School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Information Systems Development & Security

Implementing Privacy-Preserving Neural Network Inference via Distributed Stream Processing

Despina Kanakidou
p3312207

Supervisor: Prof. Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business

Committee: Prof. Dimitrios Gunopulos
Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

Prof. Dimitrios Gritzalis
Department of Informatics
Athens University of Economics and Business

March 2025

https://doi.org/10.26219/heal.aueb.5198

Despina Kanakidou

Implementing Privacy-Preserving Neural Network Inference via Distributed Stream Processing

March 2025

Supervisor: Prof. Vana Kalogeraki

Athens University of Economics and Business
School of Information Sciences and Technology
Department of Informatics
Computer Systems and Communications Laboratory
Athens, Greece

Abstract

This thesis presents the implementation and evaluation of enhancements to the PP-Stream system, a hybrid framework for high-performance, privacy-preserving neural network inference. Rather than proposing a new architecture, the primary contribution of this work lies in extending PP-Stream's privacy mechanisms—specifically by improving the obfuscation techniques used during non-linear operations. The system leverages a hybrid approach that applies homomorphic encryption to secure linear computations and obfuscation strategies to protect non-linear layers, thereby addressing the common trade-off between security and performance in privacy-preserving inference.

In this study, we deploy PP-Stream within a distributed stream processing environment and focus on its ability to handle real-time data with minimal latency. To strengthen its privacy guarantees, we integrate additional obfuscation mechanisms derived from recent literature, such as dummy value generation and spatial perturbation. We conduct extensive experimentation using the MNIST dataset to evaluate the impact of these enhancements on both system performance and data confidentiality. The experimental results suggest that the proposed enhancements to PP-Stream can improve privacy during inference, with a moderate impact on accuracy and efficiency, underlining the trade-offs involved in privacy-preserving neural network deployment. This work highlights the importance of incorporating advanced obfuscation methods into existing privacy-preserving frameworks and contributes valuable insights toward the secure deployment of neural network inference in untrusted and distributed environments.

https://doi.org/10.26219/heal.aueb.5198

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει την υλοποίηση και αξιολόγηση επεκτάσεων στο σύστημα PP-Stream, ένα υβριδικό πλαίσιο για αποδοτική και ιδιωτική εκτέλεση προβλέψεων με χρήση νευρωνικών δικτύων. Αντί να προτείνει μία νέα αρχιτεκτονική, η βασική συμβολή αυτής της εργασίας έγκειται στην ενίσχυση των μηχανισμών ιδιωτικότητας του PP-Stream, με έμφαση στη βελτίωση των τεχνικών απόκρυψης που εφαρμόζονται στις μη γραμμικές λειτουργίες του νευρωνικού δικτύου. Το σύστημα υιοθετεί μια υβριδική προσέγγιση, συνδυάζοντας την ομομορφική κρυπτογράφηση για τις γραμμικές λειτουργίες και μηχανισμούς απόκρυψης για τις μη γραμμικές, με στόχο την εξισορρόπηση μεταξύ ιδιωτικότητας και απόδοσης.

Στο πλαίσιο αυτής της μελέτης, το PP-Stream αναπτύσσεται και αξιολογείται σε κατανεμημένο περιβάλλον επεξεργασίας ροών δεδομένων, με έμφαση στη δυνατότητα επεξεργασίας δεδομένων σε πραγματικό χρόνο και με χαμηλή καθυστέρηση. Για την περαιτέρω ενίσχυση των εγγυήσεων ιδιωτικότητας, ενσωματώνονται και αξιολογούνται πρόσθετοι μηχανισμοί απόκρυψης που προκύπτουν από σύγχρονη βιβλιογραφία, όπως η δημιουργία ψευδών τιμών και η χωρική παραμόρφωση. Μέσω πειραμάτων με το σύνολο δεδομένων MNIST, αξιολογείται η επίδραση των ενισχυμένων μηχανισμών τόσο στην απόδοση του συστήματος όσο και στη διατήρηση της εμπιστευτικότητας των δεδομένων. Τα αποτελέσματα δείχνουν ότι οι προτεινόμενες επεκτάσεις του PP-Stream μπορούν να ενισχύσουν την ιδιωτικότητα κατά τη φάση της εκτίμησης, διατηρώντας μέτρια ακρίβεια και αποδοτικότητα συγκριτικά με την υπάρχουσα υλοποίηση, υπογραμμίζοντας με αυτόν τον τρόπο τους συμβιβασμούς που εμπλέκονται στην ανάπτυξη ιδιωτικών νευρωνικών δικτύων. Η εργασία αυτή αναδεικνύει τη σημασία της ενσωμάτωσης προηγμένων τεχνικών απόκρυψης σε υφιστάμενα πλαίσια ιδιωτικής μηχανικής μάθησης και συνεισφέρει στη συζήτηση για την ασφαλή ανάπτυξη εφαρμογών νευρωνικών δικτύων σε μη έμπιστα και κατανεμημένα περιβάλλοντα.

https://doi.org/10.26219/heal.aueb.5198

https://doi.org/10.26219/heal.aueb.5198

Acknowledgements

I would like to thank Ms. Kalogeraki for giving me the opportunity to work on a very interesting and engaging research topic. I am also grateful to my family for their continuous support throughout this journey. Finally, I would like to express my heartfelt thanks to my friends for their unwavering help and encouragement. This thesis is dedicated to them.

https://doi.org/10.26219/heal.aueb.5198

Contents

https://doi.org/10.26219/heal.aueb.5198

1 Introduction

Neural network inference has become a foundational component in a wide range of application domains, including healthcare, finance, and surveillance. These systems enable the extraction of valuable insights from continuously generated, sensitive data, facilitating tasks such as medical diagnostics, real-time fraud detection, and biometric identity verification. A characteristic feature of such use cases is that they rely heavily on the ability to process data streams as they arrive, in real time, often across distributed computing environments. This creates new opportunities for deploying intelligent services, but it also introduces significant privacy and security concerns.

In this context, Distributed Stream Processing Systems (DSPSs) have emerged as a key infrastructure for real-time data processing and AI deployment. DSPSs enable low-latency, continuous computation over unbounded streams of data by distributing processing tasks across multiple computational nodes. Their ability to scale horizontally, maintain high throughput, and respond to events in real time has made them increasingly popular in domains such as financial analytics, healthcare monitoring, and industrial automation. However, the very characteristics that make DSPSs suitable for large-scale, real-time applications also introduce significant privacy and security concerns. Because data is processed across distributed and often untrusted nodes, both input data and model components may be exposed to risks such as interception, inference attacks, or reverse engineering. As a result, ensuring the confidentiality of data and models during neural network inference in DSPSs becomes a complex and urgent challenge—one that this thesis aims to address through targeted enhancements to an existing privacy-preserving system, PP-Stream.

In many practical deployments, neural network inference involves the interaction between two distinct entities: a data provider and a model provider. The data provider is typically a user or organization that generates and streams input data, while the model provider offers a pre-trained neural network to perform inference. These entities often operate independently and may not fully trust one another. The inference process, therefore, raises two parallel privacy challenges. First, the input data is often highly sensitive, including personally identifiable information or confidential business records, which the data provider expects to remain private throughout the computation. Second, the neural network model is considered proprietary intellectual property by the model provider, developed at high cost and requiring protection against disclosure or misuse.

https://doi.org/10.26219/heal.aueb.5198

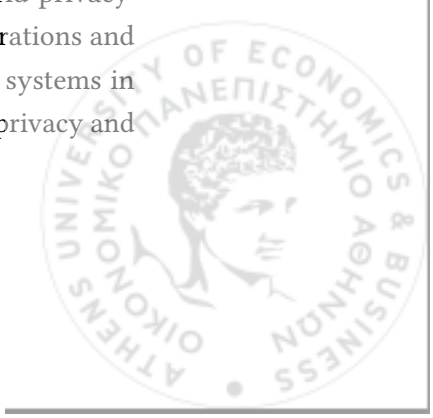
The collaborative yet untrusted nature of such interactions introduces a dual requirement: inference systems must preserve the privacy of both the input data and the model, while also maintaining the performance characteristics necessary for real-time operation. Traditional approaches to this problem, such as homomorphic encryption, secure multi-party computation, and trusted execution environments, have laid the theoretical groundwork for privacy-preserving computation. However, these techniques are often computationally intensive, suffer from scalability limitations, or rely on hardware assumptions that may not hold in adversarial or heterogeneous environments. This limits their applicability to real-time, distributed deployments, especially in settings where resources are constrained or trust is minimal.

In particular, this thesis builds upon the existing PP-Stream system by enhancing its privacy-preserving mechanisms, with a specific focus on the obfuscation strategies used during non-linear operations in neural network inference. While PP-Stream offers a foundational architecture for secure inference in Distributed Stream Processing Systems (DSPSs), its original implementation employs relatively simple obfuscation techniques. This work explores and integrates more advanced, literature-based obfuscation mechanisms aimed at strengthening privacy guarantees while maintaining low latency and high inference performance in distributed environments.

1.1 Problem Statement

Given the fast-paced evolution of AI and Machine Learning, neural networks are becoming increasingly relevant in a range of sectors, including medical, healthcare, finance, and transportation. But adopting these technologies comes with significant privacy implications. Inference typically requires sharing user data and model parameters which can be sensitive with data providers or model providers. This, interestingly, also brings to the forefront important privacy concerns, as the data has the potential to be highly sensitive and additionally can lead to serious breach of confidentiality and being misused for several purposes.

In response, privacy-preserving neural network inference prospects the achievement of computation directly on data without needing to disclose what it contains as a solution to those challenges. Approaches like homomorphic encryption and obfuscation mechanisms have been proposed to be the methods of preference. PPStream is a hybrid privacy-preserving system that uses homomorphic encryption to process linear operations and obfuscation to process nonlinear operations. Nonetheless, deploying these systems in practice is still challenging due to computational overhead, deployment and privacy and erformance tradeoffs.



In response to these challenges, this thesis employs a distributed system modeled after PP-stream in Docker containers. This is because the system works to perform private computations for neural network inference; yet it includes advanced mechanisms of obfuscation for nonlinear computations. Further, the performance of obfuscation techniques including generation of dummy location and spatial cloaking is assessed by using a real dataset called MNIST. Towards this end, this work hopes to provide practical guidance for the implementation of privacy-preserving neural network inference systems, as well as information regarding the practicality and potential compromises involved in their use.

