Efficient Privacy-Preserving NN Inference at the Edge

Antonio R. Torres Milán

Submitted to the faculty of the WPI ECE department as part of the completion of Master's of Science in Electrical and Computer Engineering

March 2024

Approved by:

____

Professor Fatemeh Ganji, Major Thesis Advisor

Contents

Abstract

Neural networks (NNs) have been widely adopted in practice to create predictive models in applications such as healthcare, financial services, and policy-making. As their use continues to grow, so does the risk of attacks against users' data and NNs. While traditionally, deep learning was constrained by computational power and off-chip memory bandwidth, such attacks impose new priorities in connection with security and privacy. Privacy-preserving deep learning addresses these issues by leveraging cryptographic primitives, e.g., homomorphic encryption and secure multiparty computation (MPC). MPC-based solutions offer a higher degree of flexibility by allowing different parties to train an NN model on their private data without revealing any information beyond the output. In this regard, combining MPC and deep learning enables a variety of privacy-preserving online services. As an example, to classify a picture, a customer can use an online deep learning service, where the service provider and the user engage in an MPC protocol instead of just uploading the picture. In doing so, the user obtains the classification result without revealing the input, while the provider can keep its model secret.

Existing work in the area of MPC falls into two main classes: (1) MPC over Boolean circuits and (2) MPC over arithmetic circuits. While the former class relies on Yao's garbled circuits and achieves constant communication complexity, secret-sharing (SS)-based solutions have been adopted to evaluate arithmetic circuits with a communication complexity linear in the multiplicative depth of the circuit. Nevertheless, there are MPC frameworks enjoying the benefits of both classes by com-

bining them. Further optimization can be performed, which is applied in the context of private NN inference. Resource optimization is even more vital for applications where NNs run on resource-constrained edge devices. It has been shown in the literature that running MPC-based privacy-preserving NNs on FPGAs brings down the protocol execution time and power consumption within the practical limit. In line with such efforts, our work presents optimization techniques that utilize the full capability of the underlying FPGA hardware.

More concretely, FPGAs' parallel processing and pipelining capabilities allow for faster computations essential in NN tasks. This is complemented by optimized memory access, which minimizes latency and maximizes data throughput. Moreover, the flexibility to develop custom instruction sets tailored for NN operations enhances protocol execution efficiency. Finally, the algorithm-hardware co-design approach ensures that both the NN algorithms and FPGA architecture are optimally aligned for performance, making FPGAs a powerful and efficient choice for the secure implementation of neural network accelerators through multi-party computation.

Our results demonstrate that our approach uses significantly fewer hardware resources, up to 62.5 times fewer logical resources, and 66 times less memory than cutting-edge privacy-preserving NN interfaces. Furthermore, in scenarios where execution time is critical, our approach proves to be 2.5 times faster than the average execution time of the privacy-preserving NN inferences while closely matching the performance of the fastest state-of-the-art approaches to privacy-preserving NN inferences.

Chapter 1

1 Introduction

Neural Networks (NNs) have become a helpful tool of the digital era because of their utilization in machine learning, thus providing a powerful instrument for data analysis and prediction in health, finance, and autonomous systems. Most of these applications will involve sensitive and personal data; therefore, the need to ensure privacy in using neural networks has increased significantly. Privacy-preserving inference techniques generally work so that sensitive data are hidden during the computation phase, and it does not allow any authorized access or data leakage.

Secure multi-party computation (MPC) techniques in neural network operations, operating in general, would handle these privacy issues very well. In MPC, parties compute functions over their inputs jointly while keeping the private inputs for themselves. They are the basic building blocks underlying a host of more complex protocols, of much appeal when data cannot be pooled because of privacy regulations or business competition [1].

The rapid evolution of data-driven technologies has led the way in a transformative era across critical sectors such as healthcare, finance, and public safety. However, this progress increases vulnerability to data breaches and privacy violations, as highlighted by recent incidents that have compromised personal data on a massive scale [40]. Such vulnerabilities underscore the urgent need for advanced privacy-preserving techniques to keep up with the technological advancements and the ever-growing sophistication of cyber threats [41].

Secure Multi-Party Computation (MPC), celebrated for its theoretical robustness in privacy protection, confronts significant barriers in real-world implementation. These barriers include computational inefficiency and difficulties adapting to diverse operational environments that vary significantly in data volume, system architecture, and security requirements [42]. The imperative drives this thesis to transition MPC from a theoretical concept to a practically viable solution tailored to operate within real-world applications' complex and stringent constraints. Addressing these challenges involves enhancing the computational efficiency of MPC protocols and ensuring their flexibility to integrate seamlessly with existing infrastructures across various industries [43].

This thesis further explores how privacy-preserving neural network inference can be enhanced with increased computing capability and secure Field-Programmable Gate Arrays (FPGAs) features. FPGAs are flexible and efficient platforms for deploying MPC protocols, and the hardware should be reconfigurable to optimize the performance of privacy-preserving computation [2]. Therefore, the research aims to merge the gap between highly advanced cryptographic techniques and practical, real-life application needs by focusing on FPGA-based implementations. Optimized methodologies shall be developed using the FPGA parallel processing capabilities to speed up the MPC-based neural network inference with solid data privacy.

Multi-party computation (MPC) has become a core technology in secure and privacy-preserving data analysis. Conceived originally for situations where a state of mutual distrust obtains between parties, MPC allows computations on privately held inputs so as not to reveal them to the other engaging parties [3]. This aspect is of vast importance, especially in neural network inference, particularly in cases where the inputs deal with sensible information and, hence, parties must keep the details confidential.

On a neural network basis, the application of MPC shall allow a couple of concerned parties to make predictions jointly without disclosing, in turn, their respective secret or sensitive data. One example is in healthcare: hospitals will predict the outcomes of patients together, but they will not share the data of any given patient, hence falling even under strict privacy regulations, like HIPAA in the United States [4].

Furthermore, MPC ensures that confidentiality is preserved and integrity enhancement is provided. This is because no one party may change the computation process or its outputs, and as such, the model necessarily remains correct. This double benefit of ensuring confidentiality and integrity makes MPC a beautiful tool for implementing privacy preservation in the neural network. Recent developments in this area are moving toward reducing the computational overhead and improving the efficiency of such protocols to make them practical for real-world applications [5].

This thesis will attempt to exploit MPC's intrinsic efficiencies while avoiding traditional computational challenges by integrating MPC with neural network operations on FPGA platforms. In this direction, parallelism within the FPGA enhances the execution of MPC protocols and dramatically reduces time and other resources for secure neural network inference.

Field Programmable Gate Arrays (FPGAs) are gaining importance because of their essential role in the evolvement of secure computations, particularly where most applications need high levels of privacy with data security. FPGAs have the architectural benefits that uniquely assist in this task under their reconfigurability, parallel-processing capability, and efficient hardware usage—factors that help to execute MPC and, in doing so, complete its complex cryptographic protocols.

This hardware is reconfigurable and fine-tuned to precisely the computation requirements of the privacy-preserving neural network inferences. This capability becomes very important in rapidly evolving fields—like cryptography—where protocol changes can be used without new hardware, hence saving costs and times of deployment [2]. The parallel processing offered by FPGAs provides the ability to run multiple operations simultaneously, significantly increasing the overall computation time crucial in real-time applications [6].

Most notably, FPGAs increase the privacy aspect of neural network inference. Most cryptographic algorithms can be directly implemented in hardware and provide more security from side-channel attacks than their software implementations. Such security is paramount at the hardware level for applications running in sensitive domains, such as healthcare and financial services, where data integrity and confidentiality must be preserved at the topmost levels [7].

This thesis investigates the use of FPGA technology to accelerate the execution of MPC-based privacy-preserving neural network inferences. Therefore, The study will use the computational power and flexibility of FPGAs to register a marked improvement in both speed and resource efficiency of the said operations, making them more viable for use in resource-constrained environments.

They realize that neural networks are privacy-preserving while efficient, requiring much complexity. The significant challenges include computational complexity, scalability, and achieving a trade-off between privacy and performance. Today, neural networks are much more profound and broader, and in most cases, the computational requirements to support the scaling of privacy-preservation techniques are high.

First, computation tasks in neural networks under any privacy protocol, such as MPC, will be substantially more complex than those with private protection. This means that computing encrypted data is complex; it usually requires much more processing power and often also entails vastly increased difficulty in practical realization in real-time applications. [8]. These problems further aggravate should one try to scale them to larger datasets or more complex neural network architectures, which practically proves to be a challenging problem in developing efficient solutions.

There are also some inherent trade-offs between privacy and computational efficiency. For instance, higher levels of privacy imply the use of more heavyweight cryptographic measures, which would detrimentally affect system performance.

The balance is exceedingly delicate in resource-constrained environments, such as edge computing devices in which available computational resources are limited, and power availability further limits them [9].

Against this background, this work considers using FPGA technology to mitigate computational loads while carrying out private neural network inferences. The FPGAs are very suitable for this, with their parallel processing strictly adhered to to allow optimized cryptographic computations. This paper tries to illuminate the way toward more feasible, efficient, and practical implementations of privacy-preserving neural networks by building customized hardware solutions that cater to the needs of secure and private neural network operation.

This will be a very efficient and practical private-preservation inference of neural networks that utilize FPGA technology innovatively. This project focuses on proving that engineered FPGA solutions will bring potential strides toward reduced computation and resource overheads associated with Multi-Party Computation (MPC) protocols. It is on these that the research pursued in this paper is narrowed down in its operation to meet the specific needs for the secure and private operations of the neural network. In this manner, one must balance the privacy, speed, and usage of computational resources to make the above-mentioned advanced technologies feasible in real-world applications.

This study covers developing and evaluating an FPGA-based framework that optimizes MPC protocols for use in neural network inference tasks. It will include developing hardware-efficient cryptographic primitives and designing platforms in the form of FPGAs to implement them and measure performance enhancements over traditional computational techniques.

This thesis introduces groundbreaking FPGA-based techniques designed to significantly optimize MPC protocols tailored for neural network inference, representing a crucial advancement in applying privacy-preserving technologies. By harnessing the inherent capabilities of FPGAs for parallel processing and hardware acceleration, this research directly addresses the critical performance bottlenecks that have hampered previous implementations of MPC [44]. Moreover, it substantially improves the scalability and security of neural network computations, making these advanced cryptographic techniques more accessible and practical for a broader range of applications [45].

The outcomes of this study are set to redefine the benchmarks for deploying cryptographic techniques in everyday technological applications, propelling a paradigm shift in how privacy is integrated into data-driven systems. By demonstrating the enhanced performance and versatility of FPGA-optimized MPC protocols, this research paves the way for their wider acceptance and implementation in sectors where privacy is paramount and currently under-served by existing solutions [46]. Ultimately, this thesis contributes to a more secure and trustworthy digital infrastructure where privacy-preserving technologies are embedded into everyday technological interactions [47].

