ITUJournal
Future and evolving technologies

ITU Journal on Future and Evolving Technologies, Volume 6, Issue 3, September 2025

ARTICLE

A practical homomorphic encryption approach for GDPR-compliant machine learning full training protocol

***Hyukki Lee*[1], *Jungho Moon*[1], *Donghoon Yoo*[1]**

[1] *DESILO Inc., Korea (Republic of)*

Corresponding author: Donghoon Yoo, donghoon.yoo@desilo.ai

This work introduces a novel privacy-preserving full training protocol for deep learning models using external data. It addresses the critical, yet under-explored, challenge of data privacy, especially under regulations like GDPR. While homomorphic encryption has been suggested as a breakthrough for utilizing private external datasets in machine learning, naively applying it to model training from scratch is impractical due to huge computational costs. The proposed method strategically encrypts only the minimal layers required to prevent privacy breaches. This approach is carefully designed to minimize the computational overhead of homomorphic encryption, making it efficient and scalable for larger models. Comprehensive analysis on benchmark datasets (ResNet-20/110 with CIFAR-10/100) confirms compliance with GDPR requirements without significant loss of accuracy. Experiments show a 1000x reduction in training time compared to models encrypted across all layers, which is the first published benchmark as far as we know.

Keywords: Fully homomorphic encryption, GDPR compliance, privacy-preserving machine learning

1. INTRODUCTION

In the development of artificial intelligence technologies, the acquisition of high-quality data poses a significant challenge, especially when it involves personal information that falls under strict privacy regulations such as the General Data Protection Regulation (GDPR) [1, 2]. Data-rich institutions often lack specialized personnel and cutting-edge computing infrastructure, including GPUs, which are essential for machine learning operations [3]. Moreover, machine learning experts frequently encounter barriers in accessing high-value private datasets. This gap poses a critical challenge: If data providers and model developers operate independently, how can the machine learning training process be facilitated to maintain both data privacy and model confidentiality?

The privacy legal frameworks enforce rigorous constraints on the collection and processing of data, which hinders AI researchers and developers from accessing the vast amounts of personal data necessary to train sophisticated models. Even when intending to acquire valuable health or financial data, the decision to transfer data is not solely at the discretion of the data controllers and must satisfy some conditions. One commonly accepted and practical approach is obtaining consent from data subjects as specified under GDPR Article 6.1(a)[1]. However, the process of acquiring new consent for sharing data with third parties, not initially agreed upon at the point of data collection, is not only costly but also tends to result in low participation rates [4, 5]. This low engagement significantly limits the effectiveness of data sharing initiatives. Another method involves the data provider anonymizing the input data before sharing it with the modeler; yet, this approach often compromises the quality of the input data [6].

[1] Processing shall be lawful only if, and to the extent that, at least one of the following applies: the data subject has given consent to the processing of his or her personal data for one or more specific purposes.

Alternatively, data collected without the consent of the individuals may be used for purposes other than those originally intended if these purposes are deemed compatible with the initial collection purposes (GDPR Recital 50). The use of personal data for scientific research or statistical analysis is deemed compatible under these regulations. This process must naturally be accompanied by appropriate safeguards to ensure the protection of the rights of the data subjects.

Most current Privacy-Preserving Machine Learning (PPML) techniques have focused on addressing privacy issues that can arise during the inference stage rather than during data collection for training [9]. These approaches include preventing model inversion attacks, which could deduce training data from a model's inference results, ensuring private inference by protecting input queries while still producing inference results, and thwarting model extraction attacks aimed at revealing a model's structure or parameters through its inference outputs [10]. Techniques such as federated learning and split learning have been explored within PPML to address privacy concerns related to training data [11]. However, these methods require participants like data providers and modelers to share the model, ensuring the privacy of training data but not without its challenges. Specifically, data providers must engage in some of the computations, which can be burdensome, and these approaches do not guarantee the confidentiality of the model. Although the GDPR permits the receipt of data for the development of machine learning models to a limited extent, it is challenging to find methods that provide data in compliance with these provisions for the current model training. The techniques previously mentioned, such as data anonymization, sharing outcomes from local training or utilizing public data, have primarily evolved to avoid the applicability of GDPR. These methods are effective ways to utilize data while protecting privacy, but they present several challenging issues that are difficult to resolve, as detailed below:

- **Data quality**: When training data is anonymized, information loss is inevitable. Particularly when applying differential privacy [12], perturbing the data significantly impacts accuracy, leading most techniques to synthesize data, creating new datasets. Even in these cases, a considerable amount of information loss invariably occurs, making it challenging to ensure the quality of data when it is anonymized and shared with third parties [13].
- **Data reuse**: The repurposing or resale of such data, even if anonymized, can lead to economic issues, including potential financial exploitation and unfair market practices. Moreover, in the case of pseudonymized data, the reuse is strictly bound by the requirement that it be used only for predefined purposes. This constraint is critical to ensure compliance with GDPR. If pseudonymized data is repurposed for uses other than those initially intended without appropriate measures, there is a significant risk of violating privacy laws.
- **Model confidentiality**: Ensuring model confidentiality in collaborative environments involves significant challenges. Techniques like federated learning can mitigate the risk by keeping the data localized, but they expose the model to potential threats, such as evasion attacks [14]. If the model is shared, there is no certainty that the receiving institution will use it only for statistical purposes; they could potentially use it for re-identification. Therefore, sharing a model must be accompanied by additional safeguards for model confidentiality.
- **Local resource**: In PPML methods like federated learning, the dependence on substantial local resources is particularly pronounced. Federated learning mandates that the training of models be conducted locally on each participant's device, leveraging their own data without transferring it to a centralized server. This process requires significant computational power at each node, which can introduce inefficiencies and inflate costs, especially in environments that lack advanced computational infrastructure. Addressing these resource constraints is critical to enabling broader adoption of privacy-preserving techniques.

