

FLAR-SVD: Fast and Latency-Aware Singular Value Decomposition for Model Compression

Moritz Thoma^{1,3†*} Jorge Villasante^{1,3†} Emad Aghajanzadeh^{1,3} Shambhavi Balamuthu Sampath^{1,3}
Pierpaolo Mori³ Maximilian Groetzinger^{1,3} Daniil Dylkin^{1,3} Manoj-Rohit Vemparala³ Nael Fasfous³
Alexander Frickenstein³ Daniel Mueller-Gritschneider² Ulf Schlichtmann¹

¹Technical University of Munich ²TU Wien ³BMW Group

Abstract

Advanced deep learning architectures have achieved exceptional prediction performance but come with significant computational demands, posing challenges for deployment on resource-constrained devices such as edge devices. While pruning techniques offer a way to reduce model complexity, they often lead to substantial accuracy loss and can require extensive retraining. Alternatively, Singular Value Decomposition (SVD) provides a promising solution by decomposing model weights into lower-dimensional representations, thus maintaining a closer representation of the original features and preserving accuracy. Despite progress in this domain, approaches targeted on vision model architectures typically rely on uniform compression or slow, computationally expensive rank search methods that do not account for latency improvements. In this paper, we introduce Fast, Latency-Aware Rank Singular Value Decomposition (FLAR-SVD), a novel approach that leverages inherent SVD properties to accelerate the rank search process and incorporates latency tuning to further optimize performance for hardware targets. We demonstrate the capability of our approach across CNN, ViT and Mamba architectures on both server and edge hardware. For DeiT we achieve 81.0 % accuracy on ImageNet with only 1 epoch of fine-tuning, while reducing latency by 30 % over the baseline. Our code is available in <https://github.com/MoritzTho/FLAR-SVD>.

1. Introduction

Deep learning models have achieved remarkable success across a wide range of applications [10, 17, 21]. However, as these models grow in size and complexity, their computational and memory demands pose significant challenges for real-world deployment, especially on resource-constrained

mobile devices. Structured pruning has emerged as an effective technique for reducing the computational burden of deep learning models. By removing model substructures, such as entire neurons or filters, these methods can achieve substantial reductions in model size and processing time. However, this often comes at the cost of significant accuracy degradation, which necessitates extensive fine-tuning to recover lost performance. Hence, while structured pruning effectively reduces latency, the trade-off between compression and accuracy remains a critical challenge [7, 8].

Low-rank factorization techniques based on Singular Value Decomposition (SVD), offer a promising alternative for compressing deep learning models. SVD-based compression decomposes weight matrices into two smaller sublayers with a smaller intermediate rank dimension, preserving critical structural information while reducing the number of parameters and floating-point operations (FLOPs). Unlike structured pruning, SVD-based methods introduce a more gradual transformation of model weights, leading to a less destructive impact on model performance and, in many cases, reduced fine-tuning costs [24, 28]. Building on these advantages, recent research has explored using activation-based weight importance to improve decomposition quality and optimize rank selection for an even better trade-off between FLOPs and accuracy.

However, despite their theoretical efficiency, current SVD-based methods face several critical challenges: First, while reducing FLOPs is an important metric for model efficiency, it does not necessarily translate into real hardware acceleration [3]. Second, existing rank search strategies are often relying on slow, or ineffective optimization to determine rank choices for each layer [18, 25, 28]. Third, current vision model compression techniques [9, 12, 25] do not leverage advanced decomposition strategies that have been applied in large language models (LLMs), leaving potential efficiency gains unexplored.

Backed by observations presented in Figure 2, to address these challenges, we present a novel approach to compression that combines fast, latency aware rank search with data

*Corresponding author moritz.thoma@tum.de

[†]Equal Contribution.

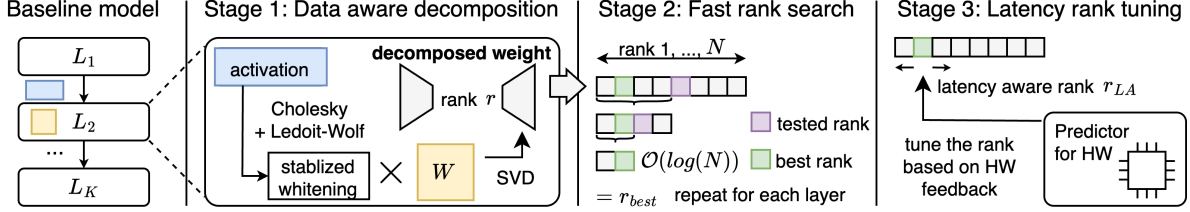


Figure 1. Overview of our FLAR-SVD approach, showing the three stages of (1) enhanced decomposition, (2) our feature similarity binary search and (3) latency rank adjustment.

advanced SVD techniques for vision models. Our approach introduces three key innovations. First, we **integrate the Ledoit-Wolf Shrinkage Estimator into data enhanced Cholesky SVD** to improve its stability and use it for vision models (Figure 2a). Second, we propose a **feature-similarity-driven rank search algorithm** that rapidly determines efficient rank settings leveraging inherent properties of Cholesky SVD, balancing compression and accuracy while avoiding the computational overhead of gradient-based optimization (Figure 2b). Finally, we incorporate a **hardware-aware component** into our rank selection process, ensuring that compression decisions lead to actual latency reductions rather than relying solely on FLOP-based approximations (Figure 2c). Through extensive experimentation across a diverse set of models and hardware targets, we demonstrate that our approach sets a new state-of-the-art in SVD model compression.

