RACE-HE: ROTATION-AWARE COMPRESSION FOR EFFICIENT PRIVATE INFERENCE WITH HOMOMORPHIC ENCRYPTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Homomorphic encryption (HE) enables privacy-preserving machine learning inference by allowing an untrusted server to compute over encrypted client data. However, its practical deployment remains constrained by the high cost of homomorphic linear layers, particularly the ciphertext rotations required for packed matrix multiplication. This paper presents **RACE-HE**, a model compression framework for HE-based private inference that restructures weight matrices before encrypted evaluation. RACE-HE approximates selected weight matrices using a rotation-aware diagonal-sparse low-rank (DSLR) decomposition, $W \approx S + AB$. RACE-HE exploits the property that S and B can share input rotations, allowing the diagonal term S to recover information lost by the low-rank approximation while limiting additional input rotations. It then applies rotation-aware diagonal pruning, jointly removing packed diagonals that share an input rotation. On ViT-Tiny with DermaMNIST, RACE-HE reduces rotations by 75.4% with a 1.1 percentage point drop in accuracy. On GPT-2 and Llama 3.1-8B, RACE-HE reduces rotations by 29.0% and 34.8%, respectively, with perplexity increases of 2.5 and 1.8, achieving better quality–rotation trade-offs than low-rank and sparse-plus-low-rank baselines. When combined with structurally auto-tuned BSGS in OpenFHE, RACE-HE reduces end-to-end ViT-Tiny latency by 24.2%. By restructuring plaintext weight matrices, RACE-HE complements existing HE execution optimizations.

1 INTRODUCTION

Fully homomorphic encryption (FHE) enables untrusted servers to evaluate machine learning models directly over encrypted data, offering a mathematically rigorous foundation for privacy-preserving machine learning (PPML) (Cheon et al., 2017; Brakerski, 2012; Fan & Vercauteren, 2012; Brakerski & Vaikuntanathan, 2014; Brakerski et al., 2014). In a standard client–server PPML setup, FHE allows the client's inputs and intermediate activations to remain encrypted during server-side computation, while the server keeps its model weights locally in plaintext. Among existing FHE schemes, CKKS (Cheon et al., 2017) is particularly well suited to privacy-preserving neural inference since it supports approximate arithmetic over real-valued data, matching the error-tolerant nature of modern neural networks. However, despite these guarantees, the substantial computational cost of homomorphic linear layers still bottlenecks practical deployment.

This bottleneck is primarily driven by large matrix multiplications (MatMuls). In typical FHE-based inference, the server evaluates linear layers by computing plaintext-ciphertext MatMuls (PC-MMs) between its plaintext model weights and encrypted activations. To efficiently evaluate these linear layers, modern FHE schemes pack multiple data elements into a single ciphertext in a single instruction, multiple data (SIMD) manner (Halevi & Shoup, 2014). While the packing layout fixes the position of each value and multiplication by a plaintext vector operates slot by slot without changing these positions, matrix multiplication requires the same input value to be multiplied by weights at different output positions. The server therefore uses ciphertext rotations to cyclically shift the encrypted values into the required positions. These rotations are among the most computationally

Code is available at `https://anonymous.4open.science/r/RACE-HE-4837`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

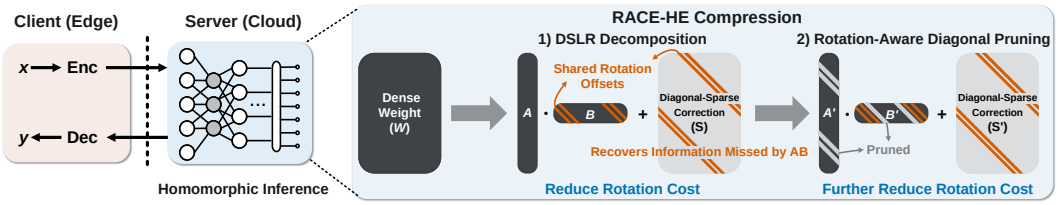


Figure 1: Overview of RACE-HE.

expensive operations in FHE, and each distinct rotation offset additionally requires a corresponding rotation key (Galois key) (Mert et al., 2022). Consequently, for the massive weight matrices in modern Transformer models, the resulting volume of rotations dominates the cost of homomorphic linear layers, rendering standard FHE deployment impractical (Lee et al., 2023a).

To reduce the cost of homomorphic matrix operations, prior work has explored several optimization directions. Algorithmic techniques such as baby-step giant-step (BSGS) (Halevi & Shoup, 2018; 2014), improved packing and rotation-encoding methods (Ma et al., 2025; Mahon & Kosieradzki, 2025), and compiler and systems frameworks (Bian et al., 2025; Chen et al., 2025) all reduce the cost of evaluating a fixed MatMul with HE. On the other hand, model compression offers a complementary way to reduce the cost by approximating the weight matrix with low-rank or sparse components. The resulting computations can then be evaluated using the same HE execution optimizations.

Conventional model compression optimizes parameter count, floating-point operations, or plaintext latency (Han et al., 2015; Yuan et al., 2023), but those objectives do not capture the cost structure of homomorphic execution, e.g., pruning individual weights leaves every packed diagonal active and saves no rotation. HE-aware low-rank (Lu et al., 2021) and pruning methods (Cai et al., 2022; Jeon et al., 2023) target rotations, but only by shrinking rank or emptying diagonals of a single operator which can discard information needed for model quality.

A useful property of HE offers an opportunity to recover some of this information with little additional computation. When two linear computations require the same rotation offset on the same input ciphertext, the server can perform that rotation once and reuse it in the next computations. Each computation applies its own plaintext weights, requiring separate plaintext multiplications but sharing the input rotation. This motivates adding a term that recovers some of the information lost by a low-rank approximation while reusing input rotations from the low-rank computation.

Based on this observation, we propose RACE-HE, a rotation-aware compression framework that restructures model weights to reduce homomorphic inference cost while preserving model quality (Figure 1). RACE-HE first replaces each dense weight matrix using a rotation-aware diagonal-sparse low-rank (DSLR) decomposition, $W \approx S + AB$. Here, AB is a low-rank approximation, and S is a diagonal-sparse correction that captures information that AB alone cannot represent, helping preserve model quality. Since S and B are multiplied by the same input ciphertext x, we construct S in a way that reuses the input rotations required by B, minimizing additional rotations. It then applies rotation-aware diagonal pruning to further reduce rotation cost. All transformations are performed offline on the server's plaintext weights, and online inference uses the resulting model.

We evaluate RACE-HE on ViT-Tiny (Touvron et al., 2021), GPT-2 (Radford et al., 2019), and Llama 3.1-8B (Grattafiori et al., 2024). On ViT-Tiny, DSLR decomposition and rotation-aware pruning jointly reduce rotations by 75.4% with only a 1.1 percentage point (pp) accuracy drop relative to the corresponding dense baseline. On GPT-2 and Llama 3.1-8B, RACE-HE reduces rotations by 29.0% and 34.8%, respectively, with perplexity increases of 2.5 and 1.8. In end-to-end OpenFHE execution, ViT-Tiny inference latency is reduced by 24.2%.

2 Background and Related Work

2.1 Cost Model for Homomorphic Inference

The CKKS scheme supports approximate arithmetic over vectors packed into ciphertext slots. In neural inference, linear layers are primarily evaluated as PC-MMs between plaintext model weights and encrypted activations. Since multiplication by a plaintext vector pairs each encrypted value with

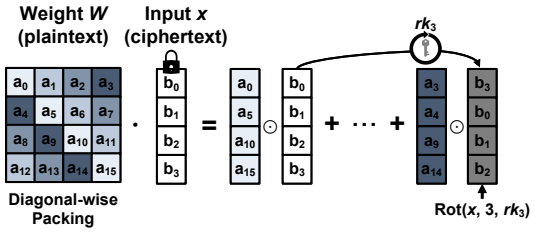


Figure 2: Example of matrix multiplication under SIMD packing.

the weight in the same slot, the server uses ciphertext rotations to cyclically shift the inputs and align them with weights at different output positions during matrix multiplication. These rotations involve key switching using rotation keys, which adds substantial computational overhead.

The diagonal method (Halevi & Shoup, 2014) organizes this computation by grouping weights that use the same rotated input, as illustrated in Figure 2. To explain this grouping, consider a square weight matrix $W \in \mathbb{R}^{q\times q}$ with its input packed into q cyclic slots. A wrapped diagonal follows a matrix diagonal and continues from the opposite edge when it reaches the matrix boundary, where each wrapped diagonal is encoded as a plaintext vector. The weights on each such diagonal align with their corresponding input values through the same rotation. Multiplying each diagonal vector by its rotated input and summing the results gives $y = \sum_{\ell=0}^{q-1} \mathbf{w}^{(\ell)} \odot \mathsf{Rot}(x, \ell)$, where x and y denote the packed encrypted input and output vectors, respectively. $\mathbf{w}^{(\ell)}$ contains the plaintext weights on wrapped diagonal ℓ, and $\odot$ denotes slotwise multiplication. $\mathsf{Rot}(x, \ell)$ cyclically shifts the input by ℓ slots, where ℓ is the rotation offset. A diagonal is active if it contains a nonzero weight. Direct evaluation requires one rotation per active nonzero offset, while offset 0 uses the original input.

For a rectangular matrix $W \in \mathbb{R}^{m\times n}$, we use $q = \min(m, n)$ logical wrapped diagonals to define the structural cost. An entry $W_{i,j}$ belongs to diagonal $\ell = (j - i) \bmod q$, where i and j are its row and column indices. This grouping wraps along the shorter matrix dimension and partitions the weights into q diagonals. Our structural cost model counts each active nonzero logical offset, giving a dense baseline of $q - 1$ rotations. Under the same grouping, factors of sizes $m \times k$ and $k \times n$, with $k \leq q$, each have at most k active diagonals.

These structural counts describe the logical diagonal groups of an operator. The realized rotation count depends not only on the active diagonal groups but also on the ciphertext layout and execution schedule, including input replication, output folding, and repacking.

2.2 Related Work

Efficient homomorphic linear transformations. BSGS scheduling reduces the number of rotations required for packed linear transformations, while subsequent work improves packing, permutation decomposition, and rotation encodings (Jiang et al., 2018; Rizomiliotis & Triakosia, 2022; Ma et al., 2025; Mahon & Kosieradzki, 2025). Compiler and system frameworks further optimize vectorization, rotation scheduling, arithmetic kernels, and distributed execution (Bian et al., 2025; Chen et al., 2025; Bae et al., 2024). These techniques optimize the evaluation of a fixed plaintext weight matrix. RACE-HE instead transforms the weight structure itself, shrinking the workload they subsequently handle.