In this paper, we propose a novel privacy-preserving training protocol that utilizes Homomorphic Encryption (HE) for pseudonymization to address the challenges outlined above. This protocol is applicable in the training phase of deep learning models, which corresponds to a data-driven statistical modeling approach. HE is particularly notable because it allows for operations on encrypted data without necessitating decryption, providing robust cryptographic security [15]. By encrypting the training data, HE enables modelers to use it for machine learning purposes without exposing the raw data. However, a considerable limitation of using HE is its substantial computational overhead. Training models with encrypted data using HE requires significantly more processing power, often hundreds to thousands of times more than what is needed for training with plaintext [15]. According to the *state-of-the-art* algorithm [7, 8, 16, 17], even just performing forward pass, not training, on a single image requires a significant amount of time. As shown in Table 1, under optimal conditions, it takes 1.40 seconds to perform a forward pass on a single image using ResNet-20. Assuming the CIFAR-10 dataset is used for training, performing a forward pass on 50000 images would take approximately 70000 seconds (about 19.4 hours). Roughly assuming that backpropagation requires twice the time, it would take 39 hours to complete one epoch. Even conducting only 100 epochs would take over five months. If training a deeper model like ResNet-110 is considered, it would take several times longer, making it highly impractical for training purposes.

Table 1 – Execution time of HE forward pass

Execution time (s)	Only CPU [7]	With GPU [8]
ResNet-20 forward pass (per image)	15.8	1.4
ResNet-20 forward pass on the CIFAR-10 Dataset (per epoch)	790,654 (9.15 days)	70,000 (19.4 hours)

The core concept of the proposed method to overcome these problems involves a targeted encryption strategy: homomorphically encrypt the few targeted layers for the training process, using the data provider's secret key. After processing through the few encrypted layers, the output is decrypted with the same secret key by the data provider, allowing the remainder of the training to proceed with this plaintext data. This targeted encryption approach, despite the higher computational cost for HE operations, maintains overall computational time within manageable bounds by restricting encryption solely to certain phases of the training. Even if operations on homomorphically encrypted data take much longer compared to plaintext operations, the overall computational time remains manageable if only a small portion of the entire training is conducted this way. Considering the increasing size of machine learning models, especially those with numerous layers like Large Language Models (LLMs) [18, 19, 20] and ResNet [21], the time required to train encrypted layers could be hundreds of times longer. Yet, even with this increase, the extended time for training encrypted layers in such extensive models results in the total training duration being manageably more than that of conventional HE methods, making this approach practically viable.

Naïve adoption of this concept without proper safeguards could be vulnerable, thus we take measures to mitigate this risk. The risk of raw training data reconstruction by a modeler is markedly facilitated if the values of the weights in the encrypted layers are known, as this could enable reverse-calculation of the input data during the training process. To prevent this, we ensure that the weights of the encrypted layers remain unknown by adding noise prior to decryption. This addition of noise effectively increases the entropy, making it infeasible to reverse-calculate the input data. Additionally, by refraining from decrypting the encrypted layers during the training process, we prevent potential exposure of its information. This comprehensive strategy effectively precludes the modeler from reconstructing the training data, thereby ensuring the privacy of the data throughout the training process. HE is considered a form of pseudonymization under GDPR because the data cannot be attributed to a specific data subject without the secret key, which serves as the additional information, and the machine learning model trained for statistical purposes such as classification can be used legally without consent from data subjects, thereby making the proposed protocol compliant with GDPR.

1.1 Contribution

- **Flexible and efficient privacy-preserving full training protocol without sacrificing data quality:** We propose a pioneering privacy-preserving full training protocol maintaining data quality. The proposed approach applies HE to only a few targeted layers of the training model, significantly reducing computational demands while still safeguarding sensitive training data. Additionally, the protocol is adaptable to any machine learning model. To the best of our knowledge, this is the first instance of training a model from scratch using HE.
- **Advanced solutions for GDPR compliance through diverse privacy-enhancing technologies:** While HE alone may not meet stringent privacy regulations such as GDPR, the proposed protocol introduces an integration of Privacy-Enhancing Technologies (PETs). This significantly reduces the risk of input data being reverse-calculated. This advanced safeguard not only elevates privacy protection but also ensures compliance with GDPR standards. The protocol expands the capability of utilizing personal data for machine learning training to third-party entities without access to the original data, a role previously restricted to data controllers. Furthermore, it can be orthogonally applied with traditional PPML techniques, such as Differentially Private Stochastic Gradient Descent (DP-SGD) [22, 23], to create anonymized models as needed.
- **Unveiling the first benchmark of full training with HE:** Through computational analysis and experimental validation, we demonstrate that the proposed protocol is not only feasible for use with deep learning models but also maintains manageable computational times in real-world settings. We provide empirical evidence showing that the proposed approach preserves model performance while boosting computational efficiency, distinguishing it from traditional techniques and establishing its practical advantage.

2. PRELIMINARIES

2.1 Homomorphic encryption

HE is a form of encryption that enables computation on ciphertexts. The operations performed on ciphertexts yield an encrypted result that, when decrypted, corresponds to the result of applying the same operations to the plaintext. This characteristic of HE allows for the processing of encrypted data while maintaining its confidentiality.

An essential feature of HE is its use of public key infrastructure. In this system, users are provided with a pair of cryptographic keys: a public key, which is shared openly, and a private key, which remains confidential. The public key enables the encryption of data, while the private key is essential for decryption. This structure ensures that data encrypted with a user's public key can only be decrypted with their corresponding private key, adding an additional layer of security.

