of the digit the network predicts.

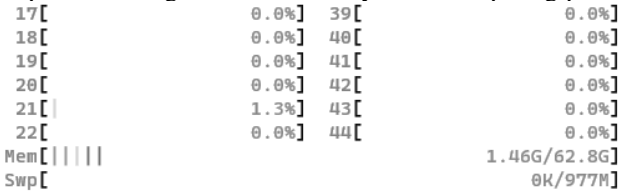


C. USE OF STANDARDS

For our project, we chose to use the C++ programming language as it would have the most compatibility with the PALISADE library, which uses C++. Additionally, we opted to use a Linux-based operating system for our system. Most of our code is written using C based system calls, namely for network communication. The server used in our project contains two Intel Xeon 2699v4 processors and 64GB of RAM, which allowed us to scale the server application to all 44 cores.

D. EXPERIMENT/PRODUCT RESULTS

We were able to successfully achieve 94% accuracy on the MNIST handwritten digit data while maintaining full encryption in the network. In doing this we observed very high CPU and memory usage as a result of the encrypted values. This presents and issues for large scale integration as the framework requires far more computational power than an unencrypted network. We we're able to slightly reduce the memory usage through optimization of the precision of the calculations and the hyper-parameters for the network. Additionally the time required to preform a single inference is approximately 100 times slower than that of an unencrypted network. These problems may be overlooked in situations where real-time performance is not strictly required. Performance could also be improved through the use of vastly more computing power.



III. COST AND SUSTAINABILITY ANALYSIS

A. ECONOMIC

The widespread use of this framework would require a large investment in the number of servers and available compute for these applications. This cost would be hard to justify as the overall data throughput would decrease without greatly increasing the number of servers assigned to the application. Additionally the additional power consumption of the data centers where these servers would be located would also increase the operating costs significantly in addition to the upfront investment. Given the current shortage of computer components this increase in scale would be hard to meet at market prices.

B. ENVIRONMENTAL

The increase computational resources required to run this framework have a large environmental impact as the number of servers required would greatly increase while achieving the same data throughput. The raw resources required to manufacture these servers would drastically increase with the additional manufacturing of more servers. The e-waste generated would see a large increase as more servers would need to be upgraded and replaced as technology improves. Adding more high power servers to a data center would also greatly increase the cooling demands of the data center. These data centers do not tend to use sustainable methods for cooling their servers and thus would have a large increase in power consumption for air conditioning and cooling. The servers themselves would also require far more power and would further increase the power demands of the data center.

C. SOCIAL

The framework would have far reaching social impact as more of the systems we interact with become integrated with machine learning techniques. The use of homomorphic encryption would allow for user to feel more secure with their data as the companies processing and storing it would not be able to see and use the data for other purposes. This system addresses the need for data security and privacy in fields where such protection is required by regulation or simply desired by the users. This would allow for the use of machine learning in additional fields such as medical records as well as the improvement in privacy for applications such as spam detection. This has to potential to enable more industries to make use of machine learning in processes that otherwise would have been forbidden by data privacy regulations.

IV. CONCLUSION/SUMMARY

We believe that this framework has the potential to enable far broader use of machine learning techniques in industry and research. Many applications have seen huge advancements from the use of these networks, we hope that by enabling a more secure method of preforming these calculations we can bring machine learning in to new fields which were previously unable to preform these expensive calculations while maintaining data security. In addition we hope that this system can be implemented into existing applications to improve user privacy. For the system to become more practical for existing applications, there are certain improvements which are implementable with more research and testing. Namely, the use of sigmoid on the client is a limiting factor for the security and efficiency of our project. We were able to experimentally apply the sigmoid activation function fully encrypted through the use of an approximation polynomial. We opted to not go about this route due to constant segmentation faults caused within the library which can be attributed to memory usage. Additionally, the computation time needed to perform an approximate sigmoid made this option impractical for our use. Some other improvements include general improvements to the encryption parameters. These improvements would allow for existing applications to more easily implement our project.

ACKNOWLEDGMENTS

We would like to thank Professor Sheng Wei for his guidance and assistance throughout this project. We would also like to thank the Rutgers ECE Department and Professor Hana Godrich for the opportunity and support over the past year.

REFERENCES

[1] "PALISADE Lattice Cryptography Library (release 1.11.5)," https://palisade-crypto.org/, Sep. 2021.

[2] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," 2021. [Online]. Available: https://arxiv.org/abs/2106.07229