Secure-inference protocols. Interactive two-party protocols combine HE with other secure-computation primitives. GAZELLE (Juvekar et al., 2018) and Cheetah (Huang et al., 2022) target CNN workloads, while IRON, BOLT, and BumbleBee (Hao et al., 2022; Pang et al., 2024; Lu et al., 2023) develop efficient protocols for Transformer inference, trading local homomorphic computation for online communication between the parties. Within the non-interactive setting, NEXUS (Zhang et al., 2024) optimizes the execution layer, including ciphertext compression, amortized offline–online matrix multiplication, non-linear kernels, and bootstrapping placement. RACE-HE complements both lines of work by transforming the model weights without adding online interaction.

Low-rank and structured model compression. Low-rank decomposition and pruning are widely used to reduce neural network parameter and arithmetic costs (Han et al., 2015), while recent

SVD-based LLM compression improves the quality–compression trade-off through activation- or truncation-aware decompositions (Yuan et al., 2023; Wang et al., 2025). LoSparse (Li et al., 2023) combines low-rank and sparse components to retain information that low-rank factors alone cannot represent. These approaches target plaintext execution. HE-aware structured pruning eliminates rotation-inducing patterns from dense operators. Specifically, Hunter (Cai et al., 2022) prunes weight diagonals of fully connected layers alongside convolution-specific structures in interactive BFV frameworks, while Artemis (Jeon et al., 2023) learns such diagonal sparsity during training. Whereas the SVD-based methods above approximate weights using low-rank factors alone, RACE-HE adds a diagonal-sparse correction to recover information those factors miss, choosing its structure to encourage input-rotation reuse. Unlike Hunter and Artemis, which prune diagonals of dense operators, RACE-HE jointly removes diagonals from the correction and low-rank branches when they share an input rotation, scoring each removal by its combined effect on the projection output.

The complete system and threat model, additional CKKS preliminaries, and extended discussion of related work are provided in Appendices A and B.

3 RACE-HE FRAMEWORK

RACE-HE consists of two compression stages, rotation-aware DSLR decomposition (Section 3.1) and rotation-aware diagonal pruning on the resulting factors (Section 3.2).

3.1 ROTATION-AWARE DIAGONAL-SPARSE LOW-RANK (DSLR) DECOMPOSITION

Standard low-rank compression approximates a weight matrix $W \in \mathbb{R}^{m\times n}$ as $W \approx AB$, with $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. A smaller rank k can reduce rotation cost but also degrade model quality by limiting the information that AB can represent. RACE-HE adds a diagonal-sparse correction S to compensate for this loss,

$$W \approx S + AB, \tag{1}$$

where S contains values primarily at positions that allow input-rotation reuse with B. The correction captures information that AB alone misses, reducing the loss in task quality without increasing k.

DSLR approximates the original output as $y = Wx \approx Sx + A(Bx)$, where Sx and Bx use the same encrypted input x. When both computations require a rotation of x by r slots, we compute $z_r = \mathsf{Rot}(x, r)$ once and reuse it in both. The A stage uses the intermediate result Bx as its input, so it cannot reuse these rotated copies of x and requires separate rotations.

We construct S from the residual $W - AB$, which represents the part of the weight matrix not captured by the current low-rank approximation AB. The residual entries are grouped into wrapped diagonals, with each diagonal's energy measured as the sum of its squared entries. We select diagonals with the largest residual energy, prioritizing those whose logical offsets are also used by B and allowing additional offsets only when needed to meet the reconstruction error budget. The selected residual values are copied into S, while all other entries of S are set to zero. After updating S, we refit AB to approximate the remaining matrix $W - S$ using rank-k truncated SVD, and these two updates are repeated alternately.

We use a reconstruction error budget to choose the rank and the number of diagonals retained in S. For each weight matrix W in Transformer block b, we require the decomposition to satisfy

$$\frac{\|W - S - AB\|_F^2}{\|W\|_F^2} \le 1 - \tau^{(b)}, \tag{2}$$

where $\tau^{(b)} \in (0, 1]$ is a threshold shared by the target projections in block b. A larger $\tau^{(b)}$ requires a closer approximation to the original weights. We compare fitted candidates using the structural rotation cost $R_{\mathrm{dir}} = 2(k-1) + e$, where e counts the logical offsets of S outside those used by B. Among candidates that meet the reconstruction budget, we select the one with the lowest R_{dir}, breaking ties by the total number of active wrapped diagonals and then by rank. If the search finds no candidate that meets the budget and reduces this cost relative to the dense projection, we retain the original weight matrix W. For language models, we additionally evaluate each compressed block on a held-out set and retain the entire block in dense form if the quality loss exceeds a preset tolerance.

When Sx and Bx require the same rotated input, the server computes it once and multiplies it separately by the plaintext weights of S and B. The correction therefore requires additional multiplications by S's weights, but does not repeat the shared input rotations. Since Sx can be computed in parallel with Bx and added to $A(Bx)$, the correction adds no sequential multiplication stage to the low-rank branch. For the two-stage low-rank branch, we use split-scale encoding with $\Delta_A \Delta_B = \Delta_S = \Delta_W$, where Δ_W is the encoding scale of the dense operator. This allows us to add Sx and $A(Bx)$ before a single rescale and stay within the same level budget. The trade-off is reduced encoding precision for each low-rank factor (Lee et al., 2022; 2023b).

Details of the algorithm, threshold schedule, and block validation are provided in Appendix C.

3.2 Rotation-Aware Diagonal Pruning

After DSLR decomposition, we further reduce rotation cost by removing the rotations with the smallest effect on the projection output. Pruning individual weights does not guarantee the removal of a rotation, as other weights may still require the same rotated input.

We define pruning units under the direct-diagonal cost model. For S and B, removing a diagonal from B does not eliminate a rotation if S still requires the same rotated input. For each nonzero logical offset used by S or B, we treat the corresponding diagonals as one rotation unit. We remove both diagonals when both matrices use the same offset and only one diagonal when a single matrix uses it. Each active diagonal of A at a nonzero offset forms a separate unit since A uses the intermediate result Bx as its input. Removing a unit sets all its weights to zero and reduces R_{dir} by one. Offset 0 is excluded since it requires no rotation.

We measure the importance of a rotation unit by the expected squared change in the projection output when that unit is removed. We normalize this quantity by the projection's output energy so that scores can be compared across projections. For projection l, let $\widehat{W}_l = S_l + A_l B_l$ denote the current DSLR approximation. The scores for a diagonal of A_l and an input-side offset used by S_l or B_l are, respectively,

$$\mathcal{S}_l^A(\ell) = \frac{\mathbb{E}_x \|A_l|_\ell B_l x\|_2^2}{\mathbb{E}_x \|\widehat{W}_l x\|_2^2}, \qquad \mathcal{S}_l^{\mathrm{in}}(\ell) = \frac{\mathbb{E}_x \|(S_l|_\ell + A_l B_l|_\ell)x\|_2^2}{\mathbb{E}_x \|\widehat{W}_l x\|_2^2}, \tag{3}$$

where $A_l|_\ell$, $B_l|_\ell$, and $S_l|_\ell$ retain only wrapped diagonal ℓ of their respective matrices, with all other entries set to zero. For a shared input-side unit, the score accounts for the combined output change from removing the corresponding diagonals of both S_l and B_l. We approximate the expectations using input statistics from a small calibration set. Lower scores indicate smaller relative output changes.

Given a pruning ratio $\rho \in [0, 1]$, we prune the $K = \lfloor \rho R_{\mathrm{fac}} \rfloor$ lowest-scoring units across all factorized projections, where R_{fac} is their total number of rotation units after DSLR decomposition. We periodically recompute the scores to account for the effects of earlier removals. We also prune projections in blocks kept dense by validation, treating each active diagonal at a nonzero offset as one rotation unit. These projections use the same output-based importance criterion, with a separate per-projection pruning ratio of $\gamma\rho$, where $\gamma \in [0, 1]$ controls how conservatively these projections are pruned.

After pruning, we fine-tune the model with all pruning masks fixed, so that the removed weights remain zero. Details of score computation and the pruning algorithm are provided in Appendix C.

4 Experimental Results

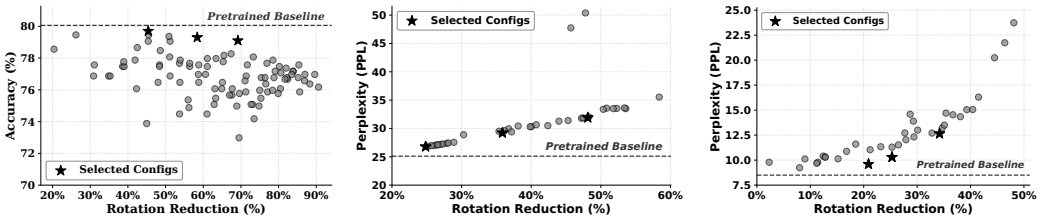
We evaluate ViT-Tiny on DermaMNIST (Yang et al., 2023) using accuracy (higher is better), and GPT-2 and Llama 3.1-8B on WikiText-2 (Merity et al., 2016) using perplexity (lower is better). Decomposition configurations are denoted *ViT-Cn*, *GPT-Cn*, and *Llama-Cn*, each corresponding to a specific $(\tau_{\mathrm{start}}, \alpha)$ setting.

We explore compression configurations through plaintext (PT) evaluation and conduct encrypted evaluation only for selected configurations, as in prior FHE systems (Ao & Boddeti, 2024; Lee et al., 2023b; 2022; Cheon et al., 2024). For each configuration, we measure layer-wise CKKS

Table 1: Task quality and rotation cost of RACE-HE compared with pure SVD, ASVD (Yuan et al., 2023), and pure $S + AB$ across all evaluated models. ViT-Tiny is evaluated on DermaMNIST (accuracy), and GPT-2 and Llama 3.1-8B on WikiText-2 (perplexity). Rotation reductions are relative to each model's dense baseline.