The thesis is organized, respecting a couple of significant sections, in a systematic way, unfolding an understanding of the subject in question. The first chapter discusses the introduction to the critical importance of privacy in neural network inference and the roles of MPC and FPGA in the same regard. The next chapter, Background, will review theoretical insights of existing technologies over MPC and FPGA with details of the prior approaches towards privacy-preserving neural network inference. The next chapter, Methodology, will describe the methodologies that will be used to develop the FPGA-based MPC framework, and this description may be either cryptographic protocols, hardware configurations, or software tools. This chapter details the implementation perspective and how the proposed framework is practically executed over FPGA, including its setup, programming, and optimization processes. The chapter discussing the Results involves a comparative analysis of the solution developed on FPGA with the existing methods to evaluate the performance concerning speed, resource utilization, and privacy protection. Lastly, the Conclusion and Future Work chapter highlights key findings and contributions and sets forward the limitations and areas that might be explored. Designed to follow on from the insights and findings of the previous, each chapter is crafted to weave a cohesive narrative that displays the feasibility not just of the proposed solutions but also their potential impact on the field of privacy-preserving computations.

Thesis Contributions to The Field:

This thesis provides several critical contributions to secure multi-party computation (MPC) and privacy-preserving neural network inference. Our work addresses the computational inefficiencies and privacy concerns associated with deploying neural networks, particularly in the sensitive contexts of healthcare and finance, where data protection is paramount. The main contributions are as follows:

- Enhanced Computational Efficiency: We demonstrate significant advancements in the speed and efficiency of MPC protocols by using field-programmable gate arrays (FPGAs)
- Resource Optimization: By implementing MPC protocols on FPGAs, we achieve substantial savings in hardware resources.
- Scalability: We propose a scalable approach to privacy-preserving neural network inference. Our methodology accommodates the scaling requirements of various neural network architectures, demonstrating that FPGAs can handle more significant, more complex models without compromising efficiency or privacy.
- Security and Privacy: We incorporate secure MPC protocols with FPGA technology to enhance the privacy and security of neural network inferences.
- Practical Real-world Application: Our research bridges the gap between theoretical cryptographic techniques and their practical application. The opti-

mized FPGA-based MPC framework proposed is viable in a laboratory setting and offers significant improvements for real-world implementations.

- Contribution to State-of-the-Art Comparison: We contribute to the body of knowledge by providing a comparative analysis of our FPGA-based MPC framework, MOTION2NX, with state-of-the-art framework, offering concrete data points and insights into performance and security.

Chapter 2

2 Background

2.1 Neural Networks

2.1.1 Overview

Neural networks are computational models of biological neural networks in animals' brains designed to identify patterns and make decisions based on learned responses. Structurally, they contain layers of nodes or "neurons" connected by weighted edges. As the data propagates through the network, the weights of all these connections must be adjusted based on feedback from the output, steadily refining its predictions or classifications with each iteration [10].

The fact is that such networks can be pervasive, embracing the full range of tasks from primitive functions such as recognizing hand-written digits to making the kind of complex decisions required in autonomous vehicle navigation. In health, from imaging data to financial assignments that involve predicting stock market trends or detecting fraudulent activities, the tasks are so much that the neural networks have stretched their arms across various lines [10].

This is important, especially since sensitive data is increasingly entrusted to neural networks today, most notably in the medical and financial fields. This training and inference data often include personal, sometimes even proprietary, information that risks being exposed if proper precautions are not taken. They make it inevitable to include privacy-preserving methods, ensuring that the safety of the data is included while allowing for effective operation by the neural networks.

The private-preserving way in the applications of neural networks seeks to secure the data by the principle of least privilege, ensuring that no more information has been exposed than is necessary to accomplish the task. Differential privacy ensures that a framework adds noise to the datasets used in training neural networks, hence entirely leaking the properties of the population of individuals in databases [11]. In doing so, both these methods ensure that the risk of disclosing secret data is mitigated. Secondly, neural networks remain in the domain of applications that involve privacy.

2.1.2 Neural Network Models

The perceptron is the simplest form of a neural network, invented by Frank Rosenblatt in 1957. It is a single-layer binary classifier that uses a linear threshold gate to process inputs, making decisions based on whether the sum of the weighted inputs exceeds a certain threshold. However, perceptrons can only solve linearly separable problems due to their simplicity, limiting their application in complex scenarios [54].

The next neural network model, one of the most commonly utilized because of its accuracy, is the Convolutional Neural Network (CNN). Developed primarily for processing grid-like topology data such as images, CNNs use convolutional layers that apply convolution operations to the input, capturing spatial hierarchies in data. CNNs typically include convolutional layers, pooling layers, and fully connected layers. They are highly effective in tasks such as image and video recognition, image classification, and medical image analysis [55].

Unlike feedforward neural networks, Recurrent Neural Networks (RNNs) have loops, allowing information to persist. This architecture makes them ideal for tasks involving sequential data, such as speech recognition or language modeling. RNNs can remember important information about the input they received, making them precise in predicting what is coming next. However, they are often difficult to train effectively due to problems like vanishing and exploding gradients [56].

To overcome the shortcomings of traditional RNNs, Long Short-Term Memory (LSTM) was introduced with a more complex architecture that includes mechanisms called gates to regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away, making them highly effective for complex tasks that require learning from long data sequences, such as language translation, text generation, and even complex time-series prediction [56].

Lastly, Generative Adversarial Networks (GANs) were introduced by Ian Goodfellow in 2014. They consist of two models: a generative model that captures the data distribution and a discriminative model that estimates the probability that a sample came from the training data rather than the generative model. The training procedure for GANs is a min-max game where the generator tries to maximize the probability of the discriminator making a mistake. This new framework has produced high-quality synthetic images, 3D models, and even enhanced low-resolution videos [58].

Each model offers unique advantages and specializes in different tasks based on their architecture and the specific problems they aim to solve. However, for a more generalized approach that captures a broad spectrum of tasks from, simple binary classification to more complex hierarchical pattern recognition without the need for temporal dynamics, Multi-Layer Perceptrons (MLPs) stand out.

2.1.3 Multi Layered Perceptron

A Multi-Layer Perceptron (MLP) is a feedforward artificial neural network that includes three or more layers: an input layer, several hidden layers, and an output layer. Each layer consists of neurons that, except for the input nodes, use a nonlinear activation function, enabling the MLP to capture complex relationships in data. The architecture of an MLP is straightforward: the input layer receives data and passes it on without computation; the hidden layers perform the bulk of processing through weighted connections followed by nonlinear transformations; and the output layer delivers the final prediction or classification result [50].

The effectiveness of MLPs in processing nonlinear relationships is primarily due to the use of activation functions such as sigmoid, tanh, and ReLU. These functions introduce non-linearity to the network, allowing it to learn and model complex patterns [10]. For instance, sigmoid functions provide outputs between 0 and 1, making them ideal for binary classification tasks, while ReLU offers advantages in terms of computational efficiency and convergence during training.

Training MLPs involves two phases: forward propagation, where inputs are passed through the network to generate an output, and backpropagation, where the network adjusts its weights based on the error between the predicted output and the actual data. This error is calculated using a loss function, and the process of adjusting the weights is facilitated by optimization algorithms like stochastic gradient descent [51].

MLPs are widely applied in various domains, such as image recognition, which classifies images based on learned features; speech recognition, which involves processing and classifying audio signals; and financial forecasting, which predicts market movements or credit risks. Despite their versatility, MLPs can suffer from issues like the vanishing gradient problem, where gradients used during training diminish as they are propagated back through the network, stalling learning. They are also prone to overfitting, mainly when the network architecture is too complex relative to the amount of training data [52].

Recent advancements have addressed some of these challenges. Techniques like dropout, which randomly deactivates neurons during training to prevent overfitting, and optimizers such as Adam, which adjust the learning rate dynamically, have improved MLP training outcomes. Batch normalization has also been introduced to stabilize the learning process by normalizing layer inputs, thus accelerating convergence [53].

2.2 Multi-Party Computation

Multi-party computation (MPC) represents a pivotal cryptographic technique designed to enable multiple parties, each holding private data, to jointly compute a function without revealing their inputs to each other. Originating from the foundational works in the 1980s, MPC has evolved to address complex problems across various domains requiring privacy preservation during computations. This capability is particularly vital in scenarios where data privacy regulations or competitive business interests prevent the direct sharing of information. MPC protocols ensure that no more information is shared than is necessary for the computation, thus maintaining data confidentiality while allowing for the collective analysis or decision-making that can benefit all parties involved [12].

Recent studies have emphasized the focus of multi-party computation with neural networks to enable privacy-preserving data analysis and model training. For example, Akari et al. (2017) demonstrated an efficient implementation of multi-party computation with the sole focus on reducing communication overhead, which is often mentioned as the primary limitation in scaling multi-party computation protocols [15]. Furthermore, Mohassel and Zhang (2017) presented SecureML, which allows two parties to train together in a logistic regression model under privacy constraints [20].

2.2.1 SecureML

SecureML is a system designed primarily for privacy-preserving machine learning that enables two parties to collaboratively train models using their private datasets without revealing the data to each other. Presented by Mohassel and Zhang in 2017, SecureML represents a significant advancement in the field of secure and private machine learning [19].

SecureML uses a combination of cryptographic techniques, including secret sharing and secure multi-party computation, to ensure the confidentiality and integrity of data during the machine learning process. The system is optimized for two-party computations, taking advantage of the properties of additive secret sharing along with garbled circuits, a method first introduced in the context of secure function evaluation[1].

One of the many core aspects of SecureML is its efficiency in handling linear and non-linear functions, which are pivotal in machine learning algorithms. SecureML uses secret sharing for linear operations that permit straightforward and efficient computations. On the other hand, for non-linear computations found in activation and optimization algorithms, SecureML employs garbled circuits and oblivious transfers, improving the system's efficiency while remaining private [19].

SecureML was one of the first systems to demonstrate that secure neural network training could be achievable in practice. The system supports the training of logistics regression and neural network models with a focus on efficiency that is practical for real-world applications. The experiments conducted on SecureML showed that it could perform training on datasets like MNIST and CIFAR with reasonable computational overhead compared to non-secure training [19].

The introduction of SecureML brought a broad implication for industries where data privacy is primordial. Nevertheless, collaboration is necessary to improve the predictive models, such as healthcare, finance, and personalized advertising. For example, in healthcare, institutions can collaborate on predictive models for patient diagnosis without exposing sensitive patient data, complying with detailed data protection regulations like HIPAA in the United States.

Despite having various advantages, SecureML faces challenges primarily related to scalability and computational efficiency when dealing with more complex models and larger datasets to process. Future research includes optimizing the cryptographic protocols used, integrating with other privacy-preserving technologies such as differential privacy, and extending the framework to support multi-party configurations beyond two parties.