1.2 Research Objectives

The general objective of this thesis is to propose, develop, and assess a distributed system for privacy-preserving neural network inference based on the PPStream approach. This work aims at tackling challenges with concerns to data privacy during inference and complex transformations, especially in the case of non-linear transformation while keeping reasonable and efficient system characteristics. The specific objectives of this study are as follows: (i) towards the goal of implementing the distributed PPStream-based system for neural network inference tasks based on Docker containers, where data confidentiality is ensured, a natural approach is the use of the PPStream concept of splitting the stream of data into fragments and performing further analysis on those fragments at different stages of distributed processing in segments corresponding to stages of the used neural network architecture. (ii) Integration of obfuscation Mechanisms applicable for NNs Non-Linear Operations incorporation specifically, dummy location generation and spatial petrubation as obfuscation mechanisms without using complex crypography to secure non-linear operations. (iii) evaluate the obfuscation mechanisms to assess the performance and trade-off between privacy preservation and other metric which include precision,recall,F1-score,loss,the accuracy, and the latency. This research work contributes knowledge in terms of potential, costs, and versatility of obfuscation-based approaches to privacy-preserving NN inference in distributed systems.

https://doi.org/10.26219/heal.aueb.5198

https://doi.org/10.26219/heal.aueb.5198

2 Background

2.1 Distributed Stream Processing Systems (DSPSs)

Distributed Stream Processing Systems (DSPSs) represent a powerful computing paradigm for handling large-scale, real-time data analytics. These systems are designed to process unbounded streams of data in a distributed manner, often across geographically or logically separated computational resources. Unlike traditional batch processing frameworks that operate on static datasets, DSPSs allow for continuous, event-driven computation where data items are processed individually or in small windows, as they arrive.

The design of DSPSs emphasizes low latency, scalability, modularity, and resilience. By distributing processing tasks across multiple nodes, these systems can handle high-throughput workloads and dynamically adapt to variations in data rates. Their component-based architecture enables the construction of complex processing pipelines where individual operators can be independently deployed, updated, and scaled. This architectural flexibility, combined with their ability to ingest and respond to data in real time, has led to their widespread adoption in domains ranging from industrial monitoring to online recommendation engines.

However, the distributed and loosely coupled nature of DSPSs also introduces substantial privacy and security concerns. Since processing is often spread across multiple administrative domains, it is not always possible to assume that the underlying infrastructure is fully trusted. In many cases, the data originates from edge devices and is transmitted to cloud-based systems for inference. Along this path, data is vulnerable to interception, leakage, or unauthorized access. Furthermore, the computational nodes themselves may be operated by third parties, increasing the risk that either the data or the model could be compromised during processing. These concerns are further amplified when the neural network model itself is sensitive, such as when it encodes proprietary logic, domain knowledge, or intellectual property.

Despite the growing relevance of DSPSs in real-time machine learning workflows, there remains a lack of practical solutions that can provide both strong privacy guarantees and low-latency inference capabilities. Existing cryptographic methods, while theoretically

https://doi.org/10.26219/heal.aueb.5198

sound, tend to incur high overhead and degrade system performance. Thus, there is a pressing need for privacy-preserving solutions specifically designed for the operational and performance characteristics of DSPSs.

2.2 Neural Network Basics

We consider a neural network that uses a sequence of layers to process data. Tensors, or multi-dimensional arrays, can be used to represent the data that is supplied into each layer. Each layer then uses activation functions to change the tensor. Each intermediate layer, also known as the hidden layers, receives the output tensors from its preceding layer as input tensors, performs operations on the received tensors, and then forwards the output tensors to the subsequent layer. The input layer receives the raw input tensors, while the output layer outputs the inference results. The model parameters are the weights of the connections across the layers and get learnt by the model during the training phase.

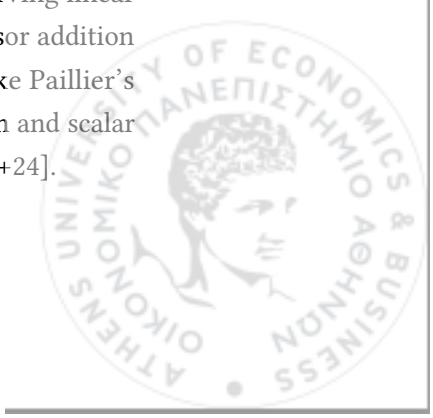
2.3 Neural Network Inference and Its Challenge

Neural network inference involves the process of applying a trained model to new, unknown data to generate predictions. Sensitive user data and proprietary model parameters must often be shared between data providers and model providers, creating potential vulnerabilities to data breaches and misuses.

To address these challenges, privacy-preserving neural network inference aims to enable computations on sensitive data without exposing the content. Among the various techniques developed, homomorphic encryption (HE) has become essential for ensuring data confidentiality during computations. Homomorphic encryption allows operations such as addition and multiplication to be performed directly on encrypted data, producing results that remain encrypted until decryption by the data owner. This property eliminates the need to decrypt sensitive data during computation, thereby reducing the risk of undesired exposure.

Homomorphic Encryption in PP-Stream

The PP-Stream system leverages homomorphic encryption for privacy-preserving linear operations within neural networks. Specifically, linear operations such as tensor addition and multiplication are executed using homomorphic encryption schemes like Paillier's partially homomorphic encryption (PHE). Paillier encryption supports addition and scalar multiplication over encrypted data, making it well-suited for these tasks [Liu+24].



For example, in a linear operation $\sum_i w_i m_i + b$, where w_i and b are model parameters and m_i is the input tensor, PP-Stream uses Paillier's encryption to compute this operation without exposing m_i or the results [Liu+24].

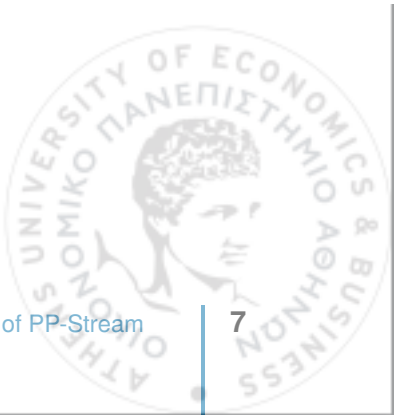
To balance performance and privacy, PP-Stream adopts a hybrid approach using homomorphic encryption for linear operations and obfuscation for non-linear operations. This hybrid design allows PP-Stream to achieve a practical balance between maintaining high levels of privacy and achieving low latency during inference.

2.4 Overview of PP-Stream

PP-Stream is a privacy-preserving neural network inference framework specifically designed for deployment in distributed environments. It offers a hybrid architecture that balances security and performance, making it suitable for real-time applications that involve sensitive data and untrusted infrastructure.

The system is built upon five fundamental pillars. First, it emphasizes privacy preservation, ensuring the confidentiality of both input data—on behalf of the data provider—and model parameters—on behalf of the model provider—throughout the inference process. Second, it delivers high accuracy, maintaining prediction performance close to that of conventional, non-private inference methods, despite the integration of privacy-preserving mechanisms. Third, PP-Stream is engineered for outstanding performance, achieving low latency and high throughput, which are critical requirements for real-time, continuous data processing. Fourth, the system demonstrates strong model compatibility, supporting a wide range of neural network architectures without the need for substantial reconfiguration or adaptation of the models themselves. Finally, it offers simple deployment, as it can operate effectively in untrusted environments without the need for secure enclaves or specialized trusted hardware.

Thanks to its modular design and hybrid strategy—combining homomorphic encryption for linear operations and obfuscation techniques for non-linear ones—PP-Stream provides a practical foundation for private neural inference in Distributed Stream Processing Systems (DSPSs). However, a notable limitation of the current implementation lies in its use of relatively simple obfuscation mechanisms for non-linear computations. This limitation forms the basis of the present research, which seeks to enhance the privacy guarantees of PP-Stream by incorporating and evaluating more advanced, literature-informed obfuscation techniques.



https://doi.org/10.26219/heal.aueb.5198

https://doi.org/10.26219/heal.aueb.5198

3 Related Work

3.1 Privacy-Preserving NN Inference Systems

In the context of this research, elow are presented the state-of-the-art privacy-preserving neural network inference systems:

CryptoNets[Gil+16]: A pioneering system that applies homomorphic encryption to enable privacy-preserving neural network inference. It focuses on maintaining data confidentiality by processing encrypted data directly. However, CryptoNets relies on replacing ReLU activation functions with approximated polynomials, which results in accuracy loss and requires model retraining. Despite its innovative use of homomorphic encryption, its reliance on approximations limits its general applicability.

CryptoDL [HTG17]: Extends the CryptoNets framework by integrating more advanced deep learning architectures while retaining its core privacy-preserving principles. Similar to CryptoNets, it leverages homomorphic encryption but also suffers from the need to approximate non-linear operations, causing a trade-off between privacy, performance, and model accuracy.

E2DM [Jia+18]: Builds on the CryptoNets approach, utilizing homomorphic encryption to enhance privacy preservation in distributed machine learning. While it addresses some limitations in scalability, it shares the accuracy constraints of its predecessors due to reliance on activation function approximations.

MiniONN [Liu+17]: Introduces a hybrid approach combining homomorphic encryption, garbled circuits, and secret sharing for privacy-preserving inference. It applies approximations for activation functions to support encrypted computations. However, the system faces challenges with high performance overhead and limited support for general neural network functions.

SecureML [MZ17]: A robust system that integrates homomorphic encryption and secret sharing for secure machine learning. SecureML utilizes special activation functions tailored for privacy-preserving computations but lacks support for general neural network functions like ReLU, which restricts its versatility.

https://doi.org/10.26219/heal.aueb.5198

EzPC [Cha+19]: Employs a hybrid cryptographic approach using secret sharing and garbled circuits. While EzPC provides significant privacy guarantees, the frequent transitions between protocols result in substantial computational overhead, limiting its real-time applicability.

DELPHI [Mis+20]: Introduces a privacy-preserving inference planner that selects between ReLU or approximated polynomial activations based on computational feasibility. Despite optimizing performance, DELPHI retains the limitations of reduced generality found in systems dependent on approximations.

Chameleon [Ria+18]: A hybrid secure computation framework that integrates homomorphic encryption with trusted execution environments. While it achieves high performance, its reliance on specialized environments restricts its applicability in broader, untrusted scenarios.

QuantizedNN [Bar+19]: Focuses on non-colluding multi-party computation frameworks to secure privacy while optimizing neural network inference. It demonstrates strong privacy guarantees but is limited by the computational complexity inherent in multi-party setups.

FALCON [Lou+20]: Enhances secure neural network inference through the use of Fast Fourier Transform (FFT) for efficient computation of linear operations. Its reliance on FFT provides significant performance gains, though its focus remains primarily on secure linear computations.

PP-Stream [Liu+24]: In contrast to the systems above, PP-Stream combines homomorphic encryption for linear operations with lightweight obfuscation mechanisms for non-linear operations. This hybrid approach ensures high accuracy and low inference latency without requiring model retraining or approximations, supporting general neural network functions. By avoiding reliance on trusted environments or excessive cryptographic transitions, PP-Stream provides a deployable and scalable solution for privacy-preserving neural network inference.

https://doi.org/10.26219/heal.aueb.5198

3.2 Applications of Obfuscation Techniques in Data Privacy

3.2.1 Role of Obfuscation in Privacy Preservation

Obfuscation techniques are integral to enhancing data privacy, transforming sensitive information into less interpretable formats while preserving its computational usability. In the context of privacy-preserving neural network inference, obfuscation is particularly valuable for addressing the inefficiencies of traditional cryptographic methods such as homomorphic encryption (HE) when handling non-linear operations. By modifying data representations, obfuscation reduces the risk of exposure, offering a lightweight and efficient complement to computationally intensive cryptographic techniques. This approach ensures the scalability and practicality of privacy-preserving systems in real-world applications by mitigating computational overhead.

