Learning Fine-grained Parameter Sharing via Sparse Tensor Decomposition

Anonymous authors
Paper under double-blind review

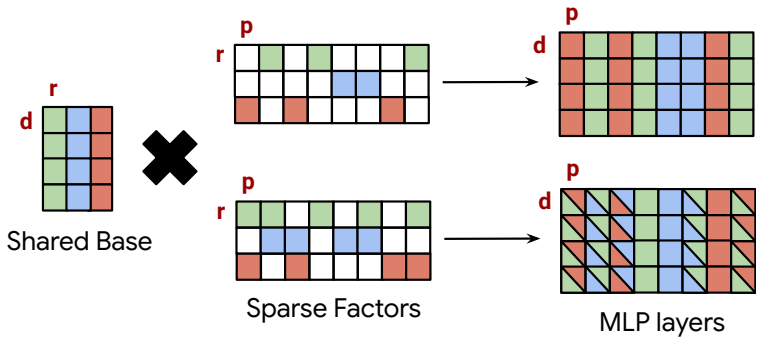


Figure 1: **Fi**ne-grained **P**arameter **S**haring (FiPS).

Abstract

Large neural networks achieve state-of-the-art performance on many tasks, yet their sheer size hinders deployment on resource-constrained devices. Among existing compression approaches, cross-layer parameter sharing remains relatively unexplored for transformer models. In this paper, we introduce Fine-grained Parameter Sharing (FiPS), a unified framework for compressing transformer Multi-Layer Perceptrons (MLPs) that combines cross-block parameter sharing, low-rank factorization, and sparsity in a single optimization. FiPS concatenates MLP weight matrices across a group of transformer blocks and factorizes them into a shared basis and sparse, layer-specific projection matrices. Both factors are initialized via singular value decomposition (SVD) and jointly optimized by block-wise reconstruction error minimization. FiPS compresses Vision Transformers (ViTs) by up to 33% with less than 1% top-1 accuracy loss on ImageNet-1k, and by up to 57% when combined with fine-tuning. It also compresses Large Language Models (LLMs) by up to 20% while outperforming existing SVD-based methods in perplexity and downstream benchmarks at matched compression. Combined with Quantization-Aware Training (QAT), 3-bit FiPS on Gemma-2-2B achieves lower perplexity than 2-bit QAT alone while matching the same 8× compression. These results establish fine-grained parameter sharing as a practical and effective approach for transformer MLP compression.

1 Introduction

Over the past decade, large neural networks have delivered impressive performance through scaling of datasets and model sizes. However, this trend has introduced substantial computational, memory, and storage burdens, highlighting the need for efficient model compression to reduce overhead and enable deployment on resource-constrained devices such as mobile phones and embedded systems. In response, researchers have explored various strategies, including tensor decomposition, quantization, distillation, sparsity, adaptive computing methods, and *parameter sharing* (Cheng et al., 2020). While most of these techniques are well-studied and widely adopted, parameter sharing in neural networks remains underexplored and has not yet been leveraged

successfully to compress Transformer models, despite its potential for significant parameter-count reduction. Transformer architectures are a particularly natural target for parameter sharing because they are composed of repeated homogeneous blocks whose Multi-Layer Perceptron (MLP) modules have identically shaped fully connected (FC) layers across depth, enabling direct concatenation and joint factorization of weights across blocks.

Sharing parameters across layers reduces memory usage and improves cache efficiency. Prior works have investigated reusing entire Transformer blocks (Lan et al., 2020; Takase & Kiyono, 2023; Lin et al., 2023), yielding more efficient models. Although directly sharing unmodified weights is promising, we hypothesize that a more fine-grained approach could yield superior compression.

We therefore focus on sharing neurons across layers by introducing a shared basis, with each neuron expressed as a linear combination of basis vectors via a projection matrix. We, then, find that enforcing sparsity in the projection matrix is essential for the effectiveness of this approach. This insight leads to our novel parameter sharing algorithm, **Fi**ne-grained **P**arameter **S**haring (FiPS), which, as we demonstrate, effectively compresses large Vision Transformers (ViTs) and Large Language Models (LLMs)[1]. Our contributions include:

- **Systematic Analysis of Cross-Block MLP Sharing.** We systematically explore strategies for sharing bases and neurons across transformer MLP modules, examining sharing granularity, concatenation schemes, and sparsity patterns. We identify when neuron sharing is most effective and how global versus local and structured versus unstructured sparsity interact with cross-block sharing.
- **FiPS Algorithm[2].** A shared basis and sparse layer-specific projection matrices are initialized via SVD and jointly optimized by block-wise reconstruction error minimization, followed by optional end-to-end fine-tuning (FT).
- **State-of-the-Art ViT & LLM Compression.** FiPS delivers substantial compression with minimal performance loss, surpassing recent baselines. It compresses DeiT-B and Swin-L by up to 33% with <1% top-1 accuracy drop across five vision benchmarks (up to 57% with fine-tuning), and compresses Llama-7B and Llama-3.1-8B by up to 20% while outperforming existing SVD-based methods on 10 NLP benchmarks.
- **Quantization-Aware Training (QAT).** We demonstrate that 3-bit QAT combined with FiPS compresses Gemma-2-2B effectively, achieving the same 8× compression as 2-bit quantization but with markedly better language modeling. This confirms that FiPS is orthogonal to QAT.

The remainder of this paper is organized as follows: Section 2 formalizes the integration of low-rank factorization and sparsity for parameter sharing; Section 3 details the FiPS algorithm; Section 4 presents our empirical findings; and Section 5 provides ablation studies on key design choices. Extensive supplementary material is provided in the Appendix, including additional experiments, algorithmic details, and hyperparameter sweeps.

2 Parameter Sharing Through Sparse Tensor Decomposition

Consider a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times p}$, where $p > d$, that projects feature vectors from a d-dimensional space to a p-dimensional space with neurons represented by the columns of $\mathbf{W}$. Our objective is to share weights among a subset of these p neurons, reducing the number of unique neurons to $r < p$. In other words, only r columns of $\mathbf{W}$ will contain unique values. These r unique neurons are represented using a shared basis $\mathbf{U} \in \mathbb{R}^{d \times r}$, which is orthogonal at initialization (via SVD) when $r \leq d$. The original matrix $\mathbf{W}$ is then reconstructed by mapping each of its p columns to an r-dimensional one-hot vector via a projection matrix $\mathbf{V} \in \mathbb{R}^{r \times p}$, i.e., $\mathbf{W} = \mathbf{UV}$. This "one-hot" approach is illustrated in the upper part of Figure 1. However, limiting the number of unique neurons to r constrains the representational capacity of $\mathbf{W}$. To address this limitation, we can increase the number of non-zero elements in $\mathbf{V}$, effectively creating linear combinations

[1]All model links are available in § A.3.
[2]Source code will be released upon acceptance.

of basis neurons and generating a significantly larger set of unique neuron representations, as shown in the lower part of Figure 1.