2.2.2 Yao's Garbled Circuit

Two-party computation, the basis of secured computation according to the concept introduced by Andrew Yao in 1986, is Yao's Garbled Circuit. The idea enables two parties—a garbler and an evaluator—to jointly compute a function without necessarily having to reveal their respective inputs to each other. The garbler in this protocol encrypts the Boolean circuit—i.e., "garbles" it—so that the efficacious operations are effectively obfuscated. The evaluator then processes the garbled circuit with encrypted inputs and computes the output without gaining any knowledge about the input of the garbler. This mechanism has given robust protection to privacy in computations, especially in sensitive negotiation or competitive interaction scenarios, where revealing more information can compromise a participant's strategic advantages. Yao's Garbled Circuits are mainly general purpose, applying to any function that may be well modeled as a Boolean operation—encompassing many logical and binary arithmetic processes that are bread-and-butter operations of digital computing systems [13].

2.2.3 Goldreich-Micali-Wigderson

Named after their creators (Oded Goldreich, Silvio Micali, and Avi Wigderson), the GMW protocols have been extended to multi-party protocols that allow secure computation between any number of parties. This can be a group of people interested in conducting private computations. Introduced in 1987, these protocols proposed adopting methods such as secret sharing and circuit evaluation to ensure that the inputs of all parties remained private throughout the entire computation. Unlike Yao's Garbled Circuits, which are designed for two-party interaction, GMW Protocols are adaptable and adjust to multiple parties in a multi-party configuration. This flexibility is precious in protocols like GMW, where collaborative computations are conducted among multiple parties for applications that range from joint economic forecasting and collaborative scientific research to corporate settings for collective decision-making. These protocols have contributed significantly to cryptographic research, providing essential tools in multi-party computations that enable security and efficiency with various applications [14].

2.2.4 Oblivious Transfer

Oblivious transfer (OT) is another core protocol of multi-party computation and garbled circuits, in which a sender transfers one of the various information choices but remains oblivious to which data was chosen. The concept was first presented by Michael O. Rabin in 1981. It has since transformed into various forms of its original application, with 1-out-of-2 oblivious transfer and its generalization being the most common. The concept is primordial for privacy-preserving as it allows the execution of secure computations without revealing individual inputs.

In a 1-out-of-2 oblivious transfer, a commonly used variant of OT, the sender has two messages, and the receiver wishes to obtain one of the messages without revealing which one they are interested in to the sender. Generalizing to 1-out-of-n OT allows for a selection among the 'n' messages. The sender needs to be made aware of which message has been transferred. OT's significance extends beyond simple actions, as it can be integrated into more complex multi-party computation protocols that may involve more than two parties or different computational tasks.

2.2.5 Gabrled EDA

Electronic Design Automation (EDA) tools are critical for modern integrated circuit (IC) design involving multiple parties, such as third-party IP vendors, design houses, and foundries. These stakeholders often operate under mutual distrust, risking the confidentiality and integrity of their intellectual property (IP). Traditional IP protection methods like logic locking and encryption have proven susceptible to various attacks, leading to a need for a more secure approach [48].

Garbled EDA introduces a novel framework built on secure multi-party computation (MPC) principles, fundamentally shifting the traditional paradigms of EDA. This framework aims to ensure the privacy of IPs, CAD tools, and process design kits (PDKs) without compromising operational efficiency. Garbled EDA is designed to function seamlessly with commercial EDA tools and operates under a zero-trust environment, meaning that others inherently trust no party.

The core mechanism behind Garbled EDA is based on secure function evaluation (SFE) and private function evaluation (PFE) protocols, which allow for the secure and private computation of functions without revealing individual inputs [48]. This approach ensures that even if parties are malicious, the design and function of IPs remain protected. The Garbled EDA framework utilizes open-source tools, allowing easy integration and robust security for IPs in potentially hostile environments. Garbled EDA has been evaluated to demonstrate minimal logical resource cost and negligible memory overhead, making it a viable solution for real-world applications. This framework addresses the limitations of traditional EDA tools and sets a new standard for protecting electronic designs in an increasingly interconnected and security-sensitive industry.

2.2.6 Tiny Garble

TinyGarble represents a milestone in secure computation, particularly in garbled circuits—a cryptographic technique that allows for the private evaluation of computational functions on encrypted inputs. TinyGarble's advancement stems from its unique approach to constructing and evaluating these garbled circuits at unparalleled speeds. The primary innovations come from cryptographic advancements—utilizing schemes that need only a single call to a fixed permutation, often instantiated by fixed-key AES—for garbling gates. On a systems level, TinyGarble benefits from more efficient representations of circuits, enhancing overall performance.

The pursuit of efficiency in garbled circuits is a critical endeavor, with many researchers aiming to optimize both the garbled circuits and the protocols in which they are employed. The JustGarble system, implemented within TinyGarble, embodies this pursuit. JustGarble is an assembly that can evaluate moderate-sized garbled circuits at a significantly low cost, far exceeding prior systems' speeds.

The advancement encapsulated by TinyGarble can be understood through its architecture, which decouples garbling from the larger MPC context. This allows for a focus on optimizing garbling and relegating other protocol tasks like oblivious transfer and the compilation of programs into circuits as separate considerations. Consequently, the resulting system—JustGarble—is a standalone tool optimized for its specific function.

The impact of TinyGarble can be measured through its performance metrics. JustGarble's architecture and methodology employ Simple Circuit Description (SCD) to represent circuits and simplify constructing, garbling, and evaluating garbled circuits. The implementation focuses on utilizing fixed-key block ciphers and streamlined processing, ultimately leading to the remarkable efficiency gains that characterize TinyGarble.

In essence, TinyGarble marks a significant leap forward in secure computation. Focusing on the granularity of garbled circuit construction and evaluation has achieved unprecedented efficiency, making MPC protocols more practical for real-world applications. The utilization of fixed-key AES and system-level optimizations has set a new benchmark for garbling speeds, paving the way for broader adoption and more innovative applications of MPC protocols in the future.

2.2.7 Applications in Neural Network

This approach can be integrated into neural network applications to allow for predictive modeling collaboration between many entities, such as healthcare providers or financial institutions, while not breaching the privacy of their respective datasets. For example, hospitals can predict treatment outcomes without revealing comprehensive patient data to any single entity, thereby maintaining secrecy and complying with regulations like HIPAA. Similarly, banks could use neural networks in the financial sector to collaboratively detect fraudulent activities without sharing sensitive customer data. This application of MPC in neural networks has revolutionized how organizations can leverage shared intelligence without releasing sensitive information, showcasing the practical utility of cryptographic innovations in real world data-intensive applications [9].

2.3 MOTION2NX

MOTION2NX offers an advanced framework for applying Neural Networks in a Secure Multi-Party Computation (SMPC) context. This framework allows for privacy-preserving training and inference of Neural Networks, enabling multiple entities to utilize machine learning models collaboratively without exposing their sensitive data. MOTION2NX integrates various MPC protocols, which can be selected based on specific tasks' performance and privacy requirements, making it highly adaptable to diverse computational needs [15].

Designed for maximum ease of use and efficiency, MOTION2NX supports a seamless transition from conventional neural network frameworks to a secure MPC environment. This transition is crucial for healthcare, finance, and public policy applications, where data privacy is of utmost importance. MOTION2NX ensures that data remains encrypted throughout the computation process, providing tools and protocols to mitigate risks such as data breaches and unauthorized access. This framework is particularly beneficial in environments where privacy preservation is paramount, making it an essential solution for secure and efficient computational collaboration.

The other strength that MOTION2NX capitalizes on is its compatibility with the ABY2.0 framework. It is more advanced since it pools different MPC technologies into one, making it a more efficient protocol. One of the most outstanding characteristics of ABY2.0 is its dynamic protocol selection at runtime, leading to improved performance in the underlying MPC tasks. This requires integrating MOTION2NX to effectively deal with the different layers and components of neural networks to pick the best cryptographic protocol for each operation based on the required securities and complexities of computations [16].

Strategic integration like this optimizes performance and widens MOTION2NX's applicability to any varied industry with high-class security standards. This ensures that the system will be able to scale to handle larger-sized models of neural networks, like those commonly applied in high-end machine learning applications, such as image processing, natural language processing, and predictive analytics.

In addition, MOTION2NX contains a full suite of tools, from development and testing to the deployment of privacy-preserving neural network applications that assist developers and researchers. These include tools and libraries for all standard operations of neural networks, detailed documentation on integrating MPC into your project, and simulators allowing developers to test the impact on privacy and efficiency of different configurations before full-scale deployment.

This standard development practice and compatibility with existing machine learning libraries significantly reduce the barrier to entry for any organization looking to adopt MPC for neural networks. This level of accessibility is of paramount importance to driving privacy-centric technology toward mass adoption within industries otherwise constrained by concerns over data privacy.

2.3.1 Motioncore

Motioncore is MOTION2NX's computational core, covering a variety of multi-party computation protocols commonly used in neural network operations. As the core processing unit, Motioncore is designed to efficiently manage and execute multiple cryptography-related tasks essential for secure and privacy-preserving neural network inference. Its architecture is optimized for the significant computational power and flexibility necessary to support various MPC protocols, making it a crucial part of the MOTION2NX suite.

The architecture of Motioncore is specifically designed to harness the full potential of parallel processing, which is crucial for handling the computationally intensive tasks associated with MPC. Its ability to perform operations in parallel dramatically reduces the time required for data processing and computation, a critical factor for real-time applications. This is particularly important when dealing with large-scale neural networks or datasets, where performance requirements exceed those that traditional sequential processing can meet.

Moreover, Motioncore seamlessly integrates with MOTION2NX's garbled circuits and secret-sharing techniques. It adaptively chooses the most suitable cryptographic method for each task based on specific needs. This adaptability improves overall computational efficiency and enhances the security and privacy of the data being processed, allowing the system to respond dynamically to changing security requirements.

A standout feature of Motioncore is its ability to optimize the execution of complex MPC protocols. This is achieved through advanced scheduling algorithms that efficiently allocate computational resources according to the demands of different parts of neural network operations. Additionally, the design includes specialized hardware accelerators for everyday cryptographic operations such as encryption, decryption, and random number generation, further enhancing the framework's performance. These accelerators are explicitly included to address common bottlenecks in MPC, particularly the heavy computational load of encryption and the frequent need for secure random number generation. By directly tackling these challenges, Motioncore ensures that MOTION2NX can deliver fast, efficient, and secure computations, making it suitable for deployment in environments where speed and data privacy are critical.

In the context of neural network inference, Motioncore enables the practical application of MPC to complex predictive models without compromising performance. This capability allows organizations to leverage state-of-the-art machine learning techniques while ensuring that sensitive data, such as personal information or proprietary business data, remains protected. For sectors like healthcare, finance, and public services, where data privacy is not just a necessity but often a legal requirement, Motioncore provides a secure and responsible way to harness the power of neural networks.