2. Related Work

Improved Singular Value Decomposition Algorithms. Beyond the capabilities of SVD to derive a matrix into orthonormal singular vectors that explain variance proportionally to the truncation rank, performing standard factorization only considers the magnitude of the weights, without weighing-in the importance of said weights for a particular task. Hence, when solely focusing on weight reconstruction via Low Rank Factorization (LRF), the vanilla-SVD method results are suboptimal, as it does not take into account feature distribution on the lower-rank matrix [12, 20]. To compensate for this loss of information, Hsu et al. [12] introduced FWSVD with the Fisher information [19] as a 2nd-moment weighting mechanism to enhance decomposition. As promising as it is, this method requires access to gradients to obtain the Fisher information and apply Fisher weighting in a row-by-row fashion. Following this path, Yuan et al. [28] proposed ASVD, leveraging the feature’s activation distribution. This method aims to absorb the outliers in the weight matrix and thereby enhance decomposition accuracy. Similarly, Azizi et al. [1] propose MemoryViT that uses activation to improve their SVD representativeness. Even though vanilla-SVD ensures the singular

vectors are orthonormal, this does not mean that the previously enhanced feature map will be reconstructed linearly with the matrix reconstruction - a problem that persists despite activation awareness [1, 28]. Taking this into consideration, Wang et al. [24] present a truncation-aware LRF weighing-in the Cholesky Decomposition of input activation to directly map the compression loss to singular values. Going in a different direction, Wang et al. [23] explore the benefits of performing SVD factorization on 1st-order unstructured pruned matrices, reducing the complexity while weighing-in the sparsity. Other methods try to reduce the error of the decomposition by using an additional sparse matrix [20, 29] or an additional low rank module [1].

Recent advancements in LRF [24] have been limited to LLMs, while approaches for vision models have not adopted them [9, 18] or have added additional parameters to compensate for the error instead [1, 20, 29], unnecessarily increasing complexity. Differently, we adopt recent advancements in LLM LRF to vision models, improve on them, and use their intrinsic properties for our approach.

Rank selection in SVD. Recent LRF approaches use fixed selection based on a global compression rate [9, 13, 29]. However, rank selection is an important consideration when performing SVD, considering that not all matrices are equally compressible. Yuan et al. [28] set a varying thresholds searching process to truncate singular values based on the layer sensitivity. In a similar way, Wang et al. [23] introduce a learnable parameter to signal the importance of each singular value, based on L_0 norm regularization. Similarly, ComCat [25] uses gradient based updates as foundation to determine the best rank configurations. FLoRA [2] introduces a two-stage neural architecture search (NAS) approach that decouples training and search, speeding up the search process. Lastly, Azizi et al. [1] and Luo et al. [18] optimize ranks using greedy search based on a cosine similarity criterion between original and reconstructed matrices and an activation aware energy loss criterion, respectively.

While all of these approaches achieve rank reduction, they come with substantial upfront cost [2], use lengthy gradient optimization [23, 25], or rely on criteria with limited effectiveness [1, 18, 28]. Moreover, no consideration has been made into hardware aware rank search, with only

FLOPs advantages shown. Differently we introduce a fast, binary rank search that is built upon inherent properties of SVD and uses latency considerations in its rank choices.

3. Methodology

Recent advancements in low-rank decomposition methods have primarily focused on using FLOPs as a proxy for hardware efficiency, relying on slow or suboptimal search algorithms, and do not apply SOTA decomposition techniques on vision models. Backed by observations presented in Figure 2, to address these challenges, we introduce Fast, Latency-Aware Rank Singular Value Decomposition (FLAR-SVD), a novel compression approach that addresses these limitations. FLAR-SVD incorporates three key innovations: (1) a stabilized Cholesky SVD for more robust low-rank decomposition, (2) an efficient rank search that exploits the properties of SVD, and (3) a latency-aware rank adjustment that optimizes performance across different hardware platforms. An overview of our approach is provided in Figure 1.

3.1. Preliminaries

Singular Value Decomposition (SVD) is a factorization method in linear algebra that decomposes a matrix A ($m \times n$) into three matrices U , Σ , and V^T , allowing the analysis of the matrix properties and the reconstruction of the original matrix with reduced dimensionality.

- U ($m \times m$) contains the left singular vectors which form an orthonormal basis.
- Σ ($m \times n$) is a diagonal matrix with non-negative singular values, which are ordered in a hierarchical manner to represent the importance of each singular vector in terms of the variance in the reconstruction error.
- V ($n \times n$) contains the right singular vectors which form an orthonormal basis.

We can reconstruct the original matrix with the matrix multiplication $A = U\Sigma V^T$. Keeping in mind the properties of U , S , and V , matrix A can as well - up to a certain extent - be reconstructed with a lower rank k (hence low-rank decomposition): $A' = U_k \Sigma_k V_k^T$. Here, U_k ($m \times k$), Σ_k ($k \times k$), and V_k ($k \times n$) are truncating to an arbitrary k number of singular vectors before reconstructing A . Truncating k in turn lowers the parameters and computational complexity with respect to A , thereby resulting in compression.

3.2. Data aware matrix decomposition

Performing vanilla SVD, without any importance metric will base its reconstruction loss solely on the weight's magnitude. However, keeping in mind the non-linear behavior of DNNs, weight's importance lie as well in the interaction with activations. In other words, there is an obvious need to couple the optimization objectives of compression loss and task performance [12, 24].

