Published: 19 November 2025

Citation in BibTeX format

CCS '25: ACM SIGSAC Conference on Computer and Communications Security
October 13 - 17, 2025
Taipei, Taiwan

Conference Sponsors:
SIGSAC

POSTER

Poster: Scalable Privacy-Preserving Linear Regression Training via Homomorphic Encryption

YENA CHO, Sungkyunkwan University, Seoul, South Korea

HYOUNGSCHICK KIM, Sungkyunkwan University, Seoul, South Korea

Open Access Support provided by:

Sungkyunkwan University

Poster: Scalable Privacy-Preserving Linear Regression Training via Homomorphic Encryption

Yena Cho
Sungkyunkwan University
Suwon, Republic of Korea
yena0511@skku.edu

Hyoungshick Kim
Sungkyunkwan University
Suwon, Republic of Korea
hyoung@skku.edu

Abstract

Linear regression is a fundamental model in data analysis, but training it over encrypted data remains expensive due to the overhead of homomorphic encryption (HE). We present PP-LR, an end-to-end CKKS-based training protocol for linear regression that supports encrypted gradient descent with feature-level parallelism and conditional bootstrapping. Compared to a standard HE implementation, it trains up to 15.7 times faster while maintaining accuracy within 0.2% of plaintext models on four real-world datasets.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**; **Public key (asymmetric) techniques**.

Keywords

Homomorphic Encryption; Privacy-Preserving Machine Learning

ACM Reference Format:
Yena Cho and Hyoungshick Kim. 2025. Poster: Scalable Privacy-Preserving Linear Regression Training via Homomorphic Encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3719027.3760740

1 Introduction

Linear regression is fundamental to healthcare, finance, and scientific research, yet privacy regulations such as GDPR and HIPAA increasingly restrict access to sensitive training data. In privacy-critical domains—e.g., medical diagnosis and financial risk modeling—predictive tasks often require multivariate regression over high-dimensional data. Homomorphic encryption (HE) enables computation directly on encrypted data, and the CKKS scheme [2] supports approximate arithmetic over real numbers, making it well-suited for encrypted machine learning.

However, securely training linear regression models under HE remains challenging due to the high computational cost. Most practical HE frameworks, such as Concrete ML (https://github.com/zama-ai/concrete-ml), therefore support only inference. Normal equation approaches [1] either decrypt the matrix inverse for plaintext computation, undermining fully encrypted computation, or encrypt only part of the features, providing incomplete protection by design.

Gradient descent over encrypted data involves repeated matrix-vector operations that consume multiplicative depth, limiting scalability. To avoid bootstrapping complexity, existing methods [3, 4] rely on online settings where ciphertexts are refreshed through re-encryption. While Prantl *et al.* [4] also explored offline settings, the lack of optimization resulted in impractical long runtimes, making online modes preferable despite their communication overhead. Additionally, some approaches (e.g., [3]) support only univariate regression, which restricts their real-world applicability. Under our single-server assumption, our protocol enables efficient offline training of multivariate models entirely over encrypted data, allowing practical deployment without additional trust assumptions.

To address these limitations, we present PP-LR, a CKKS-based training protocol that ensures full data encryption while achieving practical scalability for multivariate linear regression. PP-LR enhances performance by exploiting the Single Instruction, Multiple Data (SIMD) capability of CKKS for *feature-level parallelism* and applying *conditional bootstrapping* based on the available multiplicative depth, thereby reducing unnecessary computation and mitigating the runtime overhead typically associated with offline settings, all without compromising accuracy.

Our main contributions are: (1) PP-LR, a secure multivariate linear regression training protocol over CKKS that enables encrypted gradient descent with two key optimizations—feature-level parallelism via SIMD packing and conditional bootstrapping guided by multiplicative depth; and (2) empirical evidence that PP-LR achieves up to 15.7× speedup over a standard HE baseline with less than 0.2% accuracy loss on four real-world datasets, demonstrating both scalability and practicality.

2 Preliminaries

2.1 Linear Regression with Gradient Descent

Linear regression models the relationship between input features and continuous targets. For multivariate regression with input vectors $\mathbf{x}_i$ and targets y_i, we learn the linear model $\hat{y}_i = \mathbf{x}_i^\top \mathbf{w} + b$, where $\mathbf{w} \in \mathbb{R}^f$ is the weight vector and $b \in \mathbb{R}$ is the bias.