This approach readily extends from sharing neurons within a single weight matrix $\mathbf{W}$ to multiple weight matrices $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_N\}$ via concatenation (see Figure 6). The collection $\mathcal{W}$ is naturally a 3rd-order tensor in $\mathbb{R}^{d \times p \times N}$; our long-axis concatenation corresponds to its mode-2 unfolding, and the resulting factorization into a shared $\mathbf{U}$ and layer-specific sparse $\mathbf{V}_i$ is the order-2 (matrix) case of a Tucker-1 decomposition with sparse core slices—distinct from higher-order methods such as tensor train or CP decomposition. Specifically, fine-grained parameter sharing across multiple layers can be achieved by expanding the projection matrix $\mathbf{V}$ and the shared basis $\mathbf{U}$. Sharing neurons between layers in this manner may be viewed as a factorization of $\mathcal{W}$, where the first factor $\mathbf{U}$ is shared across N layers and the second, layer-specific factor $\mathbf{V}$ is sparse. Existing low-rank decomposition techniques can therefore be employed to obtain a shared basis—orthogonal at initialization via SVD, though not constrained to remain so during optimization—while sparsity in the projection matrices is induced using standard pruning and sparse training methods. When the parameter budget yields $r \leq d$, this is a genuine low-rank factorization of $\mathcal{W}$ with an orthogonal basis $\mathbf{U}$ at initialization; at higher budgets, r may exceed d, in which case $\mathbf{U}$ acts as an overcomplete shared dictionary that is no longer orthogonal, and parameter reduction arises from sharing and sparsity rather than rank reduction (discussed further in §3).

In the following sections, we investigate optimal layer-tying strategies within our framework, using a 12-block DeiT-B encoder with a single MLP module per block, pretrained on ImageNet-1k (Deng et al., 2009). Specifically, we focus on MLP modules, which account for the majority of parameters (e.g., 70.5% in Gemma-2-9B (Team et al., 2024)) and comprise two fully connected (FC) layers (i.e., FC-1 and FC-2 for DeiT-B) of dimensions $\mathbb{R}^{d \times p}$ and $\mathbb{R}^{p \times d}$ with $p = 4d$. The "parameter budget" denotes the fraction of nonzero parameters retained after truncated SVD and sparsification; e.g., 25% retains one-quarter of each MLP's weights. We measure the overall compression ratio as the percentage reduction in model size in bits, including sparsity metadata; e.g., 20% compression reduces storage by 20%. Concretely, the compressed model size (in bits) is

$$|\theta_{\text{non-MLP}}| \cdot b \;+\; G \cdot |\mathbf{U}| \cdot b \;+\; \textstyle\sum_i \big[(1-s) \cdot |\mathbf{V}_i| \cdot b \;+\; |\mathbf{V}_i|\big], \tag{1}$$

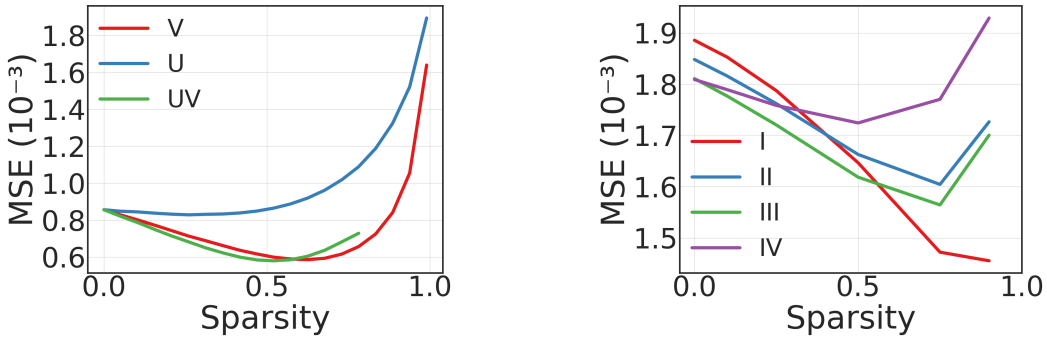
where G is the number of groups, s the sparsity level, b the bits per parameter, and the final $|\mathbf{V}_i|$ term accounts for the 1-bit mask per element in unstructured sparsity. The compression ratio is then 1 minus this quantity divided by the original model size $|\theta_{\text{original}}| \cdot b_{\text{original}}$. Note that non-MLP parameters (attention, embeddings, layer norms) remain uncompressed and are counted at full precision (full derivation in Appendix A.2). The rank r of the shared basis is not a free hyperparameter—given a parameter budget $b_{\%}$, the number of FC layers in the group N (number of blocks $\times$ FCs per block), and target sparsity s, it is uniquely determined by:

$$r = \left\lfloor \frac{b_{\%}\, N\, d\, p}{d + N(1-s)\, p} \right\rfloor. \tag{2}$$

Conceptually, FiPS differs from standard low-rank compression in three key aspects: (i) the basis is shared across layers rather than learned per layer; (ii) sparsity is imposed on the projection factors rather than the basis, enabling flexible neuron recombination; and (iii) optimization is performed jointly across layers using block-wise reconstruction loss. Together, these components transform low-rank approximation into a fine-grained parameter sharing mechanism.

2.1 Optimal Sparsity Allocation Across Factors

Before implementing parameter sharing via shared bases, we decompose individual FC layers of MLP modules using truncated SVD at a 25% parameter budget. Subsequently, sparsity is introduced by pruning low-magnitude values. Specifically, we examine sparsity induction in: (1) $\mathbf{U}$, (2) $\mathbf{V}$, and (3) both $\mathbf{U}$ and $\mathbf{V}$. Throughout this process, we vary the sparsity levels of the matrices while maintaining a constant total number of non-zero parameters. The resulting reconstruction errors are presented in Figure 2a. Our experiments show that the lowest reconstruction errors occur at sparsity levels between 60% and 80%, as confirmed by a sparsity sweep on ImageNet-1k with DeiT-B (Figure 4), particularly when sparsity is imposed on the



(a) FC-1

(b) Sharing Strategies

Figure 2: **Initial Experiments.** (a) Reconstruction error with varying levels of sparsity on different factors of the low-rank decomposition of FC-1 under 25% parameter budget. Results are analogous for FC-2, i.e., inducing sparsity on the larger factor yields a higher rank and, thus, lower reconstruction error. (b) Mean reconstruction error across four FCs of two distinct encoder blocks' MLPs under various parameter sharing schemes and sparsities. See § 2.2 for details.

larger factor matrix $\mathbf{V}$. We attribute this to the higher redundancy in larger matrices, which facilitates more efficient pruning.

2.2 Weight Concatenation and Sharing Dimensions

We investigate parameter sharing across multiple layers by analyzing four FC layers drawn from two distinct MLP modules. To align their dimensions, we transpose the second FC layer of each module, representing every layer as $\mathbf{W} \in \mathbb{R}^{d \times 4d}$. We then examine four concatenation strategies for constructing a shared weight block $\mathbf{W}_s$:

(I) **Long-axis concatenation.** All four weight matrices $\mathbf{W}$ are concatenated along their output (neuron) dimension, resulting in $\mathbf{W}_s \in \mathbb{R}^{d \times 16d}$.

(II) **Module-wise long, inter-module short.** Inside each MLP, its two FC layers are first concatenated along the output dimension, producing $\mathbf{W}_s \in \mathbb{R}^{2d \times 8d}$.

