Proceedings of the 3rd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2025.31672

ABS: An Adaptive Bootstrapping Scheduler for Efficient and Accurate Logistic Regression Training over Homomorphic Encryption

Kaishuo Wang

School of the Gifted Young University of Science and Technology, Hefei, China
titus@mail.ustc.edu.cn

Abstract. Homomorphic Encryption (HE) presents a com pelling paradigm for privacy-preserving machine learning; how ever, its practical adoption is hindered by substantial computa tional overhead. For iterative algorithms such as Logistic Regres sion (LR), the main performance bottleneck is bootstrapping, a noise-refreshing operation essential for maintaining correctness. Prevailing approaches employ static bootstrapping schedules, forcing practitioners into a suboptimal trade-off: either frequent bootstrapping with minimal parameters, which incurs a massive overhead, or infrequent bootstrapping with large parameters, which slows down all other operations. This study introduces the Adaptive Bootstrapping Scheduler (ABS), an algorithm that resolves this false dichotomy by reframing static scheduling as a dynamic optimization prob lem. ABS intelligently co-designs the HE parameterization and bootstrapping schedule to minimize the execution time while maximizing the model precision. At its core, the ABS leverages a fine-grained performance model to navigate the complex, non linear relationship between operational costs and bootstrapping frequency. Furthermore, we investigate a subtle but critical design choice: which intermediate value to use for bootstrapping. Our analysis, supported by prior work on noisy optimization, reveals that bootstrapping the gradient—a counter-intuitive but optimal strategy—significantly mitigates error accumulation via learning rate scaling. Implemented over the Poseidon library, our experiments on multiple public datasets demonstrate that ABS consistently accelerates LR training by up to 1.77× and achieves superior model fidelity compared to state-of-the-art static schedulers.

Keywords: the Adaptive Bootstrapping Scheduler, Logistic Regres, Homomorphic Encryption，fine-grained performance model，non linear relationship

1. Introduction

The modern data economy thrives on the vast amounts of data generated daily, which powers advances in machine learning and artificial intelligence. However, this progress creates profound tensions regarding fundamental privacy rights. The field of Privacy-Preserving Machine Learning (PPML) has emerged to resolve this conflict, developing techniques that enable computation on sensitive data without revealing the data itself [1]. As illustrated in Figure 1, major paradigms in

DOI: 10.54254/2755-2721/2025.31672

PPML include Secure Multi-Party Computation (MPC) [2], which distributes computation among multiple non-trusting parties, and Differential Privacy (DP) [3], which adds statistical noise to query results to protect individual records.

Among these, Fully Homomorphic Encryption (FHE) [4] offers a unique and innovative solution: a client can encrypt their data and send them to an untrusted server, which can then perform arbitrary computations (e.g., model training) on the ciphertext and return an encrypted result. This is particularly appealing for Machine Learning as a Service (MLaaS). FHE schemes have evolved significantly, from exact arithmetic (BGV [5], BFV [6]) to approximate real-number arithmetic, such as CKKS [7]. CKKS, particularly its RNS variant, which has native support for floating-point operations, has made the application of FHE to ML algorithms such as Logistic Regression (LR) a practical reality [8,9].

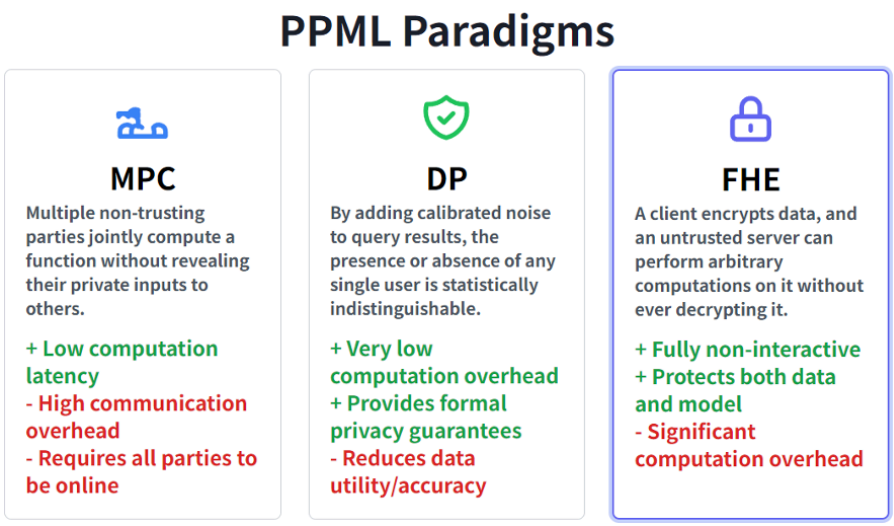

Figure 1. Comparison of major PPML paradigms. FHE offers a unique non interactive model suitable for untrusted cloud scenarios, which is the focus of this work

This template is modified using Microsoft Word. To simplify the formatting as much as possible, this template provides a complete set of pre-modified styles. It is highly recommended that the authors directly use this document to prepare their manuscripts and use these styles for ease of formatting. The styles can be found on the Home tab in the Styles group, or you may also view the Style pane by clicking "Alt + Ctrl + Shift +S". The authors may take a look at this "Apply styles" tutorial video provided by Microsoft Word if they are not familiar with how to apply styles.