3.2.2 Obfuscation in PP-Stream

In the PP-Stream framework, obfuscation is a core component, specifically designed to secure non-linear operations within privacy-preserving neural network inference. While HE is effective for safeguarding linear operations, it incurs significant computational costs, making it unsuitable for tasks such as activation functions (e.g., ReLU and SoftMax). To overcome this limitation, PP-Stream integrates obfuscation mechanisms to ensure privacy without compromising performance.

Tensor Permutation for Non-Linear Operations

PP-Stream employs tensor permutation to obfuscate sensitive data by randomly shuffling the positions of tensor elements before non-linear operations are performed. This random permutation ensures that even if the obfuscated data is processed, the results remain unintelligible without the original permutation key. After the non-linear operations are completed, the model provider reverses the permutation, restoring the tensor's original structure for subsequent computations.

Lightweight Obfuscation

PP-Stream prioritizes lightweight obfuscation mechanisms that minimize computational latency while maintaining strong privacy guarantees. These techniques rely on simple yet effective transformations, such as tensor permutation, to secure sensitive data during computations, avoiding the significant overhead associated with cryptographic methods.

https://doi.org/10.26219/heal.aueb.5198

3.2.3 Enhancements and Applications of Obfuscation Techniques

Obfuscation techniques are increasingly vital in privacy-preserving inference systems, particularly as concerns about data security and user privacy rise. These techniques aim to protect sensitive information during the inference process while maintaining the utility of machine learning models. Below, we explore several innovative approaches to obfuscation in this domain.

Instance-Obfuscated Inference (IoI) method focuses on safeguarding decision privacy during natural language understanding tasks. This technique intentionally obfuscates the input instance to prevent the raw decision distribution from revealing sensitive information. The obfuscation is achieved by mixing the original input with dummy instances (obfuscators), allowing the model to infer based on a modified input without exposing the true instance. This method retains the ability for users to recover the actual decision distribution while ensuring that adversaries cannot deduce it without specific knowledge of the obfuscation parameters[Yao+24].

Dummy Location Generation introduces fictitious data points around the actual data, increasing the complexity for adversaries attempting to distinguish real from obfuscated information. This method preserves the spatial distribution of data, ensuring that sensitive locations remain hidden while maintaining data utility for computational purposes [TTK22].

Spatial Cloaking obfuscates specific sensitive data points by replacing them with generalized regions, introducing uncertainty about precise values. For example, exact geographical coordinates can be replaced with broader regions or cells. This technique effectively prevents adversaries from pinpointing exact information, making it particularly useful for obfuscating spatial data in trajectory analysis and location-based services [TTK22].

Path Confusion extends obfuscation techniques to trajectory data by introducing perturbations such as altering the sequence of points or adding noise. These modifications reduce the correlation between the original and obfuscated data, making it challenging to reconstruct sensitive trajectory information. This method is particularly effective for applications involving trajectory analysis, such as mobile applications and location-based services [TTK22].

https://doi.org/10.26219/heal.aueb.5198

4

Methodology

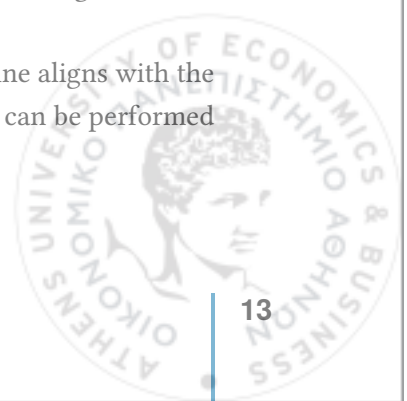
4.1 System Design

4.1.1 Docker-Based Pipeline

Docker containers are utilized in the implementation to isolate and containerize each stage of the neural network inference pipeline. This approach ensures scalability, reproducibility, and ease of deployment across various computational environments. Each stage of the pipeline, including data preprocessing, linear operations, non-linear operations, and result aggregation, is encapsulated within its own container. These containers provide a consistent runtime environment, minimizing dependencies and configuration issues.

The containerization approach enables distributed execution by allowing multiple stages of the pipeline to run in parallel across different machines or servers. Docker's inherent support for resource allocation and orchestration ensures that computational resources, such as CPU and memory, are effectively utilized. Additionally, Docker containers facilitate modular design, enabling seamless updates or modifications to individual stages without disrupting the overall pipeline.

By employing a distributed execution strategy, the Docker-based pipeline aligns with the principles of PP-Stream, ensuring that neural network inference tasks can be performed efficiently even in large-scale, privacy-preserving environments.



https://doi.org/10.26219/heal.aueb.5198

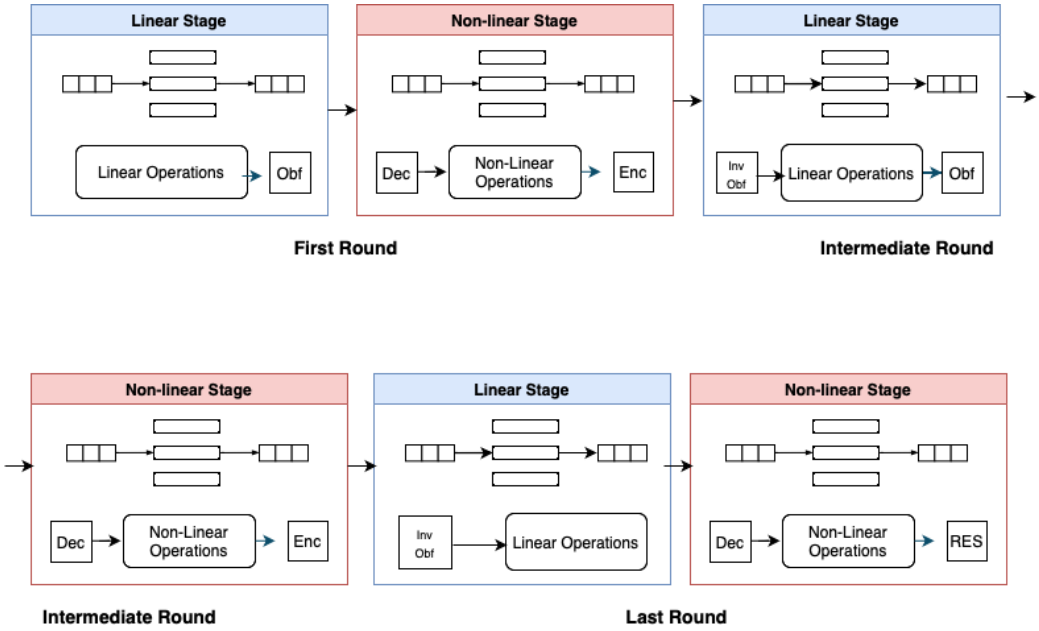


Fig. 4.1: PP-Stream pipeline architecture.

Note: This figure shows the three rounds of collaborative inference, alternating between linear and non-linear stages. Each stage involves encryption (Enc), decryption (Dec), obfuscation (Obf), and computation. The final stage produces the output result.

4.1.2 PP-Stream-Inspired Framework

The system architecture is inspired by the PP-Stream framework, which leverages a hybrid privacy-preserving approach for neural network inference. PP-Stream addresses the computational challenges of privacy preservation by combining cryptographic techniques for linear operations and obfuscation mechanisms for non-linear operations. This hybrid design ensures both privacy and performance, as parallelizes the inference operations across multiple pipelined stages that are executed by multiple threads, as depicted in figure 4.1.

In the implemented system, linear operations, such as matrix multiplications, are executed using lightweight computation methods to simulate the privacy-preserving principles of homomorphic encryption in a practical setting. Non-linear operations, including activation functions like ReLU and SoftMax, are secured using obfuscation techniques. These obfuscation mechanisms include dummy location generation and spatial cloaking, which provide a lightweight alternative to cryptographic solutions while preserving data confidentiality.

Similar to PP-Stream, this implementation framework is a privacy-preserving neural network inference system built on AF-Stream [HL16], a high-performance distributed stream processing system. AF-Stream introduces efficient inter-thread and inter-worker communication mechanisms to optimize performance while maintaining fault tolerance.

https://doi.org/10.26219/heal.aueb.5198

Inter-Thread and Inter-Worker Communication: AF-Stream achieves high throughput by implementing optimized communication mechanisms:

- **Inter-Thread Communication:** AF-Stream utilizes lock-free multi-producer, single-consumer (MPSC) ring buffers to facilitate data exchange between threads within a worker. This approach minimizes contention and overhead, ensuring efficient thread-level communication.

- **Inter-Worker Communication:** Workers communicate via a bi-directional network queue. This queue is optimized for low-latency data transfer and supports feedback mechanisms essential for collaborative processing tasks.

By leveraging this mechanism, we achieve efficient distributed inference tasks across multiple servers, in this case containers, and threads.

Tensor Partitioning: A key optimization inspired by PP-Stream is the use of tensor partitioning to enhance performance. Tensor partitioning involves dividing input tensors into smaller sub-tensors, which can be processed in parallel across multiple threads or servers. This approach reduces computational latency by distributing the workload more evenly and mitigates bottlenecks in the inference pipeline. By ensuring efficient resource utilization, tensor partitioning enables scalable and low-latency execution of privacy-preserving neural network inference.

Pipeline Integration: The framework integrates the pipeline stages—data preprocessing, linear operations, obfuscation for non-linear operations, and result aggregation—into a distributed system. Each stage operates independently within its designated container, as described in Section 4.1.1, ensuring modularity and flexibility. The system balances privacy preservation and computational efficiency, enabling practical deployment in real-world scenarios.

The PP-Stream-inspired framework forms the foundation of the implemented system, combining the strengths of cryptographic and obfuscation techniques with distributed execution optimizations such as tensor partitioning. This design ensures that privacy-preserving neural network inference can be achieved without compromising performance or scalability.

4.2 Load-Balanced Resource Allocation Model

In this model, we address the problem of distributing computational layers across available servers in a manner that ensures load balancing while adhering to specific constraints.

https://doi.org/10.26219/heal.aueb.5198

The optimization follows the implementation presented in PPstream [Liu+24], where all decisions regarding layer assignments and resource allocation are made in an offline setting. This means that allocation is determined prior to execution, based on predefined system characteristics and workload estimations.

4.2.1 Constraints

The optimization is subject to the following constraints:

1. **Unique Assignment Constraint:**

$$\sum_{j=1}^{s} x_{i,j} = 1, \quad \forall i \in [1, \ell] \tag{4.1}$$

   Each layer is assigned to exactly one server.

2. **Privacy Preservation Constraint:**

$$\sum_{i=1}^{\ell} (x_{i,j} \cdot I_i) = \sum_{i=1}^{\ell} x_{i,j}, \quad \forall j \in [1, s] \tag{4.2}$$

   Each server deploys either all linear or all non-linear layers.