2.2 CKKS HE scheme [24]

The CKKS HE scheme is known for its efficiency in performing operations on floating-point data. It offers homomorphic additions, multiplications, and rotation operations. The rotation operation enables a homomorphic cyclic shift of packed messages by a designated step. Homomorphic multiplications and rotations require key-switching, which is the most resource-intensive process in the CKKS scheme.

The efficiency of the CKKS scheme is significantly enhanced by incorporating *Single Instruction Multiple Data* (SIMD) parallelism since the scheme uses a cyclotomic polynomial ring with dimension N for encoding and encryption. SIMD enables the concurrent execution of identical operations across multiple data points, effectively leveraging the parallelism inherent in modern computational architectures. In the context of HE, this translates to performing parallel homomorphic operations on numerous data elements simultaneously, thus improving throughput and computational efficiency. This parallel processing capability is crucial for handling large-scale data operations. By accelerating computational tasks, SIMD makes CKKS-based HE more practical for real-world applications, where processing large volumes of encrypted data efficiently is paramount.

3. THE PROPOSED TRAINING PROTOCOL

3.1 Threat model

In the proposed privacy-preserving training protocol, we focus on the interactions between two primary groups: data providers (also known as data controllers in GDPR) and machine learning modelers, under the assumption of a *semi-honest* (or *honest-but-curious*) setting. This means that while both parties are expected to adhere to the protocol, their curiosity might drive them to glean additional information from the data or model. Due to their semi-honest nature, we assume that neither party will engage in *poisoning attacks* [25], which involve intentionally disturbing the data or model to affect the training process. This assumption aligns with GDPR, as pseudonymized data is still considered personal data.

Given this context, the threats to data and models in the scenario can be identified as follows:

- **Reconstruction of raw training data**: This threat involves the possibility that machine learning modelers might be able to reconstruct the original, sensitive training data from the training process. Such a scenario poses a significant risk to the privacy of the data subjects and the integrity of the data providers.
- **Evasion attacks [14]**: Evasion attacks represent a significant threat in the context of machine learning. In these scenarios, attackers might manipulate the model's input to either evade detection or produce incorrect outputs, compromising the model's reliability and effectiveness.
- **Data misuse**: The risk of data misuse is especially high when dealing with sensitive information. Data misuse encompasses unauthorized repurposing, reselling, or other forms of exploitation of the shared data, leading to privacy breaches and potential health and financial damage.

3.2 Privacy-preserving training strategy

The proposed training strategy employs a targeted layer encryption method, encrypting only a few layers of the machine learning model using the data provider's secret key. After encrypted computation, the result is decrypted on the provider side and returned to the modeler for further training. All layers except the targeted layers are updated with plaintext. The gradient of the targeted layers' weights is calculated using encrypted data, and these results are then used to update the encrypted layer. However, as HE alone cannot guarantee the privacy of training data and model confidentiality, noise addition is also utilized for additional protection. The detailed process is outlined step by step below as depicted in Fig. 1.