However, a formidable obstacle remains: noise. Every ho momorphic operation, particularly multiplication, injects noise into the ciphertext. The solution is bootstrapping [10], a com putationally demanding but essential noise-reduction mecha nism in FHE. The decryption algorithm is homomorphically applied to a noisy ciphertext using an encrypted secret key to refresh its noise budget and restore its effective modulus. Nevertheless, the time cost of bootstrap operations cannot be ignored, posing a key design challenge for any iterative HE application: When should one bootstrap?

The prevailing approach in prior studies has been to adopt a static schedule. This creates a false dichotomy between two suboptimal extremes: a Minimum Latency Strategy (MLS) that uses minimal parameters but requires frequent, wasteful bootstrapping [11], and a Minimum Bootstrapping Strategy (MBS) that avoids bootstrapping at the cost of using large, slow parameters for all operations [12,13].

DOI: 10.54254/2755-2721/2025.31672

We argue that treating the bootstrapping schedule as a static hyperparameter represents a fundamental limitation of the method. Instead, it should be treated as a dynamic variable in holistic optimization problems. This study introduces the Adaptive Bootstrapping Scheduler (ABS), a framework that precisely achieves this. Our contributions are:

• We model the HE-based LR training process, captur ing the intricate, non-linear relationship between cryp to graphic parameters, operational latency, and bootstrap ping frequency.

• We propose the ABS algorithm, which leverages this model to automatically derive a near-optimal schedule and corresponding parameters, minimizing end-to-end training time.

• Weidentify and analyze a crucial, yet overlooked, aspect of precision management: the choice of which inter mediate value to bootstrap. We propose a strategy that demonstrably reduces the final-model error.

• We provide a comprehensive evaluation showing that ABS significantly outperforms the MLS and MBS base lines in both speed and precision.

2. Background

2.1. The RNS-CKKS scheme

Modern HE schemes, such as CKKS, are built upon the hardness of lattice problems, specifically the Ring Learning with Errors (Ring-LWE) problem [14]. The CKKS scheme [15] is designed for approximate arithmetic on real and complex numbers. It encodes a vector of numbers $z \in \mathbb{C}^{N/2}$ into a plaintext polynomial $m(X)$ in a polynomial ring $\mathscr{R} = \mathbb{Z}[X]/(X^N + 1)$. A ciphertext is a pair of polynomials $(b, a) \in \mathscr{R}_Q^2$, where Q is the ciphertext modulus. A message is encrypted as $ct = (b, a) = (-a \cdot s + e, a)$, where s is the secret key, and e is a small error polynomial.

The RNS variant [7] represents polynomials in the Residue Number System form, which decomposes large integer arith metic into smaller, parallel computations modulo a set of co-prime integers$\{q_0, \ldots, q_L\}$, where$Q = \prod_{i=0}^{L} q_i$. This is highly efficient for modern central processing units (CPUs).

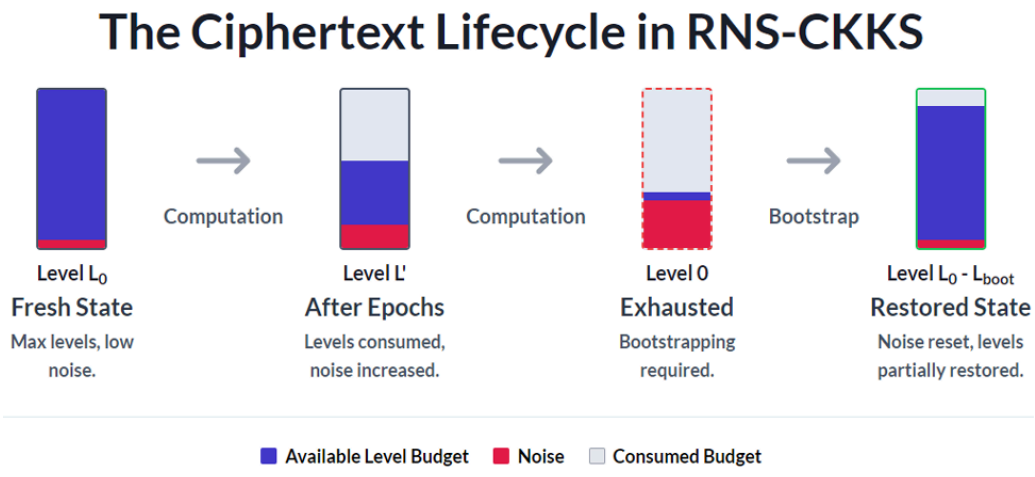

Figure 2. Conceptual model of noise and level management in CKKS. Each multiplication increases noise and consumes a level from the modulus chain. Bootstrapping resets both