To this end, SVD-LLM [24] introduced a scaling matrix S obtained by applying Cholesky decomposition to the variance-covariance matrix of the input activations for each weight matrix. Thus, a whitened scaling matrix is applied to the weight matrix and the product is low-rank decomposed; following $Y = XS^{-1}(SW)$, where SVD was applied to the terms in parentheses and S^{-1} ensures that the scaling matrix does not affect the output. This has shown to provide SOTA results, outperforming other scaling methods. Nevertheless, this method requires a certain arbitrary number of calibration samples to obtain the activations, and it can degrade or even fail in certain cases such as when the resulting variance-covariance matrix is not positive definite (requisite for Cholesky) or when the whitened matrix is non-invertible.

Thus, as an enhancement, we propose using the Ledoit-Wolf Shrinkage Estimator to stabilize the resulting variance-covariance matrix obtained through several samples across a handful of batches [14]. Hence, by using Ledoit-Wolf Shrinkage, we tackle the challenges observed in the sample variance-covariance matrix, especially when the proportion of observations to features is relatively small or when the data contains outliers.

For the implementation, we calculate the raw scaling matrix from the input activation's features per sample and average through the whole batch.

$$S_{raw} = \frac{(X - \bar{X})^T (X - \bar{X})}{n} \quad (1)$$

Then, we compute the target, in this case identity matrix, scaled by the average variance per feature to represent a more stable estimate of the covariance.

$$\text{Target} = I * \bar{\text{diag}}(S_{raw}) \quad (2)$$

The Optimal Shrinkage Intensity determines how much adjustment is required towards the shrinkage target.

$$\text{Shrink}_{\text{int}} = \frac{\text{Trace}(S) - \text{Trace}(\text{Target})}{|S - \text{Target}|_F^2} \quad (3)$$

Now, we can apply the shrinkage to obtain S_{shrinked} , which we can finally use for obtaining the whitened scaling matrix using Cholesky decomposition $\text{cholesky}(S_{\text{shrinked}})$.

$$S_{\text{shrinked}} = (1 - \text{Shrink}_{\text{int}}) \cdot S + \text{Shrink}_{\text{int}} \cdot \text{Target} \quad (4)$$

The enhancements of the shrinkage with respect with the plain variance-covariance implementation are:

- **Improved estimation stability and reliability** achieved by shrinkage of extreme eigenvalues of the covariance matrix towards more centered ones, effectively reducing noise and outliers in data.

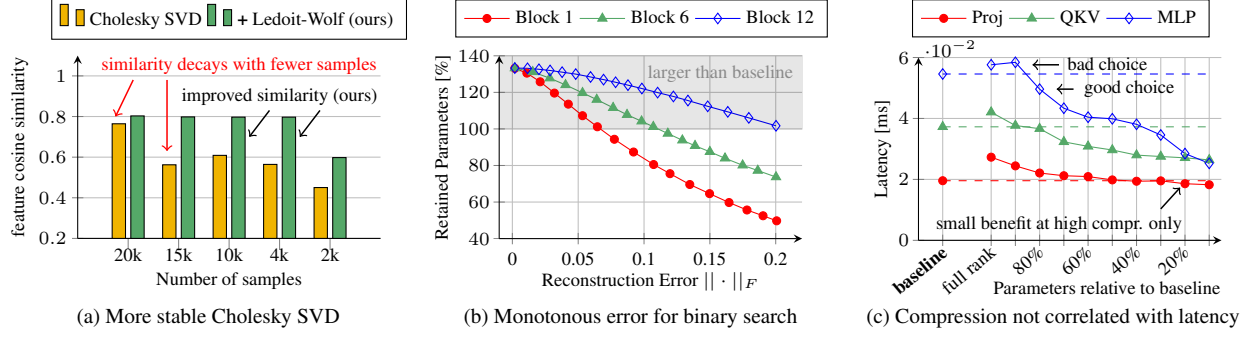


Figure 2. 2a shows the impact of reduced sample pool using vanilla Cholesky and our Ledoit-Wolf enhanced version. 2b and 2c show Frobenius norm w.r.t. uncompressed layer across different blocks and latency impact across different layers at varying compression ratios.

- **Positive definiteness** ensured by this method: all eigenvalues are positive, enabling a cleaner subsequent implementation of the Cholesky decomposition.
- **Robustness in high dimensions** yielded by a variance which is significantly reduced (through shrinkage), which is important when data samples are exceeded by the number of features.

The Ledoit-Wolf Shrinkage Estimator improves understanding of the feature’s relationships enhancing the performance of the decomposition, especially when dealing with limited calibration samples as shown in Figure 2a.

3.3. Searching for Ranks

Addressing the limitations of existing rank search methods, we present an optimized, latency-aware search algorithm that integrates (1) search criteria, (2) search algorithm, and (3) dynamic latency adjustments for enhanced performance.

3.3.1. Search Criteria

For the layer-wise rank search, we propose two different criteria:

- one for a faster deduction of a suitable rank,
- and the other one with a more precise and informed context on the effects of compression to that specific rank.

