ROCKET: Rapid Optimization via Calibration-guided Knapsack-Enhanced Truncation for Efficient Model Compression

Ammar Ali[1,2*], **Baher Mohammad**[1,2*], **Denis Makhov**[1,2],
Dmitriy Shopkhoev[1], **Magauiya Zhussip**[1,2], **Stamatios Lefkimmiatis**[1,2]
[1]MWS AI [2]ITMO
ammarali32@itmo.ru

Abstract

We present **ROCKET**, a training-free model compression method that achieves state-of-the-art performance compared with factorization, structured-sparsification, and dynamic compression baselines. Operating under a global compression budget, ROCKET comprises two key innovations. First, it formulates layer-wise compression allocation as a multi-choice knapsack problem, selecting the best compression configuration for each layer to minimize total reconstruction error while adhering to a target model size. Second, it introduces a single-pass sparse matrix factorization that executes what is, conceptually, one iteration of the classical alternating dictionary-learning template with both half-steps replaced by closed-form, calibration-aware computations: a spectral initialization followed by importance-weighted hard thresholding that accounts for the metric distortion between whitened and original weight spaces, and a closed-form dictionary update via least squares, bypassing iterative optimization, sparse coding, or backpropagation entirely. ROCKET consistently outperforms existing compression approaches across different model architectures at 20–50% compression rates. Notably, it retains over 90% of the model's performance at 30% compression without any fine-tuning. Code implementing ROCKET is available at `https://github.com/mts-ai/ROCKET/tree/main`.

1 Introduction

Transformers have achieved unprecedented success across vision and language tasks, but modern LLMs now scale to billions and trillions of parameters, imposing prohibitive computational and memory demands that hinder deployment on edge devices and latency-sensitive applications. To address this, research focuses on three primary compression strategies: pruning [12], distillation [15], and post-training weight factorization [39]. Among these, post-training weight factorization has emerged as a particularly promising direction, enabling substantial parameter reduction without the need for costly retraining or fine-tuning. A dominant paradigm in this area is low-rank approximation using truncated Singular Value Decomposition (SVD), which approximates each weight matrix as the product of two smaller dense matrices. However, this strategy imposes a rigid structural constraint forcing all columns of the weight matrix to lie in a single shared low-dimensional subspace. This often limits the representational capacity and leads to significant performance degradation under moderate to high compression. This limitation has spurred the development of methods that go beyond a single shared subspace representation, adopting instead a union-of-subspaces framework akin to dictionary learning. In such models, a weight matrix is expressed as a combination of a subset of basis

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

matrices [48], or alternatively, its individual columns are represented as sparse linear combinations of atoms from a shared dictionary [23]. These formulations provide greater flexibility by capturing the heterogeneous local structures present within the weight matrix. Despite their theoretical appeal, practical adoption of these methods faces severe computational challenges: conventional dictionary learning algorithms rely on iterative alternating minimization between sparse coding and dictionary update steps [1], which do not scale well for large-scale LLM weight matrices.

In this work, we propose **ROCKET**, a fast, training-free compression method that addresses the representational rigidity of low-rank factorization while avoiding the computational burden of iterative dictionary learning. Our approach introduces two key contributions. First, ROCKET compresses weight matrices via what is, conceptually, a **single iteration of the classical alternating dictionary-learning template**, in which both half-steps are replaced by closed-form, calibration-aware computations. This yields a factorization that inherits the expressive power of union-of-subspaces models yet operates substantially faster than alternating minimization. Second, rather than applying uniform compression or relying on heuristic layer-wise sensitivity estimates, ROCKET formulates global compression allocation as a multi-choice knapsack problem. For each layer, it selects the best compression configuration from a set of precomputed candidates to minimize total weight reconstruction error under a target model size. Together, these components enable ROCKET to produce compact models that achieve substantially higher accuracy compared to existing post-training compression methods.

The contributions of this work are summarized as follows: (1) We propose ROCKET, an efficient, training-free LLM compression method that factorizes weight matrices into a sparse dictionary representation computable in a single closed-form pass, without iterative optimization; (2) We introduce a calibration-guided criterion for sparsifying the coefficient matrix that accounts for the metric distortion between original and whitened weight spaces to preserve salient directional information; (3) We formulate layer-wise compression allocation as a multi-choice knapsack problem, enabling dynamic, performance-aware distribution of the global compression budget across layers; (4) Through extensive experiments, we demonstrate that ROCKET consistently outperforms state-of-the-art compression methods including structured sparsification, low-rank factorization, and adaptive budget allocation techniques across multiple modalities (text, vision, and audio).

2 Related Work

This work intersects three primary research directions in model compression: (1) dynamic per-layer allocation of compression budgets, (2) structured matrix factorization for weight approximation, and (3) sparsification techniques. We review recent advances in each area, with emphasis on methods most relevant to our training-free, reconstruction-aware compression framework.

Structured Matrix Factorization for Weight Approximation Early approaches employed truncated SVD to obtain low-rank approximations of transformer weights. However, several studies [5, 39, 44] demonstrated that weight matrices themselves are not inherently low-rank; instead, activations exhibit low-rank structure. These works proposed data-aware low-rank approximations using a whitening transform estimated from a small calibration dataset, yielding significantly more effective compression. A more general representation was recently introduced in [23], where weights are expressed as sparse linear combinations of dictionary atoms in a whitened space, computed via K-SVD and Orthogonal Matching Pursuit (OMP) updates. This approach overcomes the limitation of fixed, layer-invariant bases in low-rank methods since it allows each column of the weight matrix to reside in a different low-dimensional subspace, effectively promoting a more flexible union-of-subspaces modeling strategy. ROCKET bridges SVD efficiency and dictionary-learning expressiveness [1, 8] through a single closed-form, calibration-aware approximation.

Budget Allocation and Layer Importance Many early compression methods apply uniform compression across all layers, implicitly assuming equal layer importance. Recent work challenges this assumption. LLM-Pruner [21] estimates the importance of coupled layer groups using gradient and Hessian-based metrics, pruning less critical groups. ARS [10] proposes an adaptive rank selection mechanism using differentiable binary masks, regularized to respect the ordering of singular values from SVD, thereby allocating more capacity to important layers. Dobi-SVD [38] introduces a learnable truncation threshold k per weight matrix, optimized during training via a multi-objective loss balancing task performance and global compression ratio. Similarly, ARA [42] dynamically assigns ranks to linear modules by learning a monotonic probabilistic mask over singular values, guided by a loss that accounts for cases where full-rank retention is more efficient than decomposition. Hardware-aware pruning methods such as HALP [32] also formulate compression as a constrained selection

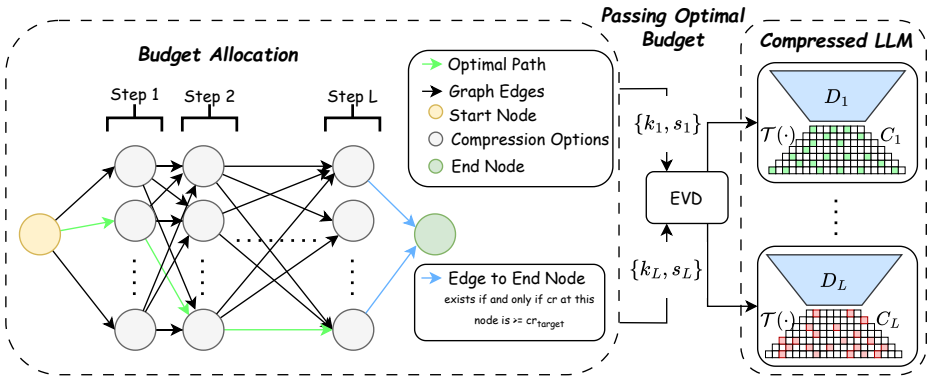


Figure 1: Overview of the proposed method. **Left:** Budget allocation formulated as a shortest-path problem on a directed graph, where nodes represent compression options and edges encode cost (reconstruction error), solved via dynamic programming to find the best sequence of operations over the discretized candidate set. **Right:** The selected path determines per-layer compression parameters (rank k_ℓ and sparsity s_ℓ), which are then applied to each layer via eigendecomposition followed by structured hard thresholding sparsification ($\mathcal{T}(\cdot)$) of coefficients.

problem, using latency estimates and saliency scores to choose pruned structures. These methods are typically training-aware and operate on binary prune/retain decisions over channels or filters. In contrast, our method performs allocation in a purely post-training setting via a multi-choice knapsack formulation. Using dynamic programming (DP), we select the best configuration over the evaluated candidates minimizing total reconstruction error under a per-layer error cap. Unlike prior DP approaches for CNN sparsity [9] that rely on heuristic sensitivity scores, ROCKET uses an analytical, calibration-guided error from activation-aware eigendecomposition. Our dual-constraint formulation maps to an exact shortest-path problem on a Directed Acyclic Graph (DAG) with dominated-state pruning, yielding reduced computational cost in practice.

Sparsification Methods We emphasize that ROCKET is not a pruning method: sparsity is applied to the coefficient matrix factor of the weights, not to their original entries, and the resulting product $\mathbf{FH}$ is preserved for inference. While structured sparse factorizations offer native hardware compatibility, recent studies show that unstructured sparsity starting at roughly 30% can also deliver practical speedups [22], highlighting a growing area of interest.

3 Method

Recent training-free compression methods for LLMs exhibit a fundamental trade-off between computational efficiency and representational flexibility. On one end, truncated SVD (SVD-LLM; [39]) enforces a rigid, globally shared low-rank subspace where compression is fast, but reconstruction fidelity degrades severely under aggressive ratios. On the other end, CoSpaDi [23] employs sparse dictionary learning (K-SVD + OMP [1]) to realize a union-of-subspaces model, at the cost of a runtime that scales poorly to multi-billion-parameter models (646 min for Llama3.2-1B). ROCKET bridges this gap through the half-step replacement with the closed-form updates; and its allocation stage distributes a global parameter budget across layers by solving a constrained knapsack problem over the profiled candidates (Sections 3.5–3.5).

3.1 Problem Setup and Whitened Activation Space

Given a calibration dataset $\mathbf{X} \in \mathbb{R}^{N \times d_1}$ and a weight matrix $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$, we seek an approximation $\widetilde{\mathbf{W}}$ that minimizes the activation-aware reconstruction error

$$\underset{\widetilde{\mathbf{W}} \in \mathcal{C}}{\arg\min} \left\| \mathbf{X}\mathbf{W} - \mathbf{X}\widetilde{\mathbf{W}} \right\|_F \equiv \operatorname{tr}\left(\left(\mathbf{W} - \widetilde{\mathbf{W}} \right)^\top \mathbf{X}^\top \mathbf{X} \left(\mathbf{W} - \widetilde{\mathbf{W}} \right) \right)^{\frac{1}{2}} \tag{1}$$

where $\mathcal{C}$ imposes structural constraints on $\widetilde{\mathbf{W}}$. Following standard practices in data-aware compression, we consider the regularized Gram matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X} + \epsilon \mathbf{I}$ ($\epsilon = 10^{-6}$), to ensure that it is positive definite, and we define $\mathbf{L}$ to be its Cholesky factor, that is $\mathbf{A} = \mathbf{L}^\top \mathbf{L}$. Then the original

objective above can be equivalently expressed as $\left\|\mathbf{W}_L - \widetilde{\mathbf{W}}_L\right\|_F$ with $\widetilde{\mathbf{W}}_L = \mathbf{L}\widetilde{\mathbf{W}}$. This problem reformulation allows us to work directly on the whitened activation space, which encodes all necessary information from the calibration data.

3.2 Dictionary Learning without the Loop

In the whitened space the structural constraint takes the form of a sparse dictionary factorization $\widetilde{\mathbf{W}}_L = \mathbf{DS}$:

$$\min_{\mathbf{D}\in\mathbb{R}^{d_1\times k},\, \mathbf{S}\in\mathbb{R}^{k\times d_2}} \|\mathbf{W}_L - \mathbf{DS}\|_F^2 \quad \text{s.t.} \quad \mathbf{S}_{:j} \text{ is } s\text{-sparse for every column } j. \tag{2}$$

The joint problem equation 2 is non-convex and is classically solved by *alternating minimization* between (a) a sparse coding step (updating $\mathbf{S}$ for fixed $\mathbf{D}$), attacked greedily by OMP-type pursuit [37], and (b) a dictionary update (updating $\mathbf{D}$ for fixed $\mathbf{S}$), either via the Method of Optimal Directions (MOD [8]) or by K-SVD [1]. Both half-steps are expensive and many alternations are required; this is precisely what limits the scalability of CoSpaDi [23].

ROCKET is built on a simple observation:

Observation 1 (Column-orthogonal dictionaries make sparse-coding half-step closed-form)**.** If $\mathbf{D}^\top\mathbf{D} = \mathbf{I}_k$, then (a) the sparse coding step of equation 2 decouples across columns and is solved by hard thresholding (Appendix C.4), and (b) dropping the imposed orthogonality constraint, the dictionary update is a ridge least-squares problem with a closed-form solution (Section 3.5).

ROCKET therefore executes *exactly one* iteration of the alternating template, with each half-step replaced by its closed-form, calibration-aware counterpart:

[**S1**] *Spectral initialization:* the optimal column-orthogonal dictionary for the requested rank is computed, with the coefficients obtained as orthogonal projections of the whitened weights. (Section 3.3);

[**S2**] *Calibrated hard thresholding:* An importance-weighted thresholding rule that accounts for the un-whitening distortion is applied on the coefficients. (Section 3.4);

[**S3**] *Dictionary refinement:* MOD dictionary update is performed with the orthogonality constraint being dropped; this constraint was initially imposed solely to simplify the sparse-coding step and it does not represent a modeling requirement (Section 3.5).

The method deliberately terminates after this single pass: step S3 destroys the column-orthogonality that made step S2 a closed-form solver, so any subsequent sparse-coding step would require iterative pursuit reintroducing precisely the computational cost that ROCKET eliminates.