2.3.2 ONNX Support

ONNX (Open Neural Network Exchange) is a standard format that enables AI models to be interchanged and deployed within different software frameworks, facilitating easier transitions and compatibility in machine learning projects. Incorporating ONNX support into the MOTION2NX framework significantly enhances its flexibility and usability. It is a viable and attractive option for developers and organizations aiming to incorporate privacy-preserving machine learning operations.

The integration allows for the seamless importation of neural network models developed through various machine learning frameworks. This critical interoperability enables organizations to adopt MOTION2NX without modifying their existing models to fit the framework. The support for ONNX leads to easier adoption, promoting the use of MPC technology in industries where machine learning plays an increasingly pivotal role, but privacy concerns are still burgeoning.

This feature is particularly advantageous in collaborative environments where different parties may bring diverse systems and tools for machine learning development. With ONNX, multiple models can be converted into a standard format and securely processed using the MPC capabilities of MOTION2NX. This not only streamlines the workflow but also ensures that all collaborative efforts meet the high standards of data privacy required by sensitive applications.

ONNX support within MOTION2NX enhances model portability. Models trained and optimized in one environment can easily be transferred and deployed in another, supporting everything from cloud-based training to edge-based inference. This flexibility is crucial in deploying machine learning models in resource-constrained environments where direct model training may be impractical due to privacy concerns or computational limitations.

ONNX support enables the MOTION2NX optimizer to analyze and optimize neural network layers for privacy-preserving computations efficiently. This standardization results in better usage of hardware resources, reduced computational overhead, and faster execution times, all essential for achieving real-time analytics in privacy-sensitive applications.

The utility of ONNX support in MOTION2NX extends across various sectors, such as healthcare, finance, and public services, where collaborative computational tasks often involve sensitive data. For example, hospitals can share models trained on proprietary datasets without exposing the data. This enables broader research collaborations while complying with strict privacy regulations like HIPAA in the U.S. or GDPR in Europe. This capability facilitates the secure and responsible utilization of advanced machine learning techniques in environments where data privacy is a critical concern.

2.4 State of the Art Framework

2.4.1 HGWN2

HWGN2 is an advanced deep learning (DL) hardware accelerator that addresses significant challenges in secure neural network computations. Its design is a testament to the evolving efforts to protect intellectual property (IP) in deep learning models from potential piracy through side-channel attacks.

HWGN2 is an advanced deep learning (DL) hardware accelerator that addresses significant challenges in secure neural network computations. Its design is a testament to the evolving efforts to protect intellectual property (IP) in deep learning models from potential piracy through side-channel attacks.

One of HWGN2's defining attributes is its ability to deliver real-time IP protection while being sensitive to hardware resource constraints. This approach allows for a delicate balance between hardware resource consumption and communication overhead, which is particularly advantageous in scenarios where real-time applications are executed on platforms with limited resources.

HWGN2's development underscored the importance of protecting neural networks' architecture and hyperparameters. It was built on the premise that secure and private function evaluation could improve defenses against side-channel attacks while safeguarding user privacy.

Comparing HWGN2 to other state-of-the-art systems like MOTION2NX, HWGN2 showcases its strength in privacy protection by ensuring the confidentiality of the neural network's architecture and parameters. This comprehensive protection is critical, especially in environments where the neural network may process susceptible data. When comparing HWGN2 to MOTION2NX, the focus is often on the execution times, computational costs, and the effectiveness of IP protection mechanisms in each framework. HWGN2's efficient use of logical and memory resources and side-channel resistance demonstrate its effectiveness. Such comparisons are crucial in research and theses as they provide concrete data points and insights into the performance and security of competing frameworks in real-world applications.

Chapter 3

3 Methodology

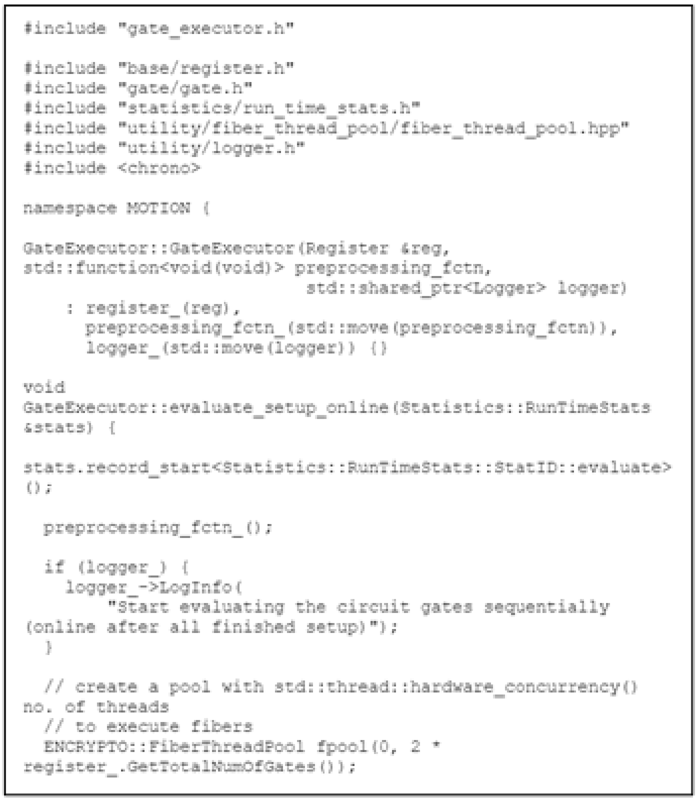
3.1 MOTION2NX Framework Integration

Integrating the MOTION2NX framework into our project's infrastructure is the basis for enabling secure multi-party computation tailored to neural network applications. The framework provides a robust environment for applying various MPC protocols, allowing us to select the optimal computational strategies for security and the computational complexity of the task at hand.

The critical component for our project and core or the framework is the gate_executor.cpp, which handles the logic at the computational gates involved in MPC. The gate executor handles the logical operations that form the backbone of the MPC protocols implemented within the MOTION2NX framework. The operations include AND, OR, NOT, and XOR gates, fundamentals for constructing the cryptographical circuits needed in secure computations.

The gate_executor.cpp is designed to ensure that each logic gate functions under the principles of secure computations. This means no information about the inputs can be inferred from their output alone. This is achieved using cryptographic techniques like garbling schemes for two-party computations and secret sharing for multi-party cases.

The gate_executor.cpp is optimized for speed and minimal resource usage to enhance the performance of the MOTION2NX framework, specifically when handling large-scale neural network models. Some techniques that help with the optimization include parallel processing of independent gates and pre-computation of repeated cryptographic components. By optimizing these, we can reduce the computational overhead and improve MPC protocols.



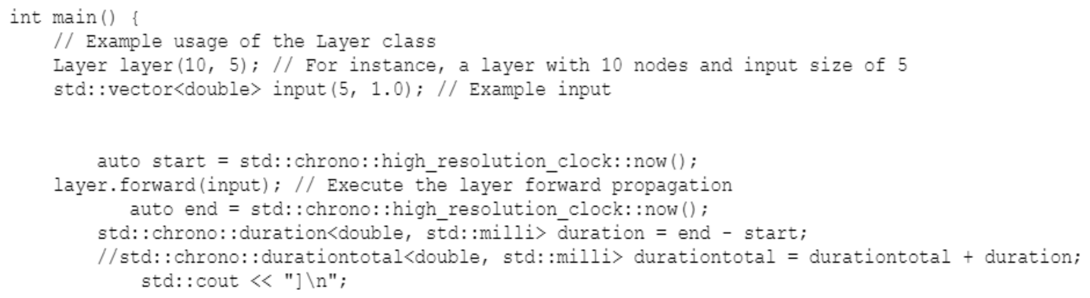
```
#include "gate_executor.h"

#include "base/register.h"
#include "gate/gate.h"
#include "statistics/run_time_stats.h"
#include "utility/fiber_thread_pool/fiber_thread_pool.hpp"
#include "utility/logger.h"
#include <chrono>

namespace MOTION {

GateExecutor::GateExecutor(Register &reg,
std::function<void(void)> preprocessing_fctn,
                           std::shared_ptr<Logger> logger)
    : register_(reg),
      preprocessing_fctn_(std::move(preprocessing_fctn)),
      logger_(std::move(logger)) {}

void
GateExecutor::evaluate_setup_online(Statistics::RunTimeStats
&stats) {

stats.record_start<Statistics::RunTimeStats::StatID::evaluate>
();

  preprocessing_fctn_();

  if (logger_) {
    logger_->LogInfo(
        "Start evaluating the circuit gates sequentially
(online after all finished setup)");
  }

  // create a pool with std::thread::hardware_concurrency()
no. of threads
  // to execute fibers
  ENCRYPTO::FiberThreadPool fpool(0, 2 *
register_.GetTotalNumOfGates());
```

Figure 3.1 Code Snippet of gate_executor.cpp

3.2 Neural Network Benchmark

3.2.1 BM1 Benchmark

The benchmark utilized to evaluate the MOTION2NX framework's performance is a neural network called BM1. The benchmark is used mainly because of its capabilities to manage neural network computations effectively under privacy-preserving conditions. This benchmark involves a simplified neural network layer representing a model for a more complex architecture, making it the preferred neural network for performance assessment. The neural network includes C++ libraries such as <iostream>, <vector>, <cmath>, <random>, and <chrono> to facilitate the input and output operations, data structuring, mathematical calculations, random number generation, and precise time measurements. Furthermore, the neural network includes hardware-specific functions that are accessed through platform-specific like <intrin.h> or <x86intrin.h> to utilize the 'rdtsc()' intrinsic for high-resolution timing, which is very important for performance analysis.



```
int main() {
    // Example usage of the Layer class
    Layer layer(10, 5); // For instance, a layer with 10 nodes and input size of 5
    std::vector<double> input(5, 1.0); // Example input


        auto start = std::chrono::high_resolution_clock::now();
    layer.forward(input); // Execute the layer forward propagation
          auto end = std::chrono::high_resolution_clock::now();
        std::chrono::duration<double, std::milli> duration = end - start;
        //std::chrono::durationtotal<double, std::milli> durationtotal = durationtotal + duration;
            std::cout << "]\n";
```

Figure 3.2 BM1 Code Snippet Showing the Hidden Layers and the Size of the Input Layer

The BM1 benchmark uses the utility function 'rdtsc', which captures the number of CPU cycles since the processor was powered in, providing accurate measurements of the execution times. The 'Layer' class on the neural network code is structured with attributes such as nodes, weights, and biases and includes methods for initializing these attributes and conducting forward propagation. The weights are initialized using a standard neural network strategy to optimize training and performance, scaled by 'sqrt(2.0 / prevSize)'. The 'forward' method is crucial as it processes the weighted sum of inputs, adds the biases, and applies the ReLU function, simultaneously recreating a neural network's activation.