For the first criteria, we use cosine similarity between original matrix and the reconstructed matrix as a proxy for performance (Eq. 5). This does not require additional calibration data and provides a comprehensive compressed set of matrices as the reconstructed matrix already holds some data insights through our SVD. However, this does not take into account the specific layer’s importance in the model’s context. Therefore, setting cosine similarity as a global criterion to reproduce the original weight will result in important or even critical layers being reconstructed with the same criteria as less important ones that could tolerate some more deviation from the original weights.

$$thresh_{sim} = cos_{sim}(A, U_k S_k V_k^T) \quad (5)$$

For the second criterion, we utilize Mean Squared Error (MSE) between the last feature of the model in both the compressed and uncompressed states. Using the last feature helps normalize the error, as it propagates from layer k to the end, providing a standardized metric to measure the compression impact of all layers of the same model against a single arbitrary threshold. Unlike the first criterion, this approach requires additional calibration data, which means more time per rank option (per layer) to process and obtain sufficient logits and feature differences to make an informed decision. However, with the assistance of our fast and latency-aware rank search algorithm, we can quickly filter out compression rates that do not provide benefits and search the remaining ones very rapidly. Thus, even though it requires forward passes, it can be executed in a matter of minutes (the exact time depends on the number of batches used).

3.3.2. Fast Rank Search via Binary Search

Several SOTA papers evaluating low-rank decomposition methods on DNNs perform uniform rank selection based on a global compression rate for the sake of simplicity. Hence, the rank is selected as a fraction of k_{equiv} , selected to maintain an equivalent number of parameters as the original matrix A ($m \times n$):

$$k_{equiv} = \frac{m \times n}{m + n} \quad (6)$$

Nevertheless, this is proven to be ineffective as different layers have different importance in the overall model structure. Furthermore, the features extracted from layers deeper in the model may get harder to represent with low rank structures as they become more informative and condensed. As seen in Figure 2b, the same layer structure (i.e. QKV matrix in DEIT-Base model) follows a distinct response to compression depending on the block it pertains to. Also, we can observe that when reconstructing from the equivalent rank (noted as the grey-white interface), the error triples (for this specific set of layers) depending on which block it comes from.

Unfortunately, doing rank search to avoid these hurdles can be extenuating and hardware costly. Keeping in mind that we found our maximum limit at rank k_{equiv} - the rank that results in the same number of parameters as the original matrix A - and our minimum at rank 1, we can proceed with the search algorithm. Since layer susceptibility and complexity varies, doing the search run up-down or down-up would be a mere guess and can be beneficial in specific models and for specific criteria, but cannot be generalizable.

To tackle this, we propose a fast rank search using a binary decision tree to accelerate iterations, as shown in Algorithm 1. This way, we reduce the search space from $k_{equiv} - 1$ to $\log_2(k_{equiv})$, massively accelerating the search process. As an example, for a rank $k_{equiv} = 1024 = 2^{10}$, we only need to perform 10 iterations to find the optimal solution. However, binary search requires sorted values in the search space to work. By design, we provide this property with our previous contribution of enhanced Cholesky SVD which ensures monotonicity in feature error vs compression rate (Figure 2b), perfectly linking our compression and search.

Algorithm 1: Fast, binary rank search

Require: A matrix A , an equivalent rank k_{equiv} and a target similarity threshold $thresh_{sim}$

- 1: $U, \Sigma, V^T = \text{SVD}(A)$
- 2: $low \leftarrow 1$
- 3: $high \leftarrow k_{equiv}$
- 4: **while** $low \leq high$ **do**
- 5: $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
- 6: $A_{rec} \leftarrow U_{mid} \Sigma_{mid} V_{mid}^T$ {Low-rank A reconstructed}
- 7: $model_{lr-A} \leftarrow model.replace(A, A_{rec})$
- 8: $v_{sim} \leftarrow sim_{feat}(model, model_{lr-A})$ {feature similarity w.r.t original}
- 9: **if** $v_{sim} < thresh_{sim}$ **then**
- 10: $low \leftarrow mid + 1$ {Search right half}
- 11: **else**
- 12: $high \leftarrow mid - 1$ {Search left half}
- 13: **end if**
- 14: **end while**
- 15: **return** mid {Found rank that maintains $thresh_{sim}$ }

3.3.3. Latency aware rank search

When compressing different layers into different ranks, we found that some can maintain performance at lower ratios than others. A naive assumption might be that all of these layers contribute the same to the latency once compressed and that 50% uniform compression is equivalent to compressing (distributively) all layers around 50% as long as the compression does not lie beyond k_{equiv} . However, this is also not true, as the surplus in layer-wise throughput will

depend on several factors such as weight size, shape, input, and - most critically - the deploying hardware. Figure 2c showcases the latency values of different blocks in the first layer of Swin at different compression rates. A first observation is that all of the layers perform worse in the full-rank representation. Second, we can see varying ratios at which the original latency is recovered and enhanced by the low-rank reconstruction. A third, noteworthy observation is that the projection matrix (sized 128×128) is not even achieving this inflection point at 10%, a very low rank we are not covering in the scope of this research paper.

Based on these findings, we propose a layer-specific criteria to update the Algorithm 1. In the "while" loop, before performing the similarity operation, which might be somewhat costly, we propose evaluating the rank's effect on latency with a predictor trained on hardware-specific data. This sophisticated regressor requires input shape, output features and the selected rank, and will return a latency recovery ratio over 1 if not favorable or under 1 if it enhances throughput. Furthermore, some hardware settings have an enhanced response when it comes to matrix multiplications with matrices shaped with multiples of 8. Hence, a minimal update to enforce each evaluated rank to lie between hardware friendly increments (i.e., 8) would be beneficial. With all these considerations, we can update the code with Algorithm 2.