DOI: 10.54254/2755-2721/2025.31672

The key operations include addition, multiplication, and rotation. The noise in the ciphertext increases with each op eration. To manage this, CKKS employs a modulus-switching technique, as shown in Figure 2. A fresh ciphertext has a large modulus Q. After multiplication, a rescaling operation is performed. This operation divides the ciphertext polynomial coefficients and the modulus by the last prime in the chain, q_L , effectively reducing the modulus to $Q' = \prod_{i=0}^{L-1} q_i$. While this controls the magnitude of the plaintext values, it also reduces the signal-to-noise ratio, and careful parameter selection is required to maintain precision [16]. The number of available levels, L+1, determines the multiplicative depth of the circuit that can be evaluated.

2.2. Bootstrapping in RNS-CKKS

When bootstrapping starts, it takes a ciphertext ct at a small modulus q_0 and "refreshes" it into a new ciphertext ct' encrypting approximately the same message, but with a large modulus Q. This allows for virtually unlimited computational depth. This process is complex and can be conceptually divided into four steps.

1) ModRaise: The modulus of the ciphertext is raised fromq_0 to a much larger modulus q_{boot} required for the subsequent steps.

2) CoeffsToSlots: This step transforms the polynomial's coefficients into plaintext slots, essentially making the coefficients themselves the message to be operated on.

3) EvalMod: This is the core step. It homomorphically evaluates a function that approximates the modular re duction operation. In CKKS, this is achieved by approximating a sine wave function, sin(x), with a polynomial [10], which clears the high-order bits of the encrypted message, where the noise resides.

4) SlotsToCoeffs: This step reverses CoeffsToSlots, con verting the processed slots back into the coefficients of the refreshed plaintext polynomial.

2.3. Logistic Regression via Gradient Descent

Logistic Regression is a model for binary classification. Given a feature vector $x \in \mathbb{R}^d$, it predicts the probability of the label being 1 as $p(x) = \sigma(w^T x)$, where $w \in \mathbb{R}^d$ are the model weights and $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function.

Training involves finding the optimal weights w using Gradient Descent (GD). In each iteration t, the weights are updated as follows:

$$w_{t+1} = w_t - \eta \nabla L(w_t) \tag{1}$$

where η is the learning rate, and $\nabla L(w_t)$ is the gradient. For a dataset of N samples $\{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$, the gradient is:

$$\nabla L(w_t) = \frac{1}{N} \sum (\sigma(w_t^T x_i) - y_i) x_i \tag{2}$$

To evaluate this in an HE setting, the non-polynomial sig moid function $\sigma(z)$ must be approximated using a low-degree polynomial. A common choice is a degree-3 approximation: $\sigma(\mathrm{z}) \approx 0.5 + 0.197\mathrm{z} - 0.004z^3$. The degree of this polynomial dictates the multiplicative depth required for each gradient descent step.

DOI: 10.54254/2755-2721/2025.31672

3. Related work

Our study is situated within the broader field of privacy preserving machine learning and builds on prior research on FHE-based computation.

1) HE for Machine Learning: The application of FHE to machine learning was first explored in works such as CryptoNets [9], which demonstrated the feasibility of neural network inference with encrypted data. For model training, Logistic Regression has been a popular benchmark owing to its iterative nature and reliance on basic arithmetic. Kim et al. [11] provided one of the first practical implementations of LR training on encrypted data, carefully selecting parameters for a single epoch. Han et al. [12] later improved this by focusing on batching techniques and using a large initial modulus to minimize the bootstrap frequency. These seminal works established the viability of FHE-based LR but relied on static scheduling, which our study aims to improve.

2) Automated HE Configuration: The challenge of managing FHE parameters and bootstrapping has recently attracted attention. The most relevant work is DaCapo [17], a compiler-based framework that automatically inserts bootstrapping op erations into an arbitrary HE-program. Our ABS framework is similar in its goal of automation but differs fundamentally in its approach and scope: it is an application-aware, model driven co-design framework, whereas DaCapo is a general purpose, compiler-driven insertion tool. In essence, ABS trades generality for higher performance in the target domain of iterative ML training.

3) System-Level Context and Orthogonal Research: It is important to understand the context in which the FHE operates. Other important paradigms in PPML, such as MPC [2] and FL [18], offer different trade-offs between computation, com munication, and trust assumptions. Our FHE-based approach provides a fully non-interactive model, which is advantageous in many client-server scenarios. Furthermore, our work at the application software layer is orthogonal and complementary to research on hardware acceleration [19,20]. Our contribu tion is a pure-software, algorithmic optimization that can be deployed on any standard hardware platform and can benefit from future hardware advancements.

4. Methodology: the ABS algorithmi

Our methodology centers on the ABS algorithm, which sys tematically solves the scheduling-optimization problem. The approach involves two key components: building an accurate performance model and using this model to guide an efficient search for the optimal schedule.