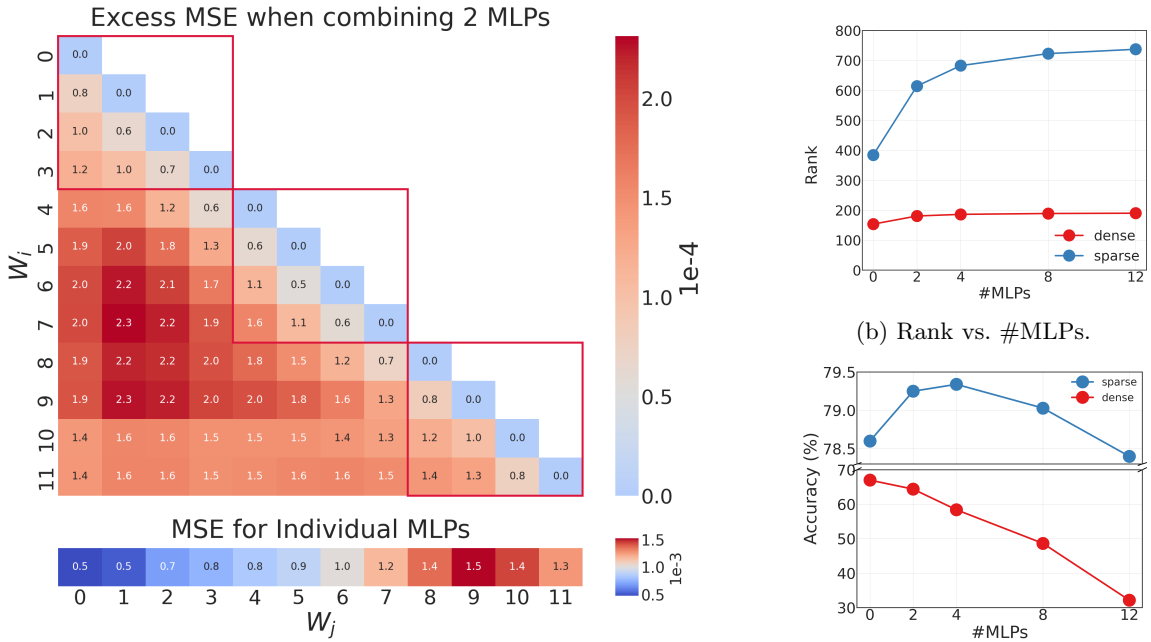
(III) **Module-wise short, inter-module long.** Inside each MLP, its two FC layers are first concatenated along the input dimension, again yielding $\mathbf{W}_s \in \mathbb{R}^{2d \times 8d}$.

(IV) **Short-axis concatenation.** All four $\mathbf{W}$ matrices are concatenated along their input (feature) dimension, aligning features rather than neurons and resulting in $\mathbf{W}_s \in \mathbb{R}^{4d \times 4d}$.

For each concatenated block $\mathbf{W}_s$, we perform truncated SVD to retain the top r singular vectors, followed by sparsification of the right singular matrix $\mathbf{V}$ via magnitude pruning (see § 2.1). Reconstruction is then obtained using the resulting shared basis, and mean squared error (MSE) is reported in Figure 2b. Empirically, concatenation along the longer dimension consistently achieves the lowest reconstruction error—particularly under high sparsity—motivating our choice of full long-axis concatenation throughout. Further implementation details are provided in § A.1.1.

2.3 Parameter Sharing Across Layers

This section examines redundancy and interdependencies among MLP modules to identify optimal parameter sharing groupings. We first decompose each module individually at rank $r = 180$ and plot the resulting mean squared error in the lower panel of Figure 3a. The error increases nearly monotonically with module depth, indicating that deeper layers require greater representational capacity. Next, we evaluate pairwise parameter sharing between modules i and j through a shared basis $\mathbf{U}$. Parameter sharing reduces the total number



(a) MSE when compressing MLP pairs.

(b) Rank vs. #MLPs.

(c) Accuracy vs. #MLPs.

Figure 3: **Parameter Sharing Groups.** (a top) MSE increases when sharing $\mathbf{U}$ across different MLP modules; red squares over the diagonal indicate that sharing adjacent modules enhances reconstruction. (a bottom) MSE for compressing individual MLP modules, showing that sharing $\mathbf{U}$ among consecutive layers typically results in the lowest error. (b) For a fixed parameter budget, the rank of the shared basis $\mathbf{U}$ stabilizes around four MLP modules, aligning with the optimal group size (c) for maximizing accuracy in DeiT-B, while the dense counterpart continuously worsens.

of unique parameters but increases the reconstruction error for each module. We denote the average error increase due to parameter sharing between modules i and j as $MSE^{\downarrow}_{i,j}$:

$$MSE^{\downarrow}_{i,j} = \frac{(MSE_{i,j} - MSE_i) + (MSE_{j,i} - MSE_j)}{2},$$

where $MSE_{i,j}$ denotes the error of module i when sharing a basis with module j. Figure 3a shows that adjacent modules exhibit the smallest error increase due to parameter sharing, motivating the practice of grouping consecutive layers.

We then explore the effect of grouping multiple MLP modules. Increasing group size allows a higher rank for the shared basis $\mathbf{U}$, as shown in Figure 3b. This benefit is most pronounced when the projection matrices $\mathbf{V}$ are sparsified. However, a higher rank does not always improve task performance, since $\mathbf{U}$ must capture a larger set of neurons. Figure 3c demonstrates that sharing across four consecutive MLP modules yields the highest post-compression accuracy.

Overall, our results reveal that (i) deeper modules require greater capacity when compressed in isolation—an effect we confirm with global pruning experiments in § 5; (ii) parameter sharing between adjacent layers curbs the rise in reconstruction error; and (iii) there exists an optimal group size that balances basis rank against sparsity. We encode this choice in the *grouping hyperparameter* $\beta = [\beta_1, \ldots, \beta_G]$, an ordered list where β_g gives the number of consecutive blocks whose MLP weights are tied in group g, with $\sum_{g=1}^{G} \beta_g = L$ (total blocks). For DeiT-B, $\beta = [4, 4, 4]$ achieves the best trade-off; for LLMs a tapered grouping with smaller groups at the network extremes works best (e.g., $\beta = [4, 6, 6, 6, 6, 4]$ for Llama-7B). A full sweep is reported in § A.8.1.

3 Fine-grained Parameter Sharing

The insights from § 2 motivate FiPS, an efficient cross-block parameter sharing algorithm grounded in sparse tensor decomposition. FiPS comprises three stages:

1. **Shared Initialization.** Tie the FC layers within each MLP group and apply truncated SVD to their concatenation (see Figure 6), yielding a shared basis $\mathbf{U}$ and projection matrices $\{\mathbf{V}_i\}$.
2. **Local Error Minimization.** Using a small calibration dataset D (§ 4.1), optimize $\mathbf{U}$ and each $\mathbf{V}_i$ to minimize the ℓ_2 discrepancy between original and compressed activations while enforcing target sparsity in $\mathbf{V}_i$.
3. **Global Error Minimization (Optional).** Fine-tune the compressed model end-to-end under a dynamic sparse training regime to recover performance at higher compression ratios.