Algorithm 2: Latency-aware update

Require: Input size X , matrix A to evaluate, and a potential rank mid

- 1: $lat_{recov} \leftarrow predict_{lat}(X, A, mid)$
- 2: **if** $lat_{recov} > 1$ **then**
- 3: **return** 0 {Discard right half, search lower ranks}
- 4: **else**
- 5: $proceed \leftarrow mid$ {proceed with SVD and similarity evaluation of current rank}
- 6: **end if**
- 7: **return** $proceed$ {Current rank supports latency reduction}

4. Evaluation

4.1. Experimental Setup

Our evaluation is done on the popular ImageNet-1K [6] dataset. All compression approaches start from the same model checkpoints using similar compression targets. For ComCat [25] we reuse their official implementation¹, other SVD compression approaches are reimplemented. In cases where finetuning is performed we rely on the standard DeiT

¹<https://github.com/jinqixiao/ComCAT>

recipe [21], with learning rate modified to 1/15 of the original one, weight decay set to 1e-8 and a shortened lr schedule. The finetuning is run for 1 or 5 epochs. To evaluate our approach on hardware, we assess latency on a single Nvidia V100, Nvidia Jetson Orin and Qualcomm Snapdragon 8 Gen 2. On Nvidia hardware we use TensorRT 8.6.1.6 for runtime evaluation while using Qualcomm AI Direct SDK v2.22 for Qualcomm compilation and deployment. For the reported latency we average 200 inference runs at 3x224x224 and batch size 20 for Nvidia V100 and batch size 1 for the other two. For accuracy we report the values seen before deployment. For our rank search we construct a latency predictor for the targets, details are given in the supplementary in Section 7. Results on downstream tasks may also be found in the supplementary in Section 6.

4.2. Comparisons to State-of-the-Art

4.2.1. FLAR-SVD on ImageNet-1K Classification

We compare the compression capability of FLAR-SVD with a wide range state-of-the-art SVD compression approaches on Convolutional, Vision Transformer, and a novel Mamba architectures on Nvidia V100 in Table 1. We further evaluate our approach on both Jetson Orin and Qualcomm Snapdragon 8 Gen 2 edge platforms in Table 2.

Comparison on Server Hardware. FLAR-SVD manages to get consistent improvements in both accuracy and latency. The accuracy improvements over other methods across all models are ranging from a modest 1.0 p.p. for Swin to an impressive 11.6 p.p. for ConvNeXt. Moreover, without any adjustments to our method, our FLAR-SVD also works on the novel MambaVision architecture, improving by 2.2 p.p. showcasing its versatility. Impressively, while achieving these accuracy improvements, our models still result in better latency on hardware. For ConvNeXt, which has the largest accuracy increase, the latency improves by 3.9 ms (5%↓) compared to the second-best approach and 43.23 ms (41%↓) compared to the baseline. Similar relative latency improvements can be seen on Swin (3%↓ 2nd best, 27%↓ baseline), DeiT-III (9%↓ 2nd best, 40%↓ baseline) and MambaVision (4%↓ 2nd best, 23%↓ baseline). DeiT is the only architecture, where our approach’s latency comes in second to ComCat, all be it at significantly higher accuracy (+30.6 p.p.). ComCat achieves this latency with a smart attention specific decomposition strategy, that - while creating impressive latency benefits on specific attention modules - is hard to or cannot at all be generalized to architectures such as ConvNeXt and is therefore absent for the other architectures.

Comparison on Edge Targets. To generalize our comparisons beyond server hardware this section evaluates the advancements on Jetson and Qualcomm targets for the DeiT model. Given that the two targets behave differently, and our search is hardware aware, we opt for two different vari-

Table 1. Accuracy and latency of various models on ImageNet-1K [6] with 224x224 resolution on Nvidia V100 without finetuning.

Model	Method	Params(M)	GFLOPs	Top-1 ↑	Latency ↓
ConvNext-Base	Base [17]	88.6	30.8	85.8	102.48
	PELA [9]	46.6	15.9	19.6	63.26
	FW-SVD [12]	47.4	15.8	52.4	65.03
	ASVD [28]	46.9	15.6	54.1	63.0
	SVD-LLM [24]	46.6	15.9	63.0	63.13
	ours	48.6	15.4	74.6	60.25
DeiT-Base	Base [21]	86.6	33.7	81.8	58.83
	PELA [9]	44.1	17.0	61.1	40.56
	FW-SVD [12]	43.9	16.9	67.8	40.81
	ASVD [28]	43.7	16.9	67.2	40.73
	SVD-LLM [24]	44.1	17.0	68.7	40.67
	ComCat [25]	41.2	8.17	48.3	33.69
	ours	49.2	19.0	78.9	40.38
Swin-Large	Base [16]	196.5	68.1	86.2	124.9
	PELA [9]	121.0	41.4	71.0	97.94
	FW-SVD [12]	120.9	39.8	78.2	94.99
	ASVD [28]	119.5	39.6	77.2	94.25
	SVD-LLM [24]	121.0	40.4	80.2	98.2
	ours	127.4	40.0	81.2	91.47
DeiT-III-Large	Base [22]	304.4	119.4	86.8	196.50
	PELA [9]	153.2	59.8	50.7	130.58
	FW-SVD [12]	154.1	60.2	75.0	135.07
	ASVD [28]	152.9	59.7	72.0	131.33
	SVD-LLM [24]	153.2	59.8	68.5	130.77
	ours	141.9	55.4	76.8	118.62
MambaVision	Base [10]	97.7	30.0	84.2	45.5
	PELA [9]	59.0	22.3	73.7	36.76
	FW-SVD [12]	56.9	22.3	71.3	36.92
	ASVD [28]	57.1	22.3	73.9	37.06
	SVD-LLM [24]	59.0	22.3	73.5	36.52
	ours	61.9	21.4	76.1	35.05