Model	Config.	Pure SVD	ASVD	Pure $S + AB$	RACE-HE (DSLR)	
		R_{dir} / PT Quality			R_{dir}	PT / CT Quality
ViT-Tiny (Acc. %, ↑)	Dense	–	–	–	9,174	80.1 / 80.0
	ViT-C1	5,008 / 77.2	5,009 / 78.4	5,008 / 78.2	5,009 (45.4%↓)	**79.7** / 79.7
	ViT-C2	3,812 / 75.4	3,811 / 76.5	3,811 / 76.3	3,814 (58.4%↓)	**79.3** / 79.2
	ViT-C3	2,820 / 77.1	2,822 / 78.4	2,822 / 78.4	2,824 (69.2%↓)	**79.1** / 79.1
GPT-2 (PPL, ↓)	Dense	–	–	–	36,816	25.1 / 25.1
	GPT-C1	27,656 / 34.1	27,675 / 28.8	27,674 / 29.2	27,675 (24.8%↓)	**26.8** / 26.8
	GPT-C2	23,610 / 36.3	23,610 / 31.3	23,611 / 32.1	23,610 (35.9%↓)	**29.2** / 29.2
	GPT-C3	19,098 / 40.0	19,098 / 37.8	19,096 / 38.2	19,098 (48.1%↓)	**31.9** / 32.0
Llama 3.1-8B (PPL, ↓)	Dense	–	–	–	720,672	8.5 / 8.5
	Llama-C1	570,142 / 12.5	570,148 / 12.0	570,144 / 12.4	570,150 (20.9%↓)	**9.6** / 9.6
	Llama-C2	538,132 / 13.3	538,130 / 13.2	538,130 / 13.3	538,132 (25.3%↓)	**10.3** / 10.3
	Llama-C3	474,201 / 16.7	474,210 / 16.9	474,208 / 16.9	474,202 (34.2%↓)	**12.6** / 12.7

Note. Pure SVD denotes the standard truncated SVD. Baselines are compared at approximately matched rotation budgets for each RACE-HE configuration, so the comparison emphasizes task quality at comparable rotation budgets.



(a) ViT-Tiny (b) GPT-2 (c) Llama 3.1-8B

Figure 3: Quality–rotation trade-off under different decomposition schedules. Each point denotes one decomposition configuration, and stars indicate selected configurations.

errors in OpenFHE (Al Badawi et al., 2022) using 20 test vectors. We then estimate ciphertext-domain (CT) quality by injecting zero-mean Gaussian noise with the measured layer-wise variances into a full-dataset simulator. These CT quality estimates closely track plaintext (PT) quality for the selected DSLR configurations (Table 1).

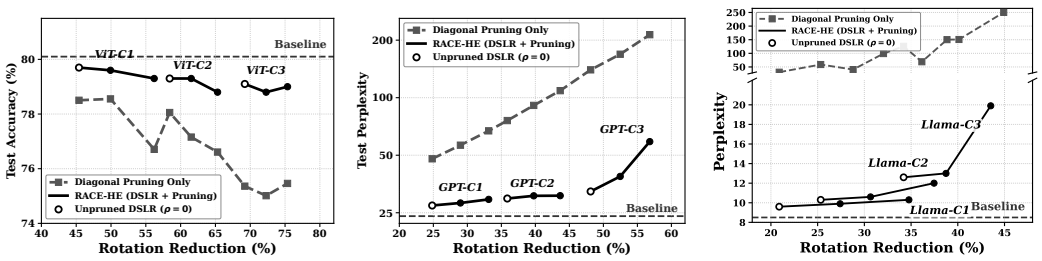
4.1 Effectiveness of DSLR Decomposition

This subsection evaluates DSLR decomposition before applying rotation-aware diagonal pruning. We compare DSLR with pure SVD (standard truncated SVD), activation-aware SVD (ASVD) (Yuan et al., 2023), and pure $S + AB$. ASVD uses activation statistics to rescale the weights before applying SVD, to reduce the quality loss caused by low-rank compression. As a controlled ablation of rotation-aware diagonal selection, we evaluate pure $S + AB$, which uses the same diagonal-sparse-plus-low-rank structure as DSLR but does not prioritize offsets shared with B. Correction diagonals are selected solely by residual energy, defined as the sum of squared residual entries on each diagonal, without preferring offsets shared with B.

Table 1 compares task quality at approximately matched structural rotation budgets to evaluate how well DSLR preserves model quality. For language models in Table 1, sensitive blocks are retained in dense form through the block-level validation described in Section 3.1. Within each comparison, these dense blocks are fixed across methods. All methods use the same recovery fine-tuning protocol, with LoRA (Hu et al., 2022) used for Llama 3.1-8B. The LoRA updates are merged into the

Table 2: Ablation of the diagonal-sparse correction and its selection rule under the same reconstruction threshold schedule, rank grid, fixed dense blocks, and recovery protocol. Pure $S + AB$ selects correction diagonals by residual energy alone. Quality is measured after recovery fine-tuning.

	ViT-C3 (Acc. %, ↑)			***GPT-C3*** (PPL, ↓)			***Llama-C3*** (PPL, ↓)		
Variant	R_{dir}	C_{diag}	**Quality**	R_{dir}	C_{diag}	**Quality**	R_{dir}	C_{diag}	**Quality**
Pure SVD	4,536	4,633	78.6	26,397	26,484	32.0	647,442	647,880	12.8
Pure $S + AB$	3,864	4,103	78.7	23,704	24,544	32.4	578,568	579,356	13.0
RACE-HE (DSLR)	**2,824**	4,299	79.1	**19,098**	25,643	31.9	**474,202**	560,852	12.6



(a) ViT-Tiny (b) GPT-2 (c) Llama 3.1-8B

Figure 4: Task quality versus total rotation reduction relative to the dense model for RACE-HE and dense diagonal pruning at matched R_{dir} budgets on (a) ViT-Tiny, (b) GPT-2, and (c) Llama 3.1-8B.

corresponding factor or dense weights before evaluation and export, while preserving the pruning supports. No additional adapter branch is evaluated during HE inference.

At approximately matched structural rotation budgets, DSLR achieves better task quality than pure SVD, ASVD, and pure $S + AB$ at all reported operating points. For instance, at *ViT-C1*, DSLR improves accuracy by 2.5, 1.3, and 1.5 percentage points over these baselines, respectively. The comparison with pure $S + AB$ indicates that rotation-aware diagonal selection helps preserve task quality within a limited rotation budget. Compared to the dense baseline, the selected ViT-Tiny configurations range from a 45.4% rotation reduction with a 0.4 pp accuracy drop (*ViT-C1*) to 69.2% with a 1.0 pp drop (*ViT-C3*). GPT-2 ranges from 24.8% at $+1.7$ perplexity (*GPT-C1*) to 48.1% at $+6.8$ (*GPT-C3*), while Llama 3.1-8B spans 20.9% at $+1.1$ (*Llama-C1*) to 34.2% at $+4.1$ (*Llama-C3*).

Figure 3 shows the quality–rotation trade-off across the DSLR decomposition sweep. Stars mark the configurations reported in Table 1, selected to cover different levels of rotation reduction with relatively good task quality at each level. Their threshold settings, numbers of compressed and skipped blocks, and average (k, t, e) values are provided in Table 5 in Appendix C.2.

We next examine how much of the rotation reduction comes from adding the diagonal-sparse correction and how much comes from its rotation-aware selection. Table 2 compares pure SVD, pure $S+AB$, and DSLR under the same reconstruction threshold schedule, rank grid, fixed dense blocks, and recovery protocol. Each method selects its rank and, where applicable, correction diagonals to meet the reconstruction budget, allowing us to compare the resulting rotation and diagonal costs. Task quality is evaluated after recovery fine-tuning.

Under the same reconstruction budget, DSLR achieves the lowest structural rotation cost across all three models. At *ViT-C3*, DSLR requires 2,824 rotations, compared with 4,536 for pure SVD and 3,864 for pure $S + AB$, while the corresponding post-recovery accuracies differ by at most 0.5 percentage points. Relative to pure $S + AB$, DSLR reduces rotations by 26.9%, 19.4%, and 18.0% on ViT-Tiny, GPT-2, and Llama 3.1-8B, respectively, with comparable post-recovery task quality. These results suggest that rotation-aware diagonal selection provides additional rotation savings beyond those obtained by adding the correction alone.

Table 3: Encrypted ViT-Tiny inference in OpenFHE with BSGS scheduling. The DSLR-based variants use support selection adapted from the *ViT-C3* configuration for BSGS execution.

	Dense (BSGS)	**Pure SVD**	**DSLR**	**DSLR + Prune (RACE-HE)**
PT accuracy (%, ↑)	80.1	79.1	80.0	79.9
R_{dir} / C_{diag} (total)	9,174 / 9,223	2,824 / 2,917	2,824 / 4,299	2,261 / 3,665
Scheduled rotations (total)	2,715	2,021	1,949	1,559
Bootstrappings (total)	49	49	49	49
Linear (PC-MM) latency (total, s)	10,784	2,445	2,409	**2,040** (81.1% ↓)
End-to-end latency (s)	39,410	31,369	30,136	**29,884** (24.2% ↓)

Note. Intel Xeon Silver 4210R, ring dimension $N = 131{,}072$, 128-bit security. Factorized variants use split-scale encoding with one rescale (Section 3.1). PT accuracy is evaluated using the weights and pruning masks of each timed configuration. Latencies are averaged over three runs; all entries refer to the same exported checkpoint and pruning masks for each configuration.

4.2 Impact of Rotation-Aware Diagonal Pruning

We compare pruning after DSLR decomposition in RACE-HE with a baseline that prunes whole wrapped diagonals directly from dense weight matrices, following the approach of Hunter and Artemis (Cai et al., 2022; Jeon et al., 2023). Both methods use the output-based importance criterion of Eq. 3 and the same calibration set, recovery fine-tuning, and checkpoint selection, and are evaluated at approximately matched direct rotation budgets. Each RACE-HE operating point combines a decomposition configuration from the sweep of Section 4.1 with a pruning ratio $\rho \in \{0.0, 0.1, 0.2\}$, so that decomposition carries a large portion of the reduction and pruning removes the least important remaining rotation units. For GPT-2 and Llama 3.1-8B, projections in blocks kept dense by skipping use a per-projection pruning ratio of $\gamma\rho$, with $\gamma = 0.5$.

Figure 4 shows the results. On ViT-Tiny, RACE-HE lies above dense diagonal pruning at all nine matched budgets, and the margin is largest at the tightest budgets. At 75.4% reduction, RACE-HE retains 79.0% accuracy where dense diagonal pruning retains 75.5%. On GPT-2, RACE-HE reduces rotations by 29.0% with a perplexity increase of 2.5 relative to the dense model. At a higher rotation reduction of 43.6%, RACE-HE achieves a perplexity of 30.7, compared with 108.8 for dense diagonal pruning. For Llama 3.1-8B, RACE-HE achieves a perplexity of 10.3 at a 34.8% rotation reduction, compared with 69.0 for dense diagonal pruning at a 36.2% reduction.

Pruning also extends the rotation reductions achieved by DSLR decomposition alone. For *ViT-C3*, setting $\rho = 0.2$ increases the reduction from 69.2% to 75.4%, with an additional accuracy loss of only 0.1 percentage points. DSLR first reduces rotation cost through low-rank approximation and a diagonal-sparse correction. Pruning then removes remaining rotation units with small measured contributions to the projection output, providing further savings with limited quality loss.