The central challenge in designing iterative HE applications lies in the trade-off between the initial modulus chain depth, L, and the bootstrapping frequency. A larger L allows more epochs to be executed before a bootstrap is needed, thus reducing the total number of costly bootstrap operations. However, the latency of all homomorphic operations increases non-trivially with L.

DOI: 10.54254/2755-2721/2025.31672

4.1. Performance modeling of the non-linear trade-offs

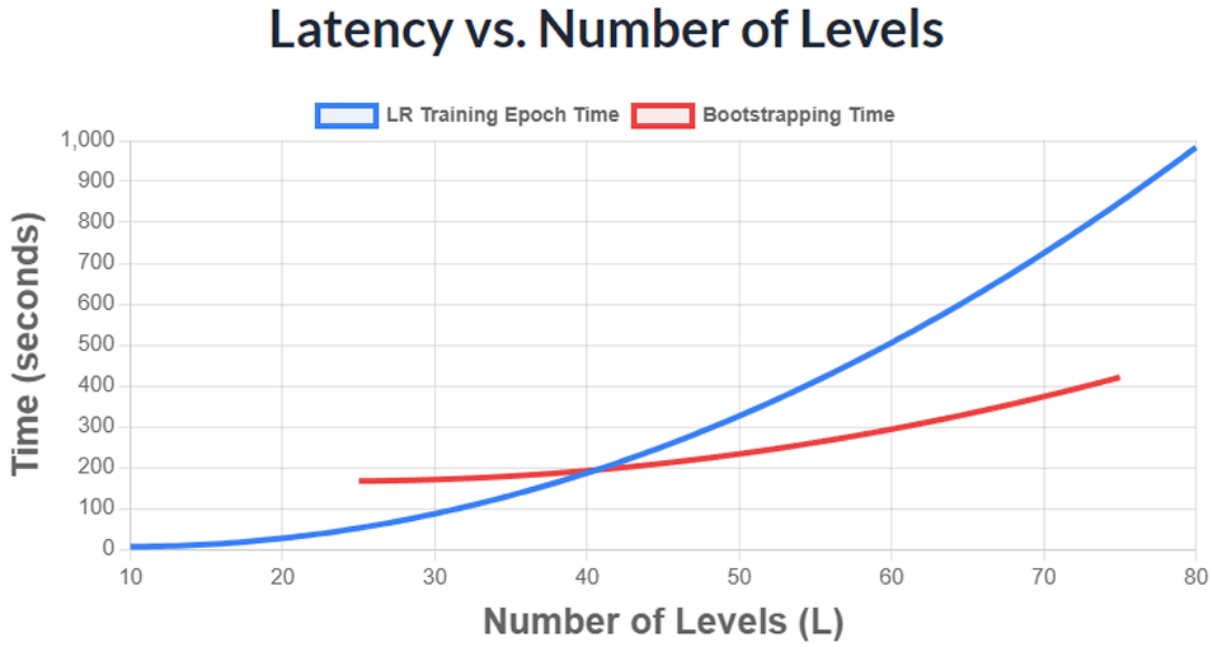

Figure 3. The non-linear trade-off between initial level depth (L) and operation latency. As L increases, both epoch time and bootstrap time grow quadrati cally, with epoch time growing at a faster rate

Crucially, this relationship was nonlinear. We modeled the latencies of the key operations as quadratic functions of the number of levels based on empirical profiling. This is because a larger level depth implies larger RNS moduli, which in turn increases the complexity of the core Number Theoretic Transform (NTT) and polynomial multiplication operations.

• Bootstrap Time (T_{boot}): The time for a single bootstrap is a quadratic function of the initial level depth L: $T_{boot}(L) \approx c_b L_2 + d_b L + e_b$.

• Epoch Time (T_{epoch}): The time for a single epoch is a quadratic function of the remaining levels, Lrem, at the start of that epoch: $T_{\text{epoch}}(L) \approx c_e L_{rem}^2 + d_e L_{rem} + e_e$.

This leads to a fascinating dynamic, as illustrated in Figure 3. At low values of L, the bootstrap time dominates. However, as L increases, the epoch time grows more rapidly than the bootstrap time, eventually surpassing it. This complex, nonlinear relationship invalidates simplistic strategies. The true optimum lies between these two values.

Thus, the optimization problem is to find a pair $(L, \mathscr{B})$ that minimizes the total time: $T_{total}\left(L, \mathscr{B}\right) = \sum_{i=1}^{E} T_{\text{epoch}}(L_{rem,i}) + \left|\mathscr{B}\right| \cdot T_{\text{epoch}}(L)$

where $L_{rem,i}$ is the remaining level depth at the start of epoch i .