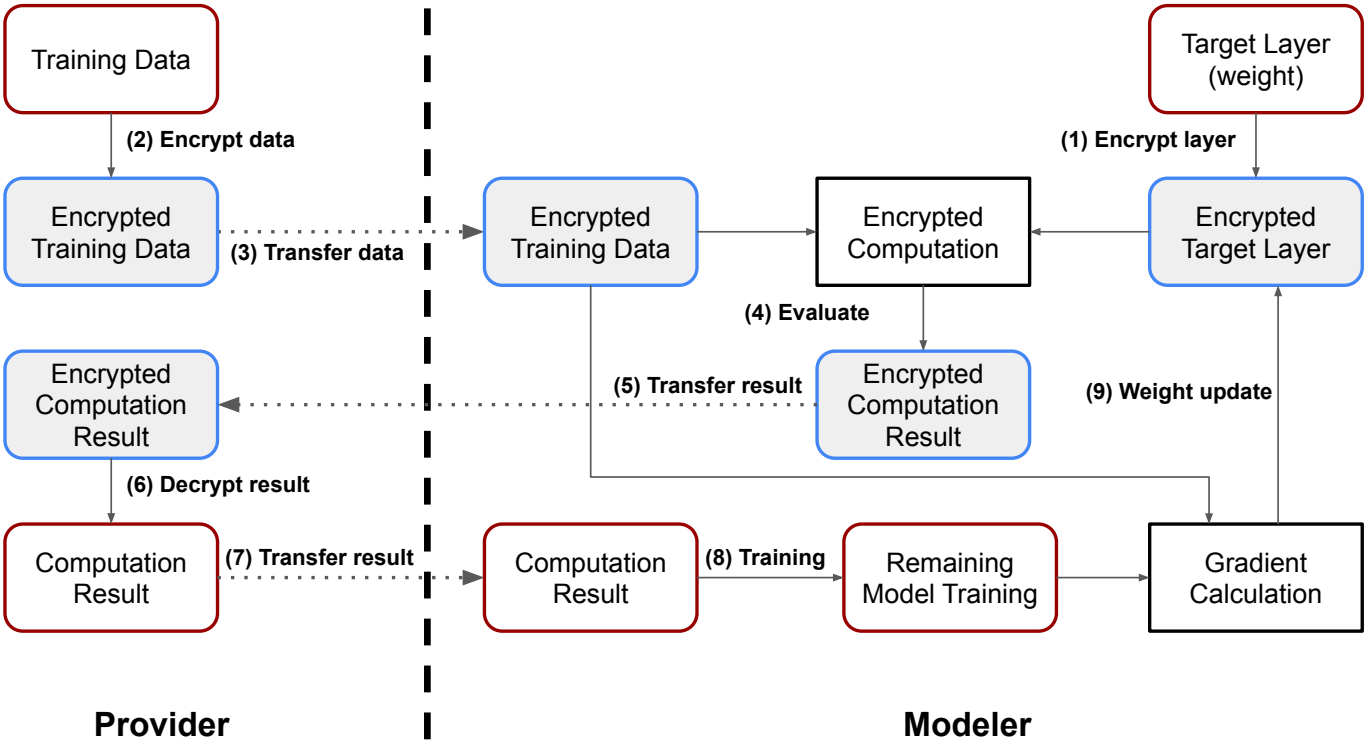


Figure 1 – Privacy-preserving training protocol

(1) Encrypt layer: The primary goal during this stage is to ensure that the modeler cannot determine the weight values of the encrypted targeted layers. The protocol permits only specific operations on the training data, and the results must be decrypted by the provider, making attacks like chosen-plaintext attacks infeasible. Consequently, even if the outputs processed through the encrypted targeted layers are decrypted, their high entropy makes it difficult to infer the training data. However, if the modeler were to know the exact weights of the encrypted targeted layers, they could potentially reconstruct the training data as learning progresses. Therefore, the protocol aims to perturb the weights of the encrypted targeted layers without revealing them to the modeler. One method to achieve this is by having the provider initialize the weights of the targeted layer and encrypt them before passing them to the modeler. If the modeler intends to utilize their own targeted layers, the modeler must transfer the relevant information about these layers to the provider. The provider then calculates the necessary noise, encrypts it, and returns it to the modeler. The modeler can subsequently apply this encrypted noise to the targeted layers using homomorphic addition, thereby integrating it into the training protocol.

(2) Encrypt data: The training data is encrypted using the provider's public key. The encryption method employed for the training data is pivotal in determining the efficiency of encrypted computations in this protocol. To exploit the efficiency of SIMD parallelism, the protocol encrypts multiple data points simultaneously. We primarily utilizes large mini-batch gradient descent [26]. In contrast, stochastic gradient descent [27], which updates weights using one data point at a time, necessitates weight updates for each input, complicating the effective use of SIMD operations. Consequently, to optimize computational efficiency, encryption tailored for SIMD operations is performed in accordance with the batch size of the training data.

(3) Transfer data: The encrypted training data is transferred to the modeler. Since the training data does not change, steps (2) and (3) of the process need only be performed once in the training protocol.

(4) Evaluate: The evaluation, referred to as homomorphically encrypted computation, is performed using encrypted training data and the encrypted weights of the targeted layers. The evaluation produces the result, which has passed through the targeted layer, as ciphertext.

(5) Transfer encrypted result: This phase involves the transmission of the evaluation results to the data provider. A critical consideration here is the risk of the provider gaining continuous knowledge about the gradient, which could lead to information that facilitates evasion attacks on the model. To mitigate this risk, the protocol requires that noise, known only to the modeler, be added to the encrypted results before they are sent to the provider. The

scale of this noise is crucial; it must be large enough to adequately conceal the original computation values. The modeler then transfers the noise-augmented evaluation results to the provider.

(6) Decrypt result and (7) transfer decrypted result: The data provider decrypts the evaluation results using their own secret key. After decrypting, the data provider sends the results to the modeler.

(8) Training: After receiving the computation results involving the targeted layers' weights and the training data, the modeler first removes the noise added in step (5) to recover the original output values. Subsequently, the modeler proceeds with training the machine learning model in plaintext. This process involves updating the weights in a manner typical of standard machine learning models, with the exception of the weights in the targeted layers.

(9) Weight update: The weights of the targeted layers remain encrypted until the end of the training process. To update these weights, the gradients are calculated using the encrypted input data and the gradients from the previous (plaintext) layer through homomorphic evaluations. This gradient information is not shared with the data provider, allowing the modeler to update the targeted layers' weights directly. Consequently, the weights of the targeted layers are updated while remaining unknown to both the modeler and the provider.

Training proceeds by iterating through steps (4) to (9). Once training is complete, the modeler needs to acquire the trained model, which requires the data provider to decrypt the encrypted targeted layers and transfer them to the modeler. As the machine learning model converges, the variations in the weights of the targeted layers decrease. During this convergence phase, if the modeler gains access to the original values of the weights of the targeted layers, there is a risk of reverse-calculating the input data. To address this risk, the proposed protocol adds random noise to the decrypted weights of the targeted layers. After decrypting these weights, the data provider perturbs them with noise before transferring them to the modeler. Consequently, even if the modeler uses the decrypted weights and the evaluation results from the previous phase, it becomes significantly more challenging to deduce the training data values.

4. IMPLEMENTATION

This section elaborates on the method for training a machine learning model using ResNet [21] by applying the proposed protocol. In this implementation, the first layer is vulnerable as it directly interacts with the training data. Therefore, the targeted layer is the first layer, which operates as a convolutional layer with its weights serving as filters.

4.1 Optimizing HE parameters for ResNet

When performing multiplications between ciphertexts, additional steps such as *relinearization* (key-switching) and *rescaling* become necessary for multiple multiplications. However, in cases where only a single multiplication is required for convolution computation, these additional evaluations are unnecessary. Typically, in the conventional method, the data provider encrypts messages for a one-depth multiplication, specifying two modulus spaces for a single multiplication. In a more straightforward approach, where only one multiplication is needed, it suffices to define only a base modulus for computation. Consequently, this approach reduces both the ciphertext size and the time required for evaluations, as shown in Table 2.

Since the rotation operation involves time-intensive key-switching procedures, minimizing these operations is advantageous. To achieve this, we ensure that the number of slots in a single ciphertext aligns with the batch size. Each ciphertext stores pixels from identical locations across all images within a single batch. Fig. 2 provides a representation of the process of packing a batch of images into ciphertexts. When encrypting a batch of 4096 images with 32×32 size and 3 channels, 3 ciphertexts are generated per pixel, resulting in a total of 3072 ciphertexts for all pixels. This alignment effectively eliminates the need for rotation evaluations during convolution operations.