3.3 Spectral Initialization

We compute the top-k eigenvectors of $\mathbf{W}_L\mathbf{W}_L^\top$:

$$\mathbf{W}_L\mathbf{W}_L^\top \approx \mathbf{B}\mathbf{\Lambda}_k\mathbf{B}^\top, \qquad \mathbf{B}\in\mathbb{R}^{d_1\times k},\ \mathbf{B}^\top\mathbf{B} = \mathbf{I}_k, \qquad \mathbf{C} := \mathbf{B}^\top\mathbf{W}_L \in \mathbb{R}^{k\times d_2}, \tag{3}$$

so that $\mathbf{W}_L \approx \mathbf{BC}$. By the Eckart–Young–Mirsky theorem, $\mathbf{B}$ is the optimal column-orthogonal basis of rank k: no rank-k approximation achieves a smaller whitened reconstruction error, and the associated coefficients $\mathbf{C}$ come for free as orthogonal projections, with no pursuit required.

Because $\mathbf{B}$ is semi-orthogonal, zeroing c_{ij} deactivates the i-th latent direction for the j-th output dimension. Since sparsity is applied independently per column, each output may activate a *distinct* subset of basis vectors, realizing a union-of-subspaces model similar to dictionary learning, while retaining a single shared basis $\mathbf{B}$ that is reused across all candidate configurations (Section 3.5).

3.4 Support Selection by Calibrated Hard Thresholding

Why plain thresholding is not enough. For an orthonormal basis $\mathbf{B}$, the whitened reconstruction error satisfies the Pythagorean decomposition $\|\mathbf{W}_L - \mathbf{BS}\|_F^2 = \|\mathbf{W}_L - \mathbf{BC}\|_F^2 + \|\mathbf{C} - \mathbf{S}\|_F^2$, so hard thresholding of $|c_{ij}|$ is the exact best-s-term solver [7, 24]. However, the whitening transform is estimated from a finite calibration set, so it captures only the observed statistics of the input activations and cannot fully cover the input distribution. To avoid relying exclusively on this

calibration-dependent whitened geometry, we balance reconstruction fidelity in the whitened space with reconstruction quality in the original weight space. Since $\mathbf{L}^{-1}$ is generally non-orthogonal, two coefficients with identical whitened magnitudes may contribute very differently to the reconstructed weight matrix. Specifically, an error of unit whitened norm along $\mathbf{b}_i$ incurs an original-space cost $a_i := \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2$.

Fused importance score. To correct for this metric mismatch we define two complementary importance measures:

$$\mathrm{imp}_{ij}^{\mathrm{white}} = |c_{ij}|, \qquad \mathrm{imp}_{ij}^{\mathrm{orig}} = |c_{ij}| \cdot \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2, \tag{4}$$

and fuse them via a scale-invariant geometric interpolation with balance parameter $\lambda \in [0, 1]$:

$$\mathrm{imp}_{ij} = \left(\mathrm{imp}_{ij}^{\mathrm{white}}\right)^{1-\lambda}\left(\mathrm{imp}_{ij}^{\mathrm{orig}}\right)^{\lambda} = |c_{ij}| \cdot \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2^{\lambda}. \tag{5}$$

Let $\boldsymbol{\nu} \in \mathbb{R}^k$ with $\nu_i = \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2^{\lambda}$; the full importance matrix is

$$\mathbf{Imp} = |\mathbf{C}| \odot \left(\boldsymbol{\nu}\,\mathbf{1}_{d_2}^{\top}\right). \tag{6}$$

This scoring is the solution of a weighted sparse approximation problem in which the weights encode the distortion induced by $\mathbf{L}^{-1}$; the derivation, the exact quadratic form, and its diagonal surrogate are given in Appendix C.5. The surrogate neglects off-diagonal couplings in $\mathbf{G} = \mathbf{B}^{\top}\mathbf{A}^{-1}\mathbf{B}$; these couplings vanish when the un-whitened atoms $\{\mathbf{L}^{-1}\mathbf{b}_i\}$ are mutually orthogonal, and empirically the trade-off is favourable (Section 5).

Two-stage thresholding. We apply column-wise hard thresholding on $\mathbf{C}$, retaining the top-s entries per column according to imp_{ij}. To allow global flexibility beyond fixed per-column budgets, we initially over-sparsify $\mathbf{C}$ by a small percentage γ ($\gamma = 5 \times 10^{-3}$ in all experiments), then reactivate the most important masked coefficients across the entire matrix until the exact target compression ratio cr is reached. For every fixed λ, thresholding by imp_{ij} is the optimizer of the stated weighted objective; the selection step never requires a greedy pursuit, in contrast to OMP-based coding for general dictionaries. This yields the sparse coefficient matrix $\mathbf{C}_{\mathrm{sparse}}$. We set $\lambda = 0.5$ as default; the full sensitivity analysis over $\lambda \in [0, 1]$ is provided in Section 5.

3.5 Dictionary Refinement

Orthogonality of $\mathbf{B}$ was needed only to make the thresholding step closed-form. Once the support of $\mathbf{C}_{\mathrm{sparse}}$ is fixed, the dictionary can be fit freely. We compute the final dictionary $\mathbf{D}_{\mathrm{final}}$ by solving a ridge-regularized least-squares problem:

$$\mathbf{D}_{\mathrm{final}} = \arg\min_{\mathbf{D}} \ \|\mathbf{W}_L - \mathbf{D}\mathbf{C}_{\mathrm{sparse}}\|_F^2 + \mu \|\mathbf{D}\|_F^2, \tag{7}$$

which admits a closed-form Cholesky-based solution; the adaptive ridge μ is specified in Appendix H.4. This is the MOD dictionary update [8] computed for the fixed code $\mathbf{C}_{\mathrm{sparse}}$: whereas K-SVD updates atoms sequentially, ROCKET evaluates the joint minimizer given the codes, yielding a better fit without increasing inference cost. The final compressed weight is recovered as

$$\widetilde{\mathbf{W}} = \mathbf{L}^{-1}\mathbf{D}_{\mathrm{final}}\mathbf{C}_{\mathrm{sparse}}, \qquad \text{stored as } \ \mathbf{F} = \mathbf{L}^{-1}\mathbf{D}_{\mathrm{final}}, \ \ \mathbf{H} = \mathbf{C}_{\mathrm{sparse}}. \tag{8}$$

When no sparsity is applied ($s = k$), thresholding reduces to the identity operator and ROCKET recovers standard data-aware low-rank SVD (Appendix C.6).

Because each step is closed-form and the basis is computed once per layer, the entire factorization of a 448-layer model completes in under 2 minutes on a single CPU thread (Appendix F), compared to 646 min for iterative K-SVD/OMP (CoSpaDi).

Compression-ratio convention and parameter counting. Throughout the paper, the compression ratio (CR) denotes the fraction of parameters *removed*. Thus, $\mathrm{CR} = 0.2$ means that 20% of the targeted compressible parameters are removed and 80% are retained. For a linear layer $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ compressed with dictionary size k and column sparsity s, where s is the number of nonzeros retained per column of $\mathbf{C}_{\mathrm{sparse}}$, ROCKET stores the dense factor $\mathbf{F} \in \mathbb{R}^{d_1 \times k}$ and the sparse coefficient matrix

$\mathbf{H} = \mathbf{C}_{\text{sparse}} \in \mathbb{R}^{k \times d_2}$. The number of stored numerical parameters is therefore $P_{\text{stored}} = d_1 k + s d_2$, and the corresponding layer-wise removed fraction is $\text{CR}_\ell = 1 - \frac{d_1 k + s d_2}{d_1 d_2}$. The retained fraction is $1 - \text{CR}$. Sparse indices for $\mathbf{H}$ are accounted for by truncating the 16-bit floating-point representation by two bits, following CoSpaDi [23]. For a global target compression ratio CR, the global kept-parameter budget is $(1 - \text{CR}) P_{\text{orig}}$, where P_{orig} is the total number of parameters in the compressible linear layers.

Layer Profiling. To select the best compression configuration under a global parameter budget, ROCKET first performs a lightweight *layer-profiling* pass over all compressible layers. For each layer ℓ, we consider a predefined set of candidate configurations, where each candidate i is specified by a rank $k_{\ell,i}$ and a coefficient-sparsity level $s_{\ell,i}$. For each candidate, we: (i) compute the eigendecomposition of $\mathbf{W}_{L,\ell} \mathbf{W}_{L,\ell}^\top$; (ii) retain the top $k_{\ell,i}$ basis vectors; (iii) form the coefficient matrix $\mathbf{C}_{\ell,i}$ and sparsify it using the fused importance scores; (iv) compute the refined dictionary $\mathbf{D}_{\text{final},\ell,i}$ via closed-form least squares; and (v) record the stored parameter count $c_{\ell,i} = d_{1,\ell} k_{\ell,i} + s_{\ell,i} d_{2,\ell}$, the rank-to-sparsity ratio $ks_{\ell,i} = k_{\ell,i} / s_{\ell,i}$, and the relative reconstruction error in the original space, $e_{\ell,i} = \frac{\|\mathbf{W}_\ell - \widetilde{\mathbf{W}}_{\ell,i}\|_F}{\|\mathbf{W}_\ell\|_F} \leq 1$. Here, $d_{1,\ell}$ and $d_{2,\ell}$ denote the input and output dimensions of layer ℓ, and $c_{\ell,i}$ is the number of parameters kept by candidate i. The corresponding layer-wise compression ratio is therefore $\text{CR}_{\ell,i} = 1 - \frac{c_{\ell,i}}{d_{1,\ell} d_{2,\ell}}$, where CR denotes the fraction of parameters removed. This procedure yields, for each linear layer ℓ, a discrete set of feasible compression options $\mathcal{O}_\ell = \{(c_{\ell,i}, ks_{\ell,i}, e_{\ell,i})\}_{i=1}^{K_\ell}$. Importantly, the eigendecomposition is computed only once per layer. All subsequent candidates are evaluated by reusing the precomputed spectral components and applying hard thresholding to the coefficient matrix, which avoids recomputing the decomposition and makes profiling efficient. A detailed runtime analysis is provided in Appendix F; the derivation of the reconstruction error metric is provided in Appendix C. The final output of the profiling stage is the collection $\{\mathcal{O}_\ell\}_{\ell=1}^{L}$, where each layer ℓ is associated with a discrete menu of feasible compression options. Each option records its stored-parameter cost $c_{\ell,i}$, rank-to-sparsity ratio $ks_{\ell,i}$, and reconstruction error $e_{\ell,i}$. These candidate sets are then passed to the constrained multi-choice knapsack solver, which selects exactly one option per layer under the global compression budget.

Constrained Multi-Choice Knapsack Formulation. Let there be L compressible layers. For each layer $\ell \in \{1, \ldots, L\}$, the profiling stage produces a discrete set of feasible compression options $\mathcal{O}_\ell = \{(c_{\ell,i},\, ks_{\ell,i},\, e_{\ell,i})\}_{i=1}^{K_\ell}$, Let P_{orig} be the total number of parameters in the compressible layers. For a target compression ratio CR, the global kept-parameter budget is $C_{\text{total}} = (1 - \text{CR})\, P_{\text{orig}}$. The task of selecting exactly one option per layer so as to minimize total reconstruction error while respecting this budget is a *multi-choice knapsack problem* (MCKP); an ablation on the discretization step is provided in Section 5:

$$\min_{x_{\ell,i} \in \{0,1\}} \sum_{\ell=1}^{L} \sum_{i=1}^{K_\ell} e_{\ell,i}\, x_{\ell,i} \quad \text{s.t.} \quad \sum_{\ell=1}^{L} \sum_{i=1}^{K_\ell} c_{\ell,i}\, x_{\ell,i} \leq C_{\text{total}}, \quad \sum_{i=1}^{K_\ell} x_{\ell,i} = 1,\ \forall\, \ell.$$

To prevent pathological allocations that achieve low global error by severely damaging a few layers while over-preserving others, we impose an additional per-layer error cap. Let $\bar{e}_{\text{ref}}$ denote the average reconstruction error obtained when every layer is compressed at a fixed reference ratio ρ_{ref}. We then require that the error of the selected option for each layer does not exceed a scaled version of this reference: $\sum_{i=1}^{K_\ell} e_{\ell,i}\, x_{\ell,i} \leq \alpha\, \bar{e}_{\text{ref}} \quad \forall\, \ell \in \{1, \ldots, L\}$, where α is a tunable hyperparameter. In all experiments, we set $\alpha = \alpha_{\min} := \inf\{\alpha' > 0 \mid \text{a feasible solution exists for } \alpha'\}$, i.e., the smallest cap that still admits a valid allocation. Further per-model tuning of α may yield additional improvements. With these layer-wise error caps, the problem remains an MCKP whose solution is globally optimal over the discretized candidate set and ensures local robustness of each individual layer. We reformulate this problem as a shortest-path search on a directed acyclic graph in Appendix D, which clarifies the dynamic-programming solution that follows.

4 Experiments

In this section, we describe our experimental setup and compare **ROCKET** against recent compression methods. We focus on low-rank and dictionary-based approaches, specifically SVD-LLM [39] and CoSpaDi [23], as well as budget allocation based methods including ARS [10], Dobi-SVD [38], and

Table 1: Performance comparison of ROCKET vs SOTA SVD-LLM and CoSpaDi methods on Qwen3-8B at different compression ratios (CR). Best results are highlighted with **bold**.

Method	CR	Accuracy↑									Perplexity↓	
		PIQA	HellaSwag	LAMBADA	ARC-e	ARC-c	SciQ	Race	MMLU	Avg.	WikiText	LAMBADA
Qwen3 8B	–	77.7	74.9	64.1	80.7	56.7	95.7	40.9	73.0	70.5	12	4.6
SVD-LLM	0.2	73.8	63.9	62.2	68.7	45.7	90.1	40.5	54.7	62.5	21	6.4
CoSpaDi	0.2	76.5	68.0	65.6	72.2	48.9	93.2	40.7	60.8	65.7	18	4.9
ROCKET	0.2	**77.6**	**72.9**	**66.0**	**75.8**	**53.9**	**94.5**	**41.4**	**67.2**	**68.7**	**15**	**4.7**
SVD-LLM	0.3	70.4	55.2	53.8	59.3	37.1	87.2	38.4	44.8	55.8	27	11
CoSpaDi	0.3	72.4	60.5	62.6	63.9	41.2	88.4	39.5	51.3	60.0	23	6.3
ROCKET	0.3	**75.1**	**67.2**	**68.0**	**72.1**	**47.7**	**92.9**	**41.4**	**62.3**	**65.8**	**18**	**4.4**
SVD-LLM	0.4	66.3	44.6	37.9	45.0	28.1	77.3	35.3	29.1	45.4	43	36
CoSpaDi	0.4	68.9	49.0	49.9	49.4	29.9	82.0	36.8	36.6	50.3	36	15
ROCKET	0.4	**71.0**	**58.7**	**63.7**	**65.7**	**40.4**	**86.1**	**41.9**	**52.3**	**60.0**	**24**	**5.9**
SVD-LLM	0.5	61.5	34.9	25.1	37.4	25.3	65.1	31.6	24.0	38.1	76	88
CoSpaDi	0.5	63.8	39.7	32.4	41.2	26.8	70.4	33.2	28.1	42.0	59	41
ROCKET	0.5	**68.8**	**48.4**	**47.5**	**54.8**	**33.0**	**81.6**	**38.5**	**37.9**	**51.3**	**35**	**24**

ARA [42]. We also provide comparisons with sparsification and width-pruning methods such as LLM-Pruner [21], SliceGPT [2], Bonsai [11], and Wanda [34]. Finally, we conduct ablation studies to isolate the contribution of each one of our design choices.