DOI: 10.54254/2755-2721/2025.31672

4.2. Design space exploration via ABS

Algorithm 1 Adaptive Bootstrapping Scheduler (ABS)

1: Input: Total epochs E, depth per epoch Depoch, range of levels to test $[L_{min}, L_{max}]$, latency models $[M_{\text{epoch}}, M_{\text{boot}}]$.
2: Output: Optimal initial level L^*, optimal schedule $\mathcal{B}^*$.
3: $T_{min} \leftarrow \infty$
4: $L^* \leftarrow L_{min}$, $\mathcal{B}^* \leftarrow \emptyset$
5: for L in $[L_{min}, L_{max}]$ do
6: Calculate max epochs before bootstrap:
7: $k_{max} \leftarrow \lfloor (L - L_{guard})/D_{epoch} \rfloor$
8: if $k_{max} < 1$ then
9: continue
10: end if
11: Calculate number of bootstraps needed:
12: $N_{boot} \leftarrow \lceil E/k_{max} \rceil - 1$
13: Estimate average epoch time $\bar{T}_{epoch}(L)$ over k_{max} epochs using M_{epoch}.
14: Estimate bootstrap time $T_{boot}(L)$ using M_{boot}.
15: $T_{est} \leftarrow E \cdot \bar{T}_{epoch}(L) + N_{boot} \cdot T_{boot}(L)$
16: if $T_{est} < T_{min}$ then
17: $T_{min} \leftarrow T_{est}$
18: $L^* \leftarrow L$
19: Construct schedule $\mathcal{B}^*$ with bootstraps at epochs $\{k_{max}, 2k_{max}, \dots\}$.
20: end if
21: end for
22: return L^*, $\mathcal{B}^*$

Manually determining the optimal operating point in this trade-off space is tedious and error-prone. The ABS algorithm, as detailed in Algorithm 1, automates this process. It performs a structured exploration of the design space to identify the configuration $(L^*, \mathcal{B}^*)$ that yields the minimum estimated execution time.

The algorithm's strategy is based on the insight that for a given initial level L, the optimal schedule is a uniform one that maximally utilizes the available depth before each bootstrap. This simplifies the complex 2D search for (L, $\mathcal{B}$) into an efficient 1D search over L. As detailed in Algorithm 1, the process iterates through a practical range of initial levels $[L_{min}, L_{max}]$. For each candidate L, it calculates the maximum number of consecutive epochs, kmax = $k_{max} \leftarrow \lfloor (L - L_{guard})/D_{epoch} \rfloor$. Using the profiled latencies from our performance model, we estimated the total time for a full training run. The configuration $(L^*, \mathcal{B}^*)$ that yields the minimum time is selected.

4.3. Precision-aware strategy: bootstrapping the gradient

A critical decision in our framework is which intermediate value to use for bootstrapping. Our empirical results reveal a counter-intuitive but optimal strategy: it is more advanta geous to bootstrap the computed gradient rather than the model weights.

DOI: 10.54254/2755-2721/2025.31672

Our optimal strategy, illustrated in Figure 4, applies boot strapping directly to the fully computed gradient ciphertext,∇_t , before it is used to update the weights of the model.

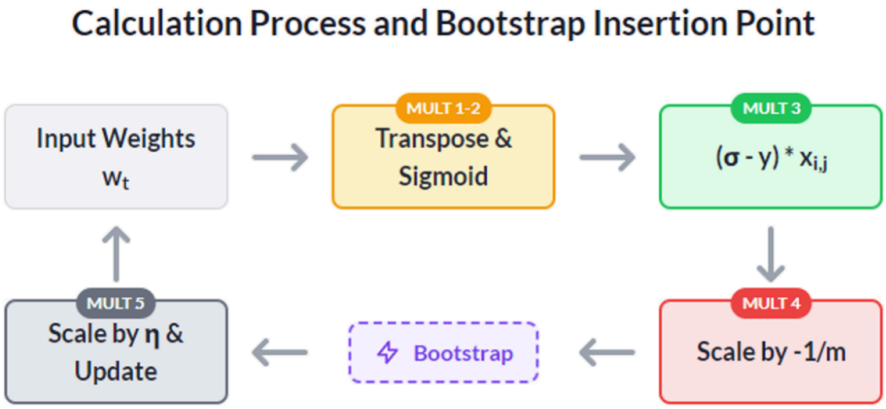

Figure 4. Our precision-aware strategy: bootstrapping the gradient. The op eration is applied after the full gradient is computed, but before it is used to update the weights. The subsequent multiplication by the learning rate η effectively dampens the bootstrapping error

The rationale for this strategy is rooted in an analysis of error propagation, supported by insights from noisy optimization algorithms such as DP-SGD [21,22]. Let the bootstrapping operation introduce an additive error δ_{boot} .

• Suboptimal Strategy (Bootstrap Weights): If we boot strap the weights w_t to get $w_t^{'} \approx w_t + \delta_{boot}$, this error propagates through the entire non-linear gradient com putation $\nabla L(w_t^{'})$, leading to a complex and potentially amplified error in the final update.