While normal equations provide closed-form solutions, they rely on operations not easily supported in encrypted computation. Gradient descent uses only additions and multiplications, allowing end-to-end computation over ciphertexts in HE schemes, though computational optimizations are necessary to minimize overhead.

We minimize the mean squared error loss $L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$ using gradient descent updates:

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} L, \quad b := b - \alpha \nabla_b L \tag{1}$$

CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3760740

where α is the learning rate and gradients are given by:

$$\nabla_{\mathbf{w}} L = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\mathbf{x}_i, \quad \nabla_b L = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{2}$$

2.2 CKKS Scheme

The CKKS scheme [2] is a fully HE scheme that supports approximate arithmetic operations on encrypted real numbers, enabling addition and multiplication on ciphertexts without decryption. CKKS supports SIMD operations that pack multiple plaintext values into a single ciphertext for parallel processing, significantly improving computational efficiency on large-scale data.

During homomorphic operations, particularly multiplication, noise accumulates and may lead to incorrect decryption. The multiplicative depth defines the maximum number of consecutive multiplications a ciphertext can support before noise exceeds the decryption threshold. To address noise accumulation, CKKS provides bootstrapping, which refreshes a ciphertext by resetting its multiplicative depth without decryption, enabling further computation.

3 PP-LR Design

HE-based *multivariate* linear regression approaches face two major challenges when scaling to high-dimensional datasets and large sample sizes: *sequential feature processing* and *naive bootstrapping*. Sequential approaches process features one by one, severely limiting parallelism. Moreover, naive bootstrapping refreshes ciphertexts at every epoch, regardless of the consumed multiplicative depth, resulting in unnecessary computational overhead.

PP-LR addresses these challenges with two key optimizations: *feature-level parallelism* and *conditional bootstrapping*.

Feature-level parallelism exploits CKKS SIMD capabilities to process all F features concurrently, overcoming the sequential bottleneck inherent in multivariate models. It is implemented by encoding all sample values of a single feature into a single ciphertext. When the number of samples S exceeds the slot capacity P, the feature is split into $B = \lceil S/P \rceil$ ciphertext blocks. Since the blocks are independent and SIMD executes the same operation simultaneously across all ciphertext slots, up to $F \cdot B$ multiplicative operations and bootstrapping steps can be performed in parallel using goroutines (lines 5–7, 9–11, 14–19 in Algorithm 1).

Conditional bootstrapping eliminates unnecessary refresh operations by triggering bootstrapping only when the remaining depth is insufficient for upcoming computations. PP-LR supports end-to-end training, where model weights are updated across epochs without refresh, unlike input ciphertexts, leading to cumulative depth consumption. Since the CKKS multiplication output level is determined by the lower-level operand, parameters must maintain sufficient levels before each computation. Our mechanism triggers bootstrapping only when both weights and bias have fewer than $3 + D_{\min}$ remaining levels (lines 5–7), as each epoch consumes three levels through three `Mul` operations. Here, $D_{\min}$ is the minimum input level that causes bootstrapping to be executed, determined by the CKKS parameters. In our setting, $D_{\min} = 1$, so bootstrapping is triggered when the remaining level falls below 4.

This policy is generalizable: for multiplicative depth D, the number of epochs that can be processed without bootstrapping is $E_{nb} = \lfloor (D - D_{min})/3 \rfloor$, and bootstrapping occurs at epochs $E_{nb} \cdot i + 1$ for $i = 1, 2, 3, \ldots$. In contrast, naive approaches bootstrap at every epoch when levels drop below the minimum threshold. Since the first epoch reduces depth from D to $D - 3$, naive bootstrapping is required starting from epoch 3, resulting in $E - 2$ total bootstraps for E epochs, under our CKKS parameter setting. For example, with $D = 10$ and $E = 10$, our mechanism performs only 3 bootstraps compared to 8 in the naive approach.