Shared Initialization. We begin by compressing the pretrained model through parameter sharing, achieved by concatenating and decomposing multiple FC layers simultaneously. For higher parameter budgets and sparsity levels (e.g., 50% and 75%, respectively, for DeiT-B), the rank of the shared factor $\mathbf{U}$ can exceed the model dimension d. In this regime, $\mathbf{U}$ is no longer a low-rank basis but an overcomplete shared dictionary: parameter reduction comes from sharing $\mathbf{U}$ across blocks and sparsity in $\mathbf{V}$, not from rank reduction. The rank r is determined by Equation 2 and remains fixed throughout optimization. To initialize the $k = r - d$ additional dimensions, we compare three strategies: (1) random growth, initializing new neurons in $\mathbf{U}$ to zero and in $\mathbf{V}$ via He et al. (2015); (2) neuron splitting, duplicating and halving top neurons following Net2Net (Chen et al., 2016); and (3) hybrid initialization, setting new neurons in $\mathbf{U}$ to zero and deriving those in $\mathbf{V}$ from the top-k singular vectors scaled by $1/\tau$, where $\tau > 1$ dampens their initial contribution so they are learned gradually (Evci et al., 2022). After sweeping τ (see § A.1.2), hybrid initialization outperforms the alternatives by 1–2 percentage points in top-1 accuracy.

Formally, the parameters of a group of FC layers across MLP modules, $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N$, are concatenated into $\mathcal{W} = [\mathbf{W}_1; \mathbf{W}_2; \ldots; \mathbf{W}_N]$, where $\mathbf{W}_i \in \mathbb{R}^{d \times p}$.[3] We then apply truncated SVD, $\mathcal{W} = \mathbf{U}\mathbf{\Sigma}\hat{\mathbf{V}}$, to obtain a low-rank approximation, where $\mathbf{U} \in \mathbb{R}^{d \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and $\hat{\mathbf{V}} \in \mathbb{R}^{r \times (N \cdot p)}$. The factor $\mathbf{U}$ is shared among all layers within the group and remains dense due to its small size. We then multiply $\hat{\mathbf{V}}$ by the singular values to obtain the projection matrix $\mathbf{V} = \mathbf{\Sigma}\hat{\mathbf{V}}$. Finally, the weights are reconstructed as $\mathbf{W'}_i = \mathbf{U}\mathbf{V}_i$, where each $\mathbf{V}_i$ is the slice of $\mathbf{V}$ corresponding to weight matrix $\mathbf{W}_i$.

Local Error Minimization. In the second phase of FiPS, we compute the input and output activations of the original FC layers using a calibration dataset D (described in § 4.1). We use these activations to optimize the compressed layers by minimizing the ℓ_2-*loss* between the original and compressed activations:

$$\underset{\mathbf{U}, \mathbf{V_i}, \ldots, \mathbf{V_N}}{\operatorname{argmin}} \sum_{i}^{N} \|\mathbf{W}_i \mathbf{X}_i - \mathbf{U}\mathbf{V}_i \mathbf{X}_i\|_2^2, \tag{3}$$

where $\mathbf{X}_i$ is the inputs to the ith original FC layer. Sparsity on each $\mathbf{V}_i$ is enforced by the *Sparsify* step of Algorithm 1 rather than as an explicit constraint in the loss; $\mathbf{U}$ is left unconstrained, as orthogonality is only a byproduct of the SVD initialization and need not be preserved.

We explore several sparse training and pruning strategies to identify a sparse $\mathbf{V}$ during optimization: (a) *Static Sparsity*, which fixes the sparsity pattern by retaining the top-magnitude connections before training (Hoefler et al., 2021); (b) *Gradual Magnitude Pruning (GMP)* (Zhu & Gupta, 2017), which progressively increases sparsity by updating the mask every T steps according to the cubic schedule of Kurtic et al. (2023); and (c) *RigL* (Evci et al., 2021), which initializes as in (a) but updates sparse connectivity every ΔT steps using both gradient and magnitude information. We adopt *GMP* as our final strategy due to its superior performance, achieving up to 4% higher top-1 accuracy on ImageNet-1k compared to the closest baseline. A detailed comparison of sparsification methods is given in Table 5.

[3]The second FC layer is transposed to match the dimensions of the first.

Algorithm 1 **Fi**ne-grained **P**arameter **S**haring

Require: MLP parameters $\mathbf{W}_1, \cdots, \mathbf{W}_N \in \mathbb{R}^{d\times p}$, MLP inputs $\mathbf{A}_i$ and MLP function $\mathbf{f}(\mathbf{W}_i,\ \mathbf{A}_i)$, Target rank r, Learning Rate η, Steps $\mathbf{T}$.

$\mathbf{U},\ [\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_N] \leftarrow \mathit{TruncatedSVD}\left([\mathbf{W}_1; \mathbf{W}_2; \cdots; \mathbf{W}_N],\ k = r\right)$
for each training iteration $t = 1$ to T **do**
  $\mathbf{G}_\mathbf{U} = 0$ ▷ Gradient accumulator for $\mathbf{U}$
  for each block i **do**
    $\mathbf{V}_i \leftarrow \mathit{Sparsify}(\mathbf{V}_i,\ t)$ ▷ Potentially increase or adjust sparsity
    $L_i \leftarrow \mathit{MSE_loss}(\mathbf{f}(\mathbf{W}_i,\ \mathbf{A}_i),\quad \mathbf{f}(\mathbf{U}\,\mathbf{V_i},\ \mathbf{A}_i))$
    $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \nabla_{\mathbf{V}_i} L_i$
    $\mathbf{G}_\mathbf{U} \leftarrow \mathbf{G}_\mathbf{U} + \nabla_\mathbf{U} L_i$
  end for
  $\mathbf{U} \leftarrow \mathbf{U} - \frac{\eta}{N}\mathbf{G}_\mathbf{U}$
end for
return $\mathbf{U},\ [\mathbf{V}_1, \ldots, \mathbf{V}_N]$

During this stage, parameters are shared across multiple MLP modules, allowing gradients to be computed one module at a time. As a result, optimization is significantly more resource-efficient than end-to-end fine-tuning.

Global Error Minimization. In this optional stage, we fine-tune the learned parameter sharing scheme end-to-end to further improve performance and leverage the masks learned via GMP. Because the factors $\mathbf{V}_i$ are sparse, we employ the dynamic sparse training method *RigL* during this stage, as it performs slightly better than *Static Sparsity* (see § 4.1).

4 Main Results

Evaluation Metrics. For ViTs we report top-1 classification accuracy on ImageNet-1k validation. For LLMs we report perplexity (PPL), defined as $\text{PPL} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T} \log p(x_t \mid x_{<t})\right)$, on WikiText-2 and C4, as well as downstream task accuracy via LM-Evaluation-Harness (Sutawika et al., 2023). During compression, reconstruction quality is measured by mean squared error (MSE) between original and compressed activations, i.e., the per-layer ℓ_2 loss in Equation (3). Lower PPL and MSE indicate better quality; higher accuracy is better.

4.1 Vision Transformers

Experimental Setup. We evaluate FiPS on DeiT-B (Touvron et al., 2021) and Swin-L (Liu et al., 2021). Each model is calibrated on 2,560 ImageNet-1k images for 20 epochs, sufficient for convergence; calibration completes in under one hour on a single NVIDIA A6000—comparable to or cheaper than existing post-training compression methods, none of which report full training cost breakdowns. For parameter sharing, we group every four MLP modules in DeiT-B. In Swin-L, which consists of four stages with 2, 2, 18, and 2 encoder blocks, we share across entire stages for the three smaller ones and use groups of six blocks in the larger stage. Additional hyperparameters are provided in § A.8; sensitivity analyses are in Figure 4.