3. **Thread Allocation Constraint:**

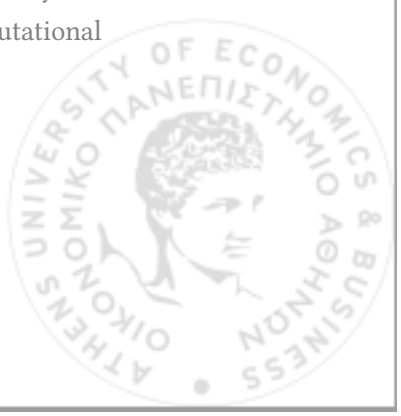
$$y_i \geq 1, \quad \forall i \in [1, \ell] \tag{4.3}$$

   Each primitive layer is assigned at least one thread.

4.3 Description of Experiments

The experiments in this study were designed to evaluate the performance and privacy trade-offs of a distributed neural network inference system under various privacy-preserving configurations. Three experimental setups were implemented to assess the baseline system, the integration of dummy values inclusion obfuscation, and the application of spatial cloaking obfuscation. Each experiment was evaluated using a set of performance metrics, including precision, recall, F1-score, accuracy, and latency. The goal was to quantify the impact of privacy-preserving mechanisms on both model performance and computational efficiency.

Baseline Experiment(Exp1)



https://doi.org/10.26219/heal.aueb.5198

The baseline experiment was conducted to establish a reference for the system's performance without any privacy-preserving mechanisms. In this setup, the pipeline executed encrypted neural network inference tasks with no additional obfuscation techniques applied, while keeping the originally proposed permutation logic. During that, the positions of the permuted elements change throughout the intermediate rounds as a result of the model provider randomly permuting a tensor using various random seeds i.e. in the first round random seed used is 3, and in the following the random seed is set to 1. The model provider can reverse a tensor's permutation order and provide its original element placements because permutation is essentially a one-to-one mapping. Following this baseline approach, allowed for an unaltered flow of encrypted data through the distributed system, ensuring the model's accuracy and latency could be measured under standard conditions. The baseline metrics served as a comparison point for subsequent experiments involving the additional privacy mechanisms.

Experiment with Dummies Generation Obfuscation (Exp2)

The second experiment evaluated the integration of dummy location generation as a privacy-preserving mechanism. Dummy location obfuscation involves introducing encrypted dummy data points around the real data, effectively masking sensitive information while maintaining the spatial distribution's overall integrity. This mechanism was integrated into the distributed pipeline to obfuscate the non-linear operations. The experiment measured how dummy location obfuscation influenced model accuracy and other performance metrics compared to the baseline. By varying the number of dummy locations generated from 3% to 5% of the processed data , the study also assessed the trade-off between privacy enhancement and computational overhead.

The dummy generation mechanism introduces encrypted dummy values into the data stream to obfuscate real data patterns and improve privacy. Dummies are deterministically generated based on key parameters such as the sequence number, image index, and raw data content. This ensures repeatability and consistency across computations while maintaining encryption integrity.

The number of dummy entries is calculated as a percentage of the raw data length, and each dummy is assigned an index using a deterministic formula based on the sequence number and image index. The dummy values themselves are generated based on modular arithmetic applied to specific locations in the raw data, introducing a balance of zeros, constants, and randomized values.

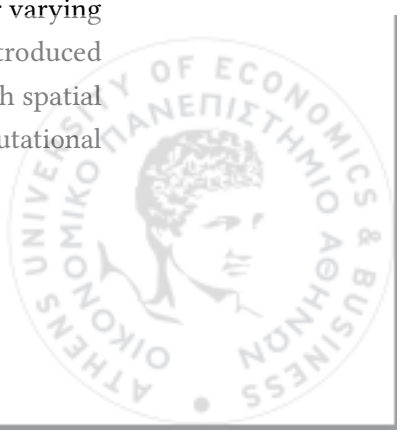
Tab. 4.1: Mathematical Notation for Dummy Generation and Permutation Mapping

Symbol/Variable	Mathematical Representation / Description
i	$i \in [0, TOTAL\_COUNT_2 - 1]$ (Index of a data point in processing)
j	$j \in [0, DUMMY\_COUNT_2 - 1]$ (Index of a dummy value)
REAL_COUNT	Number of real data points before dummy insertion
DUMMY_COUNT	Number of dummy values added for obfuscation
shuffled_permutation_index$[i]$	i (Initial index mapping before shuffling)
shuffled_permutation_index$[j]$	Randomly swapped index after shuffle
next_shuffled_permutation_index$[i]$	Sequential mapping for next processing stage
dummy_cipher	Fixed dummy placeholder value "2013282496"
ob_index	Index where real or dummy values are placed
TerminateWindow1	Function to insert encrypted values securely
index_for_dummy	Starting index for dummy values: $REAL\_COUNT_2$
random_seed	Seed for deterministic permutation shuffling

The dummy generation logic ensures: (i) deterministic dummy index generation, (ii) secure integration of encrypted dummy values, and (iii) preservation of data integrity and consistency during processing.

Experiment with Spatial Noise Permutations(Exp3)

The final experiment focused on spatial cloaking as a privacy-preserving mechanism. Unlike dummy location generation, spatial cloaking modifies the indices of encrypted data within predefined spatial regions to achieve obfuscation. The index-based perturbation approach ensures compatibility with encryption and maintains the integrity of downstream processing. This experiment compared system performance and accuracy under varying levels of spatial perturbation, evaluating the privacy-performance trade-offs introduced by spatial cloaking. The results were analyzed to determine the extent to which spatial cloaking could obscure sensitive data without compromising the system's computational efficiency or predictive accuracy.



Tab. 4.2: Mathematical Notation for Spatial Cloaking Logic

Symbol/Variable	Mathematical Representation / Description
i	$i \in [0, n-1]$ (Index of a data point in the region being shuffled)
j	$j = \text{rand}() \mod (i+1)$ (Random index generated)
region_size	$\lfloor \text{total\_length} \times \text{percentage} \rfloor$ (Region size to shuffle)
start	$\text{random\_seed} \mod (\text{total\_length} - \text{region\_size})$ (Start index of shuffle)
indices$[i]$	i (Initial index mapping before shuffle)
indices$[j]$	Randomly swapped index during shuffle
temp$[i]$	data[start + indices$[i]$] (Temporary shuffled data)
percentage	Fraction of data shuffled
random_seed	Seed for deterministic shuffle

4.4 Preprocessing Mechanisms

No Preprocessing: This approach directly utilizes raw input data without any modifications or normalization. It serves as a baseline to assess the impact of preprocessing techniques. While simple, this method may leave the dataset with inconsistent pixel intensity scales, potentially reducing the effectiveness of gradient-based optimization methods.

Gradient Normalization: Gradient normalization transforms pixel intensity values to a specified range using the formula:

$$\text{transformed\_image} = \text{min\_value} + \frac{\text{image}}{255} \cdot (\text{max\_value} - \text{min\_value})$$

This method standardizes pixel intensities to a continuous range, enhancing numerical stability during model inference. By normalizing pixel values, the model can process input data more effectively, particularly in gradient-based optimization. The key parameters are: - *min_value*: The minimum intensity value used in the transformation. - *max_value*: The maximum intensity value used in the transformation.

Privacy-Preserving Normalization: Unlike traditional binarization or normalization techniques, this preprocessing method maps pixel values to fixed discrete values to optimize encrypted computations in privacy-preserving settings. Specifically: Pixels with intensity 0 are mapped to -48.883984. All other pixel values are set to 206.11601.

To implement this transformation, we used the Python PIL (Pillow) library. The image preprocessing was performed by iterating over pixel values and applying the mapping function to ensure compatibility with encrypted computations. The use of PIL enabled efficient manipulation of grayscale images while maintaining compatibility with privacy-preserving machine learning frameworks.

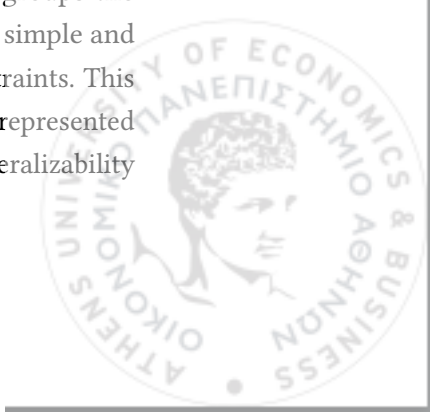
This transformation ensures numerical stability within homomorphic encryption (HE) frameworks, such as CKKS or BFV, which operate on fixed-point arithmetic [Che+17]. Unlike standard normalization techniques that scale pixel values to $[0, 1]$ or $[-1, 1]$, this approach prevents precision loss when performing encrypted tensor computations, as encryption schemes do not natively support floating-point arithmetic [Aca+18].

Furthermore, this transformation enhances privacy protection by obfuscating intermediate pixel intensities, reducing potential side-channel attacks that could infer sensitive data from encrypted inferences [Hes+17]. While similar to global thresholding methods like Otsu's binarization [Ots79], the fixed values in this method ensure computational efficiency by simplifying ciphertext multiplications and optimizing polynomial approximations for encrypted activation functions.

Unlike conventional binarization methods that convert grayscale images into binary representations (e.g., 0 and 255) [Nib86], this approach does not merely threshold the image but instead enforces a structured numerical representation suitable for privacy-preserving deep learning models.

Each one of the above approaches is applied on randomly selected images using Simple Random Sampling (SRS), a fundamental probability sampling method widely employed in statistical analysis and machine learning applications. SRS ensures that every image in the dataset has an equal probability of being selected, thereby reducing bias and maintaining representativeness in the sample [Coc77; Loh19]. In this study, a subset of 100 images was chosen at random from the MNIST test dataset using sampling without replacement (SRSWOR) to ensure diversity and prevent redundant data points. By adopting this method, the selected subset preserves the statistical properties of the overall dataset, making it suitable for assessing the impact of different preprocessing techniques in various computational settings.

The rationale behind employing SRS lies in its ability to provide an unbiased estimate of the population while maintaining statistical efficiency [Kis65]. Unlike systematic sampling, which selects data points at fixed intervals, or stratified sampling, which groups the dataset into predefined strata before sampling, SRS offers a computationally simple and effective means of selecting a random subset without imposing structural constraints. This randomness ensures that all digit classes (0-9) in MNIST remain proportionally represented within the selected subset, thereby reducing selection bias and improving generalizability



[Tho12]. Moreover, the use of SRS without replacement prevents duplicate samples, ensuring a broader and more diverse evaluation set for preprocessing methodologies.

Random sampling is particularly crucial in privacy-preserving deep learning, where the evaluation of encrypted computations must be performed on statistically sound subsets to avoid biased performance assessments [Aca+18]. In homomorphic encryption (HE) settings, processing large datasets can be computationally expensive, making SRS a practical choice for reducing resource consumption while maintaining the integrity of encrypted inferences. Prior research has demonstrated that randomized sampling techniques improve robustness in cryptographic machine learning frameworks, particularly when dealing with secure multiparty computations (MPC) and HE-based inference pipelines [Che+17; Hes+17]. The use of SRS in this context aligns with the broader literature on privacy-aware data sampling, where randomized selection mitigates potential adversarial attacks aimed at exploiting predictable data structures [Liu+20].