Algorithm 1 PP-LR Training

1: **Input:** Encrypted features $C_{in} \in (\mathcal{R}_q)^{F \times B}$, labels $Y_{in} \in \mathcal{R}_q^B$, learning rate α, epochs E
2: **Output:** Trained weights $\mathbf{w} \in \mathcal{R}_q^F$, bias $\mathbf{b} \in \mathcal{R}_q$
3: **Notation:** $\mathcal{R}_q$: CKKS ciphertext space; (f), (b): feature/block indices; `InnerSum`: slot aggregation
4: **for** $e = 0$ to $E - 1$ **do**
5: **if** $\text{Depth}(\mathbf{w}) < 3 + D_{min}$ **and** $\text{Depth}(\mathbf{b}) < 3 + D_{min}$ **then**
6: $\mathbf{w}, \mathbf{b} \leftarrow \text{Bootstrap}(\mathbf{w}), \text{Bootstrap}(\mathbf{b})$ *// Parallel*
7: **end if**
8: **for** $b = 0$ to $B - 1$ **do**
9: **for** $f = 0$ to $F - 1$ *// Parallel across features* **do**
10: $Xw^{(f,b)} \leftarrow \texttt{Mul}(C_{in}^{(f,b)}, \mathbf{w}^{(f)})$
11: **end for**
12: $Y_{err}^{(b)} \leftarrow \texttt{Sub}(\texttt{Add}(\sum_f Xw^{(b)}, \mathbf{b}), Y_{in}^{(b)})$
13: **end for**
14: **for** $f = 0$ to $F - 1$ *// Parallel across features* **do**
15: **for** $b = 0$ to $B - 1$ **do**
16: $W_{grad}^{(f,b)} \leftarrow \texttt{Mul}(C_{in}^{(f,b)}, Y_{err}^{(b)})$
17: **end for**
18: $\mathbf{w}_f \leftarrow \mathbf{w}_f - \frac{2\alpha}{n} \cdot \texttt{InnerSum}(W_{grad}^{(f,\cdot)})$
19: **end for**
20: $\mathbf{b} \leftarrow \mathbf{b} - \frac{2\alpha}{n} \cdot \texttt{InnerSum}(Y_{err})$
21: **end for**
22: **return** $\mathbf{w}, \mathbf{b}$

4 Experiments

4.1 Experimental Setup

All experiments were performed on Intel Xeon Gold 6230 CPUs (2.10 GHz) with 80 vCPUs and 567 GB of memory. We used the CKKS implementation from Lattigo (https://github.com/tuneinsight/lattigo), a Go-based HE framework optimized for goroutine-based parallelism. CKKS parameters were set to 32,768 slots, a log scale factor of 40, and a multiplicative depth of 9.

We evaluated PP-LR on four real-world multivariate datasets: Diabetes, Real Estate Price Prediction, Student Performance, and Year Prediction MSD (YPM), sourced from scikit-learn, Kaggle, and the UCI Machine Learning Repository. All datasets were standardized, with string-valued or noisy features excluded. Table 1 summarizes dataset characteristics. To reduce runtime, we used a representative subset of the YPM dataset with 100,000 samples and 30 features. Despite being a subset, its scale exceeds that used in existing studies based on the gradient descent method [3], thereby demonstrating the scalability of the proposed method. All models were trained for 5 epochs in both plaintext and encrypted settings.

Table 1: Description of datasets.

Dataset	# Samples	# Features
Diabetes	442	9
Real Estate	414	6
Student	10,000	4
YPM	515,345	90

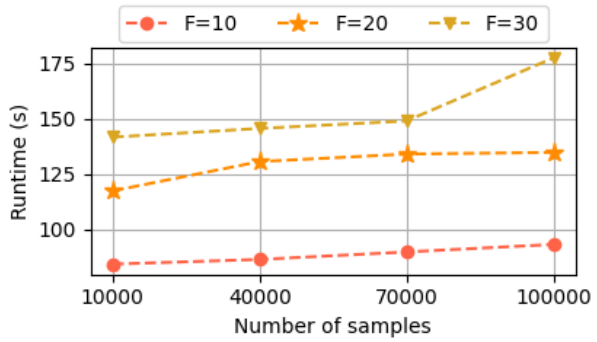

Figure 1: Training runtime vs. number of samples for different feature dimensions using the YPM dataset.