ImageNet-1k. We compare FiPS to the two closest ViT compression baselines that share its post-training, factorization-based setting without distillation or LoRA: Adaptive Atomic Feature Mimicking (AAFM), which compresses output activations rather than weights, and Global Feature Mimicking (GFM), which fine-tunes the compressed network (Yu & Wu, 2023). Both FiPS and FiPS+FT match AAFM and GFM in compute and memory budgets. At a 40% parameter budget, FiPS outperforms AAFM by 1.36 points and GFM by 0.41 points despite GFM's higher cost (Table 1). This pattern holds across all budgets: FiPS consistently achieves the highest accuracy while matching or improving upon the compute and memory budgets of AAFM and GFM. AAFM requires no fine-tuning; GFM requires full end-to-end fine-tuning; FiPS's local error

Table 1: **ViT Compression Results.** ImageNet-1k top-1 validation accuracy of DeiT-B (81.85%) (Touvron et al., 2021) and Swin-L (86.24%) (Liu et al., 2021) compressed using FiPS and AAFM/GFM across MLP parameter budgets, comparing layer-wise (FiPS) and global error minimization (FiPS+FT). Parameter budgets refer to the fraction of MLP parameters retained (see §2); the corresponding whole-model compression ratios (via Equation 1) are shown in the second row. AAFM/GFM† results are from Yu & Wu (2023); – denotes unreported metrics.

Parameter Budget (Comp. Ratio)	10% (∼57%)		25% (∼45%)		40% (∼33%)		50% (∼25%)		75% (∼6%)	
Method / Model	DeiT	Swin	DeiT	Swin	DeiT	Swin	DeiT	Swin	DeiT	Swin
AAFM †	–	–	–	–	80.33	–	81.21	85.04	81.76	85.94
GFM †	–	–	–	–	81.28	–	81.62	85.44	81.83	86.01
FiPS (ours)	70.04	74.04	80.64	84.78	**81.69**	**85.69**	**81.83**	**85.99**	**81.82**	86.21
FiPS + FT (ours)	**77.26**	**82.13**	**81.31**	**85.16**	81.54	85.68	81.54	**85.99**	**81.82**	**86.22**

Table 2: (a) Top-1 accuracy of DeiT-B and compressed variants using GFM and FiPS under different parameter budgets. GFM† and Original† results are from Yu & Wu (2023) and Touvron et al. (2021). (b) ImageNet-1k top-1 accuracy of DeiT-B with FiPS at 2:4 structured sparsity (N:M GMP (Lee et al., 2023)) versus 50% unstructured sparsity.

(a)

	Original†	GFM†		FiPS+RigL FT (ours)		
P. Budget	100	40%	50%	25%	40%	50%
(Comp. Ratio)	–	(∼33%)	(∼25%)	(∼45%)	(∼33%)	(∼25%)
CIFAR-100	90.99	90.17	90.67	90.88	91.24	**91.33**
Pets	**94.74**	93.95	94.22	94.19	94.52	94.41
Flowers102	97.77	97.02	97.45	97.84	98.14	**98.37**
iNaturalist 2019	77.39	77.13	77.56	77.26	77.58	**77.69**

(b)

P. Budget	10%	25%	40%	50%	75%
(Comp. Ratio)	(∼57%)	(∼45%)	(∼33%)	(∼25%)	(∼6%)
2:4 FiPS	52.36	76.88	80.59	81.31	81.51
FiPS	54.00	77.56	80.94	81.63	81.77

minimization is strictly cheaper than GFM and comparable to AAFM. Notably, a 10% parameter budget corresponds to roughly a 50% compression ratio, where FiPS+FT yields particularly strong gains.

Transfer Learning. For transfer learning, we fine-tune for 100 epochs on CIFAR-100, Flowers102, Oxford-III-Pets, and iNaturalist 2019 (Krizhevsky, 2009; Nilsback & Zisserman, 2008; Parkhi et al., 2012; Van Horn et al., 2018), following Yu & Wu (2023). We use *AdamW* (Loshchilov & Hutter, 2019) with learning rates selected from 12 log-spaced values. Models compressed with FiPS transfer significantly better, as shown in Table 2a.

Latency and Memory Profiling. Structured sparsity patterns enable efficient hardware implementations with minimal quality impact, as demonstrated in Table 2b using *NMGMP*. With 2:4 structured FiPS, the accuracy degradation remains just above 1% at 10% and 25% MLP parameter budgets; for all other settings, the impact is negligible. We further evaluate FiPS with alternative structured sparsity pruners—*STE*, *SR-STE* (Zhou et al., 2021), and *NMSRigL* (Lee et al., 2023; Lasby et al., 2024)—in Table 7. Latency and memory profiling with the optimal *NMGMP*+FiPS setup leverages NVIDIA's tensor core support for 2:4 sparsity (Mishra et al., 2021) on GPUs and Neural Magic's DeepSparse Engine (Neural Magic, 2021) on CPUs, as shown in Figure 7 and detailed in § A.9.

These results demonstrate that FiPS with structured sparsity translates compression into measurable inference gains on ViTs. Specifically, with a batch size of 64, FiPS with 2:4 sparsity at a 22.14% parameter budget yields a 1.31× speedup on the NVIDIA A4000 and reduces peak VRAM allocation to approximately 0.79× of the original requirement during DeiT-B inference. These gains generalize architecturally: FiPS replaces every

dense FC layer $\mathbf{W} \in \mathbb{R}^{d\times p}$ with the product $\mathbf{UV}$, changing the forward pass from a single dense matrix–vector product ($2dp$ FLOPs per token) to a sparse product followed by a dense one ($2rp(1-s) + 2dr$ FLOPs per token, where s is the sparsity level). This FLOP reduction depends only on the factorization parameters (r, s, d, p), not on whether the FC layer resides in a ViT encoder or an LLM decoder. Consequently, the latency and memory improvements measured on DeiT-B are expected to transfer to LLM architectures at matching compression configurations, subject to hardware-specific kernel availability for the relevant matrix dimensions.

4.2 Large Language Models

Experimental Setup. We evaluate FiPS on three publicly available pretrained LLMs: Llama-7B (Touvron et al., 2023), Llama-3.1-8B (Grattafiori et al., 2024), and the instruction-tuned Gemma-2-2B (Team et al., 2024). Gemma-2-2B is included solely to demonstrate that FiPS is orthogonal to QAT, while the other models are compared against the baselines described below. Unlike ViTs, these LLMs feature three FC layers per MLP in their decoder blocks. For parameter sharing, all FC layers from MLPs within the same group are concatenated. The number of MLPs per group is treated as a hyperparameter and detailed in § A.1.1. Calibration and optimization follow the ViT protocol (described in § 4.1): activations are collected from 8,192 × 20 SlimPajamas (Soboleva et al., 2023) tokens, and block-wise error minimization is run for 40 epochs—without global error minimization (i.e., no full FT)—on a single NVIDIA A100 (80GB), completing within 10 hours per model on a single GPU. To our knowledge, none of the compared baselines report end-to-end compression time; FiPS is the only method in this comparison that provides explicit cost figures.