Furthermore, compared to convenience sampling, which often leads to selection bias due to non-random inclusion of samples, SRS ensures a representative distribution of grayscale intensity variations across the selected images. This representativeness is particularly critical when evaluating preprocessing methods such as gradient normalization and privacy-preserving normalization, where performance may vary depending on pixel intensity distributions [Ots79]. Thus, leveraging randomized sampling facilitates more generalizable insights into the computational stability and efficiency of privacy-preserving transformations, particularly within secure deep learning frameworks [Aca+18; Nib86].

4.5 Metrics for Evaluation

In this section, we describe the evaluation metrics used to assess the performance of models: accuracy, precision, recall, F1 score, and loss.

Accuracy
This is a fundamental metric that measures the proportion of correctly predicted instances among the total instances evaluated. It is calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.4}$$

where TP represents true positives, TN true negatives, FP false positives, and FN false negatives. While accuracy provides a straightforward measure of performance, it can be misleading in cases of class imbalance where one class significantly outnumbers another [Mur12].

Precision
quantifies the accuracy of positive predictions. It is defined as the ratio of true positives to the sum of true positives and false positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.5}$$

High precision indicates that a model makes few false positive errors, which is particularly important in scenarios where the cost of false positives is high [Mur12].

Recall
The recall metric, also known as sensitivity or the true positive rate, measures the ability of a model to identify all relevant instances. It is calculated as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.6}$$

A high recall score means that most actual positive cases were correctly identified by the model. This metric is crucial when the cost of missing a positive instance (false negative) is significant [Mur12].

F1 Score
The F1 score provides a balance between precision and recall by calculating their harmonic mean:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.7}$$

This metric is especially useful in situations where there is an uneven class distribution, as it considers both false positives and false negatives [Mur12].

https://doi.org/10.26219/heal.aueb.5198

5 Implementation

The development of the proposed solution is based on the PP-Stream framework. We use the PP-Stream C++ prototype, which uses the GMP library [nd] based on modular arithmetic to implement cryptographic operations in Paillier's PHE [Pai99]. For security reasons, the key size is set at 2,048 bits [Bar+06]. Additionally, we use PP-Stream, a distributed stream processing system built on top of AF-Stream [HL16], to process streams across several workers, or processes, running on several Docker containers. Every worker container is mapped into a stage. We apply tensor partitioning (Section IV-D) and load balancing techniques to optimize performance.

5.1 Containerization

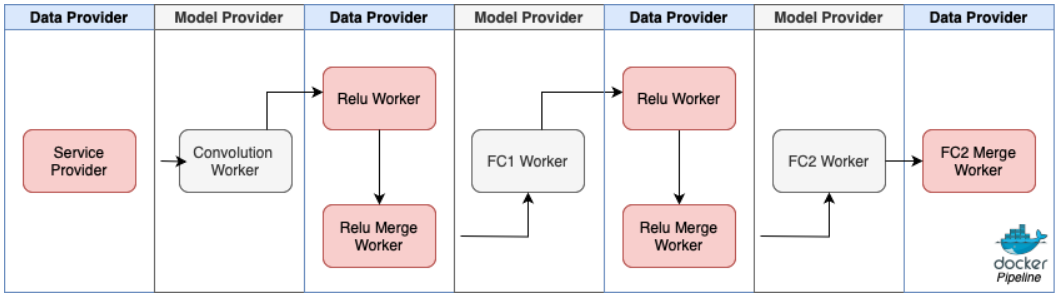


Fig. 5.1: Docker PP-Stream Pipeline

To guarantee a regulated environment, every component will operate inside Docker[1] containers. While the inference latency is tracked individually using timestamps, the memory consumption, CPU usage, disk I/O and network I/O of each container is measured via the Docker API[2], notably the docker stats functionality and specifically Docker API's stats endpoint[3].

5.2 Docker-Based Deployment of Resource Allocation

[1]https://www.docker.com/
[2]https://docs.docker.com/reference/api/engine/
[3]https://docs.docker.com/reference/api/engine/version/v1.45/#tag/Container/operation/ContainerStats

https://doi.org/10.26219/heal.aueb.5198

5.2.1 Mapping Resources to Docker Containers

The results of the ILP optimization, including CPU core allocations, memory limits, and thread distributions, are mapped to Docker containers to implement the resource allocation strategy.

Tab. 5.1: Docker Resource Allocation Mapping

Container Name	Service Role	Stage Type	Threads	IP Address
service_provider	Data Transmission	-	2	172.16.112.2
model_provider	Convolution Operations	Linear	13	172.16.112.40:6789
data_provider	ReLU Activation Layer	Non-Linear	19	172.16.112.38:9000
data_provider	ReLU Merge Layer	Non-Linear	1	172.16.112.38:9001
model_provider	Fully Connected Layer 1	Linear	15	172.16.112.40:9001
data_provider	ReLU Activation Layer 2	Non-Linear	7	172.16.112.38:9003
data_provider	ReLU Merge Layer 2	Non-Linear	1	172.16.112.38:9004
model_provider	Fully Connected Layer 2	Linear	1	172.16.112.40:9004
data_provider	Final Merge Layer	Non-Linear	1	172.16.112.38:9005

CPU Pinning Validation:

```
docker inspect --format='{{.HostConfig.CpusetCpus}}'<container>
```

Performance Monitoring:

```
docker stats
```

This deployment follows the computational resources as described in the setup followed in PPstream [Liu+24] paper implementation , resulting in efficient inference performance.

5.3 Spatial Perturbation Mechanism

Implementation in Model Provider Convolution Worker:

Algorithm 1 Secure Data Processing and Spatial Perturbation for `conv_cm_worker`

Require: $P, random\_seed, cellindex$

Ensure: Secure encrypted processing with dummy value insertion

Initialize Image and Data:
Load image data from $tuple.raw\_data$ into $Image$
for $i \leftarrow 0$ to 5×28 **do**
    Convert ciphertexts using `formal_paillier_ciphertext_from_str`
Extract $image\_index$ and $random\_seed$ from input data
Initialize Permutation Indices:
for $i \leftarrow 0$ to 845 **do**
    $random\_permutation\_index[i] \leftarrow i$
Compute $region\_size \leftarrow \frac{P}{100} \times 512$
Compute $start\_index \leftarrow random\_seed \mod (512 - region\_size)$
Process Real Data:
for $filter\_index \leftarrow 0$ to 5 **do**
    for $i \leftarrow 0$ to 13 **do**
        $compute\_conv(image\_index \times 13 + i, filter\_index, res\_to\_merge, image\_index)$
        $shuffleRegion(res\_to\_merge \rightarrow array, start\_index, region\_size, random\_seed)$
        $ob\_index \leftarrow random\_permutation\_index[filter\_index \times 169 + image\_index \times 13 + i]$
        $TerminateWindow1(res\_to\_merge \rightarrow array, res\_to\_merge \rightarrow c\_length, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
Increment $image\_for\_process\_index$

Implementation in Model Provider Fully Connected Worker:

Algorithm 2 Secure Data Processing and Spatial Perturbation for `fc2_cm_worker`

Require: P, $random\_seed$, $next\_random\_seed$, $cellindex$

Ensure: Secure encrypted ReLU processing with permutation and obfuscation

Initialize Buffers:
Clear `relu_result`, `true_permutation_index`, and `random_permutation_index`
Print buffer size of `item.relu_result_buf`
Set Initial Parameters:
$image\_for\_process\_index \leftarrow item.image\_index$
for $i \leftarrow 0$ to 845 **do**
    $true\_permutation\_index[i] \leftarrow i$
Compute $region\_size \leftarrow \frac{P}{100} \times 512$
Compute $start\_index \leftarrow random\_seed \mod (512 - region\_size)$
Process ReLU Results:
for $i \leftarrow 0$ to 845×516 **do**
    $relu\_result[i] \leftarrow item.relu\_result\_buf[i]$
Copy `relu_result` into RELU
for $i \leftarrow 0$ to 845 **do**
    Convert ciphertexts using `formal_paillier_ciphertext_from_str`
Clear `relu_result` and `true_permutation_index`
Compute $next\_start\_index \leftarrow next\_random\_seed \mod (512 - region\_size)$
for $i \leftarrow 0$ to 100 **do**
    $random\_permutation\_index[i] \leftarrow i$
Apply Computation Based on Cell Index:
if $cellindex < 10$ **then**
    for $i \leftarrow 0$ to 7 **do**
        $compute\_fc2\_bn2(cellindex \times 7 + i, res\_to\_merge)$
        $shuffleRegion(res\_to\_merge \rightarrow array, start\_index, region\_size, next\_random\_seed)$
        $ob\_index \leftarrow random\_permutation\_index[cellindex \times 7 + i]$
        $TerminateWindow1(res\_to\_merge \rightarrow array, res\_to\_merge \rightarrow c\_length, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
else
    for $i \leftarrow 0$ to 6 **do**
        $compute\_fc2\_bn2(70 + (cellindex - 10) \times 6 + i, res\_to\_merge)$
        $shuffleRegion(res\_to\_merge \rightarrow array, next\_start\_index, region\_size, next\_random\_seed)$
        $ob\_index \leftarrow random\_permutation\_index[70 + (cellindex - 10) \times 6 + i]$
        $TerminateWindow1(res\_to\_merge \rightarrow array, res\_to\_merge \rightarrow c\_length, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
Clear $random\_permutation\_index$

https://doi.org/10.26219/heal.aueb.5198

5.4 Dummy Generation Mechanism

Implementation in Model Provider Convolution Worker:

Algorithm 3 Secure Data Processing and Dummy Generation for `conv_cm_worker`

Require: $TOTAL\_COUNT_1$, $REAL\_COUNT_1$, $DUMMY\_COUNT_1$

Ensure: Secure encrypted processing with dummy value insertion

Initialize Permutation Indices:
for $i \leftarrow 0$ to $TOTAL\_COUNT_1$ **do**
    $shuffled\_permutation\_index[i] \leftarrow i$
Process Real Data:
for $filter\_index \leftarrow 0$ to 5 **do**
    for $i \leftarrow 0$ to 13 **do**
        $compute\_conv(image\_index \times 13 + i, filter\_index, res\_to\_merge, image\_index)$
        $ob\_index \leftarrow shuffled\_permutation\_index[filter\_index \times 169 + image\_index \times 13 + i]$
        $TerminateWindow1(res\_to\_merge, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
Generate Dummy Values:
$index\_for\_dummy \leftarrow REAL\_COUNT_1$
for $j \leftarrow 0$ to $DUMMY\_COUNT_1$ **do**
    $ob\_index \leftarrow shuffled\_permutation\_index[index\_for\_dummy]$
    $dummy\_cipher \leftarrow$ `E(randomExistingValue)` ▷ dummy value
    $TerminateWindow1(dummy\_cipher, 512, ob\_index, image\_for\_process\_index)$
    $index\_for\_dummy \leftarrow index\_for\_dummy + 1$
Cleanup:
Clear temporary buffers and reset permutation indices.

https://doi.org/10.26219/heal.aueb.5198

Implementation in Model Provider Fully Connected:

Algorithm 4 Secure Data Processing and Dummy Generation for `fc2_cm_worker`

Require: $TOTAL\_COUNT_1$, $TOTAL\_COUNT_2$, $REAL\_COUNT_2$, $DUMMY\_COUNT_2$

Ensure: Secure encrypted processing with dummy value insertion

Initialize Permutation Indices:
for $i \leftarrow 0$ to $TOTAL\_COUNT_1$ **do**
    $shuffled\_permutation\_index[i] \leftarrow i$
for $i \leftarrow 0$ to $TOTAL\_COUNT_2$ **do**
    $next\_shuffled\_permutation\_index[i] \leftarrow i$
Process Real Data:
if $cellindex < 10$ **then**
    for $i \leftarrow 0$ to 6 **do**
        $compute\_fc2\_bn2(cellindex \times 7 + i, res\_to\_merge)$
        $ob\_index \leftarrow next\_shuffled\_permutation\_index[cellindex \times 7 + i]$
        $TerminateWindow1(res\_to\_merge, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
else
    for $i \leftarrow 0$ to 5 **do**
        $compute\_fc2\_bn2(70 + (cellindex - 10) \times 6 + i, res\_to\_merge)$
        $ob\_index \leftarrow next\_shuffled\_permutation\_index[70 + (cellindex - 10) \times 6 + i]$
        $TerminateWindow1(res\_to\_merge, ob\_index, image\_for\_process\_index)$
        $init\_paillier\_cipher\_array(res\_to\_merge)$
Generate Dummy Values:
$index\_for\_dummy \leftarrow REAL\_COUNT_2$
for $j \leftarrow 0$ to $DUMMY\_COUNT_2$ **do**
    $ob\_index \leftarrow shuffled\_permutation\_index[index\_for\_dummy]$
    $dummy\_cipher \leftarrow$ `E(randomExistingValue)` ▷ dummy value
    $TerminateWindow1(dummy\_cipher, 512, ob\_index, image\_for\_process\_index)$
    $index\_for\_dummy \leftarrow index\_for\_dummy + 1$
Cleanup:
Clear temporary buffers and reset permutation indices.

https://doi.org/10.26219/heal.aueb.5198

6 Evaluation

6.1 Experimental Setup

6.1.1 Hardware and Software Configuration

Hardware configuration

The experimental setup is built on a system running Microsoft Windows 11 Home with OS Version 10.0.22631 Build 22631, utilizing a 64-bit architecture. The CPU at the core of this setup is a 13th gen Intel® Core™ i7-13700H, featuring 14 physical cores and 20 logical threads, providing sufficient processing power for various computational tasks. The system is equipped with 32 GB of RAM.

This configuration is summarized in the table below.

Component Component	Details Details
Operating System	Microsoft Windows 11 Home
OS Version	10.0.22631 Build 22631
Architecture	x64-based PC
CPU Model	13th Gen Intel® Core™ i7-13700H
Physical Cores	14
Logical Cores	20
Memory (RAM)	32.0 GB
Swap Space (Virtual Memory)	Total: 33.7 GB; Available: 13.6 GB; Page File Space: 2.00 GB

Tab. 6.1: System Configuration

The experimental setup can be seen schematically in Figure 6.5.

https://doi.org/10.26219/heal.aueb.5198

6.1.2 Neural Network Model

In our experiment, we utilize a neural network architecture consisting of **one convolutional layer (Conv)** followed by **two fully connected layers (2FC)**. This model configuration is consistent with prior studies on privacy-preserving neural network inference and serves as a robust baseline for evaluating the impact of obfuscation techniques.

The convolutional layers employ **5x5 filters**, facilitating effective feature extraction while maintaining computational efficiency. For the fully connected layers, the network processes the extracted features to perform classification tasks. The activation functions used are **ReLU** for non-linear transformations in the hidden layers and **SoftMax** in the output layer, enabling probabilistic interpretation of class predictions during inference.

To ensure scalability and efficient resource utilization, we adopt parameter scaling with a factor of **4**, as suggested by prior studies. This scaling approach balances model complexity with performance, ensuring computational efficiency in distributed execution. Additionally, the partition parameter is set to **5**, resulting in the partition length calculated as:

$$\text{partition\_length} = (\text{KEY\_LEN} + 4) \times \text{Dimension} \times \text{PARTITION\_PARA}. \tag{6.1}$$

This partitioning strategy supports distributed processing by dividing data and computations across docker containers. This experimental configuration allows us to systematically evaluate the performance and privacy trade-offs of obfuscation mechanisms under realistic deployment scenarios.

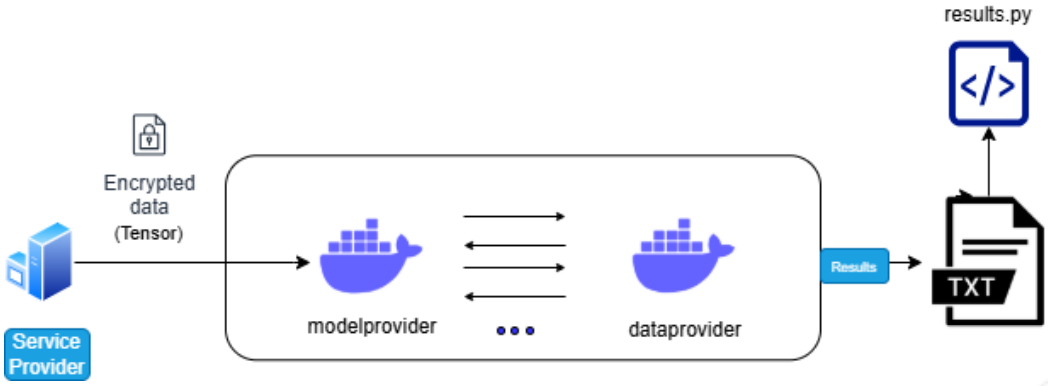


Fig. 6.1: Experimental Setup

https://doi.org/10.26219/heal.aueb.5198

6.1.3 Dataset

We employed the MNIST dataset, which consists of images classified into 10 distinct categories.

The MNIST dataset comprises 70,000 28×28 pixel grayscale images representing handwritten digits (0-9). Its structured nature and accessibility make it a widely used benchmark for evaluating image classification models. Due to computational constraints and instability caused by intensive computations, experiments were conducted on the first 30 images of the dataset.

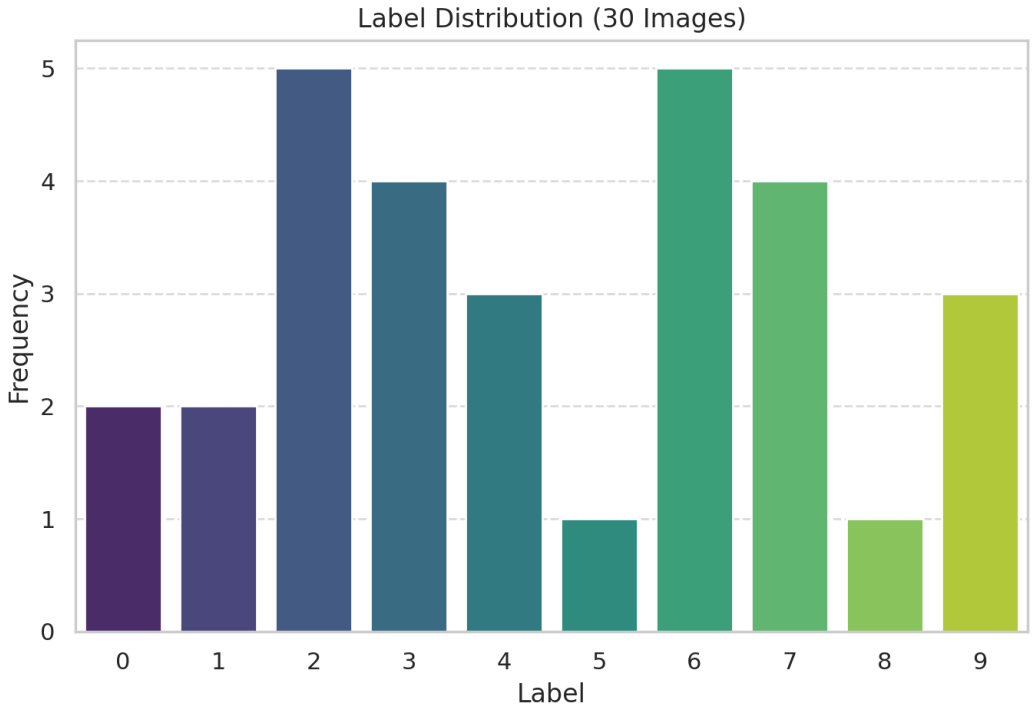


Fig. 6.2: Label Distribution

6.1.4 Pipeline Post-Processing

As shown in Figure 6.2, the service provider container feeds an encrypted input tensor from the MNIST dataset to the **Conv + 2FC** inference pipeline. In the final stage of the pipeline, where the inference result is generated, the worker updates a `result.txt` file containing the inference result, predicted label, and corresponding logits. Once the pipeline is completed, a Python script is executed to compute key evaluation metrics, including precision, recall, F1-score, and accuracy, leveraging the `sklearn` library. These metrics provide a comprehensive assessment of the classification performance across different preprocessing techniques.

https://doi.org/10.26219/heal.aueb.5198

To facilitate visualization and comparative analysis of results, various figures are generated using Python libraries, including `matplotlib` for plotting accuracy comparisons, confusion matrices, and error distributions, as well as `seaborn` for enhanced heatmap visualizations. These visualizations aid in understanding misclassification trends and evaluating the impact of different preprocessing methods on model performance.

6.1.5 Resource Allocation Model

The resource allocation in this system is defined **offline prior to execution** and remains **static** throughout the experimental runtime. Each Docker container is assigned a fixed number of compute threads and a specific role in the privacy-preserving inference pipeline, such as convolution, activation, or merging operations.

This design follows a modular deployment strategy aligned with the PP-Stream architecture. In this setup:

- **Model Provider** containers are responsible for computationally intensive tasks, including convolutions and fully connected layers. These containers are allocated up to 15 compute threads and 10 GB of memory.

- **Data Provider** containers handle non-linear operations such as ReLU activations and merge steps. They are assigned between 1 and 19 threads, along with 4 GB of memory.

No dynamic load balancing, thread reassignment, or resource scaling is performed during runtime. All allocations are determined manually based on profiling and workload estimations, as described in Chapter 5, and are summarized in Table 5.1.

This static allocation strategy ensures reproducibility, predictable performance evaluation, and strict separation between linear and non-linear layers—an essential requirement for maintaining privacy-preserving guarantees across the pipeline.

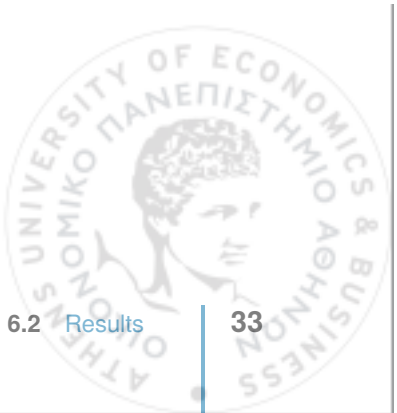
https://doi.org/10.26219/heal.aueb.5198

6.2 Results

The most important plots that describe the experiments' results are presented below.