4.1 Experimental Setup

We evaluate our method in a per-layer setting using Llama and Qwen models. All evaluations are performed in a zero-shot setting on the following benchmarks: PIQA [3], HellaSwag [46], OpenAI LAMBADA [27], ARC-Easy and ARC-Challenge [6], SciQ [41], RACE [16], and MMLU [13]. In addition, we report perplexity on WikiText [26] and LAMBADA-OpenAI. We apply compression at compression ratios ranging from 0.2 to 0.5, in steps of 0.1. For methods that need calibration data, we use 256 randomly sampled sequences from the FineWeb dataset [28] (fixed across all experiments). We also test how the choice of calibration dataset affects results in Appendix H.1. Unless otherwise noted, we compress all dense linear layers in the self-attention blocks (Q, K, V, and O projections) and the feed-forward network (gate, up, and down projections). Embedding layers and the `lm_head` are not compressed following other works. All baselines are reproduced with official code under the same setup, except CoSpaDi in Table 1, taken from [23], and the baselines in Table 2, taken from the respective papers. More experiments are reported in Appendix G.

Comparison with SVD-LLM and CoSpaDi To contextualize ROCKET's performance, we directly compare it against SVD-LLM and CoSpaDi, the two most closely related training-free compression methods. All methods are evaluated in a strictly training-free setting, no fine-tuning, healing, or data augmentation is applied post-compression. As shown in Table 1 and Fig. 2, ROCKET consistently outperforms both baselines by a significant margin across multiple architectures (Qwen3-8B, Llama3-8B and Llama3.2-1B) and compression ratios (20%–50%), in terms of both zero-shot accuracy and perplexity (we refer to Appendix G.1 for more detailed results). Notably, ROCKET exhibits superior scalability under aggressive compression and increasing model scales (details are provided in Appendix G.2): while baseline methods suffer severe degradation beyond 30% compression, ROCKET retains more robust performance even at 50% compression (e.g., 51.3 average accuracy on Qwen3-8B vs. 38.1 for SVD-LLM and 42.0 for CoSpaDi). This demonstrates that ROCKET's combination of calibration-guided factorization, importance-based sparsification and knapsack-based layer-wise budget allocation effectively preserves model fidelity under strict parameter constraints.

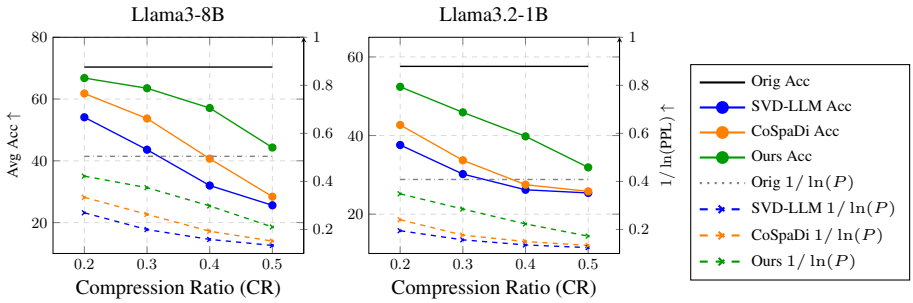


Figure 2: Comparison of Accuracy and Inverse Log Perplexity for Llama3-8B and Llama3.2-1B.

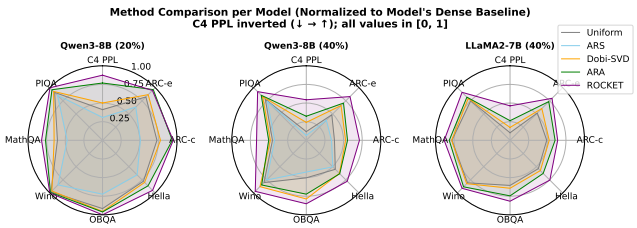


Figure 3: Comparison of ROCKET with alternative budget allocation methods (Uniform, ARS [10], Dobi-SVD [38], and ARA [42]) on three model configurations: Qwen3-8B at 20% and 40% pruning, and Llama2-7B at 40% pruning. Subplots show normalized performance on eight benchmarks (C4 perplexity is inverted, meaning higher value is better), scaled to each model's dense baseline (value=1.0). ROCKET consistently retains the best performance.

Comparison against other budget allocation methods To evaluate the effectiveness of ROCKET's layer-wise budget allocation, we compare it against four established parameter allocation strategies: SVD-LLM [39], which applies uniform compression across layers; Adaptive Rank Selection (ARS) [10]; Dobi-SVD [38]; and Adaptive Rank Allocation (ARA) [42]. Figure 3 shows normalized performance across three model-compressions configurations Qwen3-8B at 20% and 40% compression, and Llama2-7B at 40% compression with all scores scaled to their respective dense baselines. ROCKET consistently outperforms all baselines, demonstrating that globally optimizing the allocation of parameters via constrained knapsack selection preserves significantly more of the original model's capabilities than uniform or trainable strategies, especially under aggressive compression and across different architectures. To further contextualize ROCKET's knapsack-based allocation, we compare it against HALP [32], a hardware-aware latency pruning method, applied on top of dictionary learning (denoted DictionaryLearning+HALP). On Llama3.2-1B at a 20% compression ratio, DictionaryLearning+HALP obtains 30.38 average accuracy and 285 perplexity, whereas ROCKET achieves 52.4 average accuracy and 18 perplexity. This demonstrates that ROCKET's training-free multi-choice reconstruction-error allocation outperforms HALP-style structured pruning under the same compression budget.

Comparison with Depth and Sparsity Pruning Methods To demonstrate the effectiveness of our method, we compare it against several approaches that address model compression from different perspectives. These include width pruning (e.g., SliceGPT [2]), combined depth/width pruning (e.g., LLM-Pruner [21]), structured sparsification (e.g., Wanda [34], Bonsai), adaptive low-rank decomposition with quantization (e.g., Dobi-SVD [38]), and training-free modular decomposition (e.g., MoDeGPT [17]). As shown in Table 2, ROCKET achieves strong performance, outperforming all other non-quantized baselines at a 40% compression ratio (0.56 average accuracy). Since Dobi-SVD uses quantization, we additionally evaluate ROCKET with post-compression quantization to ensure a fair comparison (weight-only INT8 min-max quantization of both factors without calibration; Appendix G.5). Under this setting, ROCKET surpasses Dobi-SVD at a 40% compression ratio (0.65 vs. 0.63). At a 60% compression ratio, ROCKET again leads with an average accuracy of 0.60 compared to Dobi-SVD's 0.52. To further compare with a training-free structured decomposition baseline, we evaluate MoDeGPT on Llama3-8B at a 25% compression ratio under the same training-free protocol. The dense Llama3-8B model achieves 72.75% average accuracy, while MoDeGPT retains approximately 88% of this performance with 63.96% average accuracy. In contrast, ROCKET retains approximately 95%, achieving 69.34% average accuracy. This demonstrates that ROCKET's training-free pipeline is effective, matching or exceeding other pruning and structured decomposition methods.

Post-Compression Healing Applying a lightweight healing step that fine-tunes only unmasked parameters on 30M C4 tokens while preserving the fixed sparsity pattern recovers substantial fidelity in the 40%-compressed ROCKET-Qwen3-14B model, increasing average accuracy from 63.56 to 67.96 and reducing perplexity from 25 to 13. This healed configuration closely approaches the native Qwen3-8B baseline (70.46 accuracy, 12 perplexity), confirming that ROCKET provides a robust initialization for efficient post-compression recovery. We refer to Appendix G.6 for more details.

Generalization to Other Modalities To assess the generality of ROCKET beyond language-only models, we apply it to two transformer-based architectures from different modalities: the vision-

Table 2: Performance comparison of ROCKET against depth- and sparsity-based pruning methods on Llama3.1-8B across compression levels and benchmarks. "Avg" denotes the average accuracy across all benchmarks, "Drop" indicates the relative accuracy drop percentage compared to the dense model, and "Quant." indicates whether post-compression quantization is applied to achieve the target CR.

Models	CR	Method	Quant.	PIQA	HellaSwag	WinoGrande	ARC_e	ARC_c	Avg. (↑)	Drop (↓)
Llama3.1-8B	–	Baseline	✗	0.80	0.59	0.74	0.81	0.51	0.69	0%
	0.4	LLM-Pruner	✗	0.66	0.32	0.54	0.58	0.23	0.46	33.3%
		SliceGPT	✗	0.62	0.40	0.53	0.49	0.25	0.46	33.3%
		Bonsai	✗	0.59	0.29	0.49	0.47	0.18	0.41	40.6%
		Wanda-sp	✗	0.57	0.28	0.50	0.44	0.17	0.39	43.5%
		Dobi-SVD	✓	0.76	0.52	0.72	0.73	0.39	0.63	8.70%
		ROCKET	✗	0.72	0.43	0.66	0.64	0.33	0.56	18.84%
		ROCKET	✓	**0.78**	**0.54**	**0.72**	**0.76**	**0.42**	**0.65**	**5.79%**
	0.6	Dobi-SVD	✓	0.68	0.41	0.66	0.58	0.27	0.52	24.6%
		ROCKET	✓	**0.75**	**0.49**	**0.68**	**0.72**	**0.38**	**0.60**	**13.0%**

language model **Qwen3-VL-4B** and the speech generation model **VibeVoice** [29]. For VibeVoice, we use 256 transcriptions-only from `mls_eng_10k` dataset and validate speech results on different samples from `mls_eng_10k` [30]. For Qwen3-VL-4B, we construct a multimodal calibration set using 256 samples from the MathVista portion of the MathVerse dataset and evaluate on MMBench-en-dev (MMB) [18], MMMU-val (MMMU) [45], MMStar (MMS) [4], OCRBench [19], and RealWorldQA (RWQA). In both cases, we compress the models by 20% without any fine-tuning.

As shown in Table 3, ROCKET preserves strong performance on Qwen3-VL-4B, achieving 63.7 average accuracy (over 90% of the original model's performance). On VibeVoice, Table 4 shows near-identical speech quality: WER remains stable (0.149 vs. 0.148), and UTMOS drops only slightly (3.43 vs. 3.52), staying close to the ground-truth reference (3.73). These results demonstrate that ROCKET generalizes effectively across modalities.

Table 3: Average accuracy of Qwen3-VL-4B before and after ROCKET compression (20%).

Method	MMB	MMMU	MMS	OCR	RWQA
dense	83.76	49.44	61.85	81.70	71.50
ROCKET	78.95	44.44	54.85	74.50	65.75

Table 4: ROCKET applied to VibeVoice 7B at 20% compression.

Method	WER ↓	UTMOS ↑
Ground Truth	0.04	3.73
VibeVoice (dense)	0.148	3.52
ROCKET	0.149	3.43

5 Ablations

To evaluate the contribution of each key component in ROCKET, we conduct ablation studies on Llama3.2-1B following the evaluation protocol in Section 4.1. All variants are evaluated at a fixed global compression ratio of 20%, with all other design choices held constant, and we report average accuracy along with word-level WikiText perplexity. Ablations on the reconstruction error metric (Appendix H.5) and calibration data (Appendix H.1), per-benchmark results, and information about environmental impact and inference (Appendix E) are provided in the appendix.

Core components To assess the contribution of ROCKET's key design choices, we compare four variants: (1) SVD-LLM, which uses neither sparsification nor dynamic budget allocation; (2) CoSpaDi, which uses K-SVD-based sparse dictionary learning; (3) ROCKET with uniform compression across layers; and (4) full ROCKET, which further incorporates knapsack-based budget allocation. As shown in Table 5, using sparsification improves both average accuracy and perplexity. Moreover, replacing CoSpaDi's iterative sparsification with our closed-form, activation-aware approach is not only much faster, but also improves average accuracy from 42.7 to 45.4 and reduces perplexity from 64 to 27. Employing budget allocation yields a further significant gain, reaching 52.4 average accuracy and 18 perplexity, demonstrating that both our sparsification strategy and global parameter allocation are critical to ROCKET's performance.

Sparsification strategy We compare ROCKET against alternative approaches for sparsifying the coefficient matrix $\mathbf{C}$ (Table 6): (1) **Per-Row**, where importance scores are computed identically but sparsity is enforced independently per row (breaking the column-wise structure); (2) **Global**, which ignores structural constraints and zeros out the globally least-important entries; and (3) **Whitened-only**, which disables the original-space fidelity term by setting $\lambda = 0$ in equation 5. We further apply

Table 5: Core components. ROCKET† uses uniform budget allocation.

Method	Sparse	Budg. Alloc.	Avg. Acc. ↑	PPL ↓
Dense	—	—	57.6	12
SVD-LLM	✗	✗	37.6	170
CoSpaDi	✓	✗	42.7	64
ROCKET†	✓	✗	45.4	27
ROCKET	✓	✓	**52.4**	**18**

Table 6: Sparsification of $\mathbf{C}$: ROCKET variants (top) and pruning-style criteria after SVD (bottom).

Strategy	Inspired by	Avg. Acc. ↑	PPL ↓
Per-Row	–	45.6	32
Global	–	50.1	19
Whitened-only ($\lambda = 0$)	–	52.0	19
Uniform modular	MoDeGPT [17]	50.1	20
Outlier-weighted	OWL/Wanda [43]	34.4	187
Heavy-tailed self-reg.	AlphaPruning [20]	39.9	57
Column-wise + dual-space	**ROCKET**	**52.4**	**18**

Table 7: Step size of the knapsack CR grid.