Baselines. We restrict comparisons to methods that share FiPS's regime: post-training, factorization-based MLP compression without LoRA or distillation, so that differences reflect the factorization strategy rather than auxiliary training signals. We benchmark against three SVD-based LLM compression methods: ASVD (Yuan et al., 2023), which scales weights by activation statistics to mitigate outliers; SVD-LLM (Wang et al., 2024b), which employs truncation-aware whitening; and SVD-LLM V2 (Wang et al., 2025), which adds layer-wise rank allocation. All baseline metrics are from the original publications; we exclude LoRA (Hu et al., 2021) enhancements for fair comparison. Moreover, we compare FiPS against Basis Sharing (Wang et al., 2024a), which also pursues cross-layer parameter sharing but relies on dense coefficients. In contrast, FiPS enforces sparsity, which our ablations show to be essential (see Table 5).

Evaluation. We report perplexity on WikiText-2 (Merity et al., 2016) and C4 (Dodge et al., 2021), six classification benchmarks, and two generation tasks (TruthfulQA, GSM8K) via LM-Evaluation-Harness (Sutawika et al., 2023). At 20% compression (Table 3), FiPS achieves the lowest perplexity on both WikiText-2 and C4 for Llama-7B and Llama-3.1-8B, while matching or outperforming all baselines on downstream tasks. Results at 40% compression are provided in Table 6.

Quantization-Aware Training (QAT). To assess FiPS under low-precision regimes, we apply QAT to Gemma-2-2B. Table 4 reports WikiText-2 perplexity for 4-bit and 2-bit QAT (4× and 8× compression) as well as FiPS combined with 3-bit QAT. While 4-bit QAT matches `bfloat16` performance, 2-bit QAT degrades severely (PPL = 41.86). Combining 3-bit QAT with FiPS achieves the same 8× compression as 2-bit QAT but with substantially lower perplexity (35.43 vs. 41.86). Both approaches degrade substantially from the `bfloat16` baseline; the key finding is that FiPS recovers quality relative to aggressive quantization at the same compression factor.

Variant	Prec.	Comp.	PPL ↓
Baseline	BF16	1.0×	15.61
QAT	INT4	4.0×	16.86
QAT	INT2	8.0×	41.86
FiPS	BF16	1.5×	32.01
FiPS+QAT	INT3	8.0×	35.43

Table 4: **QAT Results.** 3-bit QAT with FiPS achieves 8× compression at lower PPL than 2-bit QAT alone.

5 Ablations

We examine the importance of various components of the FiPS algorithm when compressing DeiT-B at 25% parameter budget. We ablate the following key components:

	Method	WikiText-2↓	C4↓	Openb.	ARC-e	WinoG.	HellaS.	PIQA	MathQA	**Avg.↑**	TruthfulQA↑	GSM8K↑
LLAMA-7B	Original	5.68	7.34	0.34	0.75	0.70	0.57	0.79	0.27	0.57	0.30	0.09
	ASVD	11.14	15.93	0.29	0.53	0.64	0.41	0.68	0.17	0.45	0.21	0.04
	SVD-LLM	7.94	15.84	0.31	0.71	0.68	0.49	0.71	0.22	0.52	0.24	0.06
	SVD-LLM V2	7.12	10.47	0.32	0.72	0.70	0.52	0.75	0.24	0.54	**0.27**	**0.07**
	Basis Sharing	7.74	15.03	0.28	0.66	0.66	0.46	0.71	0.25	0.50	–	–
	FiPS (ours)	**6.06**	**8.10**	**0.32**	**0.72**	**0.70**	**0.56**	**0.78**	**0.26**	**0.56**	**0.27**	**0.07**
LLAMA-3.1-8B	Original	6.14	9.47	0.35	0.80	0.73	0.60	0.80	0.40	0.61	0.49	0.45
	ASVD	17.55	28.41	0.20	0.59	0.61	0.41	0.69	0.30	0.47	0.37	0.28
	SVD-LLM	11.82	20.05	0.29	0.77	0.64	0.51	0.72	0.30	0.54	0.45	0.31
	SVD-LLM V2	8.01	11.72	**0.33**	**0.79**	0.70	0.58	0.77	0.36	0.59	**0.46**	0.40
	FiPS (ours)	**6.88**	**10.78**	**0.33**	**0.79**	**0.72**	**0.59**	**0.78**	**0.38**	**0.60**	**0.46**	**0.42**

Table 3: **LLM Compression Results.** LLAMA-7B and LLAMA-3.1-8B at 20% compression. PPL↓: perplexity on WikiText-2 and C4; Avg.↑: mean over six classification tasks; TruthfulQA (BLEU↑) and GSM8K (exact match↑). ASVD from Yuan et al. (2023); SVD-LLM from Wang et al. (2024b); SVD-LLM V2 from Wang et al. (2025); Basis Sharing from Wang et al. (2024a). All results follow Sutawika et al. (2023); – indicates unreported metrics.

Parameter Budget (Comp. Ratio)	10% (∼57%)		25% (∼45%)		40% (∼33%)		50% (∼25%)		75% (∼6%)	
Method / Model	DEIT	SWIN	DEIT	SWIN	DEIT	SWIN	DEIT	SWIN	DEIT	SWIN
Dense	15.35	3.61	65.71	60.31	74.33	80.61	79.22	83.59	81.36	85.64
Static Sparsity	65.26	65.6	80.06	84.37	81.48	**85.69**	81.70	85.98	**81.86**	**86.23**
RigL	66.67	70.96	80.31	84.57	81.50	85.59	81.65	85.91	81.82	86.20
GMP (FiPS)	**70.04**	**74.04**	**80.64**	**84.78**	**81.69**	**85.69**	**81.83**	**85.99**	81.82	86.21

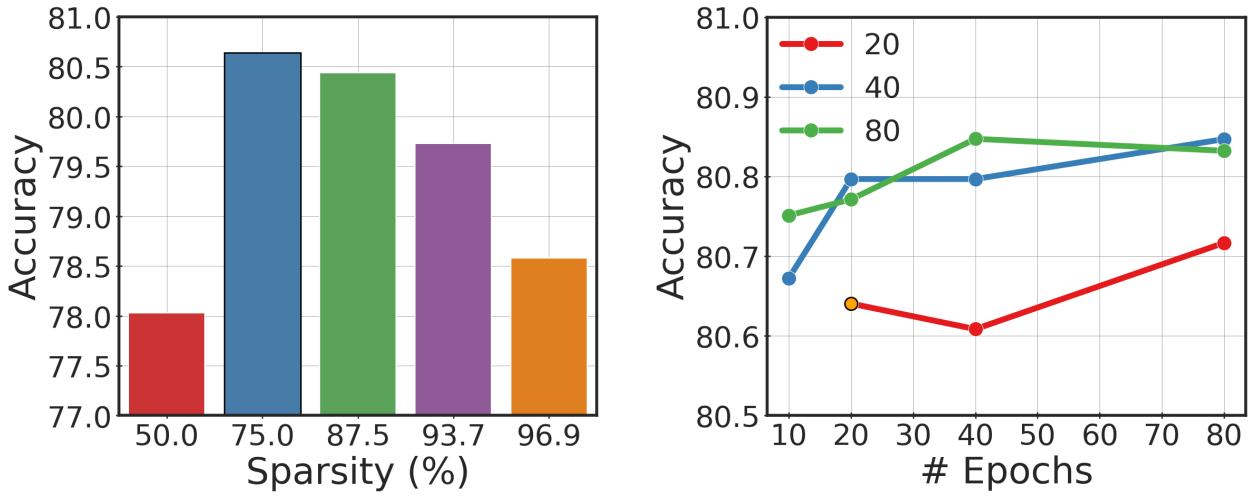
Table 5: **Sparsification Method Comparison.** ImageNet top-1 validation accuracy (%) of DEIT-B (81.85%) (Touvron et al., 2021) and SWIN-L (86.24%) (Liu et al., 2021) compressed with FiPS using different sparsity methods. GMP consistently outperforms alternatives; the dense baseline collapses at low parameter budgets.