4.3 Composition with HE Execution Optimizations

Sections 4.1 and 4.2 evaluate task quality against the structural cost R_{dir}. We now evaluate whether these savings translate into lower inference cost when combined with HE execution optimizations. Table 3 reports full ViT-Tiny inference in OpenFHE, including all twelve blocks, polynomial nonlinearities, and fully server-side bootstrapping, with BSGS auto-tuned for each operator. It compares the dense model, DSLR based on *ViT-C3*, pure SVD at approximately the same R_{dir} budget as unpruned DSLR, and RACE-HE. The BSGS experiments use backend-specific variants of Algorithms 1 and 2, selecting correction diagonals and pruning groups according to the scheduled execution cost. The target $\rho_{\text{BSGS}} = 0.2$ is defined relative to the unpruned DSLR model's total scheduled linear rotations, including the fixed output head. Appendix C.4 specifies the modified selection objective, pruning score, and physical packing, with implementation details provided in the accompanying code. Scheduled rotation counts include auxiliary layout operations, such as input replication, output folding, and inter-stage repacking, and are reported separately from R_{dir}.

The split-scale encoding of Section 3.1 keeps the two-stage branch within the same level budget as the dense operator, at the cost of reduced per-factor encoding precision. All configurations perform 49 bootstrapping operations, and the measured layer-wise CKKS errors remain small (Appendix D). Table 3 also reports full-test-set PT accuracy using the same weights and pruning masks as the

Table 4: Measured independent PC-MM kernel costs for GPT-2's block-0 attention output projection (768×768) on the NEXUS-derived coefficient-packing implementation. RACE-HE's DSLR operators use a reconstruction threshold $\tau = 0.95$, with only the retained entries of S packed according to its public sparsity pattern.

Configuration	Active S diagonals	Compressed ciphertexts	Decompression key switchings	Kernel latency (s)	Speedup	Reconstruction error
Dense (768×768)	—	72	589,752	3,098.7	$1.00\times$	—
RACE-HE ($k = 64$)	485	58	475,078	2,488.4	$1.25\times$	0.2231
RACE-HE ($k = 128$)	261	49	401,359	2,100.0	$1.48\times$	0.2232
RACE-HE ($k = 256$)	0	48	393,168	2,044.0	$1.52\times$	0.1232

Note. Latency sums the independent B, A, and S kernel passes and the encrypted addition of the two branch outputs, excluding setup, encoding, encryption, and verification. The input to the A pass is computed offline in plaintext. Reconstruction error is $\|W - S - AB\|_F / \|W\|_F$, whose square is constrained to be at most $1 - \tau = 0.05$.

encrypted runs. This evaluation uses the same plaintext operations as Table 1, including the original non-linearities. For the tested ViT-Tiny inputs, the final decrypted outputs of the complete HE pipeline show close numerical agreement with outputs from this plaintext evaluation. The encrypted runs use the same model weights and HE settings as the latency measurements.

Relative to the dense BSGS baseline, DSLR alone reduces PC-MM and end-to-end latency by 77.7% and 23.5%, respectively. Adding rotation-aware pruning increases these reductions to 81.1% and 24.2%. With input rotations shared between the B and S branches, RACE-HE reduces scheduled rotations by 42.6% (Appendix C.4). The PC-MM latency reduction is consistent with fewer scheduled rotations, plaintext multiplications, accumulations, and mask encodings. The additional pruning step saves 369 s in PC-MM time and 252 s end to end, with a 117 s increase in the combined time of the remaining components partially offsetting the PC-MM savings.

We additionally evaluate a single GPT-2 attention output projection under HE. RACE-HE reduces BSGS rotation calls by 14.7% and ciphertext–plaintext multiplications by 43.4%, yielding a $1.61\times$ PC-MM speedup while retaining one-level consumption (Table 8, Appendix C.4).

To test composition with a different packing method, we also evaluate independent PC-MM kernel passes using a coefficient-packing implementation derived from NEXUS. For a 768×768 GPT-2 projection with reconstruction threshold $\tau = 0.95$, RACE-HE's DSLR operators achieve kernel speedups of $1.25\times$, $1.48\times$, and $1.52\times$ at ranks 64, 128, and 256, respectively (Table 4). Ciphertext decompression expands packed coefficients into separate ciphertexts for matrix multiplication and requires key-switching operations. The latency reductions closely track the reduction in these operations. At ranks 64 and 128, DSLR meets the reconstruction budget using low-rank factors supplemented by the diagonal-sparse correction. Across the three tested ranks, increasing the rank reduces the correction needed, lowering the total number of matrix entries processed by this kernel. Appendix C.4 provides further analysis of the BSGS-aware selection, pruning rules, the packing layout, amortized pipeline and Llama-scale projections, with the implementation available in the accompanying code.

5 Conclusion

This paper presented RACE-HE, a rotation-aware compression framework for efficient private inference. Its DSLR decomposition augments a low-rank approximation with a diagonal-sparse correction that captures residual structure primarily on rotation offsets already used by the input-side factor. Rotation-aware diagonal pruning then removes low-importance rotation units, jointly eliminating packed diagonals that share an input rotation. Across ViT-Tiny, GPT-2, and Llama 3.1-8B, RACE-HE reduces direct rotations by 75.4%, 29.0%, and 34.8%, respectively, and achieves better quality–rotation trade-offs than compression baselines. In fully encrypted OpenFHE inference on ViT-Tiny, these structural savings translate into a 24.2% end-to-end latency reduction. These results show that rotation-aware weight compression complements HE execution optimizations and can translate structural rotation savings into lower end-to-end inference cost.

AI USE STATEMENT

Generative AI was used solely for editorial assistance (improving clarity and phrasing), minor code refinement (readability and debugging), and literature searches. It was not used to generate research ideas, design the methodology, or produce experimental results. All technical contributions and analyses were conceived and validated by the authors, who also conducted the literature review and are responsible for citing relevant prior work.

ETHICS STATEMENT

This work aims to strengthen privacy protection in machine learning. RACE-HE reduces the computational cost of FHE-based inference, in which client inputs remain encrypted end-to-end and model weights remain local to the server, thereby lowering a practical barrier to deploying privacy-preserving inference in sensitive domains. All experiments use publicly available models and benchmark datasets (DermaMNIST and WikiText-2) under their respective licenses. We did not recruit human participants or collect new personal data. As RACE-HE is a lossy compression method, deployments in high-stakes settings such as medical diagnosis should validate the chosen quality budget against application-specific requirements. Our framework exposes this trade-off explicitly through its configuration parameters.

REPRODUCIBILITY STATEMENT

We provide anonymized source code for the main experimental pipelines, including DSLR decomposition, rotation-aware diagonal pruning, CKKS projection-error calibration, and BSGS-aware OpenFHE latency evaluation. The release contains model-specific implementations for ViT-Tiny, GPT-2, and Llama 3.1-8B, together with shared calibration and evaluation utilities. The plaintext pipelines implement reconstruction-error-bounded decomposition, recovery fine-tuning, and pruning with separate factorized and dense-projection budgets. The CKKS calibration code measures projection-level numerical errors. Ciphertext-domain quality is estimated through plaintext simulation with Gaussian noise calibrated to these errors, as described in Appendix D. For the ViT-Tiny latency experiments, a common OpenFHE executable supports dense, pure SVD, DSLR, and DSLR with pruning under a shared polynomial inference circuit. The release includes BSGS-aware support selection and pruning, weight export, shared normalization calibration, Newton seed fitting, configurable bootstrapping schedules, and phase-level timing. The BSGS-aware latency configurations are generated separately from the recovered checkpoints used in Sections 4.1 and 4.2. The experiments evaluate ViT-Tiny on DermaMNIST from MedMNIST v2, and GPT-2 and Llama 3.1-8B on WikiText-2. Model weights and datasets are not bundled. Access to Llama weights requires authorization from the model provider. The accompanying READMEs specify dependencies, input checkpoints, execution commands, cryptographic parameters, measurement boundaries, and configurable random seeds.

REFERENCES

Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *proceedings of the 10th workshop on encrypted computing & applied homomorphic cryptography*, pp. 53–63, 2022.

Wei Ao and Vishnu Naresh Boddeti. {AutoFHE}: Automated adaption of {CNNs} for efficient evaluation over {FHE}. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 2173–2190, 2024.

Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. Plaintext-ciphertext matrix multiplication and fhe bootstrapping: fast and fused. In *Annual International Cryptology Conference*, pp. 387–421. Springer, 2024.

Song Bian, Weiwen Jiang, Qing Lu, Yiyu Shi, and Takashi Sato. Nass: Optimizing secure inference via neural architecture search. *arXiv preprint arXiv:2001.11854*, 2020.

Song Bian, Zian Zhao, Ruiyu Shen, Zhou Zhang, Ran Mao, Dawei Li, Yizhong Liu, Masaki Waga, Kohei Suenaga, Zhenyu Guan, et al. Chloe: Loop transformation over fully homomorphic encryption via multi-level vectorization and control-path reduction. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2395–2413. IEEE, 2025.

Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pp. 868–886. Springer, 2012.

Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

Yifei Cai, Qiao Zhang, Rui Ning, Chunsheng Xin, and Hongyi Wu. Hunter: He-friendly structured pruning for efficient privacy-preserving deep learning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pp. 931–945, 2022.

Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *International conference on financial cryptography and data security*, pp. 3–18. Springer, 2017.

Yi Chen, Qiang-Sheng Hua, Zixiao Hong, Lin Zhu, and Hai Jin. Fhe4dmm: A low-latency distributed matrix multiplication with fully homomorphic encryption. *IEEE Transactions on Parallel and Distributed Systems*, 36(4):645–658, 2025.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pp. 409–437. Springer, 2017.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 360–384. Springer, 2018.

Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. {DaCapo}: Automatic bootstrapping management for efficient fully homomorphic encryption. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 6993–7010, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Franklin Espinoza, Justo Molina, Darwin Quezada-Gaibor, Sandra Catalán, and Manuel F Dolz. Cross-platform characterisation and performance analysis of homomorphic matrix multiplication. *The Journal of Supercomputing*, 82(5):341, 2026.

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pp. 201–210. PMLR, 2016.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pp. 554–571. Springer, 2014.

Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. In *Annual International Cryptology Conference*, pp. 93–120. Springer, 2018.

Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. *Cryptology ePrint Archive*, 2020.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. *Advances in neural information processing systems*, 35:15718–15731, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *31st USENIX security symposium (USENIX Security 22)*, pp. 809–826, 2022.

Yeonsoo Jeon, Mattan Erez, and Michael Orshansky. Artemis: he-aware training for efficient privacy-preserving machine learning. *arXiv preprint arXiv:2310.01664*, 2023.

Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 1209–1222, 2018.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX security symposium (USENIX security 18)*, pp. 1651–1669, 2018.