Each weight in the filter is packed into a single ciphertext by replicating the same value to all slots. Throughout the training process, the filters remain encrypted and must be updated without being decrypted. Therefore, the weight updates (Δw) are also packed into a single ciphertext by replicating the same value to all slots to update the filters with homomorphic evaluation. For example, to calculate Δw_1, the process follows the steps shown in Fig. 3. Calculating Δw_1 requires performing homomorphic evaluation with the input data and the gradient of the previous layer, and then summing all the results. After performing the homomorphic evaluation, the resulting ciphertext contains the intermediate gradient values in each slot. These values must then be summed across all slots to obtain Δw_1, and the result needs to be replicated across all slots. To efficiently perform this operation, the Rotate-and-Sum (RotSum) technique [7] is adopted to aggregate the values within a single ciphertext, subsequently replicating Δw_1 to every single slot in the vectors.

Table 2 – Comparison with previous approach (Ring dimension $N = 2^{13}$)

	ciphertext size	multiplication time
Multiplication, rescaling and relinearization (previous)	$194KB$	$1870\mu s$
Multiplication only (proposed)	**131KB**	**154μs**

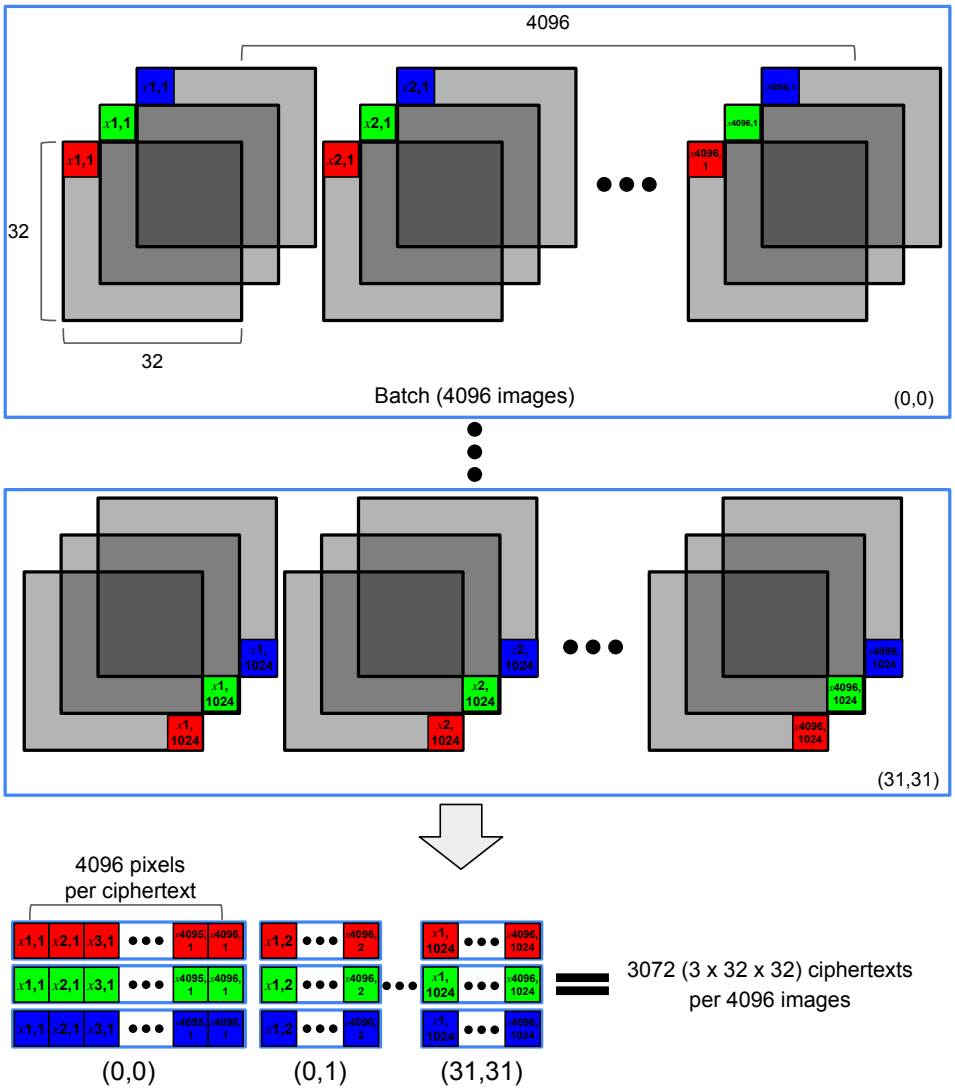


Figure 2 – Packing images for forward pass into vectors (32x32 pixels, 3 channels, 4096 images per batch, 3x3 filter)

4.2 Privacy analysis

We analyze privacy in terms of the information leakage about input data, based on output generated during each training epoch. The analysis includes an evaluation of the incremental information exposure about the images, observed through the outcomes of weight updates during each epoch. The degree of potential information leakage is quantified using the following approach:

$$h_{0,\frac{N}{M}} - h_{0,0} = -\eta \cdot \left(\sum_{k=0}^{\frac{N}{M}-1} \frac{1}{M} \sum_{i=kM}^{(k+1)M-1} \frac{\partial L}{\partial w}\bigg|_{x_i} \right) \times x_0 \qquad (1)$$

Here, $h_{0,0}$ represents the initial output for x_0 using the initial weights w_0, while $h_{0,\frac{N}{M}}$ denotes the output of the first data point x_0 after the completion of one epoch, considering all the weight updates. The learning rate, denoted by η, controls the step size in the weight update during the training process.