Step Size	Total Rec. Error	Avg. Rec. Error	Profiling Time (s)
No dynamic	37.88	0.338	0
0.20	37.74	0.3370	208
0.15	32.53	0.2904	228
0.10	31.70	0.2830	308
0.05	**31.57**	**0.2819**	**425**
0.025	31.56	0.2818	524
0.01	31.54	0.2816	1359

Table 8: Balance parameter λ in equation 5 (WikiText PPL).

λ	PPL ↓	λ	PPL ↓
0.0	17.83	0.6	17.96
0.1	17.83	0.7	17.95
0.2	17.75	0.8	18.20
0.3	**17.71**	0.9	18.52
0.4	17.72	1.0	19.06
0.5	17.75		

truncated SVD and then sparsify $\mathbf{C}$ using strategies inspired by recent pruning methods [17, 20, 43], with ROCKET's budget allocation used for all variants. ROCKET's column-aware, dual-space importance scoring outperforms others, demonstrating the efficiency of both structural awareness and dual-space scoring; transplanting pruning heuristics directly to the coefficient matrix is suboptimal, as the dictionary and coefficients must be treated jointly (detailed results in Appendix H.2).

Balance parameter λ Table 8 shows strong robustness to $\lambda \in [0, 1]$, with performance degrading only at extremes ($\lambda \geq 0.8$); we adopt $\lambda = 0.5$ as a default for simplicity and stability.

Search discretization The discretization step size governs the resolution of the compression ratio grid in the multi-choice knapsack allocator. Table 7 reports the impact of varying the step size on total reconstruction error, average per-layer error, and profiling runtime. The results justify our default choice of 0.05, as finer steps yield diminishing error improvements (< 0.03 in total reconstruction error) while increasing profiling time by up to $3.2\times$. The error plateaus at ≈ 0.282 for step sizes ≤ 0.10, confirming numerical stability and indicating that the allocation obtained over the discretized candidate set is near-optimal with respect to finer grids (candidate-set construction in Appendix H.3).

6 Conclusion and Limitations

ROCKET introduces a fast, training-free compression framework that combines calibration-guided structured factorization with principled knapsack-based budget allocation. It achieves state-of-the-art fidelity, retaining over 90% of original accuracy at 30% compression without fine-tuning. While the exact DP solver scales efficiently for dense architectures, exhaustive search becomes intensive for extreme-scale MoE models. We address this via beam search and Pareto pruning, enabling practical compression for 100K+ layers with minor optimality trade-offs (please see Appendix G.4). Moreover, support selection relies on a diagonal surrogate that discards the off-diagonal couplings of $\mathbf{G}$ (Section 3.4), and its approximation error is not characterized theoretically; the whitening also depends on the calibration data, although the observed variation is small (Appendix H.1). Finally, we note that all factorization-based methods (including ROCKET) reduce theoretical FLOPs but may incur sequential overhead on standard hardware. While ROCKET already reaches 1.07–1.37$\times$ the serving throughput of the dense model (Appendix E.3), the sparse kernel we use (MACKO) does not yet support dynamic batching, and unlocking full inference acceleration requires specialized sparse-dense kernels and tiling strategies – a promising direction for future system-level optimization.

Acknowledgments This research was supported as part of the ITMO University Research Projects in AI Initiative.

References

[1] AHARON, M., ELAD, M., AND BRUCKSTEIN, A. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing 54*, 11 (2006), 4311–4322.

[2] ASHKBOOS, S., CROCI, M. L., NASCIMENTO, M. G. D., HOEFLER, T., AND HENSMAN, J. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024* (2024), OpenReview.net.

[3] BISK, Y., ZELLERS, R., BRAS, R. L., GAO, J., AND CHOI, Y. Piqa: Reasoning about physical commonsense in natural language, 2019.

[4] CHEN, L., LI, J., DONG, X., ZHANG, P., ZANG, Y., CHEN, Z., DUAN, H., WANG, J., QIAO, Y., LIN, D., ET AL. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330* (2024).

[5] CHEN, P. H., YU, H.-F., DHILLON, I. S., AND HSIEH, C.-J. Drone: data-aware low-rank compression for large nlp models. In *Proceedings of the 35th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2021), NIPS '21, Curran Associates Inc.

[6] CLARK, P., COWHEY, I., ETZIONI, O., KHOT, T., SABHARWAL, A., SCHOENICK, C., AND TAFJORD, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[7] ELAD, M. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[8] ENGAN, K., AASE, S., AND HAKON HUSOY, J. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)* (1999), vol. 5, pp. 2443–2446 vol.5.

[9] FRANTAR, E., AND ALISTARH, D. SPDY: accurate pruning with speedup guarantees. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA* (2022), K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., Proceedings of Machine Learning Research, PMLR, pp. 6726–6743.

[10] GAO, S., HUA, T., HSU, Y.-C., SHEN, Y., AND JIN, H. Adaptive rank selections for low-rank approximation of language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (Mexico City, Mexico, June 2024), K. Duh, H. Gomez, and S. Bethard, Eds., Association for Computational Linguistics, pp. 227–241.

[11] GUPTA, M., JAIN, S., RAMANI, V., KODAMANA, H., AND RANU, S. Bonsai: Gradient-free graph condensation for node classification. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025* (2025), OpenReview.net.

[12] HASSIBI, B., AND STORK, D. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems* (1992), S. Hanson, J. Cowan, and C. Giles, Eds., vol. 5, Morgan-Kaufmann.

[13] HENDRYCKS, D., BURNS, C., BASART, S., ZOU, A., MAZEIKA, M., SONG, D., AND STEINHARDT, J. Measuring massive multitask language understanding, 2021.

[14] HENDRYCKS, D., BURNS, C., KADAVATH, S., ARORA, A., BASART, S., TANG, E., SONG, D., AND STEINHARDT, J. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual* (2021), J. Vanschoren and S. Yeung, Eds.

[15] HINTON, G. E., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *CoRR abs/1503.02531* (2015).

[16] LAI, G., XIE, Q., LIU, H., YANG, Y., AND HOVY, E. Race: Large-scale reading comprehension dataset from examinations, 2017.

[17] LIN, C., GAO, S., SMITH, J. S., PATEL, A., TULI, S., SHEN, Y., JIN, H., AND HSU, Y. Modegpt: Modular decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025* (2025), OpenReview.net.

[18] LIU, Y., DUAN, H., ZHANG, Y., LI, B., ZHANG, S., ZHAO, W., YUAN, Y., WANG, J., HE, C., LIU, Z., CHEN, K., AND LIN, D. MMBench: Is your multi-modal model an all-around player? *arXiv:2307.06281* (2023).

[19] LIU, Y., LI, Z., HUANG, M., YANG, B., YU, W., LI, C., YIN, X., LIN LIU, C., JIN, L., AND BAI, X. Ocrbench: On the hidden mystery of ocr in large multimodal models.

[20] LU, H., ZHOU, Y., LIU, S., WANG, Z., MAHONEY, M. W., AND YANG, Y. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. In *Advances in Neural Information Processing Systems* (2024), A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37, Curran Associates, Inc., pp. 9117–9152.

[21] MA, X., FANG, G., AND WANG, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems 36* (2023), 21702–21720.

[22] MACKO, V., AND BOŽA, V. Macko: Sparse matrix-vector multiplication for low sparsity. *arXiv preprint arXiv: 2511.13061* (2025).

[23] MAKHOV, D., SHOPKHOEV, D., ZHUSSIP, M., ALI, A., AND LEFKIMMIATIS, S. Cospadi: Compressing llms via calibration-guided sparse dictionary learning. *arXiv preprint arXiv: 2509.22075* (2025).

[24] MALLAT, S. *A wavelet tour of signal processing (2. ed.).* Academic Press, 1999.

[25] MARCUS, M. P., SANTORINI, B., AND MARCINKIEWICZ, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19*, 2 (1993), 313–330.

[26] MERITY, S., XIONG, C., BRADBURY, J., AND SOCHER, R. Pointer sentinel mixture models, 2016.

[27] PAPERNO, D., KRUSZEWSKI, G., LAZARIDOU, A., PHAM, Q. N., BERNARDI, R., PEZZELLE, S., BARONI, M., BOLEDA, G., AND FERNÁNDEZ, R. The lambada dataset: Word prediction requiring a broad discourse context, 2016.

[28] PENEDO, G., KYDLÍČEK, H., ALLAL, L. B., LOZHKOV, A., MITCHELL, M., RAFFEL, C., WERRA, L. V., AND WOLF, T. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2024).

[29] PENG, Z., YU, J., WANG, W., CHANG, Y., SUN, Y., DONG, L., ZHU, Y., XU, W., BAO, H., WANG, Z., HUANG, S., XIA, Y., AND WEI, F. Vibevoice technical report. *CoRR abs/2508.19205* (2025).

[30] PRATAP, V., XU, Q., SRIRAM, A., SYNNAEVE, G., AND COLLOBERT, R. Mls: A large-scale multilingual dataset for speech research. *ArXiv abs/2012.03411* (2020).

[31] REIN, D., HOU, B. L., STICKLAND, A. C., PETTY, J., PANG, R. Y., DIRANI, J., MICHAEL, J., AND BOWMAN, S. R. GPQA: A graduate-level google-proof q&a benchmark. *CoRR abs/2311.12022* (2023).

[32] SHEN, M., YIN, H., MOLCHANOV, P., MAO, L., LIU, J., AND ALVAREZ, J. M. Halp: Hardware-aware latency pruning, 2021.

[33] SPRAGUE, Z., YE, X., BOSTROM, K., CHAUDHURI, S., AND DURRETT, G. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024* (2024), OpenReview.net.

[34] SUN, M., LIU, Z., BAIR, A., AND KOLTER, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024* (2024), OpenReview.net.

[35] SUZGUN, M., SCALES, N., SCHÄRLI, N., GEHRMANN, S., TAY, Y., CHUNG, H. W., CHOWDHERY, A., LE, Q. V., CHI, E. H., ZHOU, D., AND WEI, J. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023* (2023), A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds., Association for Computational Linguistics, pp. 13003–13051.

[36] TAORI, R., GULRAJANI, I., ZHANG, T., DUBOIS, Y., LI, X., GUESTRIN, C., LIANG, P., AND HASHIMOTO, T. B. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[37] TROPP, J. A., AND GILBERT, A. C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory 53*, 12 (2007), 4655–4666.

[38] WANG, Q., KE, J., TOMIZUKA, M., KEUTZER, K., AND XU, C. Dobi-svd: Differentiable SVD for LLM compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025* (2025), OpenReview.net.

[39] WANG, X., ZHENG, Y., WAN, Z., AND ZHANG, M. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations* (2025).

[40] WANG, Y., MA, X., ZHANG, G., NI, Y., CHANDRA, A., GUO, S., REN, W., ARULRAJ, A., HE, X., JIANG, Z., LI, T., KU, M., WANG, K., ZHUANG, A., FAN, R., YUE, X., AND CHEN, W. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024* (2024), A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, Eds.

[41] WELBL, J., LIU, N. F., AND GARDNER, M. Crowdsourcing multiple choice science questions, 2017.

[42] XV, L., GAO, J., GAO, X., LIU, T., AND FU, Y. ARA: adaptive rank allocation for efficient large language model SVD compression. *CoRR abs/2510.19389* (2025).

[43] YIN, L., WU, Y., ZHANG, Z., HSIEH, C.-Y., WANG, Y., JIA, Y., LI, G., JAISWAL, A., PECHENIZKIY, M., LIANG, Y., BENDERSKY, M., WANG, Z., AND LIU, S. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity, 2023.

[44] YUAN, Z., SHANG, Y., SONG, Y., WU, Q., YAN, Y., AND SUN, G. ASVD: activation-aware singular value decomposition for compressing large language models. *CoRR abs/2312.05821* (2023).

[45] YUE, X., NI, Y., ZHANG, K., ZHENG, T., LIU, R., ZHANG, G., STEVENS, S., JIANG, D., REN, W., SUN, Y., WEI, C., YU, B., YUAN, R., SUN, R., YIN, M., ZHENG, B., YANG, Z., LIU, Y., HUANG, W., SUN, H., SU, Y., AND CHEN, W. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR* (2024).

[46] ZELLERS, R., HOLTZMAN, A., BISK, Y., FARHADI, A., AND CHOI, Y. Hellaswag: Can a machine really finish your sentence?, 2019.

[47] ZHOU, J., LU, T., MISHRA, S., BRAHMA, S., BASU, S., LUAN, Y., ZHOU, D., AND HOU, L. Instruction-following evaluation for large language models. *CoRR abs/2311.07911* (2023).

[48] ZHUSSIP, M., SHOPKHOEV, D., ALI, A., AND LEFKIMMIATIS, S. Share your attention: Transformer weight sharing via matrix-based dictionary learning. *arXiv preprint arXiv:2508.04581* (2025).

A Ethical Statement and Broader Impact

ROCKET is a training-free compression method designed to improve the efficiency and accessibility of large language models without requiring additional data or extensive computational resources for fine-tuning. By enabling high-fidelity model compression with minimal environmental and economic cost, it supports more sustainable deployment of AI systems, particularly in resource constrained settings. The method does not introduce new data collection, human annotation, or model behaviors beyond those already present in the original pretrained model; thus, it neither amplifies nor mitigates existing biases in the base model. Users should remain vigilant about the ethical implications of the underlying model's outputs, as ROCKET preserves its functional characteristics including potential biases or safety limitations. We encourage responsible deployment, including thorough evaluation and alignment measures when compressed models are used in real-world applications.

B Proposed Algorithm

In Algo. 1, we present the proposed method in the form of a formal algorithm to facilitate a clear and systematic understanding of the method's complete pipeline construction and implementation.

B.1 Theoretical Comparison with Iterative Dictionary Learning

It is important to highlight that CoSpaDi relies on unconstrained dictionary estimation via iterative K-SVD/OMP optimization, which is non-convex and lacks global convergence guarantees. In contrast, ROCKET employs a *two-stage closed-form estimator*. First, we apply the Eckart–Young–Mirsky theorem via eigendecomposition to obtain a globally optimal column-orthogonal dictionary $\mathbf{B}_k$. For this basis, sparse coding admits an *exact analytical solution* via hard thresholding, avoiding the greedy approximation errors inherent to OMP. Second, we relax orthogonality constraints and compute the global least-squares update $\mathbf{D}^* = \mathbf{W}_L\mathbf{C}^\top(\mathbf{C}\mathbf{C}^\top)^{-1}$, following the Method of Optimal Directions (MOD). While K-SVD updates atoms sequentially (coordinate descent), ROCKET computes the unique global minimizer for the dictionary given fixed codes. This sequential closed-form design combines spectral initialization stability with convex optimality, preventing error propagation and local minima typical of alternating minimization schemes.