ants for the different targets for our method, while reusing the same one for both on the other methods. As can be seen in Table 2 our method - again - achieves superior accuracy compared to other approaches. Interestingly, for both targets, the majority of approaches struggle to replicate the latency improvements seen on the V100. Differently, the hardware awareness of our approach allows to adjust and still manage to improve latency by 21 % over the baseline, outperforming all other approaches. However, optimizing for Qualcomm is harder, resulting in rank search falling back to basic uniform rank choices. While our approach has the same latency as other SVD approaches as a result of this, it still manages to outperform them by 2.5 p.p. and reduce by 25 % with respect to the baseline. Overall, our method can deliver improvements on both server and edge hardware targets, beating all other approaches.

4.2.2. Search Time Evaluation

Our FLAR-SVD approach uses a fast binary search algorithm that leverages feature properties inherent to Cholesky-SVD. In this section we compare the speed of our search approach to other SVD based approaches, that perform rank search. Results are presented in Table 3. We did not add FLoRA [2] as it requires to run 150 epochs of preconditioning training before the search, putting it at more than

Table 2. Latency of our compressed models on Nvidia Jetson and Qualcomm Snapdragon at 224x224 resolution without finetuning.

Model	Method	Params(M)	GFLOPs	Top-1	Latency	
					Snapdragon INT8	Jetson FP16
DeiT-Base	Base [21]	86.6	33.7	81.8	8.0	14.6
	PELA [9]	44.1	17.0	61.1	6.0	22.7
	FW-SVD [12]	43.9	16.9	67.8	8.8	24.4
	ASVD [28]	43.7	16.9	67.2	9.0	24.5
	SVD-LLM [24]	44.1	17.0	68.7	6.0	22.7
	ours (Snapdragon)	44.1	17.0	71.2	6.0	-
	ours (Jetson)	49.2	19.0	78.9	-	11.4

100 hours of training time before search can even begin. Compared to the remaining approaches, our FLAR-SVD shows strong search performance. While it falls slightly behind ASVD in terms of search time, our search results significantly better, pulling ahead by 11.7 p.p.. ComCat, after 1 epoch of its rank optimization, is a distant second to both ASVD and our FLAR-SVD, while taking substantially more time, making it the least effective search here. Fast-LRD [18] results are taken from the paper with accuracy including the finetuning of at least 15 epochs [18], which is why we compared it to our finetuned checkpoint as well. Compared to Fast-LRD, our method has the same performance while scoring a noticeable search time reduction of around 50%. Overall, our method is the fastest in finding good rank - accuracy trade-offs.

Table 3. Rank search time comparison to SOTA approaches. $\dagger:X$ refers to X epochs of fine tuning after search. * values are taken from the respective paper.

Methods	Top-1	Latency [ms]	Search time [h]
ComCat [25]	48.3	33.69	5.0
ASVD [28]	67.2	40.73	1.1
ours	78.9	40.38	1.3
Fast-LRD $\dagger:15$ [18]	81.0*	-	4*
ours$\dagger:1$	81.0	40.38	2.3

4.2.3. Comparison to Structured Pruning

To further evaluate our approach, we compare it to state-of-the-art structured pruning algorithms NViT (CVPR’23) [27], Isomorphic Pruning (ECCV’24) [7] and DISP-LLM (NeurIPS’24) [8], that also improve latency on HW. To compensate for the long optimization done by NViT during pruning, we finetune the other approaches for few epochs, which takes around 1 hour on the compressed models. The results are summarized in Table 4. As can be seen there, in the faster category with around 1 h of time budget our method is able to maintain high accuracy even without any finetuning, solidifying a substantial lead of 27.0 p.p. and 2.7 p.p. over the similarly complex DISP-LLM and Isomorphic Pruning, respectively. Interestingly, even though Isomorphic Pruning benefits from its very fast pruning and can squeeze in a full epoch of fine-

tuning in the given time budget, our approach still comes out ahead, although with a smaller margin compared to DISP-LLM. In addition to the superior accuracy, our approach also manages to improve latency compared to the other two approaches, highlighting its suitability for fast deployment-oriented compression tool chains. In the more time intensive category NViT exhibits strong performance without fine tuning, which can be partially attributed to it starting from the better DeiT-B *distilled* model. Regardless, the long optimization time required puts it at a disadvantage to the other methods, so that it is falling behind them when given similar amounts of time. Our method - again - exhibits strong performance, outperforming all other approaches with only a single epoch of finetuning, resulting in the best accuracy - optimization time - latency trade-off.

Table 4. Comparison of our FLAR-SVD to structured pruning approaches. $\dagger:X$ refers to X epochs of fine tuning after search. * refers to the approach starting from the stronger DeiT-B distilled baseline (83.4% accuracy).