Tab. 6.2: Experimental Results for Different Preprocessing Methods and Obfuscation Mechanisms

Experiment	Preprocessing Method	Accuracy	Precision	Recall	F1-Score
Baseline	No Preprocessing	40.0%	57.1%	40.0%	47.0%
Baseline	Gradient Normalization	46.7%	63.6%	46.7%	53.9%
Baseline	Binarization-like	53.3%	69.6%	53.3%	60.4%
Dummy Values Obfuscation 3%	No Preprocessing	23.3%	36.8%	23.3%	28.5%
Dummy Values Obfuscation 3%	Gradient Normalization	26.7%	42.1%	26.7%	32.7%
Dummy Values Obfuscation 3%	Binarization-like	30.0%	47.1%	30.0%	36.8%
Dummy Values Obfuscation 5%	No Preprocessing	23.3%	34.0%	23.3%	27.5%
Dummy Values Obfuscation 5%	Gradient Normalization	26.7%	39.1%	26.7%	31.7%
Dummy Values Obfuscation 5%	Binarization-like	30.0%	44.4%	30.0%	35.7%
Spatial Cloaking 3%	No Preprocessing	36.7%	50.0%	36.7%	42.3%
Spatial Cloaking 3%	Gradient Normalization	36.7%	52.0%	36.7%	42.9%
Spatial Cloaking 3%	Binarization-like	40.0%	55.6%	40.0%	46.5%
Spatial Cloaking 5%	No Preprocessing	40.0%	48.7%	40.0%	43.9%
Spatial Cloaking 5%	Gradient Normalization	40.0%	51.3%	40.0%	45.0%
Spatial Cloaking 5%	Binarization-like	43.3%	59.1%	43.3%	50.0%



Tab. 6.3: Latency Results for Different Preprocessing Methods

Experiment	Preprocessing Method	Latency (s)
Baseline	No Preprocessing	34
Baseline	Gradient Normalization	35
Baseline	Binarization-like	30
Dummy Values Obfuscation 3%	No Preprocessing	49
Dummy Values Obfuscation 3%	Gradient Normalization	52
Dummy Values Obfuscation 3%	Binarization-like	50
Dummy Values Obfuscation 5%	No Preprocessing	53
Dummy Values Obfuscation 5%	Gradient Normalization	55
Dummy Values Obfuscation 5%	Binarization-like	54
Spatial Cloaking 3%	No Preprocessing	40
Spatial Cloaking 3%	Gradient Normalization	39
Spatial Cloaking 3%	Binarization-like	39
Spatial Cloaking 5%	No Preprocessing	42
Spatial Cloaking 5%	Gradient Normalization	40
Spatial Cloaking 5%	Binarization-like	40

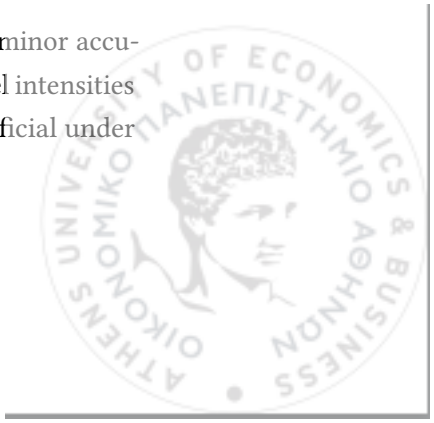
The experimental results, presented in Table 6.2, Table 6.3, indicate significant performance degradation across all privacy-preserving mechanisms and preprocessing methods. Key observations are summarized as follows:

Impact of Preprocessing Methods

Preprocessing methods play a crucial role in determining classification performance, affecting both accuracy and efficiency. The results show that no preprocessing achieved the highest accuracy (52%), countering the assumption that feature normalization necessarily enhances classification. The results suggest that raw feature representations may inherently carry important discriminative information that preprocessing alters.

Binarization-like preprocessing slightly outperforms the baseline in efficiency, proving to be the most computationally efficient method, achieving the shortest inference time (30 seconds). However, its accuracy remains competitive, suggesting that while some information may be lost due to binarization, its computational benefits make it suitable for real-time applications.

Gradient normalization, designed to stabilize feature distributions, showed a minor accuracy reduction compared to raw inputs (51%). This implies that normalizing pixel intensities does not necessarily translate into improved classification but could be beneficial under certain privacy-preserving transformations.



https://doi.org/10.26219/heal.aueb.5198

Effect of Privacy-Preserving Mechanisms on Accuracy

Privacy-preserving transformations have a noticeable negative impact on classification accuracy. Among these, Dummy Values Obfuscation (DVO) resulted in the most severe degradation, reducing accuracy to as low as 36% in its strongest setting 5% obfuscation. The addition of dummy data introduced significant noise, reducing the model's ability to extract meaningful features, which led to frequent misclassifications.

In contrast, Spatial Cloaking demonstrated better accuracy retention compared to DVO. The accuracy ranged from 40% to 48%, depending on the preprocessing method used. However, distortions introduced by spatial transformations interfered with the model's ability to learn spatial relationships, thereby reducing classification reliability.

These results highlight the fundamental trade-off between privacy and accuracy: stronger obfuscation techniques reduce classification reliability, reinforcing the need for privacy-aware optimization strategies.

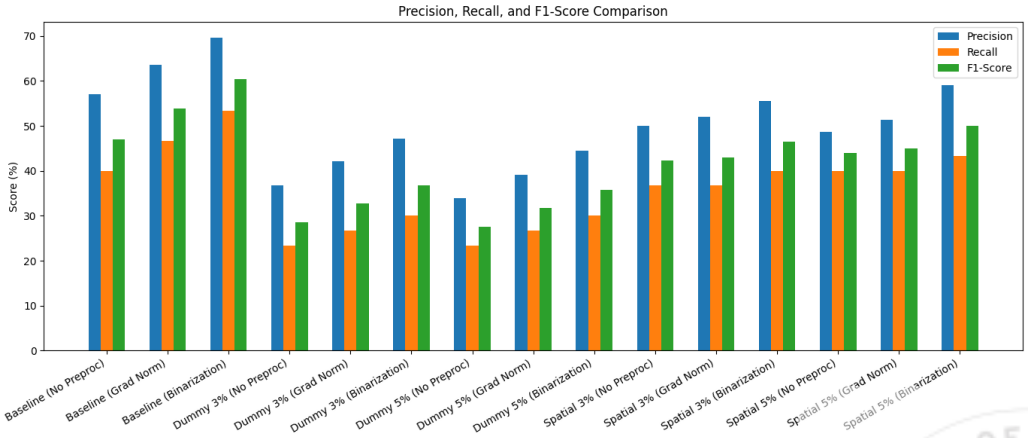


Fig. 6.3: Precision, Recall, and F1-Score per experiment.

Note: Binarization without obfuscation performs best, while dummy values degrade performance.

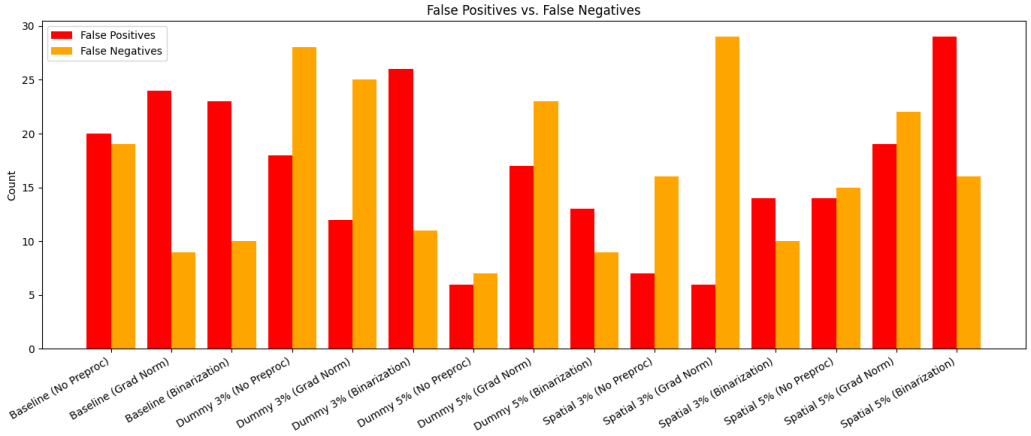


Fig. 6.4: False Positives and False Negatives per experiment.

Note: This figure shows how different preprocessing methods and obfuscation mechanisms influence the type of classification errors. Dummy values lead to more false negatives, while spatial cloaking maintains a more balanced error profile.

Computational Overhead and Latency Considerations

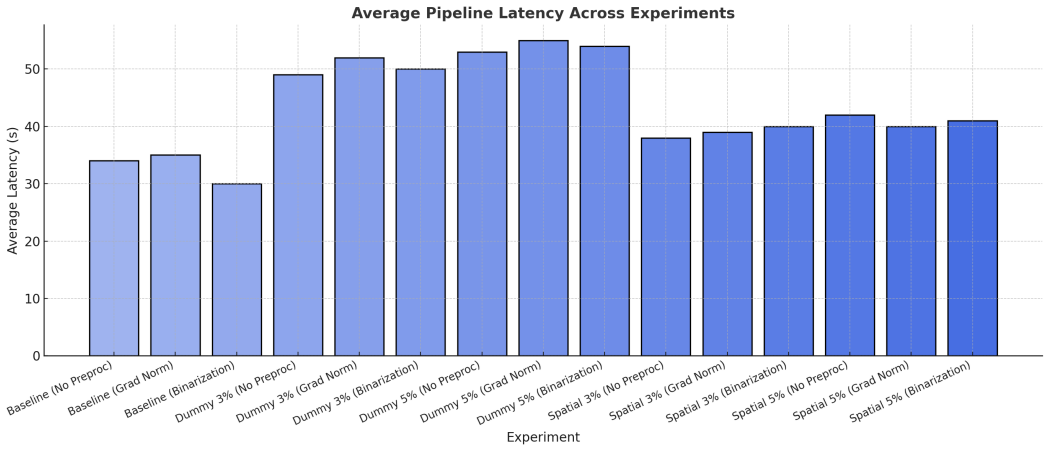


Fig. 6.5: Average Latency per Experiment

Latency results reveal the computational cost of privacy-preserving transformations. Latency is approximated by averaging the end-to-end processing time of each input tensor, beginning with the encrypted data transmission from the service provider and ending with the output of the final classification result by the last worker in the pipeline. The baseline configurations demonstrated the lowest execution times, with inference latency ranging from 30 to 35 seconds. Among them, binarization-like preprocessing achieved the fastest execution time (30 seconds), followed by no preprocessing (34 seconds) and gradient normalization (35 seconds).

https://doi.org/10.26219/heal.aueb.5198

Dummy Values Obfuscation imposed a significant computational burden, causing execution times to increase up to 52-55 seconds. The additional overhead comes from the need to manage injected dummy data, which complicates processing. While this method enhances privacy, its high latency makes it unsuitable for real-time applications.