Andes YL Kei and Sherman SM Chow. Shaft: Secure, handy, accurate, and fast transformer inference. *Cryptology ePrint Archive*, 2025.

Joon-Woo Lee, Eunsang Lee, Young-Sik Kim, and Jong-Seon No. Rotation key reduction for client-server systems of deep neural network on fully homomorphic encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 36–68. Springer, 2023a.

Yongwoo Lee, Seonyeong Heo, Seonyoung Cheon, Shinnung Jeong, Changsu Kim, Eunkyung Kim, Dongyoon Lee, and Hanjun Kim. Hecate: Performance-aware scale optimization for homomorphic encryption compiler. In *2022 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pp. 193–204. IEEE, 2022.

Yongwoo Lee, Seonyoung Cheon, Dongkwan Kim, Dongyoon Lee, and Hanjun Kim. {ELASM}:{Error-Latency-Aware} scale management for fully homomorphic encryption. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4697–4714, 2023b.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pp. 20336–20350. PMLR, 2023.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*, 2023.

Yuxiao Lu, Jie Lin, Chao Jin, Zhe Wang, Min Wu, Khin Mi Mi Aung, and Xiaoli Li. Ff-conv: Fast factorized convolutional neural network inference on encrypted data. *arXiv preprint arXiv:2102.03494*, 2021.

Xirong Ma, Junling Fang, Chunpeng Ge, Dung H Duong, Yali Jiang, Yanbin Li, Willy Susilo, and Lizhen Cui. New permutation decomposition techniques for efficient homomorphic permutation. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1068–1082, 2025.

Hannah Mahon and Shane Kosieradzki. Encrypted matrix multiplication using 3-dimensional rotations. *Cryptology ePrint Archive*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Ahmet Can Mert, Aikata, Sunmin Kwon, Youngsam Shin, Donghoon Yoo, Yongwoo Lee, Sujoy Sinha Roy, et al. Medha: Microcoded hardware accelerator for computing on encrypted data. *arXiv preprint arXiv:2210.05476*, 2022.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4753–4771. IEEE, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Panagiotis Rizomiliotis and Aikaterini Triakosia. On matrix multiplication with homomorphic encryption. In *proceedings of the 2022 on cloud computing security workshop*, pp. 53–61, 2022.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pp. 19299–19319, 2025.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific data*, 10(1):41, 2023.

Mu Yuan, Lan Zhang, Yihang Cheng, Miao-Hui Song, Guoliang Xing, and Xiang-Yang Li. Stip: Three-party privacy-preserving and lossless inference for large transformers in production. In *Network and Distributed System Security (NDSS) Symposium*, 2026.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024.

APPENDICES

A SYSTEM AND THREAT MODEL

In this paper, we consider a standard two-party setting consisting of a model owner (server) and a data owner (client). The server holds a proprietary neural network model with plaintext weights W_l. The client holds sensitive input data x and wishes to obtain the inference result $f(x)$ without revealing x to the server. The client encrypts its input under the CKKS scheme and transmits the ciphertext to the server, which evaluates the model homomorphically and returns an encrypted result.

We assume a semi-honest (honest-but-curious) adversarial model in which both parties follow the prescribed protocol correctly but may attempt to infer private information from their respective views. Under this model, RACE-HE performs model compression entirely as a preprocessing step on the server's plaintext weights before any interaction with the client. Therefore, RACE-HE introduces no additional online interaction and does not expand the adversarial view beyond that of the baseline semi-honest protocol.

B ADDITIONAL CKKS BACKGROUND AND EXTENDED RELATED WORK

B.1 CKKS OPERATIONS

The CKKS scheme is a leveled FHE scheme that supports approximate arithmetic over vectors of complex numbers. A plaintext vector $\mathbf{m} \in \mathbb{C}^{N/2}$ is encoded and encrypted into a ciphertext ct, and the resulting computation is constrained by a finite multiplicative-level budget L. The essential homomorphic operations and their relative costs are as follows:

- $\mathsf{Add}(ct_1, ct_2)$: element-wise addition; inexpensive, does not consume a multiplicative level, and is used to accumulate partial sums in linear-layer evaluation.
- $\mathsf{PMult}(ct, \mathbf{p})$: multiplication of a ciphertext ct by a plaintext vector $\mathbf{p}$; cheaper than ciphertext-ciphertext multiplication and, in CKKS, typically consumes one level, followed by rescaling. Since the server holds weights in plaintexts, linear layers are primarily implemented using PMult.
- $\mathsf{Mult}(ct_1, ct_2, evk)$: element-wise ciphertext-ciphertext multiplication; expensive, consumes one multiplicative level, and requires a relinearization key evk. It is mainly used to evaluate polynomial approximations of non-linear activation functions.
- $\mathsf{Rot}(ct, r; rk_r)$: cyclically shifts the slots of ct by r positions; does not consume a multiplicative level in the usual CKKS level-budget sense, but requires key switching and a rotation key rk_r for each distinct offset r, making rotations a major source of runtime and memory overhead.
- $\mathsf{Rescale}(ct)$: reduces the scaling factor after multiplication to control noise growth and restore the ciphertext to the expected scale. The cumulative number of rescaling operations determines how much multiplicative depth can be supported before bootstrapping becomes necessary.
- $\mathsf{Bootstrap}(ct)$: refreshes a low-level ciphertext back to a higher level when the finite level budget L is exhausted (Cheon et al., 2018), effectively extending the computable depth. It is among the most expensive operations in FHE and is an important factor in end-to-end inference efficiency.

B.2 EXTENDED RELATED WORK

HE-aware model adaptation. A complementary line of work adapts the model on the non-linear side, with CryptoNets (Gilad-Bachrach et al., 2016) replacing activations by low-degree polynomials, AutoFHE (Ao & Boddeti, 2024) searching layer-wise mixed-degree polynomials and bootstrapping placement, and NASS (Bian et al., 2020) jointly exploring architectures, quantization, and cryptographic parameters. These methods reduce bootstrapping or communication costs but leave the rotation-inducing structure of the dense weights unchanged, which is the axis RACE-HE targets.

Compiler and distributed systems. Beyond algebraic improvements to individual homomorphic operators, another major line of research optimizes how fixed FHE programs are scheduled, compiled, and executed on real systems. A conventional baseline for homomorphic linear transformations is the BSGS strategy, whose schedule and interaction with DSLR operators are analyzed in Appendix C.4, and which is combined in practice with hoisting or double-hoisting to amortize key-switching overhead. Building on such operator-level schedules, compiler frameworks seek to automate efficient code generation for encrypted programs. A representative recent example is CHLOE (Bian et al., 2025), a multi-level compiler framework for loop-containing FHE programs, which introduces loop-aware intermediate representations and applies loop segmentation, vectorization, control-path reduction, and cost-guided lowering to optimize both transparent and oblivious loops. At the systems level, FHE4DMM (Chen et al., 2025) scales encrypted matrix multiplication beyond a single machine, combining task partitioning, reuse-aware scheduling of homomorphic data movement, and implementation-level optimizations at cluster scale. A related line of work studies homomorphic matrix multiplication from a benchmarking perspective, with Espinoza et al. (2026) analyzing CKKS matrix multiplication across SEAL (Chen et al., 2017), HElib (Halevi & Shoup, 2020), and OpenFHE (Al Badawi et al., 2022) on different CPUs and showing that library and hardware choices substantially affect throughput and memory behavior. Since RACE-HE reduces the transform before scheduling, a smaller operator means fewer diagonals for BSGS to schedule, fewer loop iterations for a compiler to vectorize, and less data for a distributed runtime to move.

Alternative secure Transformer systems. Recent work has also explored secure Transformer inference under system models that differ from FHE-based secure inference. SHAFT (Kei & Chow, 2025) is a two-party framework based on secret sharing that targets the major non-linear bottlenecks of Transformer inference, designing specialized secure protocols for operations such as softmax and GELU while supporting model conversion from standard machine learning toolchains. STIP (Yuan et al., 2026) adopts a three-party deployment model involving a model developer, a cloud model server, and a data owner. It avoids heavy cryptographic computation during online inference by protecting the model and user data through a semi-symmetric permutation-based transformation with lightweight trusted execution environment (TEE) support, which preserves the original computation semantics and targets lossless accuracy on commodity cloud hardware. Resting on secret sharing and on permutation with TEE support respectively, they address a different problem from RACE-HE, which stays within the standard two-party FHE setting and reduces the structural cost of homomorphic linear layers, so that we include them to map the broader privacy–efficiency design space rather than as direct baselines.

HE-aware structured pruning. Hunter (Cai et al., 2022) prunes packing-aware structures of CNNs in interactive BFV frameworks, including the weight diagonals of fully connected layers, through iterative retraining, with each pruned structure provably eliminating one Perm operation. Artemis (Jeon et al., 2023) instead learns such sparsity during training with Group Lasso regularization, and its ablation attributes the rotation–accuracy Pareto frontier almost entirely to diagonal pruning, further supporting packed diagonals as the effective granularity for reducing rotations. Neither combines pruning with factorization, targets Transformers, or accounts for multiplicative depth. Conceptually, both prune a dense operator, whereas RACE-HE prunes rotation units of a DSLR operator whose rotation count is already reduced by factorization, removing them post hoc under a global budget rather than during training (Section 3.2). Section 4.2 compares dense diagonal pruning with RACE-HE at matched rotation budgets.

C Method Details and Algorithms

C.1 Direct-Diagonal Cost of a DSLR Operator

We derive the structural costs using the logical wrapped offsets defined in Section 2.1. Consider $\widehat{W} = S + AB$, where $A \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{k\times n}$, and $1 \le k \le \min(m, n)$. Before pruning, assume that each factor has all k logical diagonals at offsets $0, \dots, k-1$ active. Each factor then contributes $k-1$ nonzero offsets.