C Norm-Preserving Properties of Low-Rank Approximation and Sparsification

Let $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ with singular value decomposition (SVD)

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{d_1 \times r}$, $\mathbf{V} \in \mathbb{R}^{d_2 \times r}$ have orthonormal columns, $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$, $\sigma_1 \geq \cdots \geq \sigma_r > 0$, and $r = \operatorname{rank}(\mathbf{W})$.

C.1 Relative Error of Rank-k Truncated SVD

The optimal rank-k approximation ($1 \leq k \leq r$) is

$$\mathbf{W}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top,$$

with error given by the Eckart–Young–Mirsky theorem:

$$\|\mathbf{W} - \mathbf{W}_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2.$$

Since $\|\mathbf{W}\|_F^2 = \sum_{i=1}^{r} \sigma_i^2$, it follows that

$$\frac{\|\mathbf{W} - \mathbf{W}_k\|_F}{\|\mathbf{W}\|_F} = \sqrt{\frac{\sum_{i=k+1}^{r} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}} \leq 1,$$

with equality iff $k = 0$.

Algorithm 1 ROCKET: Training-Free Heterogeneous transformer Compression

Require: Pre-trained transformer with linear layers $\{\mathbf{W}^{(\ell)}\}_{\ell=1}^{L}$, output layer data $\mathbf{X}_\ell \in \mathbb{R}^{N\times d_1}$ (using calibration data $\mathbf{X} \in \mathbb{R}^{N\times d_1}$), global parameter budget C_{total}

Ensure: Compressed model with factorization $\{\mathbf{B}^{(\ell)}, \mathbf{C}^{(\ell)}\}_{\ell=1}^{L}$

for each layer $\ell = 1$ to L **do**
  Compute whitening transform: $\mathbf{L}_\ell \leftarrow \text{chol}(\mathbf{X}_\ell^\top \mathbf{X}_\ell + \epsilon \mathbf{I})$
  $\mathbf{W}_L^\ell \leftarrow \mathbf{L}_\ell \mathbf{W}^{(\ell)}$
  for each candidate rank $k \in \mathcal{K}$ and sparsity ratio $s \in \mathcal{S}$ **do**
    Compute the top-k eigenvectors $\mathbf{B}$ via eigendecomposition:

$$\mathbf{W}_L^\ell \mathbf{W}_L^{\ell\top} \approx \mathbf{B}\mathbf{\Lambda}_k \mathbf{B}^\top$$

    Form coefficient matrix: $\mathbf{C} \leftarrow \mathbf{B}^\top \mathbf{W}_L^\ell$
    Compute importance scores: $\text{imp}_{ij} \leftarrow |c_{ij}| \cdot \|\mathbf{L}_\ell^{-1}\mathbf{b}_i\|_2^\lambda \ \{\lambda = 0.5\}$
    Over-sparsify $\mathbf{C}$ to ratio $s + \gamma$, then reactivate top entries globally to reach exact sparsity s
    Solve for optimal left factor: $\mathbf{D} \leftarrow \arg\min_{\mathbf{D}} \|\mathbf{W}_L^\ell - \mathbf{D}\mathbf{C}_{\text{sparse}}\|_F^2 + \mu\|\mathbf{D}\|_F^2$
    Reconstruct weight: $\widetilde{\mathbf{W}}^\ell \leftarrow \mathbf{L}_\ell^{-1}\mathbf{D}\mathbf{C}_{\text{sparse}}$
    Record option: cost $c_{\ell,i} \leftarrow \texttt{nnz}(\mathbf{D}) + \texttt{nnz}(\mathbf{C}_{\text{sparse}})$, error $e_{\ell,i}$
    Store $(c_{\ell,i}, e_{\ell,i})$ in $\mathcal{O}_\ell$
  end for
end for
Compute reference error $\bar{e}_{\text{ref}}$ from uniform compression baseline and $\min \alpha$ that admits a solution
Initialize DP table: $\text{DP}_0[0] \leftarrow 0$, others $\leftarrow \infty$
$\beta_{\text{scale}} = \text{param_precision}/C_{\text{total}}$
for $\ell = 1$ to L **do**
  for each state p in $\text{DP}_{\ell-1}$ **do**
    for each option $i \in \mathcal{O}_\ell$ **do**
      if $e_{\ell,i} \le \alpha \cdot \bar{e}_{\text{ref}}$ **then**
        $p' \leftarrow p + \lfloor \beta_{\text{scale}} c_{\ell,i} \rfloor$
        $\text{DP}_\ell[p'] \leftarrow \min\left(\text{DP}_\ell[p'],\ \text{DP}_{\ell-1}[p] + e_{\ell,i}\right)$
      end if
    end for
  end for
  Prune dominated states in DP_ℓ: remove (p_1, err_1) if $\exists (p_2, \text{err}_2)$ with $p_2 < p_1$ and $\text{err}_2 \le \text{err}_1$
end for
Find $p^* = \arg\min_{p \le \lfloor \beta_{\text{scale}} C_{\text{total}} \rfloor} \text{DP}_L[p]$
Backtrack to recover the selected per-layer choices $\{i_\ell^*\}_{\ell=1}^{L}$
for $\ell = 1$ to L **do**
  Assign $\mathbf{D}_{\text{final}}^{(\ell)}, \mathbf{C}_{\text{final}}^{(\ell)}$ from option i_ℓ^*
end for

C.2 Sparsification of the Coefficients

Let $\mathcal{T} : \mathbb{R}^{k\times d_2} \to \mathbb{R}^{k\times d_2}$ denote an entrywise hard-thresholding operator. For any matrix $\mathbf{M}$, $\mathcal{T}(\mathbf{M})$ retains entries whose magnitudes exceed a prescribed threshold and sets the remainder to zero. Consequently, the retained component $\mathcal{T}(\mathbf{M})$ and the pruned residual $\mathbf{M} - \mathcal{T}(\mathbf{M})$ have disjoint supports, which implies orthogonality under the Frobenius inner product:

$$\langle \mathcal{T}(\mathbf{M}), \mathbf{M} - \mathcal{T}(\mathbf{M}) \rangle_F = 0.$$

By the Pythagorean theorem for the Frobenius norm,

$$\|\mathbf{M}\|_F^2 = \|\mathcal{T}(\mathbf{M})\|_F^2 + \|\mathbf{M} - \mathcal{T}(\mathbf{M})\|_F^2,$$

which yields the fundamental bound $\|\mathbf{M} - \mathcal{T}(\mathbf{M})\|_F \le \|\mathbf{M}\|_F$.

Let $\mathbf{W}_k = \mathbf{B}_k \mathbf{\Sigma}_k \mathbf{V}_k^\top$ be the rank-k approximation of $\mathbf{W}$, where $\mathbf{B}_k \in \mathbb{R}^{d_1\times k}$ has orthonormal columns ($\mathbf{B}_k^\top \mathbf{B}_k = \mathbf{I}_k$). Define the sparsified reconstruction as

$$\widetilde{\mathbf{W}} = \mathbf{B}_k \, \mathcal{T}(\mathbf{\Sigma}_k \mathbf{V}_k^\top).$$

Decompose $\mathbf{W}$ into its rank-k component and orthogonal residual: $\mathbf{W} = \mathbf{W}_k + \mathbf{W}_\perp$, with $\mathbf{W}_\perp \perp \mathbf{W}_k$. Since $\widetilde{\mathbf{W}}$ lies in the column space of $\mathbf{B}_k$, it follows that $\mathbf{W}_\perp \perp (\mathbf{W}_k - \widetilde{\mathbf{W}})$. Therefore, the total reconstruction error decomposes as

$$\|\mathbf{W} - \widetilde{\mathbf{W}}\|_F^2 = \|\mathbf{W}_\perp\|_F^2 + \|\mathbf{W}_k - \widetilde{\mathbf{W}}\|_F^2.$$

Because $\mathbf{B}_k$ has orthonormal columns, left-multiplication preserves the Frobenius norm:

$$\|\mathbf{W}_k - \widetilde{\mathbf{W}}\|_F = \|\mathbf{\Sigma}_k\mathbf{V}_k^\top - \mathcal{T}(\mathbf{\Sigma}_k\mathbf{V}_k^\top)\|_F.$$

Applying the Pythagorean bound to $\mathbf{M} = \mathbf{\Sigma}_k\mathbf{V}_k^\top$ gives $\|\mathbf{W}_k - \widetilde{\mathbf{W}}\|_F \leq \|\mathbf{\Sigma}_k\mathbf{V}_k^\top\|_F = \|\mathbf{W}_k\|_F$. Substituting this into the error decomposition yields

$$\|\mathbf{W} - \widetilde{\mathbf{W}}\|_F^2 \leq \|\mathbf{W}_\perp\|_F^2 + \|\mathbf{W}_k\|_F^2 = \|\mathbf{W}\|_F^2.$$

Taking square roots establishes the rigorous relative error bound

$$\frac{\|\mathbf{W} - \widetilde{\mathbf{W}}\|_F}{\|\mathbf{W}\|_F} \leq 1.$$

Equality holds if and only if $\mathcal{T}$ zeroes out all coefficients ($\widetilde{\mathbf{W}} = \mathbf{0}$). For any nontrivial threshold, the relative error is strictly bounded by 1.

C.3 Equivalence of Eigenvalue-Based Basis Construction and SVD in ROCKET

In ROCKET, the weight matrix $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is generally rectangular and non-symmetric, so eigendecomposition cannot be applied directly to $\mathbf{W}$. Instead, compression operates on the whitened weight $\mathbf{W}_L = \mathbf{L}\mathbf{W}$, where $\mathbf{L} = \mathrm{chol}(\mathbf{X}^\top\mathbf{X})$ whitens the input activations using a small calibration set $\mathbf{X} \in \mathbb{R}^{N \times d_1}$. The method then computes the top-k eigenvectors of the symmetric positive semi-definite matrix $\mathbf{W}_L\mathbf{W}_L^\top$. This eigendecomposition yields

$$\mathbf{W}_L\mathbf{W}_L^\top = \mathbf{B}_k\mathbf{\Lambda}_k\mathbf{B}_k^\top,$$

where $\mathbf{B}_k \in \mathbb{R}^{d_1 \times k}$ has orthonormal columns ($\mathbf{B}_k^\top\mathbf{B}_k = \mathbf{I}_k$) and $\mathbf{\Lambda}_k = \mathrm{diag}(\eta_1, \ldots, \eta_k)$ with $\eta_i \geq 0$.

Let the compact singular value decomposition of $\mathbf{W}_L$ be

$$\mathbf{W}_L = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top.$$

Then

$$\mathbf{W}_L\mathbf{W}_L^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top,$$

which is precisely the eigenvalue decomposition of $\mathbf{W}_L\mathbf{W}_L^\top$. Therefore, the eigenbasis coincides exactly with the left singular vectors: $\mathbf{B}_k = \mathbf{U}_k$ and $\eta_i = \sigma_i^2$. The coefficient matrix is obtained via orthogonal projection:

$$\mathbf{C}_k = \mathbf{B}_k^\top\mathbf{W}_L = \mathbf{U}_k^\top\mathbf{W}_L = \mathbf{\Sigma}_k\mathbf{V}_k^\top,$$

which matches the right factor in the SVD. Hence, the reconstruction

$$\mathbf{W}_L = \mathbf{B}_k\mathbf{C}_k$$

is identical to the rank-k truncated SVD of $\mathbf{W}_L$.

Sparsification is applied to $\mathbf{C}_k$ using an operator $\mathcal{T} : \mathbb{R}^{k \times d_2} \to \mathbb{R}^{k \times d_2}$ satisfying

$$\|\mathcal{T}(\mathbf{C}_k)\|_F \leq \|\mathbf{C}_k\|_F.$$

The sparsified approximation in the whitened space is defined as

$$\widetilde{\mathbf{W}}_L = \mathbf{B}_k\,\mathcal{T}(\mathbf{C}_k).$$

Since $\mathbf{B}_k$ has orthonormal columns, the Frobenius norm is preserved under multiplication:

$$\|\widetilde{\mathbf{W}}_L\|_F = \|\mathcal{T}(\mathbf{C}_k)\|_F \leq \|\mathbf{C}_k\|_F < \|\mathbf{W}_L\|_F \quad \text{for } k < r.$$

(Note: Strict inequality holds as truncation removes non-zero singular components.) Consequently, the relative reconstruction error in the whitened space satisfies

$$\frac{\|\mathbf{W}_L - \widetilde{\mathbf{W}}_L\|_F}{\|\mathbf{W}_L\|_F} \leq 1,$$

with equality only in the degenerate case $\widetilde{\mathbf{W}}_L = \mathbf{0}$. Because the eigenbasis $\mathbf{B}_k$ is mathematically identical to the left singular vectors of $\mathbf{W}_L$, all norm-preserving properties and error bounds established for truncated SVD carry over verbatim.

Finally, the compressed weight in the original space is recovered as

$$\widetilde{\mathbf{W}} = \mathbf{L}^{-1}\widetilde{\mathbf{W}}_L.$$

Although $\mathbf{L}^{-1}$ is not orthogonal, the theoretical guarantees in the whitened space where the core approximation occurs remain intact. Thus, despite using the eigendecomposition of $\mathbf{W}_L\mathbf{W}_L^\top$ for computational efficiency, ROCKET achieves the same norm constraints and relative error bound (≤ 1) as classical SVD-based low-rank approximation followed by norm-non-increasing sparsification.

C.4 Exact Thresholding for Orthonormal Bases

The following proposition justifies the use of hard thresholding as the exact sparse-coding solver when the dictionary is column-orthogonal.

Proposition C.1 (Pythagorean decomposition)**.** *Let $\mathbf{B} \in \mathbb{R}^{d_1 \times k}$ have orthonormal columns and let $\mathbf{C} = \mathbf{B}^\top \mathbf{W}_L$. For any $\mathbf{S} \in \mathbb{R}^{k \times d_2}$,*

$$\|\mathbf{W}_L - \mathbf{B}\mathbf{S}\|_F^2 = \|\mathbf{W}_L - \mathbf{B}\mathbf{C}\|_F^2 + \|\mathbf{C} - \mathbf{S}\|_F^2 . \tag{9}$$

Consequently, for fixed $\mathbf{B}$, minimizing the whitened objective over s-sparse $\mathbf{S}$ decouples across columns and is solved by retaining, in each column, the s coefficients of largest magnitude $|c_{ij}|$ — the classical best-s-term approximation property of orthonormal bases [7, 24].