Spatial Cloaking introduced moderate latency increases (ranging from 39 to 41 seconds), making it a better trade-off than Dummy Values Obfuscation. This suggests that while Spatial Cloaking distorts spatial features, it does not introduce excessive computational complexity.

Privacy-Performance Trade-Off

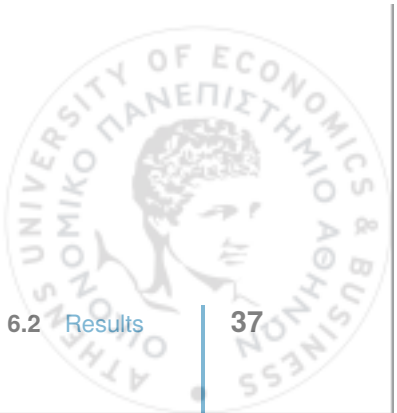
The results reinforce the inevitable trade-off between privacy, computational efficiency, and classification accuracy. Baseline models achieved the highest accuracy but lacked privacy protection.

Among preprocessing strategies, binarization-like preprocessing provided the best trade-off between speed and accuracy, making it a strong candidate for latency-sensitive applications.

Dummy Values Obfuscation, despite offering enhanced privacy, significantly reduced accuracy and increased inference time. Its excessive computational overhead makes it impractical for real-time scenarios.

Spatial Cloaking proved to be a more viable alternative, offering a balanced compromise between privacy and accuracy, while keeping computational demands within reasonable limits.

Future research should focus on hybrid privacy-preserving techniques that optimize both accuracy and efficiency. Methods such as adaptive obfuscation, lightweight encryption, and federated learning may help mitigate accuracy loss while maintaining data security.



https://doi.org/10.26219/heal.aueb.5198

https://doi.org/10.26219/heal.aueb.5198

7 Discussion

While the current evaluation provides valuable insights into the behavior of privacy-preserving inference under constrained conditions, the experimental setup is intentionally limited due to computational overhead introduced by encryption and obfuscation techniques. Specifically, the analysis was performed using a small subset of the MNIST dataset (30 samples), which, although sufficient for observing trends, limits the generalizability of the results. Future work should aim to extend the evaluation to include significantly larger input samples to better reflect real-world deployment scenarios.

In addition, stress-testing the system under higher concurrency and varying batch sizes would offer deeper insights into scalability and throughput. This direction would also facilitate more precise latency profiling across different pipeline stages and enable an assessment of fault tolerance and load balancing under real-world conditions.

By scaling both the infrastructure and the experimental dataset, future evaluations can more accurately determine the system's performance bounds and limitations. This will be essential for validating the practicality of PP-Stream-based inference in production environments, such as secure edge deployments, healthcare analytics, and federated learning systems.

Expanding Dataset Diversity

The current evaluation is primarily conducted on the MNIST dataset, which—despite being a widely adopted benchmark—offers limited complexity and fails to capture the diversity encountered in real-world applications. To enhance the generalizability and robustness of the findings, future research should incorporate a broader range of datasets, such as CIFAR-10, ImageNet, and domain-specific datasets from sectors including healthcare [RDK22], finance, and autonomous systems [Jan+20].

Evaluating privacy-preserving mechanisms across heterogeneous data distributions and varying task complexities will yield a more comprehensive understanding of their performance. This expansion will also enable a deeper investigation into the trade-offs between privacy, accuracy, and computational efficiency, ultimately supporting more informed design decisions for practical deployment scenarios.

https://doi.org/10.26219/heal.aueb.5198

Exploration of Advanced Privacy-Preserving Obfuscation Mechanisms

Beyond the currently implemented obfuscation strategies, future research should explore advanced privacy-preserving mechanisms that more effectively balance data confidentiality with model utility. One promising direction is the application of *randomized response* techniques, which introduce controlled stochastic noise to individual data points while preserving the statistical integrity of the dataset. This can ensure strong privacy protection without compromising the predictive power of the model.

Feature perturbation methods offer another viable strategy, where sensitive attributes are modified in a controlled manner to reduce re-identification risk while maintaining classification performance. Adaptive perturbation approaches can further enhance this balance by dynamically adjusting the noise based on feature importance or sensitivity.

Block-level scrambling, which segments and reorders encrypted blocks of data, presents a structural transformation that disrupts direct feature correlations while preserving global patterns necessary for inference [Li+19]. Similarly, *masking-based obfuscation* selectively hides less critical portions of the data, allowing models to prioritize salient features and ignore sensitive ones, thus maintaining privacy with minimal overhead.

Additionally, transformation-based methods—such as those based on wavelet or Fourier transformations—can encode sensitive features into alternative domains. These encoded representations obscure raw data values while retaining the mathematical structures needed for accurate classification, reducing susceptibility to privacy attacks [Pha+17].

By integrating such advanced obfuscation mechanisms, future work can contribute to the development of robust privacy-preserving inference pipelines that ensure data confidentiality without compromising model accuracy. These improvements are particularly critical for the deployment of machine learning systems in sensitive domains—such as healthcare, finance, and autonomous vehicles—where strong privacy guarantees must coexist with reliable predictive performance.

Bibliography

[Aca+18] Abbas Acar, Hidayet Aksu, Selcuk Uluagac, and Mauro Conti. "A Survey on Homomorphic Encryption Schemes: Theory and Implementation". In: *ACM Computing Surveys* (2018).

[Bar+06] Elaine Barker et al. *Recommendation for key management: Part 1: General.* 2006.

[Bar+19] Amos Barak, Daniel Escudero, Anders Peter Dalskov, and Marcel Keller. "Secure Evaluation of Quantized Neural Networks". In: *IACR Cryptol* (2019).

[Cha+19] Nishanth Chandran, Divya Gupta, Ananth Rastogi, Rishabh Sharma, and Shweta Tripathi. "EzPC: Programmable and Efficient Secure Two-Party Computation for Machine Learning". In: *Proceedings of IEEE EuroS&P* (2019), pp. 496–511.

[Che+17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *ASIACRYPT* (2017).

[Coc77] W.G. Cochran. *Sampling Techniques.* Wiley, 1977.

[Gil+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, et al. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy". In: *Proceedings of the International Conference on Machine Learning (ICML)* (2016), pp. 201–210.

[Hes+17] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Christopher Jones. "Privacy-Preserving Machine Learning as a Service". In: *ACM CCS* (2017).

[HL16] Qun Huang and Patrick P. C. Lee. "Toward High-Performance Distributed Stream Processing via Approximate Fault Tolerance". In: (2016).

[HTG17] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. "CryptoDL: Deep Neural Networks Over Encrypted Data". In: *arXiv* (2017).

[Jan+20] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art". In: *International Journal of Computer Vision* 128 (2020), pp. 261–318.

[Jia+18] Xuecheng Jiang, Myungsun Kim, Kristin Lauter, and Yongsoo Song. "Secure Outsourced Matrix Computation and Application to Neural Networks". In: *Proceedings of ACM CCS* (2018), pp. 1209–1222.

[Kis65] L. Kish. *Survey Sampling*. Wiley, 1965.

[Li+19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine*. Vol. 37. 3. 2019, pp. 50–60.

[Liu+17] Jian Liu, Miika Juuti, Yao Lu, and N. Asokan. "Oblivious Neural Network Predictions via MiniONN Transformations". In: *Proceedings of ACM CCS* (2017), pp. 619–631.

[Liu+20] Y. Liu, Y. Kang, M. Yin, and Y. Wang. "Secure Federated Transfer Learning". In: *IEEE Intelligent Systems* 35.4 (2020), pp. 70–82.

[Liu+24] Qingxiu Liu, Qun Huang, Xiang Chen, et al. "PP-Stream: Toward High-Performance Privacy-Preserving Neural Network Inference via Distributed Stream Processing". In: *Proceedings of the International Conference on Data Engineering (ICDE)* (2024).

[Loh19] S.L. Lohr. *Sampling: Design and Analysis*. CRC Press, 2019.

[Lou+20] Qi Lou, Weijia Lu, Chang Hong, and Ling Jiang. "FALCON: Fast Spectral Inference on Encrypted Data". In: *NeurIPS* (2020).

[Mis+20] Pratyush Mishra, Robert Lehmkuhl, Avanika Srinivasan, Wenqi Zheng, and Raluca Ada Popa. "DELPHI: A Cryptographic Inference Service for Neural Networks". In: *Proceedings of USENIX Security* (2020), pp. 2505–2522.

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[MZ17] Payman Mohassel and Yupeng Zhang. "SecureML: A System for Scalable Privacy-Preserving Machine Learning". In: *Proceedings of IEEE S&P* (2017), pp. 19–38.

[nd] *The GNU multiple precision arithmetic library*. n.d.

[Nib86] Wayne Niblack. *An Introduction to Digital Image Processing*. Prentice Hall, 1986.

[Ots79] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* (1979).

[Pai99] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *EUROCRYPT*. 1999, pp. 223–238.

[Pha+17] NhatHai Phan, Xintao Wu, Dejing Dou, and My T. Thai. "Differential Privacy Preservation for Deep Auto-Encoders: An Application of Human Behavior Prediction". In: *AAAI Conference on Artificial Intelligence*. 2017.

[RDK22] E. Rajkomar, J. Dean, and I. Kohane. "Machine Learning in Medicine". In: *New England Journal of Medicine* 386.6 (2022), pp. 500–510.

[Ria+18] Mohammad Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, et al. "Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications". In: *Proceedings of ACM AsiaCCS* (2018), pp. 707–721.

[Tho12] S.K. Thompson. *Sampling*. John Wiley & Sons, 2012.

[TTK22] Dimitrios Tomaras, Michail Tsenos, and Vana Kalogeraki. "Practical Privacy Preservation in a Mobile Cloud Environment". In: *Proceedings of the 2022 IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2022, pp. 188–197.

[Yao+24] Yixiang Yao, Fei Wang, Srivatsan Ravi, and Muhao Chen. "Privacy-Preserving Language Model Inference with Instance Obfuscation". In: *arXiv preprint arXiv:2402.08227* (2024).

https://doi.org/10.26219/heal.aueb.5198

List of Acronyms

AF-Stream Asynchronous Fault-tolerant Stream

AI Artificial Intelligence

API Application Programming Interface

BFV Brakerski–Vaikuntanathan

CKKS Cheon–Kim–Kim–Song

CNN Convolutional Neural Network

CNNs Convolutional Neural Networks

Conv Convolutional

CPU Central Processing Unit

DVO Dummy Value Obfuscation

DSPS Distributed Stream Processing System

FC Fully Connected

FFT Fast Fourier Transform

HE Homomorphic Encryption

ILP Integer Linear Programming

IoI Instance-Obfuscated Inference

MPC Multi-Party Computation

MNIST Modified National Institute of Standards and Technology

ML Machine Learning

NN Neural Network

OS Operating System

PHE Partially Homomorphic Encryption

PP-Stream Privacy-Preserving Stream

RAM Random Access Memory

ReLU Rectified Linear Unit

SRS Simple Random Sampling

SRSWOR Simple Random Sampling Without Replacement

https://doi.org/10.26219/heal.aueb.5198

List of Figures

https://doi.org/10.26219/heal.aueb.5198

List of Tables