• Optimal Strategy (Bootstrap Gradient): In our ap proach, we compute the gradient ∇_t and then bootstrap it, resulting in $\nabla_t^{'} \approx \nabla_t + \delta_{boot}$. The weight update then becomes:

$$w_{t+1} = w_t - \eta(\nabla_t + \delta_{boot}) = (w_t - \eta\nabla_t) - \eta\delta_{boot}$$

The crucial insight is that the bootstrapping error δ_{boot} is multiplied by the learning rate η, which is typically small (e.g., 0.1). This multiplication dampens or dilutes the error before it is applied to the weights. This significantly reduces the perturbation of the model's convergence path, leading to higher precision in the final trained weights.

5. Experimental setup

HE Library and Parameters: We implemented our en crypted LR training using the Poseidon library [23], which is a modern C++ library for homomorphic encryption. Poseidon was chosen for its flexible API for managing cryptographic parameters and bootstrapping, which facilitated the implemen tation of our adaptive scheduler. We used the RNS-CKKS scheme. The cryptographic and scheme parameters used in our experiments are listed in Table 1. These parameters were chosen to provide at least 128 bits of security.

DOI: 10.54254/2755-2721/2025.31672

Table 1. Rules to format sections cryptographic and scheme parameters

Parameter	Value
Security Level	128-bit
Polynomial Degree (N)	2^{16}
Ciphertext Modulus (Q)	Variable (determined by baselines/ABS)
Prime Sizes	32 for data, 60 for special
Initial Scale	2^{32}

Dataset and Model: We evaluated our method on two public datasets from the UCI repository: the Breast Cancer Wisconsin (Diagnostic) dataset [24], which contains 569 sam ples and 30 features, and the Pima Indians Diabetes dataset [25], which contains 768 samples and 8 features. For both datasets, the LR model was trained for 20 epochs with a learning rate $\eta = 0.05$. Among them, 0.05 can not only avoid the slow convergence caused by an extremely small learning rate, but also weaken the interference of errors. At the same time, it ADAPTS to the LR gradient characteristics. The LR gradient value is relatively small, and this learning rate can ensure that the weight update is effective within 20 iterations, avoiding oscillations or insufficient convergence. LR is a linear model. Twenty iterations are sufficient for the model to converge on the experimental dataset, without the need for additional iterations to increase the computational cost of HE.

Baselines for Comparison: We compare our ABS method against two static strategies representing the extremes of the design space:

• Minimum Latency Strategy (MLS): This strategy pri oritizes the speed of individual operations. It uses the minimum possible level depth (L) required for a single epoch and performs bootstrapping after each epoch.

• Minimum Bootstrapping Strategy (MBS): This strat egy aims to minimize the number of bootstraps. It uses a large initial level depth (L) to accommodate a fixed, large number of epochs (five in our experiments) before bootstrapping.

Hardware: All experiments were run on a server running Ubuntu 22.04 LTS, equipped with an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz (using a single thread) and 256GB of RAM.

6. Evaluation

We conducted a comprehensive set of experiments to evaluate the effectiveness of the proposed ABS algorithm. We aimed to answer the following key questions:

1) How does ABS perform against state-of-the-art static scheduling strategies in terms of training time across different datasets?

2) How accurately does ABS preserve the ideal training trajectory compared to baselines, measured by a direct vector similarity metric?

3) What is the underlying reason for the performance advantage of ABS, as revealed by an analysis of time composition?

DOI: 10.54254/2755-2721/2025.31672

6.1. Training time analysis

We first evaluated the end-to-end training time on two stan dard UCI benchmarks: the Breast Cancer Wisconsin (BCW) and Pima Indians Diabetes (PID) datasets. The precise execu tion times for each strategy are detailed in Table 2, and Figure 5 provides a visual summary.

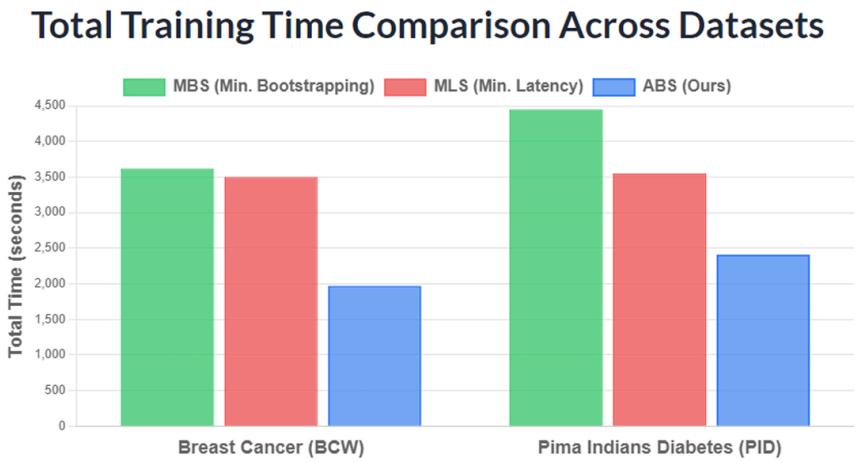

Figure 5. Total training time comparison across two datasets. ABS consistently outperforms both static baselines

Table 2. Total training time (in seconds) across datasets

Method	Breast Cancer (BCW)	Pima Diabetes (PID)
MBS (Min. Bootstrapping)	3501	3552
MLS (Min. Latency)	3617	4451
ABS (Ours)	1974	2412

The findings were consistent and compelling across both datasets. ABS consistently achieved the lowest training time. As shown in the table, the ABS outperforms the next-bestbaseline (MLS) by 1.77x on the BCW dataset and 1.47x on the PID dataset.