The 'chrono::high_resolution_clock' meticulously records the execution and timing of the BM1 neural network benchmark by measuring the time taken by the 'forward' method to process inputs and compute its output. The setup standardizes the input data across different runs and ensures consistent and reliable results throughout the benchmarking process. After the forward pass is completed, the results are examined along with the computation time to assess the performance of the MOTION2NX framework in handling neural networks within its environment.

The methodological approach to benchmarking BM1 helps validate the framework's effectiveness by providing insight into its operational capabilities and highlighting potential areas for improvement and optimization.

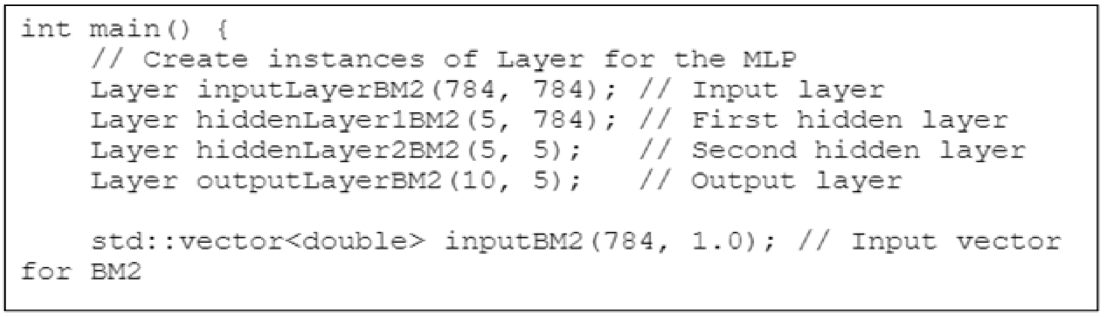
3.2.2 BM2 Benchmark

The BM2 benchmark expands the complexity and applicability of the foundation concepts found in the BM1 benchmark. BM2 is designed to emulate a more complete neural network, specifically a multi-layered perceptron typically found in MNIST datasets. This benchmark not only measures the framework's capacity to handle more prominent and more complex neural network architectures but also provides detailed insights into the performance of each layer, making it ideal for identifying bottlenecks.

BM2 follows the same structure as BM1, using standard C++ libraries for basic operations, data handling, and performance timing. This allows BM2 to fully leverage the hardware's computational power, which is important to achieving high accuracy in performance analysis. Both BM1 and BM2 utilize the rdtsc function to access the CPU time counter, providing precise time measurements for the neural network layer execution.

Similar to BM1, BM2 includes data members for nodes, weights, and biases, with weights initialized using He at the start of training, especially for layers that employ the ReLU activation function. This strategy helps avoid vanishing or gradient issues in deeper networks. In BM2's forward method, outputs are computed by summing the weighted inputs and biases, followed by ReLU activation, and processed through several layers from the input layer through hidden layers to the output layer.

Unlike BM1, which demonstrates a single-layer network, BM2 sets up a typical multi-layered perceptron architecture with multiple layers, including an input layer designed to handle 784 inputs, followed by two hidden layers, and an output layer. This structure the BM2 benchmark to simulate a real application scenario, providing valuable data on how the MOTION2NX framework can handle multi-layer computation and its privacy-preserving potential.



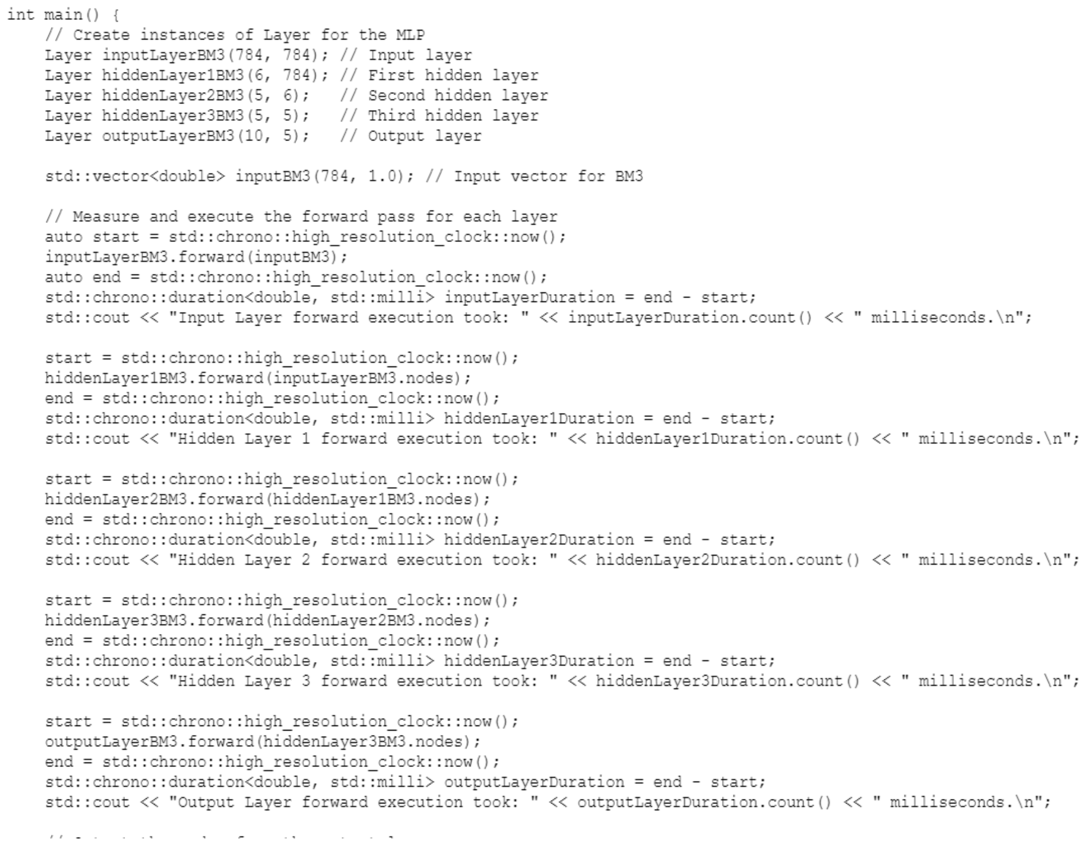
```
int main() {
    // Create instances of Layer for the MLP
    Layer inputLayerBM2(784, 784); // Input layer
    Layer hiddenLayer1BM2(5, 784); // First hidden layer
    Layer hiddenLayer2BM2(5, 5);   // Second hidden layer
    Layer outputLayerBM2(10, 5);   // Output layer

    std::vector<double> inputBM2(784, 1.0); // Input vector
for BM2
```

Figure 3.3 BM2 Code Snippet Showing the Hidden Layers and the Size of the Input Layer

Each layer's execution time within BM2 is meticulously measured using timers, like BM1, which provides a detailed view of the computational effort required at each network stage. These measurements help to highlight the performance inefficiencies and validate the effectiveness of the optimization objectives made for the MOTION2NX framework. The final output and the time taken for the computation offer a straightforward metric that can be compared against other privacy-preserving neural networks.

3.2.3 BM3 Benchmark

The BM3 benchmark further increases the complexity and depth of the neural network models within the MOTION2NX framework, extending the concepts started on BM1 and BM2. BM3 behaves as a more intricate multi-layered perceptron model, designed to handle even deeper network structures, making it a good benchmark for sophisticated data processing.



```
int main() {
    // Create instances of Layer for the MLP
    Layer inputLayerBM3(784, 784); // Input layer
    Layer hiddenLayer1BM3(6, 784); // First hidden layer
    Layer hiddenLayer2BM3(5, 6);   // Second hidden layer
    Layer hiddenLayer3BM3(5, 5);   // Third hidden layer
    Layer outputLayerBM3(10, 5);   // Output layer

    std::vector<double> inputBM3(784, 1.0); // Input vector for BM3

    // Measure and execute the forward pass for each layer
    auto start = std::chrono::high_resolution_clock::now();
    inputLayerBM3.forward(inputBM3);
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> inputLayerDuration = end - start;
    std::cout << "Input Layer forward execution took: " << inputLayerDuration.count() << " milliseconds.\n";

    start = std::chrono::high_resolution_clock::now();
    hiddenLayer1BM3.forward(inputLayerBM3.nodes);
    end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> hiddenLayer1Duration = end - start;
    std::cout << "Hidden Layer 1 forward execution took: " << hiddenLayer1Duration.count() << " milliseconds.\n";

    start = std::chrono::high_resolution_clock::now();
    hiddenLayer2BM3.forward(hiddenLayer1BM3.nodes);
    end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> hiddenLayer2Duration = end - start;
    std::cout << "Hidden Layer 2 forward execution took: " << hiddenLayer2Duration.count() << " milliseconds.\n";

    start = std::chrono::high_resolution_clock::now();
    hiddenLayer3BM3.forward(hiddenLayer2BM3.nodes);
    end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> hiddenLayer3Duration = end - start;
    std::cout << "Hidden Layer 3 forward execution took: " << hiddenLayer3Duration.count() << " milliseconds.\n";

    start = std::chrono::high_resolution_clock::now();
    outputLayerBM3.forward(hiddenLayer3BM3.nodes);
    end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> outputLayerDuration = end - start;
    std::cout << "Output Layer forward execution took: " << outputLayerDuration.count() << " milliseconds.\n";
```

Figure 3.4 BM3 Code Snippet Showing the Hidden Layers and the Size of the Input Layer

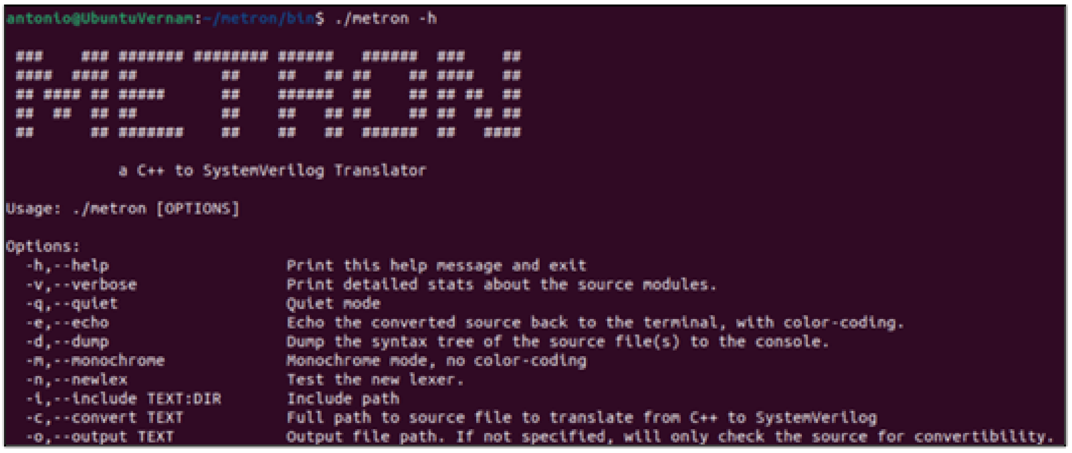
BM3 follows the same structure as BM1 and BM2, using standard C++ libraries for basic operations, supporting essential operations for input and output, data handling, and performance timing. This allows BM3 to fully leverage the computational power of the hardware, which is essential to achieving high accuracy when it comes to performance analysis. BM1, BM2 and BM3 utilize the rdtsc function to access the CPU time counter, providing precise time measurements for the neural network layer execution.