The summation term calculates the average effect of the gradients from each batch on the weight updates. Specifically, $\frac{\partial L}{\partial w}$ represents the gradient of the loss function with respect to the weights, evaluated at each data point x_i. Equation *(1)* collectively illustrates how the output for a specific data point changes over an entire epoch of

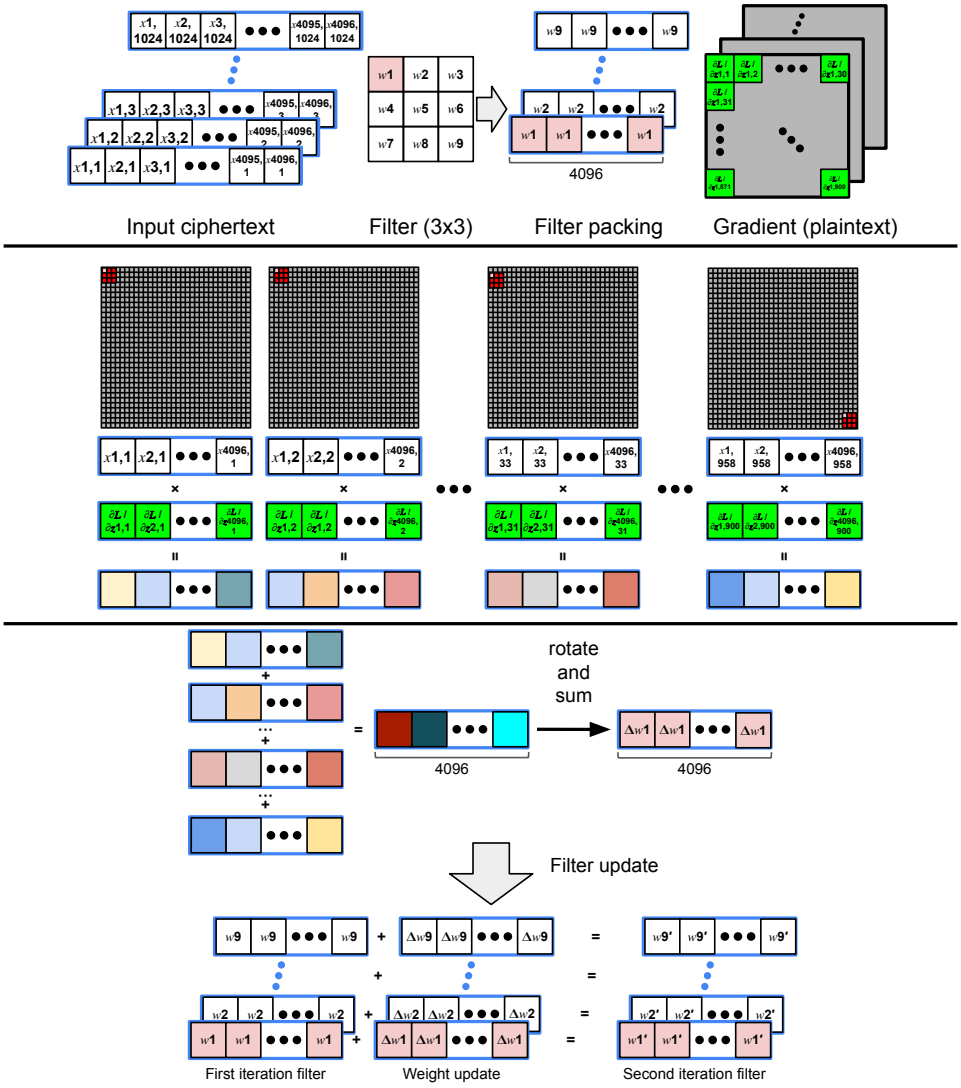


Figure 3 – Packing images for backpropagation into vectors (32x32 pixels, 3 channels, 4096 images per batch, 3x3 filter)

training, reflecting the cumulative impact of the learning process on the data point, thus providing insights into the privacy loss associated with the training.

5. EXPERIMENTS

5.1 Experimental setup

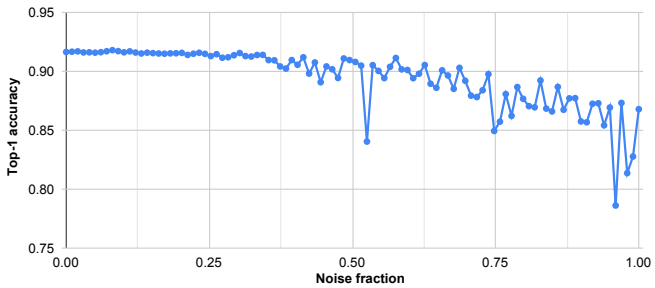
The experimental setup utilized an identical computing environment comprising both data provider and modeler, each configured with an AMD EPYC 7502 32-Core Processor and an NVIDIA RTX A6000 GPU, operating on Linux Ubuntu 18.04.6 LTS. These machines were interconnected via a 1Gbps network. For the cryptographic computations, we employed Microsoft SEAL [28], a HE library utilizing the RNS version of the CKKS scheme [29]. The GPU was used only for plaintext learning, while homomorphic evaluations were performed solely on the CPU. We selected a cyclotomic ring dimension of 2^{13}, resulting in each ciphertext having 2^{12} (4096) slots. For the machine learning tasks, we used the CIFAR-10 and CIFAR-100 datasets [30], along with the ResNet-20 and ResNet-110 architectures [21]. The first layer of the ResNet architecture, designated as the targeted layer, employs a convolutional layer with a filter size of 3×3 and a stride of 1. The batch size was set to 4096, which generates feature maps that retain the spatial dimensions of 32×32. In our experiments, we compare our proposed method with the *state-of-the-art* approach optimized for large batch forward pass [7].