The figure visually illustrates this false dichotomy of static approaches. The MBS is consistently the slowest owing to the high cost of its large-parameter operations, whereas the MLSishampered by its excessive bootstrapping overhead. Our ABS algorithm successfully navigates this trade-off, finding a "sweet spot" that yields superior end-to-end performance, demonstrating that its advantage is not an artifact of a single dataset but a generalizable solution to the problem.

6.2. Precision and convergence path analysis

A key challenge in approximate HE is that cryptographic noise can perturb the training process, causing the weight vector of the model to deviate from its ideal trajectory. To measure this deviation directly, we compare the homomor phically trained weight vector, whe, with the plaintext-trained ground-truth vector, wplain, at the end of each epoch. We used Cosine Similarity as our metric:

$$Similarity = \frac{w_{he} \cdot w_{plain}}{\| w_{he} \| \| w_{plain} \|}$$

DOI: 10.54254/2755-2721/2025.31672

This metric, valued between -1 and 1, measures the directional alignment of the two vectors. A score closer to 1 indicates that the HE-based training follows the correct convergence path, even if the magnitude of the weights is slightly different.

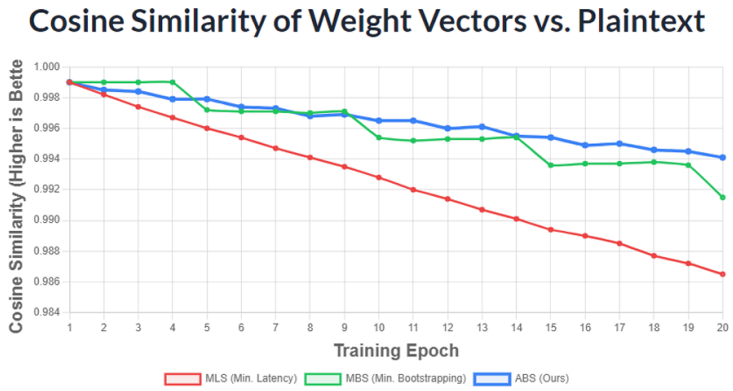

Figure 6. Cosine similarity of HE-trained weight vectors vs. plaintext-trained vectors over epochs on the BCW dataset. A higher value indicates better precision and a more accurate convergence path

Figure 6 shows the evolution of cosine similarity over 20 epochs on the PID dataset. The results clearly demonstrate the superiority of our proposed precision-aware strategy. The ABS curve remained consistently higher than both baselines, indicating a more stable and accurate convergence trajectory.

6.3. Analysis of time composition

To further investigate the source of the performance gains, we analyze the composition of the total execution time for each strategy on the PID dataset. Figure 7 visualizes the proportion of time spent on core HE operations versus the time spent on bootstrapping.

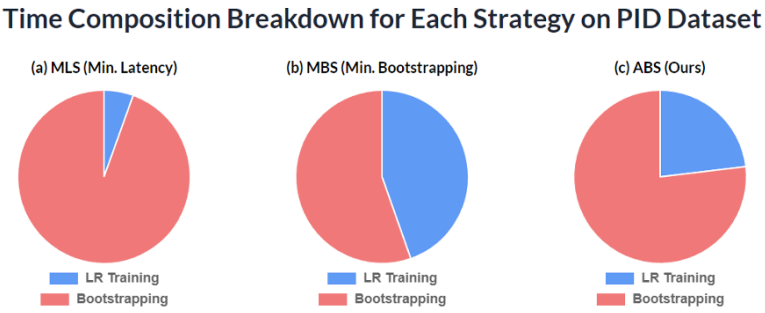

Figure 7. Time composition breakdown for each strategy on the BCW dataset. (a) MLS is dominated by bootstrapping cost. (b) MBS is dominated by HE operation cost. (c) ABS achieves a much better balance

The results are highly illuminating. For the MLS (Figure. 7a), the total time is overwhelmingly dominated by bootstrapping (94.5%). For the MBS (Figure. 7b), the situation is reversed, with the now-sluggish HE operations becoming the main bottleneck (44.6%). For our ABS (Figure. 7c), we observe a much healthier balance. This analysis visually confirms that ABS successfully navigates

DOI: 10.54254/2755-2721/2025.31672

the trade-off, finding a balanced configuration that avoids the inefficient extremes of the static strategies.