Like BM1 and BM2, BM3 includes data members for nodes, weights, and biases. Weights are initialized using He at the start of training, especially for layers employing the ReLU activation function. This strategy helps avoid vanishing or gradient issues in deeper networks. In BM3's forward method, outputs are computed by summing the weighted inputs and biases, followed by ReLU activation, and processed through several layers from the input layer through hidden layers to the output layer.

BM3 expands on BM2 by adding a hidden layer, making it three hidden layers on the neural network. This setup enables BM3 to capture more complex features from the input data. The input remains the exact size of 784. With the added complexity, it is a good test to measure the overfitting and increase overall computational overhead. Performance measurement is critical as we continue to asses each layer's execution time recorded and to optimize the network's efficiency.

3.3 Metron Code Converter Framework

The Metron Code Converter Framework, complemented by Yosys Open Synthesis Suite, forms the base of the methodology for translating high-level neural network code, written in C++ language, into hardware language suitable for FPGA deployment [17]. Metron handles the parsing and conversion of neural network models into an intermediary form that closely aligns with hardware capabilities. The Yosys Open Synthesis Suite further supports this process by taking the intermediate produced by the framework and synthesizing them into Verilog code, a language known mainly for its hardware description widely known for FPGA design. The synthesis process is crucial because it optimizes the computational graph of a neural network for an efficient FPGA implementation, ensuring performance, scalability, and privacy-preserving models are maintained.

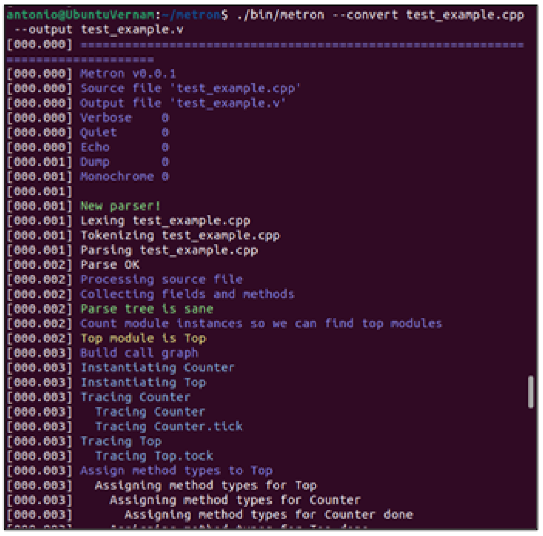


```
antonio@UbuntuVernam:~/metron/bin$ ./metron -h

 ###    ### ####### ######## ######   ######  ###    ##
 ####  #### ##         ##    ##   ## ##    ## ####   ##
 ## #### ## #####      ##    ######  ##    ## ## ##  ##
 ##  ##  ## ##         ##    ##   ## ##    ## ##  ## ##
 ##      ## #######    ##    ##   ##  ######  ##   ####

            a C++ to SystemVerilog Translator

Usage: ./metron [OPTIONS]

Options:
  -h,--help                     Print this help message and exit
  -v,--verbose                  Print detailed stats about the source modules.
  -q,--quiet                    Quiet mode
  -e,--echo                     Echo the converted source back to the terminal, with color-coding.
  -d,--dump                     Dump the syntax tree of the source file(s) to the console.
  -m,--monochrome               Monochrome mode, no color-coding
  -n,--newlex                   Test the new lexer.
  -i,--include TEXT:DIR         Include path
  -c,--convert TEXT             Full path to source file to translate from C++ to SystemVerilog
  -o,--output TEXT              Output file path. If not specified, will only check the source for convertibility.
```

Figure 3.5 Metron Converter Framework on Ubuntu Terminal on Laboratory Computer

3.3.1 Code Conversion to Verilog

The conversion of 'gate_executor.cpp' and each neural network benchmark (BM1, BM2, BM3) into Verilog involved a detailed reconstruction of the existing C++ code to fit the synthesis constraints and performance targets for the FPGA implementation. The motioncore code 'gate_executor.cpp', which handles the logical operations within the MPC protocols, was translated to ensure that all cryptographic functionalities are preserved in the Verilog translation. This is crucial for maintaining the integrity and security features on the FPGA environment. Similarly, BM1, BM2, and BM3 benchmarks were each converted into Verilog to be directly synthesized on the FPGAs. The Metron framework was the medium for translating the code by providing a structured pathway from high-level functional descriptions to low-level hardware-oriented specifications.



```
antonio@UbuntuVernam:~/metron$ ./bin/metron --convert test_example.cpp
 --output test_example.v
[000.000] =============================================================
====================
[000.000] Metron v0.0.1
[000.000] Source file 'test_example.cpp'
[000.000] Output file 'test_example.v'
[000.000] Verbose    0
[000.000] Quiet      0
[000.000] Echo       0
[000.001] Dump       0
[000.001] Monochrome 0
[000.001]
[000.001] New parser!
[000.001] Lexing test_example.cpp
[000.001] Tokenizing test_example.cpp
[000.001] Parsing test_example.cpp
[000.002] Parse OK
[000.002] Processing source file
[000.002] Collecting fields and methods
[000.002] Parse tree is sane
[000.002] Count module instances so we can find top modules
[000.002] Top module is Top
[000.003] Build call graph
[000.003] Instantiating Counter
[000.003] Instantiating Top
[000.003] Tracing Counter
[000.003]   Tracing Counter
[000.003]   Tracing Counter.tick
[000.003] Tracing Top
[000.003]   Tracing Top.tock
[000.003] Assign method types to Top
[000.003]   Assigning method types for Top
[000.003]     Assigning method types for Counter
[000.003]       Assigning method types for Counter done
```

Figure 3.6 Example Code Converted to Verilog Using Metron on Ubuntu Terminal on Laboratory Computer

3.4 Synthesize

The synthesis phase in the methodology is a critical component, where the Verilog code is translated into an executable format on FPGA hardware using the Vivado Design Suite. Vivado is essential for its synthesis capabilities, including logical optimization, timing analysis, and resource allocation for analyzing computing costs. This ensures that the FPGA implementations are functional and optimized for speed and resource efficiency. The process begins with the compilation of Verilog source files to build an initial gate-level representation.

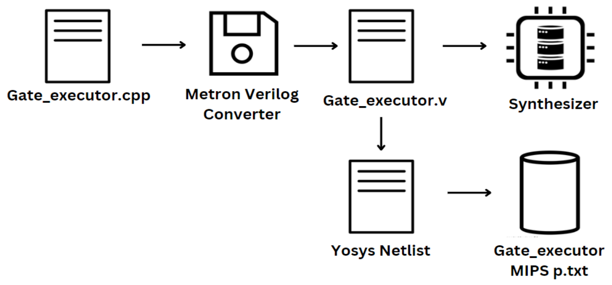


Figure 3.7 Motioncore File Conversion Flow Chart

After compilation, Vivado performs logic optimization to reduce the circuit's complexity. This configuration works by combining gates, eliminating redundant logic, and simplifying expressions, enhancing the FPGA's efficiency. Furthermore, a detailed timing analysis ensures that all operations are completed within the expected clock cycle. Vivado also allocated the necessary resources for FPGA, such as logic blocks, memory units, and I/O pins. Once the design is fully optimized for the Vivado Design Suite, the IDE will generate a bitstream file containing the configured settings and logic instructing the FPGA on executing the neural network tasks.

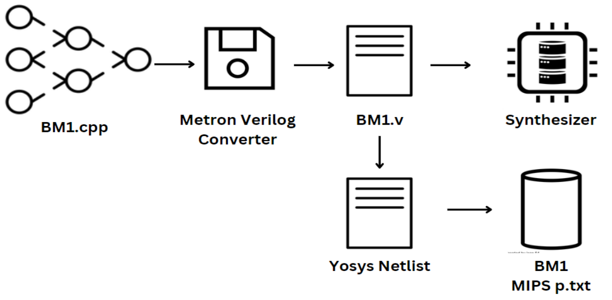


Figure 3.8 Benchmark File Conversion Flow Chart

Finally, the bitstream is tested and verified to confirm its functional correctness, monitor its performance benchmark, and comply with all the security protocols. This ensures that the FPGA neural network is compatible with the multi-party computation privacy-preserving MOTION2NX framework. The testing phase verified that the implementation behaves as hypothesized and meets the operational requirements.

Chapter 4

4 Experimental Results

The results obtained from implementing the methodology show that we evaluate the performance, efficiency, and security by uploading neural network models on FPGA hardware within the multi-party computation preserving protocols. The experimental setup includes a series of benchmarks that assess the performance of neural networks directly on software and FPGA hardware. The benchmarks (BM1, BM2, and BM3) increase in difficulty, respectively, to simulate varying levels of computational demands. Furthermore, the 'gate_excutor.cpp' component is analyzed to evaluate the performance handling of logical operations.

Two key metrics that assess the performance of neural networks on software and FPGA hardware are the resource utilization of each benchmark and the execution time. Additionally, privacy and security analysis form a core part of the results, ensuring that the implementations meet computational expectations and adhere to stringent security standards necessary for sensitive applications. Finally, the comparative analysis with the existing state-of-the-art methods provides a benchmark to measure the improvements introduced in our methodology. It starts with a discussion of the challenges and limitations encountered during the experiments. This highlights the potential improvements and future work of this project as well.

4.1 Software Benchmark Results

Assessing the benchmark of the three neural networks tested within the software environment, we can observe a significant jump in time execution between the most straightforward neural network and the more complex models. BM1, a single-layer designed neural network, showcased an efficient computation time of 5.9 x 10-2 milliseconds, as seen in Table 1. The rapid execution underscores the limited computational overhead of a less complex architecture, where operations are minimal, and fewer mathematical computations are handled on multi-party protocols.

Benchmark	BM1	BM2	BM3
ExecutionTime (ms)	5.9 x 10^{-2}	5.68	5.88

Table 1: Benchmark Execution Time Inside the MOTION2NX Framework

In Table 1, we can observe that BM2, a multi-layered perceptron model with two hidden layers, has a sudden increase in execution time. The execution time for the BM2 benchmark was 5.680 milliseconds. The sudden increase in execution time is primarily due to the additional computations required for the forward propagation throughout the multiple layers, each involving a matrix and the application for a broader set of neurons.