Proof. Write $\mathbf{W}_L - \mathbf{B}\mathbf{S} = (\mathbf{I} - \mathbf{B}\mathbf{B}^\top)\mathbf{W}_L + \mathbf{B}(\mathbf{C} - \mathbf{S})$. The two terms are orthogonal: the first lies in $\mathrm{span}(\mathbf{B})^\perp$ and the second in $\mathrm{span}(\mathbf{B})$. Moreover $\|\mathbf{B}\mathbf{M}\|_F = \|\mathbf{M}\|_F$ for any $\mathbf{M}$ because $\mathbf{B}^\top\mathbf{B} = \mathbf{I}$. □

For general (non-orthogonal) dictionaries no such exact rule exists, and one must resort to greedy pursuits such as OMP [37]. This is precisely the cost that ROCKET eliminates by restricting the sparse-coding step to an orthogonal basis.

C.5 Weighted Sparse Approximation and the Diagonal Surrogate

Proposition C.1 shows that plain thresholding of $|c_{ij}|$ is the exact solver of the whitened objective. However, the ultimate goal is to reconstruct $\mathbf{W}$, not $\mathbf{W}_L$: the un-whitening transform $\mathbf{L}^{-1}$ is generally non-orthogonal, so two coefficients with identical whitened magnitudes may contribute very differently to the final reconstruction error.

The exact quadratic form. For a candidate coefficient matrix $\mathbf{S}$, the perturbation $\mathbf{B}(\mathbf{C} - \mathbf{S})$ maps back to the original space as $\mathbf{L}^{-1}\mathbf{B}(\mathbf{C} - \mathbf{S})$, whose energy is

$$\left\|\mathbf{L}^{-1}\mathbf{B}(\mathbf{C} - \mathbf{S})\right\|_F^2 = \mathrm{tr}\big((\mathbf{C} - \mathbf{S})^\top \mathbf{G}(\mathbf{C} - \mathbf{S})\big), \qquad \mathbf{G} := \mathbf{B}^\top \mathbf{A}^{-1}\mathbf{B}, \tag{10}$$

with diagonal entries $G_{ii} = \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2^2$. The full form equation 10 couples coefficient rows through the off-diagonal entries $G_{ii'} = \langle \mathbf{L}^{-1}\mathbf{b}_i, \mathbf{L}^{-1}\mathbf{b}_{i'} \rangle$; optimizing under equation 10 directly would require combinatorial search.

The diagonal surrogate. ROCKET replaces $\mathbf{G}$ by the powered diagonal surrogate

$$\mathbf{G} \;\approx\; \mathrm{diag}(\mathbf{G})^{\lambda}, \qquad \lambda \in [0,1], \tag{11}$$

which renders the objective separable across entries. The surrogate has a precise meaning at the extremes:

- $\lambda = 0$: the surrogate is the identity, recovering the *exact* whitened error of Proposition C.1;
- $\lambda = 1$: the surrogate is the diagonal part of the *exact* original-space quadratic form equation 10;
- $\lambda = 0.5$: the geometric mean of the two error models.

Both neglected terms the off-diagonal couplings $G_{ii'}$ and the truncation-residual coupling vanish in the isotropic limit $\mathbf{A} \propto \mathbf{I}$, where $\mathbf{G} = \mathbf{I}$ and the whitened and original geometries coincide. Empirically, the surrogate is insensitive to λ over a wide range (Section 5), and the dual-space scoring consistently outperforms whitened-only scoring and transplanted pruning heuristics (Section 5).

Weighted best-subset thresholding. With $\nu_i := G_{ii}^{\lambda/2} = \left\|\mathbf{L}^{-1}\mathbf{b}_i\right\|_2^{\lambda}$, support selection under the surrogate amounts to the weighted sparse approximation problem

$$\mathbf{S}^{\star} \in \arg\min_{\mathbf{S}} \; \|\mathrm{diag}(\boldsymbol{\nu})(\mathbf{C} - \mathbf{S})\|_F^2 \quad \text{s.t.} \quad \mathrm{supp}(\mathbf{S}) \in \mathcal{F}, \tag{12}$$

where $\mathcal{F}$ encodes the per-column and global sparsity constraints. For a fixed support Ω, the optimal coefficients are $S_{ij}^{\star} = C_{ij}$ on Ω and zero elsewhere, and the objective reduces to

$$\text{const.} \;-\; \sum_{(i,j)\in\Omega} \nu_i^2\, c_{ij}^2 = \text{const.} \;-\; \sum_{(i,j)\in\Omega} \mathrm{imp}_{ij}^2, \qquad \mathrm{imp}_{ij} := \nu_i\, |c_{ij}|. \tag{13}$$

Support selection therefore reduces to maximizing the accumulated importance $\sum_{(i,j)\in\Omega} \mathrm{imp}_{ij}^2$.

Lemma C.2 (Weighted best-subset thresholding)**.** *Let $\nu_i > 0$ and $w_{ij} := \nu_i^2 c_{ij}^2 \geq 0$.*

1. *Under a per-column cardinality constraint $|\Omega_j| = s$, the optimal support keeps the s largest scores $\nu_i|c_{ij}|$ in each column.*
2. *Under the mixed constraint $|\Omega_j| \geq s_0$ for all j and $|\Omega| = K$ (with $K \geq d_2 s_0$), an optimal support is obtained by first keeping the top-s_0 entries in each column and then reactivating the globally largest remaining scores until K entries are selected.*

Proof. (i) The objective and constraint decouple across columns; within a column, keeping the s largest weights is optimal.

(ii) Let T_j denote the top-s_0 entries of column j and $T = \bigcup_j T_j$. An exchange argument shows that some optimal solution contains T: if $i \in T_j \setminus \Omega$ with $|\Omega_j| > s_0$, swapping i with a lowest-weight element of $\Omega_j \setminus T_j$ preserves feasibility and does not decrease the objective. Conditioned on $\Omega \supseteq T$, all column-wise lower bounds are satisfied, so the remaining $K - d_2 s_0$ selections are unconstrained and must be the globally largest remaining weights. □

Rule (ii) of Lemma C.2 is exactly ROCKET's two-stage thresholding procedure: over-sparsify per column to $\mathrm{cr} + \gamma$, then reactivate globally top entries until the target CR is reached.

C.6 Recovery of Standard SVD

We verify that ROCKET reduces to standard data-aware low-rank SVD when no sparsity is applied.

Case $s = k$ (no sparsity). When the column sparsity equals the dictionary size ($s = k$, equivalently $k/s = 1$), no coefficient is removed and $\mathbf{C}_{\mathrm{sparse}} = \mathbf{C} = \boldsymbol{\Sigma}_k \mathbf{V}_k^{\top}$. Taking $\mu \to 0$ in the ridge-regularized least-squares problem equation 7, the MOD update gives

$$\mathbf{D}_{\mathrm{final}} = \mathbf{W}_L \mathbf{C}^{\top}(\mathbf{C}\mathbf{C}^{\top})^{-1} = \mathbf{U}_k \boldsymbol{\Sigma}_k^2 (\boldsymbol{\Sigma}_k^2)^{-1} = \mathbf{U}_k, \tag{14}$$

so that $\widetilde{\mathbf{W}}_L = \mathbf{D}_{\text{final}}\mathbf{C} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$, which is exactly the rank-k truncated SVD of $\mathbf{W}_L$. Empirically, on Llama3.2-1B at CR $= 0.2$, ROCKET with $k/s = 1$ and uniform allocation reproduces SVD-LLM exactly (37.6 average accuracy and 170 WikiText perplexity).

With the ridge term included, $\mathbf{D}_{\text{final}} = \mathbf{U}_k\mathbf{\Sigma}_k^2(\mathbf{\Sigma}_k^2 + \mu\mathbf{I})^{-1}$, which approaches $\mathbf{U}_k$ as $\mu \to 0$. In ROCKET, μ is adaptive and negligible relative to the matrix scale (Appendix H.4).

Case $s < k$ (union of subspaces). Each column retains a different subset of s coefficients, so each reconstructed column lies in a different s-dimensional subspace. The model therefore represents a union of subspaces, strictly more expressive than the single-subspace rank-k SVD.

D Mapping MCKP to Graph Theory.

We reformulate the constrained multi-choice knapsack problem (MCKP) as a shortest-path problem on a directed acyclic graph (DAG), where the key to enforcing the global parameters kept lies in a clever *target-aware sink connectivity rule*. Each node is labeled (ℓ, p), with $\ell \in \{0, \dots, L\}$ denoting the layer index and $p \in \mathbb{Z}_{\geq 0}$ representing the scaled cumulative number of parameters kept up to layer ℓ, discretized via $p = \left\lfloor \texttt{scale} \cdot \sum_{\ell'=1}^{\ell} c_{\ell',i} \right\rfloor$ using $\texttt{scale} = \texttt{param_precision}/C_{\text{total}}$. For each feasible compression option i in layer ℓ, we add an edge from $(\ell-1, p_{\text{in}})$ to (ℓ, p_{out}) with $p_{\text{out}} = p_{\text{in}} + \lfloor \texttt{scale} \cdot c_{\ell,i} \rfloor$ and edge cost $\lfloor \texttt{error_scale_factor} \cdot e_{\ell,i} \rfloor$, while discarding any option violating the per-layer error cap $e_{\ell,i} \leq \alpha \bar{e}_{\text{ref}}$. Crucially, we connect a terminal node (L, p) to the sink *if and only if* $p \leq p_{\max} = \lfloor \texttt{scale} \cdot C_{\text{total}} \rfloor$, where $C_{\text{total}} = (1 - \rho_{\text{target}})P_{\text{orig}}$ is the kept-parameter budget and ρ_{target} is the desired compression ratio (fraction of parameters removed). This structural embedding of the global budget constraint via node naming and selective sink connectivity ensures that any path reaching the sink automatically satisfies the target compression, transforming the constrained combinatorial optimization into an unconstrained shortest-path search solvable exactly (up to discretization) by Dijkstra's algorithm. Since the MCKP is NP-hard, this exact solution is pseudo-polynomial in the number of budget bins; dominated-state (Pareto) pruning retains only non-dominated (cost, error) states, which keeps the search tractable in practice (Appendix F). The graph thus encodes both layer-wise flexibility and global feasibility, with complexity rendered tractable by limiting the candidate space per layer to a modest grid of compression ratios and k_s sparsity levels.

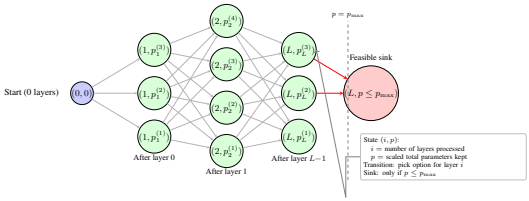


Figure 4: Exact state-space graph matching. Each state (i, p) represents having processed the first i layers with p scaled parameters retained. From (i, p), the algorithm branches to all options for layer i, producing states $(i + 1, p + \Delta p)$. The sink is reachable only from states with $p \leq p_{\max}$, enforcing the global compression ratio by construction.

This graph formulation underlies the dynamic programming procedure in Algorithm 1, where states correspond to nodes and transitions to edges.

E Inference Optimization

To accelerate inference, we leverage MACKO [22], a specialized sparse matrix–vector multiplication kernel that outperforms PyTorch's built-in implementation for structured, column-wise sparse matrices. Because ROCKET employs a calibration-guided, adaptive compression strategy, the sparsity level quantified by the ratio k/s of dictionary atoms to nonzeros per coefficient column varies across layers. This heterogeneity arises from the solution to a constrained multi-choice knapsack problem, which allocates parameters to layers based on their marginal contribution to reconstruction fidelity under a global budget.

Empirically, we observe that MLP layers (gate and up projections) are assigned significantly higher sparsity and lower rank than attention projections (query, key, value, out). This behavior is theoretically justified: MLP weight matrices are substantially larger ($d_{\text{model}} \times 4d_{\text{model}}$) than attention matrices ($d_{\text{model}} \times d_{\text{model}}$), and their calibration-aware reconstruction error increases more slowly with sparsification. Consequently, the optimizer preferentially compresses MLP layers to maximize parameter savings while preserving overall model accuracy.

Figure 5 compares the runtime of MLP projections in Qwen3-8B using either PyTorch's default sparse kernels or MACKO. For large, moderately sparse coefficient matrices (e.g., gate and up projections), MACKO provides consistent speedups; for smaller or denser matrices (e.g., down projections), both implementations perform similarly. In attention layers, where weight matrices are small and less aggressively compressed, we retain PyTorch's native implementation, as it proves faster in practice. Additionally, we fuse dictionaries and coefficients for layers sharing the same input namely, {query, key, value} in attention and {gate, up} in the MLP before applying MACKO.

In terms of theoretical floating-point operations (FLOPs), ROCKET is structurally analogous to CoSpaDi: both represent a weight matrix $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ as $\mathbf{W} \approx \mathbf{BC}$, where $\mathbf{B} \in \mathbb{R}^{d_1 \times k}$ is a dense dictionary and $\mathbf{C} \in \mathbb{R}^{k \times d_2}$ is column-wise sparse. Assuming optimal reuse of the intermediate product $\mathbf{XD}$, the FLOP count is $Nd_1K_{\text{active}} + Nsd_2$, where K_{active} is the number of distinct active atoms. Under a fixed global parameter budget, the total FLOP count may be similar between the two methods.

However, a key distinction lies in budget allocation: CoSpaDi typically enforces uniform sparsity across layers, whereas ROCKET dynamically assigns heterogeneous (k_ℓ, s_ℓ) pairs per layer based on reconstruction sensitivity. This results in lower FLOPs in large, robust layers (e.g., MLP) and higher FLOPs in small, sensitive layers (e.g., attention) even though the latter contribute minimally to total computation due to their size. The net effect is a more balanced per-layer runtime profile and better utilization of sparse kernels, which explains the consistent throughput advantage of ROCKET over CoSpaDi (Table 9) despite comparable theoretical operation counts.