7. Conclusion

In this paper, we challenged the prevailing use of static bootstrapping schedules in iterative HE applications. We intro duced ABS, an algorithm that automates the co-design of cryp tographic parameters and a dynamic bootstrapping schedule to optimize performance and precision. Our results, validated across multiple datasets, show that this application-aware approach yields substantial benefits, consistently accelerating training by up to 1.77x while simultaneously improving model precision compared to strategies representing the extremes of the design space. This work advocates for a shift in perspective: instead of treating HE as a black-box replacement for plaintext operations, we should build application logic that is cognizant of the underlying cryptographic constraints and opportunities. By doing so, we can significantly narrow the gap between the promise of privacy-preserving machine learning and its practical, efficient realization.

References

[1] H. I. Fawaz, M. Aledhari, Z. Shukur, M. A. Al-garadi, and A. Al-kaff, "A survey on privacy-preserving machine learning, " IEEE Access, vol. 9, pp. 124338–124361, 2021.
[2] A. C. Yao, "Protocols for secure computations, " in 23rd Annual Sym posium on Foundations of Computer Science (sfcs 1982). IEEE, 1982, pp. 160–164.
[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis, " in Theory of cryptography conference. Springer, 2006, pp. 265–284.
[4] C. Gentry, "Fully homomorphic encryption using ideal lattices, " in Pro ceedings of the 41st annual ACM symposium on Theory of computing, 2009, pp. 169–178.
[5] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homo morphic encryption without bootstrapping, " in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12). ACM, 2012, pp. 309–325.
[6] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from LWE and RLWE, and their applications, " Cryptology ePrint Archive, Report 2011/530, 2011.
[7] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full rns variant of approximate homomorphic encryption, " in International Conference on Selected Areas in Cryptography. Springer, 2018, pp. 347–368.
[8] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: Machine learning on encrypted data, " in International Conference on Information Security and Cryptology. Springer, 2012, pp. 1–21.
[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, " in International conference on machine learning (ICML). PMLR, 2016, pp. 201–210.
[10] J. H. Cheon and K. Han, "Bootstrapping for approximate homomorphic encryption, " in Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). Springer, 2020, pp. 360–384.
[11] M. Kim, Y. Song, S. Lee, and J. H. Cheon, "Logistic regression model training over homomorphically encrypted data, " BMC medical genomics, vol. 11, no. 4, pp. 29–38, 2018.
[12] K. Han, S. Park, S. Lee, J.-H. Seo, and J. H. Cheon, "Efficient logistic regression on large encrypted data, " in Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, 2019, pp. 209–218.
[13] J. L. H. Crawford, E. Dufour-Sans, and C. T. E. R. Schmidt, "A low depth homomorphic circuit for logistic regression model training, " IACR Cryptology ePrint Archive, Report 2023/177, 2023.
[14] V. Lyubashevsky, C. Peikert, and O. Regev, "Ideal lattices and learning with errors over rings, " in Annual international conference on the theory and applications of cryptographic techniques. Springer, 2010, pp. 1–23.
[15] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers, " in international conference on the theory and application of cryptology and information security. Springer, 2017, pp. 409–437.

DOI: 10.54254/2755-2721/2025.31672

[16] J. H. Cheon, D.-g. Lee, D. Lee, and J. Eom, "Numerical method for comparison on homomorphically encrypted numbers, " in International Conference on Selected Areas in Cryptography. Springer, 2019, pp. 415–439.
[17] S. Cheon, Y. Lee, D. Kim, K. Roh, Y. Son, D.-g. Kim, K. Han, and J. H. Cheon, "Dacapo: Automatic bootstrapping management for efficient fully homomorphic encryption, " in 32nd USENIX Security Symposium (USENIX Security 23). USENIX Association, 2023, pp. 103–120.
[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data, " in Artificial intelligence and statistics. PMLR, 2017, pp. 1273 1282.
[19] B. Reagen, U. Gupta, O. Pentakalos, M. Rastegari, F. Ghaffari, G. E. Ko, G.-Y. Wei, and D. Brooks, "Cheetah: A lightweight, fast, and robust framework for securely outsourcing linear algebra, " in 2021 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2021, pp. 1018–1032.
[20] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus, " 2021, pp. 311–336.
[21] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy, " in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.
[22] S. Lee, C. Park, M. Kim, and J. H. Cheon, "Privacy-preserving federated learning with lightweight and efficient homomorphic encryption, " IEEE Transactions on Information Forensics and Security, vol. 17, pp. 1552 1565, 2022.
[23] CryptoLab, "Poseidon: A c++ homomorphic encryption library, " https: //github.com/CryptoLab/Poseidon, 2022.
[24] D. Dua and C. Graff, "UCI machine learning repository, " 2017. [Online]. Available: http: //archive.ics.uci.edu/ml
[25] J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the adap learning algorithm to forecast the onset of diabetes mellitus, " American Medical Informatics Association, pp. 261–265, 1988.