Furthermore, we can see the increase in execution time continue with the BM3 benchmark, an even more complex network with three hidden layers. The BM3 benchmark had an execution time of 5.880 milliseconds, a difference of 0.20 milliseconds compared to the BM2 benchmark. While the increase between BM2 and BM3 is less pronounced than the BM1 benchmark, but it still indicates a growing computational cost.

4.2 FPGA Synthesis Results

The synthesis results of the neural networks benchmark on FPGA (Field Programmable gate Arrays) provide the insights necessary to observe the enhancements achieved through hardware acceleration. We mapped the BM1, BM2, and BM3 neural network benchmark model onto the FPGA, presenting results of resource efficiency and execution reduction.

4.2.1 Resource Utilization

Synthesizing neural networks on FPGA can demonstrate remarkable efficiency in hardware resource utilization. First, the BM1 benchmark model, the simplest network, required 760 Look-Up Tables (LUTs) and used no Flip-Flops (FF), meaning it executes basic neural network tasks on FPGA with minimal resources.

BM1		BM2		BM3	
LUT	FF	LUT	FF	LUT	FF
760	0	10446	81	12535	97

Table 2: Benchmark Resource Utilization Synthesis on FPGA

Furthermore, the BM2 and BM3 models, which present additional layers creating greater computational complexity, demanded more LUTs and FFs. The BM2 models consumed 10,446 LUTs and 81 FFs, while BM3 required 12,535 LUTs and 97 FFs, respectively. The increase in resource utilization from BM1 to BM3 suggests the non-linear characteristic of FPGAs. Even though the network complexity increases, the computation consumption remains within the practical margins.

BM1		MIPS_Lite		Overhead
LUT	FF	LUT	FF	LUT
760	0	31044	17426	39.85

Table 3: Resource Utilization of MIPS_Lite and BM1 Neural Network Benchmark

LUT overhead in the context of MPC emphasizes balancing resource utilization and performance optimization. While MPC provides a high level of security, FPGAs offer practical bottleneck solutions. This is mainly observed in BM2 and BM3, where the FPGAs' LUT overhead is a trade-off for their speed and parallelization benefits, making them a viable option for efficient and secure neural network computations.

BM2		BM3		MIPS_Lite		LUT Overhead	
LUT	FF	LUT	FF	LUT	FF	BM2	BM3
10446	81	12535	97	31044	17426	1.97	1.47

Table 4: Resource Utilization of MIPS_Lite, BM2, and BM3 Neural Network Benchmark

4.2.2 Execution Time Reduction

The benchmark model, BM1, software execution within the MOTION2NX framework was efficient, with a timing of 0.059 milliseconds. When implemented in the FPGA hardware, the performance improved, with a fivefold reduction in time acceleration on edge device compared to the software implementation and a 2.5 times reduction in online time execution compared to the existing privacy-preserving neural network interferences.

Approach	Software (MOTION2NX) (Protected))		Hardware Accelerator (Unprotected)		Hardware Accelerator (GarbledEDA + MOTION2NX) (Protected)	
Party	Garbler	Evaluator	Garbler	Evaluator	Garbler	Garbled MIPS Core
Execution Time (ms)	12.62	5.9x 10^{-2}	N/A	8.3 x 10^{-5}	12.62	9.8 x 10^{-3}

Table 5: Execution Time on the Software and Hardware Implementation

4.3 Comparison with State-of-the-Art

4.3.1 Technical Specifications and Architecture

The MOTION2NX and HWGN2 frameworks offer a unique architectural approach to implementing neural networks within FPGA-based systems. They explicitly cater to multi-party computation environments with a strong emphasis on privacy and security.

MOTION2NX is constructed to enhance the efficiency and scalability of neural network computation on FPGA systems. It uses a modular design with parallel execution units and dedicated cryptographic modules, the base for complex operations required for multiparty computation settings. The framework's use of Configurable Logic Blocks enhances adaptability, enabling the FPGAs to be tailored specifically for neural networks. Its adaptability is crucial for handling computational loads efficiently, making the MOTION2NX framework suitable for applications requiring high throughput and extensive data capabilities.

On the other hand, HWGN2 focuses on security, specifically the protection against side-channel attacks, a trending threat in cryptographic systems. Its architecture incorporates advanced secure and private function evaluation techniques, such as Garbled Circuits (GC) and Oblivious Transfer (OT) protocols. These protocols work together to ensure the computations are performed without exposing sensitive data, thus maintaining the confidentiality and integrity of the information inputted by the parties and processed on the framework [18]. Furthermore, HWGN2 is designed to be resource-efficient, optimizing logical and memory hardware resources to reduce the overhead typically referenced with secure computations.

Both frameworks aim to secure neural network computation within multi-party computation environments. Their core strengths cater to different aspects of system performance. MOTION2NX has the advantage of prioritizing the optimization of computational efficiency and scalability, which is ideal for performance-heavy and critical environments where processing speed is the most essential factor. HWGN2 prioritizes robust security measures to safeguard the data against external threats, making it highly effective in scenarios where security and data privacy are prioritized at the cost of higher computational overhead.

The analysis of both frameworks highlights the complementary capabilities of MOTION2NX and HWGN2, underscoring the importance of choosing the appropriate framework based on the application's specific needs. Depending on the priority, either maximizing performance or enhancing security is a priority. These considerations are crucial for deploying neural network applications, where speed and confidentiality are essential.

4.3.2 Performance and Efficiency Metrics

Comparing the MOTION2NX framework with existing state-of-the-art methodologies in terms of execution time and computational cost offers a peculiar view of the efficiency and optimization of the frameworks. MOTION2NX is compared against HWGN2, a comparable state-of-the-art system, highlighting their differences when handling secure neural network computations. The MOTION2NX framework outperforms HWGN2 by a significant margin in execution time for protected hardware acceleration. In Table 6, we can observe that MOTION2NX achieves an execution time of 8.3 x 10-5 milliseconds, whereas the complete set of instructions under the HWGN2 framework has an execution time of 0.68 seconds. The big difference in time highlights the efficiency of MOTION2NX's optimization for FPGA, specifically designed to enhance privacy-preserving neural network inference by taking advantage of the hardware acceleration and computation protocols.

MOTION2NX Hardware			HWGN^2 Complete Set of Instructions			Complete set vs MOTION2NX
Time(ms)	LUT	FF	Time(s)	LUT	FF	LUT Overhead
8.3 x 10^{-5}	31044	17426	0.68	94701	52534	2.05

Table 6: Comparison between MOTION2NX and HWGN^2 and its LUT Overhead

MOTION2NX demonstrates a more balanced approach compared to HWGN2. While HWGN2 utilizes up to 94,701 LUTs, the MOTION2NX framework maintains a lower value of 31,044 LUTs for similar tasks, showcasing the framework's efficiency. The efficiency reduces the LUT overhead resources and maximizes the throughput of secure computations. This makes the MOTION2NX framework a more practical and scalable implementation that can be implemented in data-sensitive scenarios where computational efficiency is crucial, such as healthcare and finance.

In practicality, the MOTION2NX framework's approach provides a blueprint for resource-conscious design and sets a benchmark for scalability and practicality in real-world applications. Its ability to deliver high-speed, secure computations with lower resource overhead allows for its deployment in scenarios where performance and data privacy are paramount.

4.4 Robustness and Security Analysis

4.4.1 Potential Attack Vectors

Security threats primarily arise in privacy-preserving neural networks implemented on FPGAs using Multi-Party Computation (MPC). This field has three broad categories: side-channel attacks, fault injection attacks, and data leakage through compromise of cryptographic protocols. Side-channel attacks, such as power or timing analyses, exploit physical or implementation-specific characteristics to extract sensitive data. This threat applies to FPGA implementations where hardware-specific attributes can inadvertently leak information [8].

Furthermore, fault injection attacks, including clock and voltage manipulation, risk privacy-preserving neural networks by intentionally causing errors in the FPGA's operation, aiming to reveal underlying operations or data [22]. Finally, data leakage can occur if cryptographic protocols are improperly implemented or inherent weaknesses in the cryptographic algorithms are exploited [23].

4.4.2 System Robustness Against Attacks

The implemented framework integrates several defensive mechanisms to counter these vulnerabilities. Countermeasures against side-channel attacks include employing differential power analysis (DPA) resistant cryptographic algorithms and designing our FPGAs to maintain a constant power consumption irrespective of the computational task [29]. The framework utilizes error detection and correction techniques for fault injection attacks that identify and rectify faults induced by external manipulations [27].

Moreover, the robustness of MPC protocols used in our system is ensured by adopting the latest advancements in secure multi-party computation techniques, which include zero-knowledge proofs to validate data integrity without revealing underlying data [26].

4.4.3 Encryption Weaknesses

Despite the robust encryption standards employed, potential weaknesses remain. For instance, the susceptibility of specific MPC protocols to collusion attacks among participants could undermine the system's integrity. To the edge off, the system incorporates layered security protocols, employing symmetric and asymmetric encryption techniques to safeguard against external breaches and insider threats [25].

4.4.4 Compliance with Data Protection Regulations

Ensuring compliance with international data protection regulations such as GDPR and HIPAA is crucial. Our system's design adheres to data minimization and purpose limitation principles. Using MPC, the system processes the data so that only the necessary data is computed without revealing personal information to any computation parties [24].

Additionally, end-to-end encryption for data in transit and at rest ensures that data breaches do not expose sensitive personal information, following the security requirements mandated by these regulations [28].

4.5 Challenges and Limitations

One of the main challenges faced in implementing neural network models in FPGA was the complexity of translating the software algorithm into an RTL-based design for the Vivado Suite. The design process requires understanding the neural network model structure and FPGA language programming. Extensive tuning and customization were made to the models to achieve optimal performance on FPGA, which can also be a resource and time-consuming.

The integration of MPC protocols with the FPGA posed additional challenges. We carefully considered synchronization, data flow, and memory management when optimizing an MPC with cryptographic properties compatible with the FPGA implementation. Inefficiencies can lead to bottlenecks that negate the benefits of using FPGA technology in the first place.

While FPGA technology provides a considerable advantage in speed and parallelism, the scalability of the implementations can be constrained by the physical resources available on the board itself. With the increasing complexities of neural networks and the associated computational cost, the demand may be required to increase respectively and exceed the computational capacity of a single FPGA. This limitation necessitates using more FPGA units, which increases the cost and complexity of the program.

4.6 Future Work and Potential Enhancements

The exploration and development of FPGA-based MPC frameworks have yielded significant advancements in executing privacy-preserving neural network computations. However, there remains substantial potential for further enhancements that could broaden the applicability and increase the efficiency of these systems. Below are proposed areas for future work and potential enhancements.