Let t be the number of active diagonals of S at offsets in $\{0, \dots, k-1\}$, and let e be the number at offsets outside this set. In the direct-diagonal cost model, offsets shared by S and B are counted

Algorithm 1 DSLR Structure Selection for a Single Projection

Require: Original weight $W_l \in \mathbb{R}^{m_l \times n_l}$, threshold $\tau^{(b)}$, candidate rank grid $\mathcal{K}$, iterations T
Ensure: DSLR representation (S_l, A_l, B_l) or dense fallback

$q_l \leftarrow \min(m_l, n_l)$, $\eta \leftarrow (1 - \tau^{(b)})\|W_l\|_F^2$, $\mathcal{B} \leftarrow \varnothing$
function FIT(k, t, e) {alternating fit; $\mathcal{F}_k = \{0, \ldots, k-1\}$}
  $L \leftarrow 0$
  repeat up to T times, or until the selected support stabilizes
    $S \leftarrow (W_l - L)$ restricted to the t highest-energy offsets inside $\mathcal{F}_k$ and the e highest-energy offsets outside $\mathcal{F}_k$
    $L \leftarrow$ rank-k truncated-SVD approximation of $W_l - S$
  return (S, L), accepted iff $\|W_l - S - L\|_F^2 \le \eta$
for each $k \in \mathcal{K}$ with $1 \le k \le \lfloor q_l/2 \rfloor$ **do**
  Scan $e = 0, \ldots, q_l - 2k$ in ascending order; let e_k be the first value for which FIT(k, k, e) is accepted {full pool, $t = k$}
  if no such e_k is found **then**
    continue {no accepted fit found at this rank}
  end if
  Scan $t = 0, \ldots, k$ in ascending order; let t_k be the first value for which FIT(k, t, e_k) is accepted
  Add (k, t_k, e_k) and its accepted fitted pair (S, L) to $\mathcal{B}$
end for
if $\mathcal{B} = \varnothing$ **then**
  return original W_l as dense
end if
Select the candidate in $\mathcal{B}$ with lexicographically minimum $\big(2(k-1) + e,\ 2k + t + e,\ k\big)$
Factorize its fitted matrix as $L = U\Sigma V^T$
return $S_l \leftarrow S$, $A_l \leftarrow U\Sigma$, $B_l \leftarrow V^T$

once as both branches use the same input. The t reused offsets therefore add no structural rotation cost, while the e extra offsets each add one. The input-side cost is thus $k - 1 + e$. The A stage operates on Bx and contributes another $k - 1$ offsets, giving

$$R_{\text{dir}} = 2(k-1) + e, \qquad C_{\text{diag}} = 2k + t + e. \tag{4}$$

Here, C_{diag} counts the active logical diagonals of S, A, and B separately, including diagonals that share an input offset. Thus, t increases the diagonal workload without increasing R_{dir}, whereas e increases both. If some factor diagonals are inactive, these expressions remain upper bounds.

C.2 Compression Algorithms and Block-Level Adaptation

Per-Projection DSLR Structure Selection. Algorithm 1 describes structure selection for each projection given the block threshold $\tau^{(b)}$. All residuals, low-rank updates, and reconstruction checks use the original weight matrix W_l. The reconstruction constraint in Eq. 2 applies at this stage, before subsequent recovery and pruning.

The candidate rank grid discretizes the search and does not by itself guarantee physical rotation sharing. The algorithm considers only candidates with a lower structural rotation cost than the dense projection and uses alternating fits to search for an approximation that meets the reconstruction budget. This search is approximate, and the original weight matrix is retained if no accepted fit is found.

Block-level adaptation and skipping. All target projections in Transformer block b share the reconstruction threshold

$$\tau^{(b)} = \max\left(\tau_{\min}, \tau_{\text{start}} - \alpha \frac{b}{N_{\text{blk}}}\right), \qquad b \in \{1, \ldots, N_{\text{blk}}\}, \tag{5}$$

where N_{blk} is the number of blocks, τ_{start} controls the threshold level, α controls its decrease with depth, and $\tau_{\min}$ sets a lower bound. Given this shared threshold, structure selection is performed independently for each projection.

Table 5: Threshold settings and selected DSLR structures for the configurations reported in Table 1, before rotation-aware diagonal pruning.

Model	Config.	$\tau_{\mathtt{start}}$	α	D/S	Avg. k	Avg. t	Avg. e
ViT-Tiny	*ViT-C1*	0.80	0.05	12 / 0	52	51	3
	ViT-C2	0.70	0.10	12 / 0	39	39	3
	ViT-C3	0.60	0.15	12 / 0	29	29	3
GPT-2	*GPT-C1*	0.60	0.20	4 / 8	98	97	2
	GPT-C2	0.60	0.10	6 / 6	109	108	2
	GPT-C3	0.80	0.20	10 / 2	162	161	3
Llama 3.1-8B	*Llama-C1*	0.95	0.10	30 / 2	953	943	109
	Llama-C2	0.90	0.10	25 / 7	995	987	186
	Llama-C3	0.80	0.15	22 / 10	715	711	189

Note. D/S denotes the numbers of DSLR-compressed blocks and blocks retained in dense form, respectively. Average values are computed over DSLR-compressed projections and rounded to integers. Here, k is the rank, t counts correction diagonals at logical offsets also used by B, and e counts correction diagonals at additional logical offsets.

Algorithm 2 Rotation-Aware Diagonal Pruning

Require: Factorized operators $\{\widehat{W}_l = S_l + A_l B_l\}$, dense projections $\{W_l\}$, calibration set $\mathcal{X}$, pruning ratio ρ, dense factor γ, rescoring step η
Ensure: Pruning masks for every projection
1: $\mathcal{U} \leftarrow$ rotation units of all factorized projections, $K \leftarrow \lfloor \rho\,|\mathcal{U}| \rfloor$, $\mathcal{P} \leftarrow \varnothing$ $\{|\mathcal{U}| = R_{\mathrm{fac}}$, one unit per nontrivial offset of A_l and of $\mathcal{H}(S_l) \cup \mathcal{H}(B_l)\}$
2: **while** $|\mathcal{P}| < K$ **do**
3: Estimate the input second moments $\mathbb{E}_x[x_j^2]$ of every factorized projection on $\mathcal{X}$ with the current masks applied
4: Score every unit in $\mathcal{U} \setminus \mathcal{P}$ by Eq. 3
5: Move the $\min\big(\lceil \eta\,|\mathcal{U}| \rceil,\ K - |\mathcal{P}|\big)$ lowest-scoring units to $\mathcal{P}$ and zero their diagonals, in A_l for an A-side unit and in S_l and B_l wherever active for an input-side unit
6: **end while**
7: **for** each dense projection W_l **do**
8: Zero its $\lfloor \gamma\rho\,(q_l - 1) \rfloor$ nontrivial diagonals with the lowest scores $\mathbb{E}_x \|W_l|_\ell\, x\|_2^2 \,/\, \mathbb{E}_x \|W_l\, x\|_2^2$
9: **end for**
10: **return** masks that keep the zeroed diagonals at zero during recovery fine-tuning

For language models, we additionally apply validation-based block skipping. Blocks are processed in order of depth, with earlier decisions kept in place. For GPT-2, after applying the candidate DSLR operators to block b and performing a brief block-level recovery step, we measure the increase in validation perplexity relative to $M_{\mathrm{pre}}^{(b)}$, the perplexity before modifying the block. The candidate is accepted if this increase does not exceed

$$\delta_b = \max\Big(\delta_{\mathrm{abs}}, \delta_{\mathrm{rel}} M_{\mathrm{pre}}^{(b)}\Big). \tag{6}$$

For Llama 3.1-8B, we instead evaluate the increase in validation negative log-likelihood (NLL) after applying the candidate operators but before LoRA recovery. The candidate is accepted if the NLL increase relative to the model before modifying the block does not exceed a preset absolute tolerance. For both models, a rejected candidate restores the model to its state before processing block b. The rank-grid steps, threshold floor, skipping tolerances, and recovery settings are listed in Table 6.

Rotation-aware diagonal pruning. Algorithm 2 implements the pruning procedure using the rotation units and importance scores defined in Section 3.2. Here, $\mathcal{H}(M)$ denotes the set of logical wrapped-diagonal offsets of M whose diagonals contain at least one nonzero entry. We estimate each projection's input second moments from a small calibration set with the current pruning masks applied. Let Δ denote the change in the effective weight matrix caused by removing a unit. Using a

diagonal approximation to the input second moment, we estimate the output-change energy as

$$\mathbb{E}_x\|\Delta x\|_2^2 \approx \sum_{i,j} \Delta_{ij}^2\, \mathbb{E}_x[x_j^2]. \tag{7}$$

The same approximation is used to compute the output energy in the normalization denominator. This calculation applies to both factorized and dense projections.

For comparisons across pruning ratios, factorized projections follow a shared greedy pruning path. Scores are recomputed at checkpoints fixed independently of the target ratio, and each ρ uses the corresponding prefix of this path. Recovery fine-tuning uses the same training budget and seed for every ρ, including the $\rho = 0$ control. Calibration and rescoring settings are provided in Table 6.

Hyperparameters and implementation settings. Table 6 summarizes the main hyperparameters and implementation settings. Experiment-specific reconstruction budgets are reported alongside the corresponding results, and additional settings are provided in the released code.

C.3 Worked Example on a Transformer Projection Layer

Consider a 4096×4096 projection, such as the query projection of Llama 3.1-8B. Under the structural cost model of Section 2.1, the dense matrix has $q = 4096$ active wrapped diagonals and $R_{\mathrm{dir}} = q - 1 = 4095$.

For illustration, suppose a DSLR approximation $W \approx S + AB$ with rank $k = 256$ satisfies the reconstruction budget using the full reuse pool and no extra offsets, so that $t = k$ and $e = 0$. The dense factors $A \in \mathbb{R}^{4096\times 256}$ and $B \in \mathbb{R}^{256\times 4096}$ each have 256 active wrapped diagonals. All 256 correction diagonals belong to the logical reuse pool of B, so S adds plaintext multiplications without increasing the structural rotation count. The resulting costs are $R_{\mathrm{dir}} = 2(k-1) = 510$ and $C_{\mathrm{diag}} = 2k + t = 768$. Compared to the dense baseline, this hypothetical configuration reduces R_{dir} by approximately 87.5%, as summarized in Table 7.

These counts describe the logical support structure of the assumed operators. Physical rotation counts and rotation-key requirements depend on the packing layout and execution schedule, including padding, ciphertext chunking, folding, and BSGS scheduling. Backend-specific packing and execution costs are discussed in Appendix C.4.

C.4 Composition with Rotation Scheduling, Packing, and Execution-Layer Systems

RACE-HE restructures the weight operators evaluated by existing HE implementations. This subsection analyzes how the resulting operators compose with BSGS scheduling, alternative packing layouts, and execution-layer systems, distinguishing analytical operation counts from the measured execution costs in Section 4.3.

BSGS-aware evaluation of RACE-HE. BSGS groups diagonal terms into baby steps and giant steps. Baby-step rotations are applied to the input ciphertext, while giant-step rotations are applied to weighted partial sums. The S and B branches share cached baby-step ciphertexts whenever they require the same rotation of the same input ciphertext. Their weighted giant-step rotations are evaluated separately, and the A stage uses its own rotations on the intermediate result Bx.