1. **Random Initialization (RI):** Using RI instead of SVD initialization results in a 1 percentage point drop in accuracy.
2. **Global Pruning (GP):** Using GP when sparsifying the factors $\mathbf{V}$ yields a 0.4 percentage point improvement over local pruning (LP), which enforces the same sparsity level for each group.
3. **Scaling Vectors (SV):** Following Liu et al. (2024), FC weights are normalized, and the magnitudes initialize the SV for neuron scaling. This enhances LP but is less effective than GP.

An analysis on sparsity, methods, and calibration settings using DEIT-B confirms that GMP with 75% sparsity and 20 epochs over 20 batches yields optimal performance (Figure 4).

Figure 2a shows the rank–sparsity trade-off at a fixed 25% parameter budget: as sparsity on $\mathbf{V}$ increases from 0% to ∼80%, the rank r grows to compensate, and reconstruction error decreases until diminishing returns set in beyond 80% sparsity. Table 5 compares all sparsification strategies across budgets for both DEIT-B and SWIN-L: GMP consistently achieves the highest accuracy, while the dense baseline (no sparsity on $\mathbf{V}$) collapses at low budgets (e.g., 15.35% at 10%), underscoring that sparsity is not merely helpful but essential. Figure 4 further shows that 75% sparsity is optimal and that performance is robust to calibration data volume beyond 20 batches.

Sparsity Distribution and MSE Loss. Initial experiments in Figure 3a reveal that later layers exhibit higher reconstruction error under uniform compression budgets, suggesting these layers benefit from greater



(a) Optimal Sparsity.

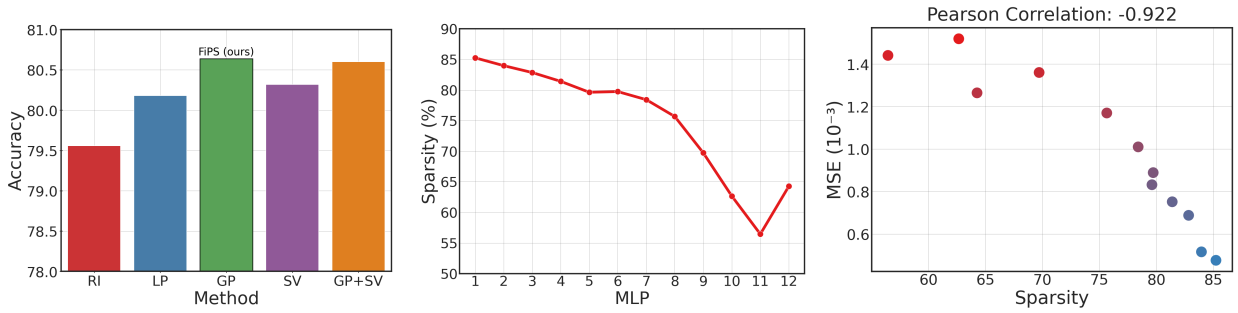
(b) Calibration Data.

Figure 4: **Sensitivity Analysis.** (a) Impact of sparsity levels on DeiT-B accuracy; 75% is optimal. (b) Effect of calibration data volume and training duration; 20 epochs over 20 batches suffices.

parameter allocation. FiPS addresses this by applying global magnitude pruning to its sparse factors, automatically directing parameters where they are most needed. As shown in Figure 5b, FiPS indeed assigns more parameters to later layers. Furthermore, Figure 5c demonstrates a strong negative correlation (-0.922) between the final sparsity pattern and the MSE losses observed in Figure 3a, confirming the effectiveness of this adaptive allocation.

6 Related Work

Vision Transformers (ViT) & Large Language Models (LLM). Recent transformer architectures extend beyond the foundational ViT models (Dosovitskiy et al., 2021), which treat image patches as token sequences. DeiT (Touvron et al., 2021) enhances data efficiency with distillation tokens, while Swin (Liu et al., 2021) introduces a hierarchical design using shifted windows. Both vision models employ two FC layers per MLP module. In contrast, decoder-only LLMs such as Llama (Touvron et al., 2023), Llama-3 (Grattafiori et al., 2024), and Gemma-2 (Team et al., 2024) achieve state-of-the-art zero-shot and instruction-following performance using three FC layers per MLP module in each block.



(a) (b) (c)

Figure 5: **DeiT-B Ablation and Global Sparsity Analysis.** (a) Component analysis of the FiPS algorithm: Random Initialization (RI), Local Pruning (LP), Global Pruning (GP), and Scaling Vectors (SV). (b) End-of-training sparsity allocation; later layers require more parameters. (c) Strong correlation between the MSE reported in Figure 3a and the parameter distribution captured by FiPS.

Sparsity in Neural Networks. Early methods involved heuristic pruning, such as removing the smallest-magnitude parameters (Thimm & Fiesler, 1995). Later approaches, such as GMP (Zhu & Gupta, 2017), increased the extent of pruning, while dynamic pruning with accelerated schedulers was explored by Kurtic et al. (2023). Static sparsity uses a preinitialized mask throughout training (Hoefler et al., 2021), whereas dynamic methods such as RigL (Evci et al., 2021) adjust the sparsity pattern during training based on gradient information.

Tensor Decomposition. Yu & Wu (2023) introduce Adaptive Atomic Feature Mimicking (AAFM) and its global variant Global Feature Mimicking (GFM), which apply truncated PCA on ViT activations followed by fine-tuning. In LLMs, Activation-aware SVD (ASVD) (Yuan et al., 2023), truncation-aware SVD-LLM (Wang et al., 2024b), and SVD-LLM V2 with rank distillation and layer-wise allocation (Wang et al., 2025) serve as our baselines.

Parameter Sharing. Several prior works explore parameter sharing in neural networks. Eban et al. (2019) introduce a Sum-Product reducer to map shared parameters, and Obukhov et al. (2021) employ TR decomposition for shared parameters in 3D tensors. At the block level, ALBERT (Lan et al., 2020) shares entire transformer blocks (all parameters across layers), reducing model size but constraining every layer to identical weights. Zhang et al. (2022) propose "Weight Multiplexing," sharing parameters between MLP modules in ViTs with distillation and learned linear projections between blocks to aid recovery; however, MiniViT does not impose sparsity on its projection factors and relies on distillation for quality. Concurrent work by Wang et al. (2024a) also explores cross-layer parameter sharing via SVD, representing weight matrices as linear combinations of shared basis vectors with dense, layer-specific coefficients.