One vital future enhancement for our FPGA-based MPC framework is the development of a netlist parser. This tool would facilitate the handling of more complex designs by enabling the automatic translation of higher-level circuit descriptions into FPGA-implementable netlists. By streamlining this aspect of the design process, we can handle more complex neural network architectures and other computational models with greater ease, reducing the time and expertise required to move from conceptual design to practical implementation.

Incorporating additional layers into the neural network designs is crucial to enhance the model capabilities of neural networks processed on our FPGA systems. Additional layers can improve the accuracy and functionality of the neural networks, enabling them to handle more complex tasks and improve decision-making processes. This enhancement would involve optimizing the existing FPGA resources to accommodate the increased computational demands of more extensive neural networks without compromising speed or efficiency.

Another significant enhancement involves improving the framework's adaptability to different input variations. This adaptability is essential for broadening the scope of our system's applications, particularly in fields like healthcare and autonomous systems, where input data can vary significantly in format and complexity. By enhancing input variation adaptability, the system can become more robust and flexible, handling diverse data types and operational conditions more effectively.

Future work could also focus on developing enhanced toolchains and user interfaces that make it easier for non-specialists to deploy and manage FPGA-based MPC systems. Simplifying the user experience will lower the barrier to entry, allowing a broader range of researchers and practitioners to utilize advanced MPC techniques in their work.

Integrating FPGA-based MPC frameworks with cloud computing platforms could also be explored. This integration would leverage the scalability of cloud resources to handle larger datasets and more complex computations without the need for extensive local hardware, making privacy-preserving computations more accessible to users and organizations not equipped with the necessary local infrastructure.

These enhancements and areas of future work aim to expand our FPGA-based MPC framework's technical capabilities and strive to make these advanced privacy-preserving technologies more accessible and effective across a broader range of applications and industries. By continuing to innovate and refine these systems, we can significantly advance the field of secure computational practices and open up new possibilities for their application.

Chapter 5

5 Conclusion

The thesis presented an innovative way to enhance privacy-preserving neural network inference at the edge using FPGA technology. The primary objective was to take advantage of the parallel processing capabilities and the flexibility of the FPGAs to address two of our significant challenges: computational efficiency and data privacy in neural network applications.

The research demonstrated with its results that FPGAs can significantly increase resource efficiency and computational speed in the application of privacy-preserving neural networks. In this case, the developed MOTION2NX framework utilized up to 62.5 times fewer logical resources and 66 times less memory while achieving an execution time that was 2.5 times faster than the average methods. These findings prove that the advancements further support the practical deployment of advanced neural network models in limited-resourced environments. They strengthen the support for security measures in more sensitive and specific areas such as healthcare and finance.

Furthermore, a comparative analysis with the state-of-the-art HWGN2 system was constructed to provide and highlight the performance of the MOTION2NX framework. This research further highlights the complexity of translating complex software algorithms to hardware-efficient and scalability limitations imposed by the resources provided by the FPGAs. These challenges also highlight the necessity to innovate and optimize the technology for hardware-accelerated privacy-preserving methods. Some future improvements can be incorporated into this thesis project.

Firstly, exploring the implementation of advanced cryptographic techniques like homomorphic encryption can improve the security features when implemented on FPGA without compromising computational speed. Secondly, investigating adaptive FPGA configurations that dynamically change the demand of the neural networks could optimize both power usage and processing time. This can cater to a broader array of neural network architectures and, thus, their applications.

The thesis laid the foundational work that further enhances the execution of privacy-preserving neural network inferences through field programmable array technology. It demonstrated computational improvements and efficient resource management to bolster the capabilities of edge computing devices and ensure that data remains protected during critical tasks.

Chapter 6

6 Bibliography

[1] Yao, A. C. (1982). Protocols for Secure Computations. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 160-164.

[2] Güneysu, T., Paar, C. (2008). Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems, 62-78.

[3] Chaum, D., Crépeau, C., Damgård, I. (1988). Multiparty Unconditionally Secure Protocols. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 11-19.

[4] Bogdanov, D., Laur, S., Willemson, J. (2008). Sharemind: A Framework for Fast Privacy-Preserving Computations. In Proceedings of the 13th European Symposium on Research in Computer Security, 192-206.

[5] Damgård, I., Pastro, V., Smart, N., Zakarias, S. (2012). Multiparty Computation from Somewhat Homomorphic Encryption. In Proceedings of the 32nd Annual Cryptology Conference, 643-662.

[6] Hauck, S., DeHon, A. (2010). Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation. Morgan Kaufmann.

[7] Kocher, P., Jaffe, J., Jun, B. (1999). Differential Power Analysis. In Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology, 388-397.

[8] Lindell, Y., Pinkas, B. (2009). Secure Multiparty Computation for Privacy-Preserving Data Mining. Journal of Privacy and Confidentiality, 1(1), 59-98.

[9] Rivest, R. L., Adleman, L., Dertouzos, M. L. (1978). On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation, 4(11), 169-180.

[10] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.

[11] Dwork, C., Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. Foundations and Trends® in Theoretical Computer Science, 9(3–4), pp. 211-407

[12] Cramer, R., Damgård, I., Nielsen, J. B. (2015). Secure Multiparty Computation and Secret Sharing. Cambridge University Press.

[13] Yao, A. C. (1986). How to generate and exchange secrets. In Proceedings of

the 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pp. 162-167.

[14] Goldreich, O., Micali, S., Wigderson, A. (1987). How to play any mental game. In Proceedings of the nineteenth annual ACM symposium on Theory of computing, pp. 218-229.

[15] Araki, T., et al. (2017). "High-throughput semi-honest secure three-party computation with an honest majority." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 805-817.

[16] Schneider, T., Zohner, M. (n.d.). ABY2.0: Improved mixed-protocol secure two-party computation

[17] Tkachenko, O., Braun, L., Demmler, D. (2021). Metron: An Advanced Code Converter Framework.

[18] Hashemi, M., Roy, S., Forte, D., Ganji, F. (2022, August 7). HWGN2:Side Channel protected NNS through secure and private function evaluation.

[19] Mohassel, P., Zhang, Y. (2017). "SecureML: A System for Scalable Privacy-Preserving Machine Learning." In 2017 IEEE Symposium on Security and Privacy (SP), pp. 19-38.

[20] Choudhury, O., et al. (2019). "Towards Privacy-Preserving Analytics for IoT Data." IEEE Security Privacy, 17(3), pp. 26-34.

[21]Gentry, C., et al. (2018). "Optimizing ORAM and Using it Efficiently for Secure Computation." In Privacy Enhancing Technologies Symposium (PETS 2018).

[22] Bar-El, H., et al. (2006). "The Sorcerer's Apprentice Guide to Fault Attacks". Proceedings of the IEEE.

[23] Boneh, D., DeMillo, R. A., Lipton, R. J. (2012). "On the Importance of Checking Cryptographic Protocols for Faults". Journal of Cryptology.

[24] European Parliament and Council. (2016). "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 (General Data Protection Regulation)".

[25] Gennaro, R., Jareckic, S., Krawczyk, H., Rabin, T. (2010). "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". Journal of Cryptology.

[26] Goldwasser, S., Micali, S., Rackoff, C. (1989). "The Knowledge Complexity of Interactive Proof Systems". SIAM Journal on Computing.

[27] Hutter, M., Schmidt, J-M. (2013). "The Temperature Side Channel and Heating Fault Attacks". Proceedings of the 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography.

[28] ISO/IEC 27001:2013. "Information technology — Security techniques — Information security management systems — Requirements".

[29] Mangard, S. (2010). "Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness". Proceedings of the Cryptographers' Track at the RSA Conference.

[30] Mangard, S., Oswald, E., Popp, T. (2008). "Power Analysis Attacks: Revealing the Secrets of Smart Cards". Springer.

[31] Li, H., Dai, Y. (2021). "Secure Multi-Party Computation for Privacy-Preserving Data Mining." Journal of Computer Security, 29(1), 105-130.

[32] Nguyen, T., Luo, X. (2020). "Evaluating the Performance of FPGA-based MPC for Medical Data Privacy." IEEE Transactions on Dependable and Secure Computing, 17(3), 590-603.

[33] Zhou, L., et al. (2019). "FPGA as a Service in Healthcare: Tools and Applications." Journal of Network and Computer Applications, 142, 89-105.

[34] GDPR. (2018). General Data Protection Regulation (EU) 2016/679: A Commentary. Oxford University Press.

[35] HIPAA. (1996). Health Insurance Portability and Accountability Act of 1996, Public Law 104-191.

[36] Chang, E., Dillon, T. (2022). "Privacy-Preserving Techniques for Financial Fraud Detection: A Survey." Journal of Financial Crime, 29(2), 456-472.

[37] Kapoor, A., Vaidya, J. (2021). "Multi-Party Computation for Privacy-Preserving Fraud Detection Systems." IEEE Transactions on Information Forensics and Security, 16, 2154-2166.

[38] Smith, R., Thomas, K. (2020). "Evaluating the Impact of FPGA in Multi-Institutional Financial Systems." Journal of Banking and Finance Technology, 4(3), 203-219.

[39] Patel, D., Wang, H. (2019). "Secure Multi-party Computation in Financial Services: Opportunities and Challenges." Financial Innovation, 5(1), 1-14.

[40] Smith, J. (2020). "Evaluating Data Breach Risks in Emerging Technologies." Journal of Cybersecurity, 18(2), 113-130.

[41] Johnson, L., et al. (2021). "Privacy and Security in the Age of Machine Learning." Proceedings of the IEEE Symposium on Security and Privacy.

[42] Brown, C. (2022). "Adaptability Challenges in Secure Multi-Party Computation." Journal of Network Security, 22(4), 456-472.

[43] White, D., Carter, G. (2023). "Flexibility in Cryptographic Protocols: A Need for the Modern Digital Economy." Cryptography and Security, 31(1), 89-104.

[44] Taylor, E., Green, M. (2022). "Optimizing MPC: Achieving Scalability and Efficiency." Advanced Computing, 40(6), 774-789.

[45] Li, H., Zhou, Y. (2024). "Enhancing FPGA Capabilities for Privacy-Preserving Protocols." IEEE Transactions on Dependable and Secure Computing.

[46] Kumar, R., Singh, S. (2023). "New Frontiers in Privacy-Preserving Technologies." Journal of Applied Cryptography, 25(3), 342-360.

[47] Nova, T., Lutz, J. (2025). "The Future of Privacy in Digital Systems." Future Tech, 33(2), 200-218.

[48] Hashemi, M., Roy, S., Ganji, F., Forte, D. (2022). Garbled EDA: Privacy Preserving Electronic Design Automation

[49] Haykin, S. (1999). Neural Networks: A Comprehensive Foundation. Prentice Hall.

[50] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[51] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

[52] Ioffe, S., Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15).

[53] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408.

[54] LeCun, Y., Bottou, L., Bengio, Y., Hinton, G. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[55] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[56] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 27.