Model	Method	Params(M)	Time [h]	Top-1	Latency [ms]
DeiT-B	Base [21]	86.6	-	81.8	58.8
	DISP-LLM [8]	53.1	1.1	51.4	48.2
	Isomorphic $\dagger:1$ [7]	53.2	1.0	76.2	42.8
	ours	49.2	1.3	78.9	40.4
	DISP-LLM $\dagger:5$ [8]	53.1	6.1	79.3	48.2
	NViT* [27]	53.6	5.7	79.7	41.7
	Isomorphic $\dagger:5$ [7]	53.2	5.0	80.0	42.8
	ours$\dagger:1$	49.2	2.3	81.0	40.4

4.3. Ablations

4.3.1. Individual Method Contributions

To assess the impact of all of the components introduced with our FLAR-SVD, this section presents the incremental improvements in Table 5. Starting from the advanced Cholesky-SVD introduced in SVD-LLM [24], we first add the Ledoit-Wolf shrinkage to improve stability with fewer samples, thereby adding 3.1 p.p. accuracy. Next, our dynamic rank search further increases accuracy by another 5.9 p.p. at a small latency increase of 0.7 ms. Adding our latency rank tuning manages to improve accuracy by 1.2 p.p. while lowering the latency by 0.6 ms. Finally, minor finetuning of 1 epoch results in the final accuracy of 81.0 % at 40.4 ms latency, which is 30 % faster than the baseline with a only minor accuracy hit of 0.8 p.p..

4.3.2. Search Criteria

Properly evaluating and making rank choices for each layer is a challenging problem requiring well thought-out evaluation metrics. This section dives into different options for metrics and their effectiveness during search. As shown in 6, Cross-entropy (CE) loss is not effective for rank selection due to its fluctuating nature as the model evolves. In contrast, cosine similarity between original and reconstructed

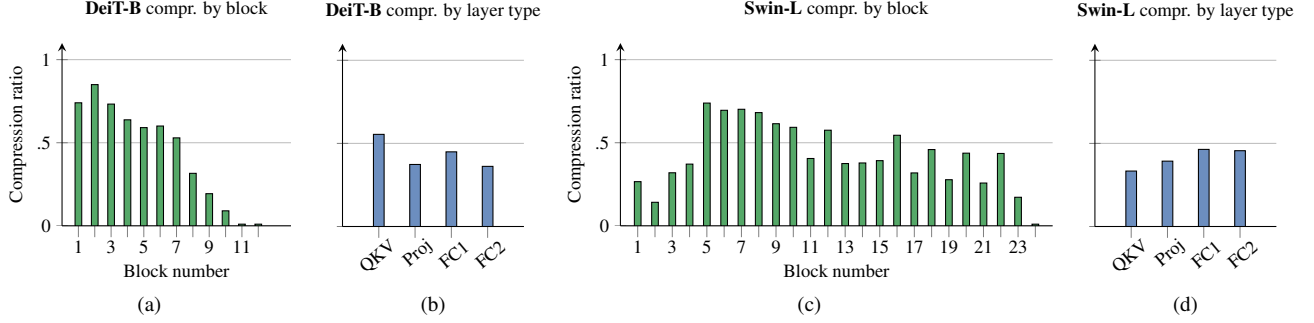


Figure 3. 3a and 3c show the averaged ratios to which different blocks were compressed with our method on DeiT-Base and Swin-Large, respectively. 3b and 3d show the averaged reduction contribution our method obtains in the subcomponents of both MHA and MLP across all blocks on DeiT-Base and Swin-Large, respectively.

Table 5. Components of our FLAR-SVD and their contribution to improved accuracy and latency.

Method contributions	Params (M)	GFLOPs	Top-1	Latency
Baseline: DeiT-Base	86.6	33.7	81.8%	58.8 ms
Cholesky-SVD (SVD-LLM [24])	44.1	17.0	68.7%	40.3 ms
+ Ledoit-Wolf shrinkage	44.1	17.0	71.8%	40.3 ms
+ dynamic rank search	43.5	16.8	77.7%	41.0 ms
+ latency rank tuning	49.2	19.0	78.9%	40.4 ms
+ 1 epoch finetuning	49.2	19.0	81.0%	40.4 ms

Table 6. Impact of applying different rank search criteria on parameter count, FLOPs and Top-1 accuracy.

Criteria	Params (M)	GFLOPs	Top-1
Baseline: DeiT-Base	86.6	33.7	81.8%
CE Loss	72.4	28.1	76.3%
$\cos_{sim}(A, A_{recon})$	46.5	17.9	78.2%
MSE(logits)	46.41	17.9	78.25%
MSE(feats _{head})	46.0	17.8	77.87%
MSE(feats _{block11.FC2})	49.2	19.0	78.9%

weights serves as a better proxy metric for rank selection. However, this metric overly focuses on the proximity to original parameters rather than the effect of deviations. To address this, we explored using mean squared error (MSE) between features and logits obtained in inference. This approach effectively compares different ranks across layers, particularly in later features. It highlights the cumulative effect this error introduces, ensuring all the contributions can be measured with the same scale and should not exceed a global threshold. Our method leverages the MSE between the last feature of the last MLP block and provides an accuracy 0.6% compared to using MSE of logits.