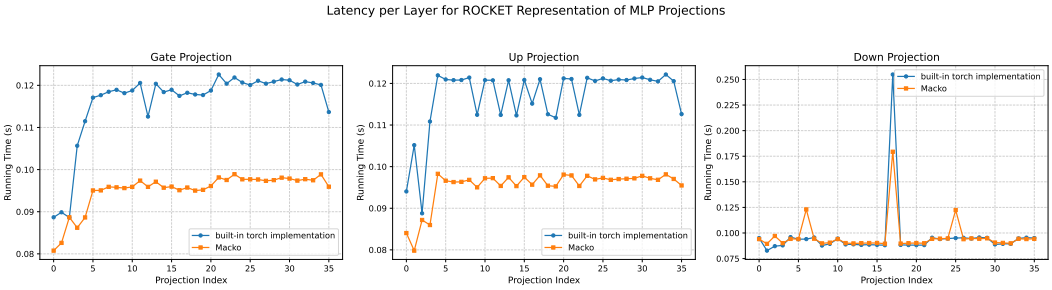


Figure 5: Comparison between PyTorch's built-in sparse-matrix vector multiplication and MACKO across MLP layers in Qwen3-8B. MACKO shows consistently better running time for large coefficients (gate and up projections), while being on par with PyTorch for smaller sparse matrices (down projection).

Table 9: Throughput (tokens/s) for Qwen3-8B with batch_size = 1 and context_length = 256.

Method	20%	30%	40%
SVD-LLM	24.36	24.31	24.72
CoSpaDi	25.45	25.76	25.62
ROCKET	26.74	26.60	26.36

E.1 Environmental Impact

As shown in Table 10, the ROCKET method not only achieves superior compression performance but also offers dramatic environmental benefits. Compared to COSPADI, ROCKET consumes over 100 times less energy, completes compression 96 times faster, and produces 23 times lower

CO_2 emissions. These results highlight ROCKET as both a high-performance and environmentally sustainable solution.

Table 10: Comparison of energy consumption, runtime, and CO_2 emissions for CoSpaDi and ROCKET using the Llama3.2-1B model.

Method	Model	GPU	CPU	Energy Consumed (kWh)	Duration (s)	CO_2 Emissions (kg eq)
CoSpaDi	Llama3.2-1B	1 × NVIDIA A100-SXM4-40GB	AMD EPYC 7742 64-Core Processor	7.88	90080.97	0.782
ROCKET	Llama3.2-1B	1 × NVIDIA A100-SXM4-40GB	AMD EPYC 7742 64-Core Processor	0.0765	930	0.0337

E.2 Hardware Efficiency and Sparse Kernel Compatibility

Recent advances in sparse linear algebra have made heterogeneous sparsity increasingly hardware-friendly. We leverage MACKO [22], a specialized kernel optimized for column-wise sparse matrices, which achieves **1.3–1.5**× throughput gains over cuBLAS at 50% density and **3.5–4.5**× at 10% density. As shown in Table 9, ROCKET consistently outperforms uniform-compression baselines under matched budgets.

While MACKO currently lacks dynamic batching support (planned for future releases), ROCKET's structural advantages persist with standard PyTorch sparse kernels. The FLOP count for a forward pass is $N(d_1 k_{\text{active}} + sd_2)$, where k_{active} denotes the number of distinct active dictionary atoms. Our knapsack allocator preferentially assigns higher sparsity to large, robust layers (e.g., MLP projections), effectively reducing k_{active} and s in compute-heavy components. Empirically, this yields **5–12% lower end-to-end latency** compared to low-rank baselines even without MACKO, confirming that heterogeneous allocation intrinsically improves hardware utilization.

E.3 Serving-Style Throughput and Memory Footprint

To assess ROCKET under realistic serving workloads, we measure total system throughput under concurrent users, following CoSpaDi's evaluation protocol. All throughput values are reported as relative tokens/s with respect to the dense model; values greater than 1.00 indicate higher throughput than the uncompressed model.

Short-prompt throughput. Table 11 reports relative throughput at prompt length $= 1$ under varying concurrency. ROCKET achieves the highest throughput at 1, 4, and 16 concurrent users and remains competitive at 8 users.

Table 11: Short-prompt serving throughput (prompt length $= 1$), relative tokens/s vs. dense model.

Method	1 user	4 users	8 users	16 users
SVD-LLM	1.29	1.33	**1.35**	0.99
CoSpaDi	1.24	1.30	1.29	1.13
ROCKET	**1.31**	**1.37**	1.31	**1.14**

Prompt-length sweep. Table 12 reports relative throughput across prompt lengths and concurrency levels (format: CoSpaDi / ROCKET). ROCKET matches or exceeds CoSpaDi in most configurations.

Table 12: Full prompt-length sweep, relative tokens/s vs. dense (format: CoSpaDi / ROCKET).

Users	Prompt 1	Prompt 16	Prompt 32	Prompt 64	Prompt 128
1	1.24 / **1.31**	1.11 / **1.19**	1.09 / **1.15**	**1.20** / 1.16	**1.14** / 1.12
4	1.30 / **1.37**	1.09 / **1.19**	**1.11** / 1.08	**1.22** / 1.20	**1.18** / 1.12
8	1.29 / **1.31**	**1.08** / 1.07	1.11 / **1.14**	1.24 / **1.25**	1.15 / 1.15
16	1.13 / **1.14**	1.09 / **1.14**	1.08 / **1.15**	1.13 / **1.16**	1.19 / 1.16

These results show that ROCKET is not only accurate but also practically competitive under realistic serving workloads. We note that MACKO currently lacks dynamic batching support; throughput gains are expected to increase once sparse-kernel batching is available.

Memory footprint. Table 13 summarizes the stored parameter count for a single layer $W \in \mathbb{R}^{d_1 \times d_2}$ under different compression strategies.

Table 13: Per-layer memory footprint comparison for $W \in \mathbb{R}^{d_1 \times d_2}$.

Method	Stored Parameters
Dense	$d_1 \cdot d_2$
Rank-k SVD	$k(d_1 + d_2)$
ROCKET	$d_1 \cdot k + s \cdot d_2$

ROCKET stores a dense factor $\mathbf{F} \in \mathbb{R}^{d_1 \times k}$ and a column-wise sparse factor $\mathbf{H} = \mathbf{C}_{\text{sparse}} \in \mathbb{R}^{k \times d_2}$ with s nonzeros per column. Sparse indices for $\mathbf{H}$ are accounted for by truncating the float16 representation by 2 bits, following CoSpaDi's convention.

F Scalability and Runtime Profiling

The computational overhead of ROCKET consists of three components: activation gathering, whitening, and the DP allocation search.

DP Allocation Cost. For each linear module, we compute a single SVD/eigendecomposition. Subsequent rank/sparsity evaluations rely on pointer traversal across singular values and closed-form matrix operations, making the DP search complexity independent of weight matrix dimensions. As shown in Table 14, DP search runtime scales super-linearly in practice (empirically $\approx O(L^{1.8})$) due to Pareto pruning overhead and state management.

Table 14: DP search runtime on a single CPU thread.

Layers	Options	Run Time (s)
112	14	9.80
224	14	35.00
336	14	69.75
448	14	115.93

Table 15: Whitening runtime for varying hidden dimensions (1024 calibration tokens).

Hidden Dim	$X^\top X$ + Cholesky	Time (s)
100	100×100	0.0005
1,000	1000×1000	0.0009
10,000	10000×10000	0.0450
28,672	28672×28672	0.3800
50,000	50000×50000	3.9600

Even for dense 70B-parameter models (448 linear layers), the solver completes in < 2 minutes on a single CPU thread.

Whitening Overhead. Cholesky-based whitening ($\mathbf{L} = \text{chol}(\mathbf{X}^\top \mathbf{X})$) is performed once per layer using a small calibration set. Table 15 demonstrates that this step is negligible compared to SVD/factorization costs.

G Further results

G.1 Detailed Comparison with CoSpaDi and SVD-LLM

We begin this section by providing the detailed per-benchmark results from Figure 2 in Tables 16, 17. As shown, ROCKET outperforms CoSpaDi and SVD-LLM by a large margin across different benchmarks and compression ratios.

Table 16: ROCKET comparison vs low-rank and SDL counterparts in data-aware scenarios on Llama3.2-1B at different compression ratios (CR). Best results are provided in **bold** All experiments are in training-free setup.

Method	CR	Accuracy↑									Perplexity↓	
		PIQA	Hella Swag	LAMBADA	ARC-e	ARC-c	SciQ	Race	MMLU	Avg.	Wiki Text	LAMBADA
Llama3.2 1B	–	74.5	63.7	63.0	60.5	36.2	88.3	37.8	37.0	57.6	12	5.7
SVD-LLM	0.2	62.1	36.4	24.4	36.0	25.1	64.9	29.0	23.0	37.6	170	170
CoSpaDi		66.1	42.9	38.4	39.9	26.0	71.6	31.7	24.8	42.7	64	35
ROCKET		**71.9**	**56.7**	**46**	**56.7**	**32.4**	**88.6**	**36.4**	**30.8**	**52.4**	**18**	**13**
SVD-LLM	0.3	55.7	30.1	9.1	30.5	21.5	45.9	25.8	23.2	30.2	590	2,500
CoSpaDi		56.9	32.4	18.2	31.9	22.1	56.7	28.0	23.1	33.7	290	660
ROCKET		**66.3**	**46.8**	**38.0**	**47.9**	**27.4**	**79.6**	**34.1**	**27.4**	**45.9**	**35**	**26**
SVD-LLM	0.4	51.8	27.3	1.3	26.9	22.9	32.3	24.4	23.0	26.2	1,600	33,000
CoSpaDi		53.5	28.2	3.8	27.8	23.0	36.9	24.0	**23.1**	27.5	800	9,200
ROCKET		**63.9**	**39.4**	**23.8**	**41.0**	**25.7**	**72.1**	**30.9**	23.0	**39.8**	**88**	**130**
SVD-LLM	0.5	51.1	26.6	0.0	26.1	**25.9**	26.1	23.9	23.0	25.4	3,100	100,000
CoSpaDi		51.7	27.0	0.3	26.3	24.0	29.5	24.2	**23.3**	25.8	1,800	73,000
ROCKET		**57.3**	**31.6**	**9.4**	**34.9**	22.6	**50**	**26.1**	22.9	**31.9**	**330**	**2,200**

Table 17: Performance comparison of ROCKET vs SOTA SVD-LLM methods on Llama3-8B at different compression ratios (CR). Best results are highlighted with **bold**.

Method	CR	Accuracy↑									Perplexity↓	
		PIQA	HellaSwag	LAMBADA	ARC-e	ARC-c	SciQ	Race	MMLU	Avg.	WikiText	LAMBADA
Llama3 8B	–	80.7	79.1	75.6	77.7	53.5	93.9	40.3	62.2	70.4	7.3	3.1
SVD-LLM	0.2	71.1	58.4	59.3	55.5	34.0	86.4	35.5	32.6	54.1	41	11
CoSpaDi		75.2	66.5	73.8	66.5	41.6	89.5	38.2	42.8	61.8	20	4.3
ROCKET		**76.9**	**74.8**	73.6	**73.7**	**47.1**	**92.7**	**40.7**	**54.9**	**66.8**	**11**	**3.8**
SVD-LLM	0.3	65.8	46.4	38.1	41.9	27.7	70.0	31.8	27.2	43.6	150	61
CoSpaDi		70.5	56.2	61.3	54.2	33.5	85.7	36.2	32.2	53.7	45	9.2
ROCKET		**76.8**	**69.5**	**70.4**	**70.5**	**42.3**	**90.9**	**39.7**	**47.8**	**63.5**	**15**	**4.4**
SVD-LLM	0.4	60.3	34.5	11.4	32.4	24.5	44.2	25.7	23.1	32.0	550	1,300
CoSpaDi		63.7	41.4	30.3	39.1	26.6	68.5	30.5	25.4	40.7	180	120
ROCKET		**71.9**	**60.4**	**59.3**	**64.4**	**36.4**	**88.1**	**36.8**	**39.7**	**57.1**	**29**	**8.1**
SVD-LLM	0.5	55.1	24.7	1.8	26.0	23.1	30.5	22.4	21.0	25.6	1,800	6,500
CoSpaDi		58.4	31.8	13.6	30.7	24.8	46.2	25.8	23.4	31.8	740	520
ROCKET		**66.9**	**45.6**	**28.4**	**48.2**	**27.7**	**78.2**	**33.2**	**26.1**	**44.3**	**116**	**90**

G.2 Scaling Behavior Across Model Sizes

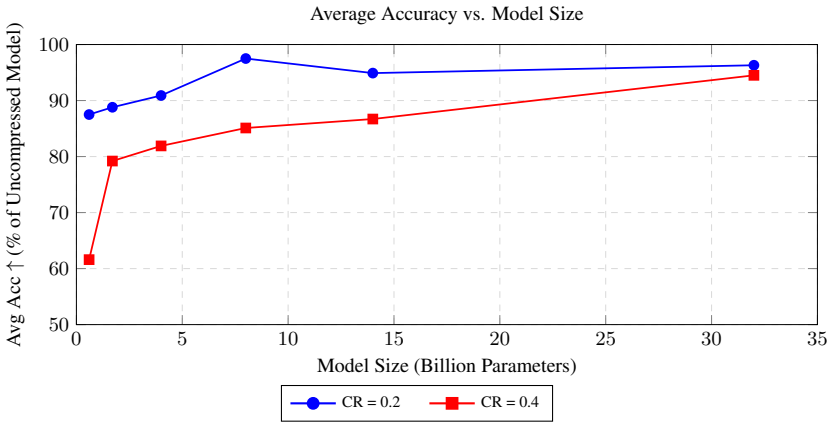


Figure 6: Average accuracy (relative to uncompressed model) as a function of model size for two compression ratios.

In Figure 6, we present the results of compressing Qwen models of varying sizes from 0.6B to 32B parameters at two compression ratios (20% and 40%). The results show that larger models retain a higher fraction of their original (uncompressed) performance after compression. This suggests that larger models may still be significantly underfitted relative to their capacity.

G.3 Evaluation on Advanced Benchmarks

Table 18: Performance comparison of ROCKET against CoSpaDi and SVD-LLM on a new set of benchmarks.