4.2 Experimental Results

4.2.1 Runtime. We evaluated PP-LR's scalability by measuring training time across different dataset sizes, running each experiment 50 times under identical settings. Figure 1 presents median training times on the YPM dataset with varying sample counts (10,000–100,000) and feature dimensions (F = 10, 20, 30). Overall, runtime increases roughly linearly with both parameters; for F = 10, it rises from 84 to 93 seconds, while for F = 30, it grows more steeply from 142 to 178 seconds due to heavier homomorphic computation. An exception occurs at F = 30 with 100,000 samples, where runtime deviates sharply because of thread scheduling bottlenecks from intensive parallelism, adding 20–40 seconds of overhead. Such delays become more frequent as workloads grow, introducing runtime variance and indicating that scalability is constrained not only by algorithmic complexity but also by scheduling instability under high parallel workloads. Although we report median values to reduce the impact of noise, achieving stable performance in encrypted training requires co-optimizing cryptographic operations and runtime behavior.

To demonstrate PP-LR's effectiveness, we compare against a standard implementation without our optimizations. The baseline processes features sequentially and triggers bootstrapping immediately upon reaching minimum depth. Using 10,000 samples and 30 features, the standard implementation requires 2,231 seconds compared to PP-LR's 142 seconds, achieving 15.7 times speedup.

4.2.2 Regression Result. We evaluated the accuracy of PP-LR by comparing its mean squared error (MSE) and coefficient of determination (R^2) with those of plaintext computation. MSE and R^2 are standard metrics for assessing regression model performance. All experimental parameters were kept identical between encrypted and plaintext settings to ensure a fair comparison.

Table 2: PP-LR vs. plaintext linear regression performance.

Dataset	PP-LR		Plaintext	
	MSE	R^2	MSE	R^2
Diabetes	0.5457	0.3751	0.5457	0.3751
Real Estate	0.4276	0.5615	0.4272	0.5624
Student	0.0124	0.9877	0.0124	0.9877
YPM	0.7788	0.1958	0.7772	0.1975

Table 3: Ablation study of PP-LR.

Model	Runtime (s)
PP-LR	128.348
w/o Feature parallelism	1376.814
w/o Conditional bootstrapping	351.103

As shown in Table 2, PP-LR achieves nearly identical performance to plaintext computation across all datasets. For the Diabetes and Student datasets, PP-LR achieves identical MSE and R^2 values as the plaintext. The Real Estate dataset shows a minimal MSE difference (0.4276 vs. 0.4272), while the YPM dataset maintains comparable performance with a slightly higher MSE (0.7788 vs. 0.7772). The maximum accuracy difference across all datasets remains within 0.2%, confirming that PP-LR preserves model utility while ensuring privacy through HE.

4.2.3 Ablation Study. We conducted an ablation study to evaluate the effectiveness of the optimization techniques in PP-LR using the YPM dataset (50,000 samples, 20 features). Median runtime results are summarized in Table 3. Without feature parallelism, runtime increases from 128.4 seconds to 1376.8 seconds (10.7 times slowdown) as feature-related computations and bootstrapping operations must be processed sequentially. Without conditional bootstrapping, runtime increases to 351.1 seconds (2.7 times slowdown) since PP-LR applies bootstrapping selectively based on multiplicative depth requirements, while the baseline performs unnecessary bootstrapping. These results show that both optimization techniques are crucial for achieving efficient encrypted training, with feature parallelism providing the most significant performance improvement.

ACKNOWLEDGMENTS

This work is supported by the IITP grants (No. RS-2024-00419073, IITP-2025-RS-2022-II221199, RS-2022-II220688, and RS-2024-00436934).

References

[1] Bijiao Chen and Xianghan Zheng. 2022. Implementing Linear Regression with Homomorphic Encryption. *Procedia Computer Science* (2022).
[2] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*.
[3] Gianluca Dini. 2024. On Implementing Linear Regression on Homomorphically Encrypted Data: A Case-Study. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES)*.
[4] Thomas Prantl, Simon Engel, Lukas Horn, Dennis Kaiser, Lukas Iffländer, André Bauer, Christian Krupitzer, and Samuel Kounev. 2023. Performance Impact Analysis of Homomorphic Encryption: A Case Study Using Linear Regression as an Example. In *Proceedings of the 18th International Conference on Information Security Practice and Experience (ISPEC)*.