4.3.3. Compression Distribution

Our dynamic rank search allocates sparsity based on feature degradation, without constraints to any particular structure of the model. This section investigates the compression distribution for DeiT-Base and Swin-Large, two different architectures. We visualize the compression ratios of the dif-

ferent blocks and respective components in Figure 3. As can be seen, DeiT (Figure 3a and 3b) favors compressing early parts of the model, with very little sparsity allocated to later blocks. While Swin’s compression distribution (Figure 3c and 3d) is much more even, it still exhibits the same overall trend. Notably, and different to DeiT the initial layers of Swin receive much fewer compression. This is attributed to the first stages in Swin, which have a higher input-to-weight size ratio. Thus, as seen in 2c, having a disproportion between input and weights sizes can introduce overheads in inference even at low rank ratios. Overall, the flexible allocation without any manual tuning highlights the adaptability of our approach to a wide range of architectures.

5. Conclusion

In this work, we introduced FLAR-SVD, a SVD approach with dynamic, latency aware rank search that allocates sparsity based on feature degradation, offering significant compression without requiring any model-specific tuning. Through extensive experiments on diverse models like ConvNeXt, DeiT-Base and MambaVision, we demonstrated that FLAR-SVD can effectively adapt to different model structures, achieving impressive compression and maintaining high accuracy. Our approach also outperforms existing state-of-the-art SVD methods, yielding up to 11.6% accuracy improvement on server hardware while reducing latency by as much as 41% compared to the baseline. Notably, FLAR-SVD’s hardware-aware design also excels on edge platforms, where it adapts to the unique constraints of devices like Jetson Orin and Qualcomm Snapdragon 8 Gen 2, surpassing other methods in both accuracy and latency. Overall, FLAR-SVD represents a fast, versatile solution for compressing deep learning models across a range of architectures and hardware platforms. Future work will focus on further optimizations on more advanced latency estimation, pushing the boundaries of model compression and performance.

References

- [1] Seyedarmin Azizi, Mahdi Nazemi, and Massoud Pedram. Memory-efficient vision transformers: An activation-aware mixed-rank compression strategy, 2024. [2](#)
- [2] Chi-Chih Chang, Yuan-Yao Sung, Shixing Yu, Ning-Chi Huang, Diana Marculescu, and Kai-Chiang Wu. Flora: Fine-grained low-rank architecture search for vision transformer. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2470–2479, 2024. [2](#), [6](#)
- [3] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don’t walk: Chasing higher flops for faster neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023. [1](#)
- [4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. [1](#)
- [5] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. <https://github.com/open-mmlab/mms Segmentation>, 2020. [1](#)
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009. [5](#), [6](#)
- [7] Gongfan Fang, Xinyin Ma, Michael Bi Mi, and Xinchao Wang. Isomorphic pruning for vision models. In *Eur. Conf. Comput. Vis.*, pages 232–250. Springer, 2024. [1](#), [7](#)
- [8] Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Displlm: Dimension-independent structural pruning for large language models. *Adv. Neural Inform. Process. Syst.*, 37: 72219–72244, 2024. [1](#), [7](#)
- [9] Yangyang Guo, Guangzhi Wang, and Mohan Kankanhalli. Pela: Learning parameter-efficient models with low-rank approximation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024. [1](#), [2](#), [6](#), [7](#)
- [10] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*, 2024. [1](#), [6](#)
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. [1](#)
- [12] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *Int. Conf. Learn. Represent.*, 2022. [1](#), [2](#), [3](#), [6](#), [7](#)
- [13] Injoon Hwang, Haewon Park, Youngwan Lee, Jooyoung Yang, and SunJae Maeng. Pc-lora: Low-rank adaptation for progressive model compression with knowledge distillation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024. [2](#)
- [14] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *UPF economics and business working paper*, (691), 2003. [3](#)
- [15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision—ECCV 2014: 13th European conference, zurich, Switzerland, September 6–12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014. [1](#)
- [16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. [6](#), [1](#)
- [17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. [1](#), [6](#)
- [18] Yuan-June Luo, Yu-Shan Tai, Ming-Guang Lin, and An-Yeu Andy Wu. Similarity-aware fast low-rank decomposition framework for vision transformers. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2024. [1](#), [2](#), [7](#)
- [19] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In *Int. Conf. Learn. Represent.*, 2014. [2](#)
- [20] Siyu Ren and Kenny Q. Zhu. Low-rank prune-and-factorize for language model compression, 2023. [2](#)
- [21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10347–10357, 2021. [1](#), [6](#), [7](#)
- [22] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Eur. Conf. Comput. Vis.*, pages 516–533, Cham, 2022. Springer Nature Switzerland. [6](#)
- [23] Haoyu Wang and Wei-Qiang Zhang. Unstructured pruning and low rank factorisation of self-supervised pre-trained speech models. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–14, 2024. [2](#)
- [24] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *Int. Conf. Learn. Represent.*, 2025. [1](#), [2](#), [3](#), [6](#), [7](#), [8](#)
- [25] Jinqi Xiao, Miao Yin, Yu Gong, Xiao Zang, Jian Ren, and Bo Yuan. COMCAT: Towards efficient compression and customization of attention-based vision models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 38125–38136. PMLR, 2023. [1](#), [2](#), [5](#), [6](#), [7](#)
- [26] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. [1](#)
- [27] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 18547–18557, 2023. [7](#)

- [28] Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guanyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2024. [1](#), [2](#), [6](#), [7](#)
- [29] Stephen Zhang and Vardan Papyan. OATS: Outlier-aware pruning through sparse and low rank decomposition. In *Int. Conf. Learn. Represent.*, 2025. [2](#)
- [30] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017. [1](#)