5.2 Experimental results

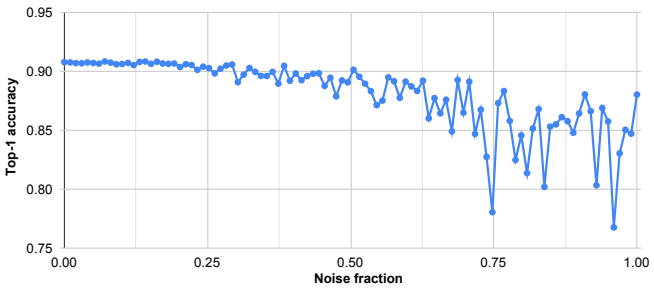
Table 3 presents the execution times. "Baseline" refers to experiments conducted with plaintext data, while the "Proposed" experiments were conducted with the protocol we introduced. The results from [7] are excerpted from their published work for comparison, and it should

Table 3 – Execution time (per epoch except initialization and after training, [7] performs forward only)

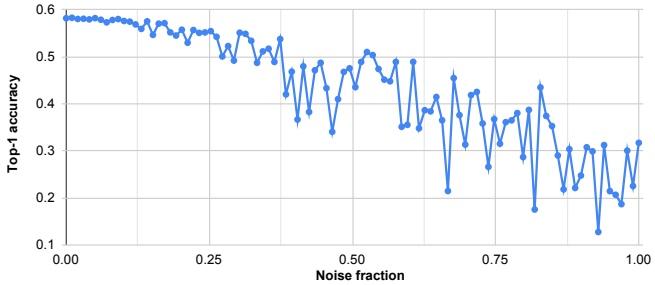
Part	ResNet-20			ResNet-110		
	Baseline	[7] (Only forward)	Proposed	Baseline	[7] (Only forward)	Proposed
Initialization & After training	-	-	108.58	-	-	108.58
Network transfer	-	-	400.14	-	-	400.14
Plaintext learning	-	-	24.7	-	-	35.1
Homomorphic evaluation	-	790,654	4,556	-	4,935,323	4,556
Execution time (per epoch)	24.7	790,654	**4,981**	35.1	4,935,323	**4,991**



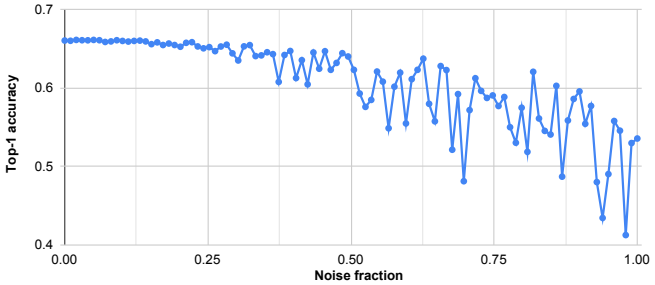
(a) CIFAR-10, ResNet-20



(b) CIFAR-10, ResNet-110



(c) CIFAR-100, ResNet-20



(d) CIFAR-100, ResNet-110

Figure 4 – CIFAR-10/100 classification accuracy with ResNet-20/110 varying noise

be noted that their method does not include backpropagation. The training time per epoch for the regular learning process is approximately 24.7 seconds for ResNet-20 and 35.1 seconds for ResNet-110. In contrast, the training time for the proposed protocol amounts to 4981 seconds for ResNet-20 and 4991 seconds for ResNet-110. As depicted in Table 3, a detailed examination reveals that the majority of the total time is consumed by network transfer and the homomorphic evaluation of the targeted layer. The initialization time includes the encryption and transfer time of the training data, and the homomorphic evaluation time encompasses all HE evaluations in both the forward pass and backpropagation. Homomorphic evaluation significantly increases execution time. According to [7], performing only a forward pass on 512 images took 8067.90 seconds, meaning a forward pass on 50000 images would require 790654 seconds. Including backpropagation, which necessitates changing HE parameters, would likely more than double the time required. In contrast, our proposed method completes training in about 4981 seconds per epoch for ResNet-20. Additionally, [7] reported that a forward pass on 512 images with ResNet-110 took 44850.24 seconds, and a forward pass for a single epoch took 4395323 seconds. The proposed technique completes a single epoch in just 4991 seconds, demonstrating a significant improvement in computational efficiency.

The accuracy results are presented in Fig. 4. These results were obtained by measuring accuracy after adding noise to each weight in ratios ranging from 0 to x. In a pseudonymization environment where shared data and the background knowledge of attackers can be sufficiently predicted and restricted, adding a certain ratio of noise alone can achieve an adequate level of privacy protection [31]. Increasing the noise reduces the possibility of reconstructing the original weights or training images, thereby enhancing privacy protection but also

decreasing accuracy. Adding noise at about 30% does not significantly degrade the experimental results. Therefore, setting the noise level by considering the privacy leakage, equation *(1)* in Section 4.2, can minimize information loss.

6. CONCLUSIONS

In this paper, we proposed a novel privacy-preserving training protocol. We introduced an algorithm that efficiently integrates HE and PETs to safeguard sensitive training data and ensure model confidentiality. By encrypting only few layers of the model, we significantly reduce computational demands while ensuring privacy protection. We believe our research represents an initial step towards utilizing HE for machine learning training, enabling modelers without direct access to data to securely utilize and train on sensitive datasets. The experimental results confirm that the protocol can be implemented in real-world settings with minimal impact on computational efficiency and model accuracy, providing a practical solution to the challenges of data privacy in AI. Furthermore, the protocol's compliance with GDPR and its applicability in sensitive sectors like healthcare and finance highlight its potential to facilitate secure and private data utilization across various industries.