The ViT-Tiny experiment in Table 3 uses backend-specific variants of Algorithm 1. The reconstruction constraint is still evaluated against the original weight matrix W, but correction candidates are constructed from physical diagonals within each input/output ciphertext chunk. The search accounts for input baby rotations already used by B and the additional baby and giant rotations required by S. For each fitted candidate, the backend retunes the BSGS grid and compares candidates lexicographically by $(R_{\mathrm{sched}}, N_{\mathrm{PMult}}, k)$, where R_{sched} includes auxiliary layout rotations and N_{PMult} counts ciphertext–plaintext multiplications. This is a heuristic search and does not guarantee a globally optimal schedule.

Within a fixed BSGS grid, a baby-step rotation disappears only when no surviving term uses it, and a giant-step rotation disappears when its weighted group becomes empty. Each candidate removal

Table 6: Key hyperparameters and implementation settings.

Setting	ViT-Tiny	GPT-2	Llama-3.1-8B
Decomposition and recovery			
Rank-grid step Δk	3	3	32
Energy-threshold floor $\tau_{\min}$	0.1	0.1	0.1
Alternating-fit iterations	4	4	2
Absolute block-skipping tolerance	—	1.0 PPL	0.15 NLL
Relative block-skipping tolerance	—	0.05	—
Validation batches for skipping	—	30	8
Maximum skipped blocks	—	No explicit cap	10
Micro-batch size $\times$ gradient accumulation	64×1	4×8	2×16
Sequence length	—	256	512
Progressive recovery epochs	—	3/4/5/7 FT	—
Progressive recovery learning rate	—	5×10^{-5}	—
Final recovery epochs	15 FT	25 FT + 5 LoRA	1 LoRA
Final FT learning rate	10^{-4}	1.5×10^{-5}	—
LoRA rank / scaling parameter / dropout	—	32/64/0.05	32/64/0.05
LoRA learning rate	—	2×10^{-4}	2×10^{-4}
Multi-anchor rotation-aware pruning			
Pruning ratios ρ	$\{0, 0.1, 0.2\}$	$\{0, 0.1, 0.2\}$	$\{0, 0.1, 0.2\}$
Dense-projection quota multiplier γ	0.5	0.5	0.5
Calibration batches per rescoring stage	4	4	16
Rescoring interval $\Delta\rho$	0.1	0.1	0.05
Pruning recovery epochs / learning rate	$8/10^{-4}$	$8/(1.5 \times 10^{-5})$	1 LoRA / 2×10^{-4}
HE latency implementation			
OpenFHE evaluation scope	ViT-Tiny encrypted tokens to encrypted logits		
OpenFHE ring degree / slots	$2^{17}/2^{16}$		
OpenFHE scaling / key switching	FLEXIBLEAUTO / HYBRID (3 large digits)		
OpenFHE scale / first-modulus bits	59/60		
OpenFHE depth / requested post-bootstrap levels	42/20		
OpenFHE bootstrap level budgets	$(4, 4)$		
OpenFHE BSGS baby-step selection	Structural auto-selection: joint S/B, separate A		
NEXUS evaluation scope	GPT-2 single-projection PC-MM		
NEXUS ring degree / coefficient-modulus bits	$2^{13}/(60, 40, 60)$		
NEXUS activation / packed-coefficient scales	$2^{40}/10^{10}$		

Notes. Recovery uses AdamW with seed 42; Llama keeps S frozen. GPT-2's progressive recovery epochs correspond to blocks 1–3, 4–6, 7–9, and 10–12. Its skipping criterion is $\Delta\mathrm{PPL} > \max(1.0, 0.05\,\mathrm{PPL}_{\mathrm{pre}})$, evaluated after progressive recovery; Llama evaluates skipping before LoRA recovery. OpenFHE uses `HEStd_128_classic`. NEXUS uses independent PC-MM passes with an offline plaintext intermediate and no security-level enforcement, not encrypted E2E inference.

is evaluated by retuning the BSGS grid, and only candidates with $\Delta R_{\mathrm{sched}} > 0$ are considered. Candidates are ranked by

$$\frac{\|M_{\mathrm{before}} - M_{\mathrm{after}}\|_F^2}{\|W\|_F^2\, \Delta R_{\mathrm{sched}}},$$

where M denotes the current projection matrix, equal to $S + AB$ for a factorized projection, and $\Delta R_{\mathrm{sched}}$ is the decrease in scheduled rotations after replanning. The score measures the change in effective weights per saved scheduled rotation, and candidates with lower scores are selected first.

In the ViT implementation, each ciphertext contains 256 feature positions with 256 token lanes each, with slot index $\phi(f, u) = 256f + u$. A feature rotation by d corresponds to a slot rotation by $256d$. The intermediate rank is padded to $p = 2^{\lceil \log_2 k \rceil}$, with $p \mid 256$ for the supported ranks $1 \le k \le 256$.

For example, the 192×768 FC2 projection with rank 48 uses three input ciphertexts and a padded rank of 64. With baby-step width $G = 16$, the coefficients $S[191, 0]$ and $B[47, 0]$ use physical

Table 7: Illustrative theoretical comparison for a single decomposed 4096×4096 Transformer projection layer under standard diagonal-wise packing. The DSLR operator uses $k = 256$ with the full reuse pool and no extra offsets ($t = k$, $e = 0$).

Metric	Dense baseline	RACE-HE
Operator form	W	$S + AB$
Active diagonals	4096	768
Nontrivial rotations	4095	510
Distinct rotation keys	4095	255
Rotation reduction	–	87.5%
Rotation key reduction	–	93.8%

Table 8: Measured HE latency for a single GPT-2 projection with OpenFHE. RACE-HE uses the BSGS-aware selection and pruning variant adapted from ***GPT-C3***, with pruning target $\rho_{\mathrm{BSGS}} = 0.2$. Both methods use BSGS scheduling.

Metric	**Dense**	**RACE-HE**
Scheduled rotation calls	156	133
Ciphertext–plaintext multiplications	2,304	1,305
Levels consumed	1	1
PC-MM latency (s)	157.84	**98.08** (1.61×)

Note. GPT-2 block-0 attention output projection (768×768, 256 tokens, RACE-HE rank 90). OpenFHE 1.4.2, ring dimension $N = 2^{17}$, multiplicative-depth parameter 4, without bootstrapping. Latency is the median of three timed runs after one warm-up. Timing includes online plaintext-mask construction and encoding, but excludes cryptographic setup, input encryption, and output decryption/verification. These are projection-level, not full-model end-to-end, measurements. The pruning target is defined relative to the scheduled rotation count of the unpruned DSLR configuration.

feature offsets 65 and 17, respectively. Both offsets have baby step 1, so the two branches reuse the same cached $\mathsf{Rot}(X_0, 256)$, while applying separate giant-step rotations. After summing the B outputs across input chunks, two rotate-and-add operations with slot offsets 16,384 and 32,768 fold the partial results and produce the repeated rank vector consumed by A. The A output aligns with Sx, so this FC2 construction requires no additional output repacking.

All 11 synthetic full-slot arithmetic tests passed, with a maximum absolute discrepancy below 3×10^{-15} against ordinary matrix multiplication, and the operation counts matched the backend planner. These tests validate the slot mapping and operation counting, rather than CKKS numerical precision or task quality. Table 3 reports the measured ViT-Tiny costs, including all auxiliary layout rotations. The accompanying code provides the BSGS planner, selection and pruning implementation, and slot-level arithmetic tests.

Non-diagonal packing layouts. The benefit of RACE-HE is not limited to diagonal-wise packing. In JKLS-style layouts, large plaintext matrices are partitioned into sub-blocks or tiles, as the full matrix cannot be mapped efficiently into the available ciphertext slots under a single packed representation. Factorizing a dense $m \times n$ matrix into factors of sizes $m \times k$ and $k \times n$ reduces the effective operand dimensions whenever $mk + kn \ll mn$, so that fewer tiles are needed to represent the operator, which in turn reduces the encoded ciphertexts, homomorphic products, and data movement of the packing scheme. The diagonal term adds $(t + e)\, n$ entries, at most a few tiles at the values of $t + e$ selected in our configurations. Therefore, the structural reduction carries over to these layouts as well.

Execution-layer systems. The same substitution extends to system-level execution frameworks. NEXUS (Zhang et al., 2024), the state-of-the-art non-interactive Transformer inference system, optimizes the execution layer through ciphertext compression, amortized offline–online PC-MM, sign-based non-linear kernels, and bootstrapping placement, while leaving the dense weight operator unchanged. Its published cost model (Table II of Zhang et al. (2024)) prices the weight-dependent components by entry count and measures compute in key switchings, which are cost-equivalent to

Table 9: Analytical composition of RACE-HE with NEXUS's amortized PC-MM protocol (Zhang et al., 2024), following the accounting of their Table II ($m_{\text{tok}} = 128$, $N = 4096$, $t_{\text{amo}} = 256$). Formula rows price the low-rank branch, and the numeric rows price the full DSLR operator, with the diagonal term entering as a third PC-MM stage of $(t + e)\, d_{\text{in}}$ entries.

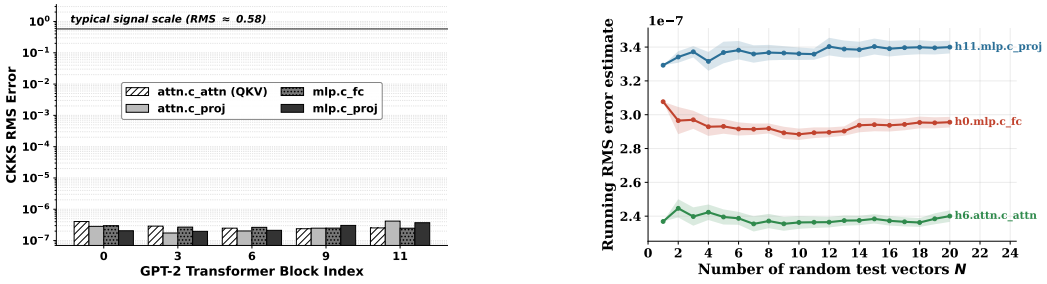
		Dense	**RACE-HE**
Cost (amortized)	Weight ciphertexts	$d_{\text{in}} d_{\text{out}}/(N t_{\text{amo}})$	$k(d_{\text{in}} + d_{\text{out}})/(N t_{\text{amo}})$
	Decompression key switchings	$(2\, d_{\text{in}} d_{\text{out}} - 2)/t_{\text{amo}}$	$\big(2k(d_{\text{in}} + d_{\text{out}}) - 4\big)/t_{\text{amo}}$
	Result ciphertexts	$m_{\text{tok}}\, d_{\text{out}}/N$	$m_{\text{tok}}(k + d_{\text{out}})/N$
GPT-2 (768×768, avg $k = 161.8$)	Weight ciphertexts	0.56	0.36
	Decompression key switchings	4,608	2,928 (36.5%↓)
	Result ciphertexts	24	29.1
Llama 3.1-8B (4096×4096, $k = 256$)	Weight ciphertexts	16	3.0
	Decompression key switchings	131,072	24,576 (81.2%↓)
	Result ciphertexts	128	136