Method	CR	Accuracy↑					
		IFEval	BBH	MATH	GPQA	MUSR	MMLU-Pro
Qwen3 8B	—	39.21	60.86	52.57	36.16	43.12	47.72
SVD-LLM	0.2	25.54	41.00	1.06	28.36	39.81	26.30
CoSpaDi	0.2	28.90	45.25	1.96	28.61	**42.06**	31.46
ROCKET	**0.2**	**31.89**	**50.33**	**11.10**	**31.45**	39.68	**36.65**
SVD-LLM	0.3	22.90	34.42	0.98	25.59	**41.40**	18.82
CoSpaDi	0.3	25.18	38.22	0.98	24.75	38.36	22.81
ROCKET	**0.3**	**25.54**	**47.54**	**2.64**	**29.19**	39.94	**32.27**
SVD-LLM	0.4	22.66	30.24	0.83	23.07	37.70	11.55
CoSpaDi	0.4	**26.14**	32.72	0.76	26.01	38.10	16.74
ROCKET	**0.4**	23.99	**40.06**	**1.28**	**28.85**	**41.00**	**25.53**

In Table 18 we compare against other methods on a new set of benchmarks. These modern benchmarks IFEval [47], BBH [35], MATH [14], GPQA [31], MuSR [33], and MMLU-Pro [40] represent an evolution in the evaluation of large language models (LLMs), targeting more rigorous, diverse, and realistic capabilities. IFEval (Instruction-Following Evaluation) assesses a model's ability to follow precise, verifiable natural language instructions. BBH (Big-Bench Hard) isolates 23 of the most challenging tasks from the original BIG-Bench suite to probe complex reasoning. MATH evaluates advanced mathematical problem-solving across algebra, geometry, and other domains. GPQA tests graduate-level scientific knowledge with high difficulty and minimal data contamination risk. MuSR (Multi-step Reasoning) focuses on multi-hop and long-context reasoning, while MMLU-Pro enhances the original MMLU by increasing answer choices (from 4 to 10) and reducing ambiguity, thereby offering a cleaner, more demanding assessment of expert knowledge. Unfortunately other papers are still following the old benchmarks therefore for fair comparison we stick with them for comparison in the main paper.

G.4 Cross-Architecture and Large-Scale Evaluations

To validate cross-architecture generality, we evaluate ROCKET on Gemma3-12B, Phi-4-14B, Qwen3-30B-A3B-MoE, and Llama3-70B. Results are summarized in Table 19.

We also compare ROCKET against structured depth/width pruning methods on Llama1-7B at 20% compression (Table 20). Despite being entirely training-free, ROCKET achieves state-of-the-art average accuracy, outperforming methods that require costly fine-tuning recovery (e.g., LLM-Pruner, LoRAShear).

G.5 Post-Compression Quantization

Dobi-SVD's reported compression ratios include quantization. To compare at matched effective compression ratios (Table 2), we apply a simple weight-only post-training quantization to ROCKET: both factors $\mathbf{F}$ and $\mathbf{H}$ are quantized independently to INT8 with min-max scaling, without calibration data. This conservative setting matches Dobi-SVD's precision and is not tuned for ROCKET. ROCKET is also compatible with stronger quantizers: applying INT8 GPTQ (via LLMCompressor) to ROCKET at 20% compression on Llama3-8B yields an effective compression of $\approx$60% with almost no additional degradation (Table 21).

Table 19: ROCKET performance across diverse LLM families and scales.

Model	CR	Avg. Accuracy (↑)	Lambada PPL (↓)
Gemma3-12B (dense)	–	72.88	4.16
SVD-LLM	0.3	51.70	74.61
ROCKET	0.3	**68.50**	**4.23**
ROCKET	0.4	**62.75**	**8.36**
Phi-4-14B (dense)	–	72.05	3.49
SVD-LLM	0.3	64.62	4.39
ROCKET	0.3	**72.39**	**3.06**
ROCKET	0.4	**68.20**	**3.67**
Qwen3-30B-A3B-MoE	–	72.90	4.13
SVD-LLM	0.3	46.12	262.19
ROCKET	0.3	**67.90**	**13.70**
ROCKET	0.4	**54.80**	**107.30**
Llama3-70B (dense)	–	77.30	2.69
SVD-LLM	0.2	72.03	2.92
ROCKET	0.2	**76.90**	**2.71**
SVD-LLM	0.4	55.87	11.78
ROCKET	0.4	**73.32**	**2.54**

Table 20: Comparison with depth/width pruning methods on Llama1-7B (20% CR).

Method	Training-free	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
Baseline	–	76.5	79.8	76.1	70.1	72.8	47.6	57.2	68.59
LLM-Pruner	✗	66.79	77.58	68.48	64.96	64.06	37.88	39.00	59.82
LoRAPrune	✗	65.82	79.31	70.00	62.76	65.87	37.69	39.14	60.05
WANDA	✓	65.75	74.70	64.52	59.35	60.65	36.26	39.40	57.23
ShortGPT	✗	68.26	72.28	61.70	63.77	60.22	39.00	41.60	58.12
LoRAShear†	✗	70.17	76.89	68.69	65.83	64.11	38.77	39.97	60.63
LoRAShear	✗	72.78	76.36	69.49	67.63	69.02	39.47	40.78	62.22
ROCKET	✓	**75.69**	**78.40**	**71.80**	**68.67**	**71.80**	**39.51**	**42.60**	**64.07**

G.6 Post-Compression Healing

In Table 22, we provide the per-benchmark results corresponding to the healing experiment. As noted, we are approaching the performance of models trained from scratch and in some cases nearly matching or surpassing them despite using a compressed model. Specifically, we compressed Qwen3-14B to an 8B model and applied a very limited fine-tuning phase (using only approximately 30 million tokens) while keeping the sparsity pattern fixed an approach that is known to be suboptimal. Nevertheless, the resulting model achieves performance comparable to the original Qwen3-8B trained from scratch. We expect that fine-tuning with higher quality, carefully curated data would further improve results. Moreover, as previously mentioned, enabling trainable sparsity patterns remains a direction for future work.

H Further Ablations

H.1 Ablation on calibration data

Table 23: Ablation on calibration data for Llama3.2-1B at 20% compression (CR = 0.2). The first row shows the uncompressed baseline. Higher accuracy and lower perplexity are better.

Calibration Data	CR	Accuracy↑									Perplexity↓	
		PIQA	HellaSwag	LAMBADA	ARC-e	ARC-c	SciQ	Race	MMLU	Avg. Acc.	$\text{WikiText}_{\text{word}}$	$\text{LAMBADA}_{\text{PPL}}$
None (Baseline)	0	74.5	63.7	63.0	60.5	36.2	88.3	37.8	37.0	57.6	12	5.7
FineWeb	0.2	71.9	56.7	46.0	56.7	32.4	**88.6**	**36.4**	**30.8**	52.4	18	13
PTB	0.2	70.3	54.9	47.3	55.2	31.9	87.8	34.5	26.7	51.1	21	13
WikiText	0.2	70.7	56.3	50.9	**57.9**	33.6	88.1	35.5	28.3	52.7	**17**	11
Alpaca	0.2	**73.7**	**57.2**	**53.2**	**57.9**	**34.9**	87.5	35.4	30.5	**53.8**	20	**9.7**

Table 21: ROCKET combined with INT8 GPTQ on Llama3-8B.

Method	Effective CR	Avg. Acc. ↑	PPL ↓
ROCKET (20% CR)	0.20	66.8	11.04
ROCKET (20% CR) + INT8 GPTQ	≈0.60	66.6	11.04

Table 22: Post-compression healing results on Qwen3-8B models.

Method	PIQA	HellaSwag	Lambada	ARC-e	ARC-c	SciQ	Race	MMLU	WikiText	Perplexity	Avg. Acc.
Qwen3-14B (dense)	79.86	78.85	67.88	82.82	60.23	96.50	43.25	77.20	11	3.7	73.32
Qwen3-8B (dense)	77.70	**74.90**	64.10	**80.70**	**56.70**	**95.70**	**40.90**	**73.00**	**12**	4.6	**70.46**
ROCKET-Qwen3-8B (training-free)	72.68	62.63	**70.26**	67.76	44.19	91.20	39.80	59.99	25	**3.8**	63.56
ROCKET-Qwen3-8B (healed)	**78.51**	74.67	65.55	75.29	53.07	93.50	37.89	65.23	13	4.7	67.96

We evaluate the sensitivity of ROCKET to the choice of calibration dataset by comparing four sources: FineWeb [28], PTB [25], WikiText [26], and Alpaca [36]. As shown in Table 23, while instruction-tuned data such as Alpaca yield slightly higher average accuracy (53.8 vs. 52.4), the differences across datasets are relatively small, confirming that ROCKET is robust to the calibration data choice. Nevertheless, to ensure a fair comparison with CoSpaDi, which uses FineWeb as its default calibration data, we adopt FineWeb for all primary experiments reported in this paper.

H.2 Ablation on the Sparsification Strategy

Table 24: Ablation on sparsification strategies for Llama3.2-1B at 20% compression (CR = 0.2). All methods use the same calibration data and global parameter budget.

Sparsification Strategy	CR	Accuracy↑									Perplexity↓	
		PIQA	HellaSwag	LAMBADA	ARC-e	ARC-c	SciQ	Race	MMLU	Avg. Acc.	$\text{WikiText}_{\text{word}}$	$\text{LAMBADA}_{\text{PPL}}$
None (Baseline)	0	74.5	63.7	63.0	60.5	36.2	88.3	37.8	37.0	57.6	12	5.7
ROCKET	0.2	**71.9**	**56.7**	46.0	**56.7**	32.4	**88.6**	**36.4**	**30.8**	**52.4**	**18**	**13**
Per-Row Sparsification	0.2	67.8	48.1	32.5	48.6	29.4	79.9	31.6	26.7	45.6	32	38
Global Importance Sparsification	0.2	69.5	54.7	42.7	53.5	30.7	85.7	35.5	28.1	50.1	19	17
Whitened-Space Importance Only	0.2	71.5	55.7	**47.3**	56.3	**32.9**	87.9	35.4	28.9	52.0	19	**13**

To evaluate the impact of our sparsification strategy, we compare ROCKET against three alternative approaches for pruning the coefficient matrix $\mathbf{C} = \mathbf{\Sigma}\mathbf{V}^\top$: (1) **Per-Row Sparsification**, where importance scores are computed identically but sparsity is enforced independently per row (breaking column-wise structure); (2) **Global Importance Sparsification**, which ignores any structural constraints and simply zeros out the globally least-important entries based on the full importance matrix; and (3) **Whitened-Space Importance Only**, which disables the original-space fidelity term by setting $\lambda = 0$ in equation 5 (theoretically optimal with respect to the whitened space). Table 24 shows that ROCKET's **column-aware, activation-and-weight-balanced sparsification** outperforms alternatives, demonstrating the efficiency of both structural awareness and dual-space importance scoring.

H.3 Search Space Construction

To maintain tractability, we decouple the sparsity ratio (k/s) from the dynamic programming (DP) search space. For each candidate cr, we greedily select the k/s pair that minimizes reconstruction error, leaving the DP solver to optimize only over the discretized cr candidates. In practice, we use a cr **step of 0.05** and restrict $k/s \in [1, 4]$ with step 0.1, yielding 14 compression candidates per layer, each evaluated across 30 sparsity configurations. Crucially, DP complexity depends on the number of layers and candidates (Appendix F), *not* on weight matrix dimensions.

H.4 Ridge Regularization of the Dictionary Update

When solving for the refined left factor $\mathbf{D}$ post-sparsification, we apply a small adaptive ridge regularization μ to prevent ill-conditioning in $\mathbf{C}\mathbf{C}^\top$:

$$\mu = \max\left(10^{-6},\ 10^{-3} \cdot \frac{\operatorname{trace}(\mathbf{C}^\top\mathbf{C})}{k}\right).$$

This keeps regularization negligible relative to matrix scale while ensuring numerical stability during least-squares inversion.

H.5 Ablation on Reconstruction Error Metric

In ROCKET, the layer profiling stage enumerates a set of candidate compression configurations per layer, where each candidate is defined by a pair (cr, ks), namely, the compression ratio and the sparsity ratio applied to the coefficient matrix. Given the original layer weights $\mathbf{W}$, each candidate (cr, ks) induces a reconstructed weight matrix $\tilde{\mathbf{W}}$. For each candidate, we compute a reconstruction error, which serves as the estimated *cost* in the constrained multi-choice knapsack problem. The optimizer then selects one candidate per layer to minimize the sum of estimated error while satisfying the global parameter budget.

We evaluate four variants for this per-candidate error estimate: (1) **relative Frobenius error** $\|\mathbf{W} - \tilde{\mathbf{W}}\|_F / \|\mathbf{W}\|_F$ (our default), (2) ℓ_1 **distance** $\|\mathbf{W} - \tilde{\mathbf{W}}\|_1$, (3) **mean cosine distance across columns**, and (4) **spectral distance** $\|\mathbf{W} - \tilde{\mathbf{W}}\|_2$. All variants run under the same configurations. As shown in Table 25, the relative Frobenius error yields the best downstream performance, while ℓ_1-based estimates lead to significant degradation, highlighting the effectiveness of the former metric for effective budget allocation.

Table 25: Ablation on reconstruction error metric for layer profiling in ROCKET for Llama3.2-1B at 20% compression.

Error Metric	**Avg. Acc.** ↑	**Perplexity** ↓
Dense (Baseline)	57.6	12
L1 Distance	35.2	180
Mean Cos Columns	51.1	19
Spectral Distance	51.3	18
Frobenius (ours)	**52.4**	**18**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: main contributions of the paper are clearly stated in the Abstract and Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: limitations are stated with the conclusions, regarding scaling to MoE architectures and inference time optimization.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide a proper flow of the math for the proposed method and a follow-up detailed explanation is added to the appendix for the readers that are not familiar with this field.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Implementation details and summary of the algorithm are provided, and the implementation of the method is publicly available at `https://github.com/mts-ai/ROCKET`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All data used on this work are publicly available and the implementation of the method is publicly available at `https://github.com/mts-ai/ROCKET`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: No training is done in the scope of this paper, testing is done using open publicly available benchmarks and libraries.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: under the same calibration data and library versions the results are exactly the same even for different seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the running time for some experiments as well environmental impact study in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that it does.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: discussed at the beginning of the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: the paper poses no such risks

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All experiments are done on publicly available models, models compressed and released using the proposed method inherits the license of the original model.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: the paper does not release new assets

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The LLM is used to polish the paper and correct grammar issues only.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.