REFERENCES

[1] Asia J Biega, Peter Potash, Hal Daumé, Fernando Diaz, and Michèle Finck. "Operationalizing the legal principle of data minimization for personalization". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2020, pp. 399–408.

[2] Bashir Rastegarpanah, Krishna Gummadi, and Mark Crovella. "Auditing black-box prediction models for data minimization compliance". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20621–20632.

[3] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. "Split learning for health: Distributed deep learning without sharing raw patient data". In: *arXiv preprint arXiv:1812.00564* (2018).

[4] Phaik Yeong Cheah, Nattapat Jatupornpimol, Borimas Hanboonkunupakarn, Napat Khirikoekkong, Podjanee Jittamala, Sasithon Pukrittayakamee, Nicholas PJ Day, Michael Parker, and Susan Bull. "Challenges arising when seeking broad consent for health research data sharing: a qualitative study of perspectives in Thailand". In: *BMC Medical Ethics* 19 (2018), pp. 1–11.

[5] Elizabeth Hutchings, Max Loomes, Phyllis Butow, and Frances M Boyle. "A systematic literature review of attitudes towards secondary use and sharing of health administrative and clinical trial data: a focus on consent". In: *Systematic Reviews* 10 (2021), pp. 1–44.

[6] Fengmei Jin, Wen Hua, Matteo Francia, Pingfu Chao, Maria E Orlowska, and Xiaofang Zhou. "A survey and experimental study on privacy-preserving trajectory data publishing". In: *IEEE Transactions on Knowledge and Data Engineering* 35.6 (2022), pp. 5577–5596.

[7] Jung Hee Cheon, Minsik Kang, Taeseong Kim, Junyoung Jung, and Yongdong Yeo. "High-Throughput Deep Convolutional Neural Networks on Fully Homomorphic Encryption Using Channel-By-Channel Packing". In: *Cryptology ePrint Archive* (2023).

[8] Donghwan Kim, Jaiyoung Park, Jongmin Kim, Sangpyo Kim, and Jung Ho Ahn. "HyPHEN: A Hybrid Packing Method and Its Optimizations for Homomorphic Encryption-based Neural Networks". In: *IEEE Access* (2023).

[9] Maria Rigaki and Sebastian Garcia. "A survey of privacy attacks in machine learning". In: *ACM Computing Surveys* 56.4 (2023), pp. 1–34.

[10] Runhua Xu, Nathalie Baracaldo, and James Joshi. "Privacy-preserving machine learning: Methods, challenges and directions". In: *arXiv preprint arXiv:2108.04417* (2021).

[11] Ziyao Liu, Jiale Guo, Wenzhuo Yang, Jiani Fan, Kwok-Yan Lam, and Jun Zhao. "Privacy-preserving aggregation in federated learning: A survey". In: *IEEE Transactions on Big Data* (2022).

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer. 2006, pp. 265–284.

[13] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. "Benchmarking differentially private synthetic data generation algorithms". In: *arXiv preprint arXiv:2112.09238* (2021).

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[15] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. "A survey on homomorphic encryption schemes: Theory and implementation". In: *ACM Computing Surveys (Csur)* 51.4 (2018), pp. 1–35.

[16] Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. "Bts: An accelerator for bootstrappable fully homomorphic encryption". In: *Proceedings of the 49th annual International Symposium on Computer Architecture.* 2022, pp. 711–725.

[17] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon yoo, Young-Sik Kim, et al. "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network". In: *IEEE Access* 10 (2022), pp. 30039–30054.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[19] Shyam Sudhakaran, Miguel González-Duque, Matthias Freiberger, Claire Glanois, Elias Najarro, and Sebastian Risi. "Mariogpt: Open-ended text2level generation through large language models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[20] Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. "PIXIU: A Comprehensive Benchmark, Instruction Dataset and Large Language Model for Finance". In: *Advances in Neural Information Processing Systems* 36 (2024).

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[22] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* 2016, pp. 308–318.

[23] Badih Ghazi, Yangsibo Huang, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. "Sparsity-Preserving Differentially Private Training of Large Embedding Models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[24] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer. 2017, pp. 409–437.

[25] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. "A comprehensive survey on poisoning attacks and countermeasures in machine learning". In: *ACM Computing Surveys* 55.8 (2022), pp. 1–35.

[26] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).

[27] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[28] Alexander J Titus, Shashwat Kishore, Todd Stavish, Stephanie M Rogers, and Karl Ni. "PySEAL: A Python wrapper implementation of the SEAL homomorphic encryption library". In: *arXiv preprint arXiv:1803.01891* (2018).

[29] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. *A Full RNS Variant of Approximate Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2018/931. https://eprint.iacr.org/2018/931. 2018. URL: https://eprint.iacr.org/2018/931.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[31] Yitao Duan. "Differential privacy for sum queries without external noise". In: *ACM Conference on Information and Knowledge Management (CIKM)*. 2009.

AUTHORS

Hyukki Lee received a Ph.D. degree in computer science from Korea University, Seoul, Korea (Republic of), in 2020. He is expected to receive a master's degree in law from Yonsei University, Seoul, Korea (Republic of). His current research interests include differential privacy, GDPR-compliant privacy engineering, and AI ethics.

Jungho Moon received a M.S. degree in mathematics from Hanyang University, Seoul, Korea (Republic of), in 2025. His current research interests include homomorphic encryption, computer arithmetic, and privacy-preserving machine learning.

Donghoon Yoo received M.S. and Ph.D. degrees in information and communication technology from the Gwangju Institute of Science and Technology, Gwangju, Korea (Republic of), in 1999 and 2005, respectively. His current research interests include homomorphic encryption, hardware acceleration, supercomputer architecture, and SW stack.