ciphertext rotations, and therefore a DSLR operator substitutes without reinterpreting the protocol. The low-rank branch is priced as two PC-MM stages, and since coefficient-based packing has no per-diagonal structure, the diagonal term enters not at $t + e$ diagonals but as a sparse third stage of $(t + e)\, d_{\text{in}}$ entries evaluated in parallel with B, with diagonal pruning transferring only through the entries it removes. Entry pricing is thus the realized-cost analogue of the active-diagonal count, the second lexicographic selection criterion in Algorithm 1, and its benefit condition over the dense operator is $C_{\text{diag}} < q$ for a square layer, which every selected configuration satisfies. Table 9 reports the composed costs with the diagonal term included, contributing about one third of the entries at the selected structures, where $t \approx k$. For GPT-2 at the average rank of the decomposed projections of *GPT-C3*, the composed key switchings fall by 36.5%, and for the Llama worked example of Appendix C.3 by 81.2%. The analytical savings depend on the retained-rank ratio and the size of the diagonal term. For GPT-2, the estimate uses the average rank of the decomposed projections in *GPT-C3* ($k/d \approx 0.21$), whereas the Llama estimate uses the illustrative rank-256 configuration of Appendix C.3 ($k/d = 0.0625$). These examples illustrate how the assumed compressed structures affect the entry-based cost. The low-rank branch also introduces intermediate ciphertexts and one additional sequential PC-MM stage, whose overhead depends on how the two stages are integrated. All values are analytical estimates, and being amortized over t_{amo} inferences, they are not comparable to the per-inference rotations of Table 1. One scoping remark is in order. At a matched direct rotation count, a pure low-rank operator holds fewer entries than DSLR, and therefore the advantage of DSLR over truncated SVD reported in Section 4.1 is specific to rotation-dominated execution under diagonal packing, whereas the gain over the dense operator carries over to entry-priced layers as shown here. Finally, NEXUS's own breakdown attributes 62.3% of its end-to-end runtime to bootstrapping, indicating that multiplicative depth remains a dominant cost even after optimizing the execution layer, which motivates the split-scale encoding of Section 3.1 that keeps the two-stage branch within the one-stage depth.

D Calibration of CKKS Error Simulation

This appendix details the layer-wise CKKS error measurement used to calibrate the full-dataset simulation in Section 4. For each selected configuration, we execute the decomposed linear layers through the actual CKKS pipeline implemented in OpenFHE. We use a representative subset of $N = 20$ test vectors from the corresponding evaluation dataset and record the RMS error between the decrypted CKKS output and the exact plaintext output for each layer.

Figure 5(a) shows the per-block RMS error for GPT-2, separated by the four linear projection types. Across all blocks, the measured CKKS error remains on the order of 10^{-7}, with no anomalous layer-level error increase. The other evaluated models show similarly low error levels. Figure 5(b) further shows that the per-layer error estimate is already stable at small sample counts, remaining essentially flat as N_{test} increases. This is expected, since each test vector already averages the RMS error over hundreds of output coordinates, and the CKKS noise is largely input-independent. Hence, even a handful of test vectors yields a stable estimate. The measurement is therefore not sensitive to the sample count, supporting our use of $N = 20$ for calibration.



(a) (b)

Figure 5: (a) Per-block CKKS RMS error for GPT-2 under GPT-C1 ($\tau_{\text{start}} = 0.60$, $\alpha = 0.20$), by linear projection type. (b) Per-layer RMS error variance as a function of the number of test vectors N_{test}.

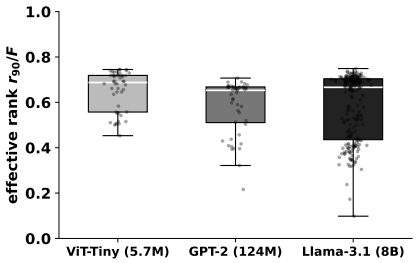


Figure 6: Distribution of per-projection effective rank $r_{90}/\min(m,n)$, where r_{90} is the number of singular components required to preserve 90% of the squared singular-value energy. Each point is one linear projection, boxes mark the interquartile range, and white lines mark the median.

Using these measurements, we calibrate the inference simulator by injecting zero-mean Gaussian noise with the measured layer-wise variance into the corresponding layers. Since the measured CKKS error is several orders of magnitude smaller than the layer outputs in the evaluated configurations, this injected noise has a negligible impact on task quality. This is consistent with the close match between plaintext-domain and calibrated ciphertext-domain results reported in Section 4. For instance, Table 1 shows that the calibrated CT perplexity differs from the PT perplexity by at most 0.1 across the selected language model configurations.

E Rank Compressibility across Model Scales

To test whether the differing reductions in Table 1 stem from a structural property of the weights, we measure the effective rank $r_{90}/\min(m,n)$ for each projection, where r_{90} is the smallest rank that preserves 90% of the squared singular value energy. As Figure 6 shows, the median effective rank is comparable across all three models, and Llama 3.1-8B even contains a larger low-rank tail. The differing reductions therefore reflect the chosen quality budgets rather than a hard limit imposed by the weight spectra. Perplexity is substantially more sensitive to perturbations in individual layers than classification accuracy, partly since activation outliers amplify low-rank approximation error (Dettmers et al., 2022; Yuan et al., 2023).

F Deployment Considerations and Future Directions

F.1 Deployment-Dependent Rotation Key Cost

The main evaluation assumes a non-interactive deployment in which the client uploads the encrypted input and the required rotation keys once. Under direct-diagonal accounting, the resident key set is indexed by the union of rotation offsets across all projections. For GPT-2, the unpruned dense output

Table 10: Analytical rotation-key transfer under stateless per-projection streaming for GPT-2 ($\rho = 0.2$, $\gamma = 0.5$, including the output head). The measured full-chain key payload is 348.01 MiB per key.

Configuration	Dense baseline (GiB)	RACE-HE (GiB)	Reduction (%)
GPT-C1	12,772.8	8,241.8	35.5
GPT-C2	12,772.8	6,670.0	47.8
GPT-C3	12,772.8	4,125.8	67.7

head retains the full offset range, and therefore compression does not reduce this resident key count. This appendix considers an alternative deployment in which keys are streamed separately for each projection.

We assume stateless per-projection streaming: immediately before evaluating a projection, the server receives its required key subset, with no key reuse across projections. Here, a streaming layer denotes one logical projection, not a Transformer block or an individual factor evaluation. All 49 GPT-2 projections, including the output head, are counted.

For a DSLR projection, the required key set is

$$K_l = (\Omega_{S_l} \cup \Omega_{B_l} \cup \Omega_{A_l}) \setminus \{0\}.$$

Before pruning, a positive-rank DSLR projection requires $|K_l| = k_l - 1 + e_l$ keys. After pruning, we count the surviving union directly. The same offset key can serve both factor stages even though they rotate different ciphertexts; consequently, key counts need not equal direct rotation-operation counts.

Under the split-budget pruning policy, rotation units of factorized projections share a global pruning budget. Each skipped dense projection instead removes $\lfloor \gamma\rho(q_l - 1) \rfloor$ nontrivial diagonals, leaving

$$|K_l| = (q_l - 1) - \lfloor \gamma\rho(q_l - 1) \rfloor, \qquad \gamma = 0.5.$$

The output head remains unpruned. The dense baseline uses $q_l - 1$ keys for every projection.

For a serialized per-key payload of s_{key} MiB, the cumulative transfer is

$$T_{\text{stream}}\ [\text{GiB}] = \frac{s_{\text{key}}}{1024} \sum_l |K_l|.$$

We count only direct wrapped-diagonal offsets and exclude BSGS, replication, folding, repacking, and other layout-specific keys from both methods. Small fixed serialization headers are also excluded.

Table 10 reports analytical transfer volumes for GPT-C1/C2/C3 at $\rho = 0.2$. We use the measured full-chain binary key payload of 348.01 MiB from OpenFHE 1.4.2 with $N = 131072$, multiplicative depth 42, and HYBRID key switching with three large digits, using the CKKS context of Section 4.3. Applying this payload to GPT-2 assumes deployment under that same context; these volumes are analytical estimates, not measured network transfers.

Relative to the dense baseline, cumulative key transfer decreases by 35.5%, 47.8%, and 67.7% for GPT-C1, GPT-C2, and GPT-C3, respectively. For GPT-C3 ($\tau_{start} = 0.80$, $\alpha = 0.20$), the cumulative key count falls from $37{,}583$ to $12{,}140$, reducing the transfer volume from $12{,}772.8$ to $4{,}125.8$ GiB.

Overlap-Aware Pruning for Key Reduction. The pruning of Section 3.2 targets rotation invocations. Its input-side units already remove an offset from S_l and B_l together, but the A-side unit at the same offset is scored and removed independently, since in the non-interactive setting the resident key set is fixed by the model-wide union of offsets. Under per-layer key streaming, the key set of a layer is $\Omega_{S_l} \cup \Omega_{B_l} \cup \Omega_{A_l}$. Hence, an offset stops requiring a key only when both its input-side unit and its A-side unit are removed. Writing (A, ℓ) and (in, ℓ) for the two units of a projection at offset ℓ, the selection can be extended with an overlap reward,

$$\min_{\mathcal{P}} \sum_{u \in \mathcal{P}} \mathcal{S}(u) \;-\; \lambda \left| \{\ell : (A, \ell) \in \mathcal{P},\ (\text{in}, \ell) \in \mathcal{P}\} \right| \qquad \text{s.t.} \qquad |\mathcal{P}| = K, \tag{8}$$

where $\mathcal{S}(u)$ is the score of Eq. 3 and $\lambda \geq 0$ rewards removing the same offset on both sides of a projection. Setting $\lambda = 0$ recovers the rotation-count objective of the main evaluation. We leave the evaluation of this variant in key-streaming deployments to future work.

F.2 Toward Optimal Allocation Schedules

RACE-HE deliberately uses simple, interpretable rules for its allocation decisions. The preserved-energy threshold follows a linear schedule over block depth, and each projection is then structured independently under its block's bound by the lexicographic selection rule in Algorithm 1, implemented approximately by a nested search with alternating fits. Pruning removes rotation units along a greedy nested path. These rules perform well empirically and avoid expensive search, but none of them is globally optimal. In particular, the per-projection reconstruction bound is a surrogate for end-to-end task quality. Thus, a joint allocation across projections and blocks, informed by singular-value spectra, activation sensitivity, and downstream error propagation, could reach better trade-offs among task quality, rotation count, and active-diagonal cost. Formulating these allocations as a principled optimization problem is a natural direction for future work.