In summary, FiPS is distinguished from all the above compression approaches by combining three properties that no prior method jointly offers: (i) cross-block weight sharing via a shared basis $\mathbf{U}$, (ii) low-rank factorization initialized by SVD, and (iii) sparsity on the projection matrices $\mathbf{V}_i$, optimized via block-wise reconstruction loss. In particular, we show in § 2.1 that enforcing sparsity in $\mathbf{V}$ is critical for achieving lower reconstruction error at the same parameter budget compared to dense coefficients (Wang et al., 2024a). Methods such as pruning combined with distillation (Kurtic et al., 2023) and LoRA-based compression (Hu et al., 2021) operate in a complementary regime—they rely on auxiliary supervision signals or learned adapters—and are orthogonal to FiPS, as we demonstrate with QAT in Table 4.

7 Conclusion

We presented FiPS, a framework for compressing transformer MLPs via fine-grained inter-layer parameter sharing that unifies cross-block weight tying, low-rank factorization, and sparsity in a single optimization. FiPS achieves state-of-the-art compression–accuracy trade-offs on MLP layers: up to 33% compression on ViTs with <1% accuracy loss (up to 57% with fine-tuning), up to 20% on LLMs while outperforming existing SVD-based methods at matched compression, and 8× on Gemma-2-2B when combined with QAT. These findings establish parameter sharing as a competitive alternative to existing compression strategies. Future work includes extending to attention layers—whose projection matrices share the same block-repeated structure that FiPS exploits—quantizing the shared bases, and developing specialized kernels that keep $\mathbf{U}$ resident in fast memory for efficient on-device inference.

Broader Impact Statement

This paper advances Machine Learning by introducing Fine-grained Parameter Sharing (FiPS), a model compression method that improves the efficiency of Vision Transformers (ViTs) and Large Language Models (LLMs). By leveraging parameter sharing, low-rank factorization, and sparsity, FiPS reduces computational and memory costs, enhancing AI accessibility on resource-constrained devices. While model compression promotes efficiency and sustainability, it may also enable broader AI deployment in sensitive domains with ethical concerns such as bias, misinformation, and privacy. Nevertheless, this work should not introduce new risks beyond those inherent in deep learning, but we encourage responsible deployment and ethical considerations in practice.

References

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer, 2016. URL https://arxiv.org/abs/1511.05641.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks, 2020. URL https://arxiv.org/abs/1710.09282.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL https://arxiv.org/abs/2104.08758.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Elad Eban, Yair Movshovitz-Attias, Hao Wu, Mark Sandler, Andrew Poon, Yerlan Idelbayev, and Miguel A. Carreira-Perpinan. Structured multi-hashing for model compression, 2019. URL https://arxiv.org/abs/1911.11177.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. URL https://arxiv.org/abs/1911.11134.

Utku Evci, Bart van Merriënboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information, 2022. URL https://arxiv.org/abs/2201.05125.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro,

Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu,

Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL https://arxiv.org/abs/1502.01852.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021. URL https://arxiv.org/abs/2102.00554.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009.

Eldar Kurtic, Torsten Hoefler, and Dan Alistarh. How to prune your language model: Recovering accuracy on the "sparsity may cry" benchmark, 2023. URL https://arxiv.org/abs/2312.13547.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. URL https://arxiv.org/abs/1909.11942.

Mike Lasby, Anna Golubeva, Utku Evci, Mihai Nica, and Yani Ioannou. Dynamic sparse training with structured sparsity. In *International Conference on Learning Representations (ICLR)*, 2024.

Joo Lee, Wonpyo Park, Nicole Mitchell, Jonathan Pilault, Johan Obando-Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, Shivani Agrawal, Suvinay Subramanian, Xin Wang, Sheng-Chun Kao, Xingyao Zhang, Trevor Gale, Aart Bik, Woohyun Han, Milen Ferev, and Utku Evci. Jaxpruner: A concise library for sparsity research, 04 2023.

Ye Lin, Mingxuan Wang, Zhexi Zhang, Xiaohui Wang, Tong Xiao, and Jingbo Zhu. Understanding parameter sharing in transformers, 2023. URL https://arxiv.org/abs/2306.09380.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024. URL https://arxiv.org/abs/2402.09353.

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, 2021. URL https://arxiv.org/abs/2103.14030.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating Sparse Deep Neural Networks, April 2021. URL http://arxiv.org/abs/2104.08378. arXiv:2104.08378 [cs].

Neural Magic. Deepsparse engine: Sparsity-aware deep learning inference runtime for CPUs, 2021. URL https://github.com/neuralmagic/deepsparse.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 722–729. IEEE, 2008.

Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, Menelaos Kanakis, Dengxin Dai, and Luc Van Gool. T-basis: a compact representation for neural networks, 2021. URL https://arxiv.org/abs/2007.06631.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. IEEE Conference on Computer Vision and Pattern Recognition, 2012. The Oxford-IIIT Pet Dataset.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 6 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Lintang Sutawika, Leo Gao, Hailey Schoelkopf, Stella Biderman, Jonathan Tow, Baber Abbasi, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Chris, Julen Etxaniz, Zdeněk Kasner, Khalid, Jeffrey Hsu, AndyZwei, Pawan Sasanka Ammanamanchi, Dirk Groeneveld, Ethan Smith, and Eric Tang. Eleutherai/lm-evaluation-harness: Major refactor, December 2023. URL https://doi.org/10.5281/zenodo.10256836.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers, 2023. URL https://arxiv.org/abs/2104.06022.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Georg Thimm and Emile Fiesler. Evaluating pruning methods. In *International Symosium on Artifical Neural Networks*, 1995. URL https://api.semanticscholar.org/CorpusID:11075297.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. URL https://arxiv.org/abs/2012.12877.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Grant Van Horn, Oisin Mac Aodha, Trevor Marquis, Steve Su, Mona Haghighi, Jason Baldridge, Subhransu Maji, and Pietro Perona. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8769–8778. IEEE, 2018.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression, 2024a. URL https://arxiv.org/abs/2410.03765.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression, 2024b. URL https://arxiv.org/abs/2403.07378.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression, 2025. URL https://arxiv.org/abs/2503.12340.

Hao Yu and Jianxin Wu. Compressingtransformers. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26304. URL https://doi.org/10.1609/aaai.v37i9.26304.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2023. URL https://arxiv.org/abs/2312.05821.

Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing, 2022. URL https://arxiv.org/abs/2204.07154.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch, 2021. URL https://arxiv.org/abs/2102.04010.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017. URL https://arxiv.org/abs/1710.01878.

A Appendix

A.1 Method

Referring to § 2.2, we detail four methods for concatenating the FC weights of two MLPs (four matrices $\mathbf{W}_{ij} \in \mathbb{R}^{d\times p}$, with $p = 4d$ and $i, j \in \{1, 2\}$):

I. **Full long-axis concatenation**, forming

$$\mathbf{W} = [\,\mathbf{W}_{11}\,\mathbf{W}_{12}\,\mathbf{W}_{21}\,\mathbf{W}_{22}\,] \;\in\; \mathbb{R}^{d\times 16d}.$$

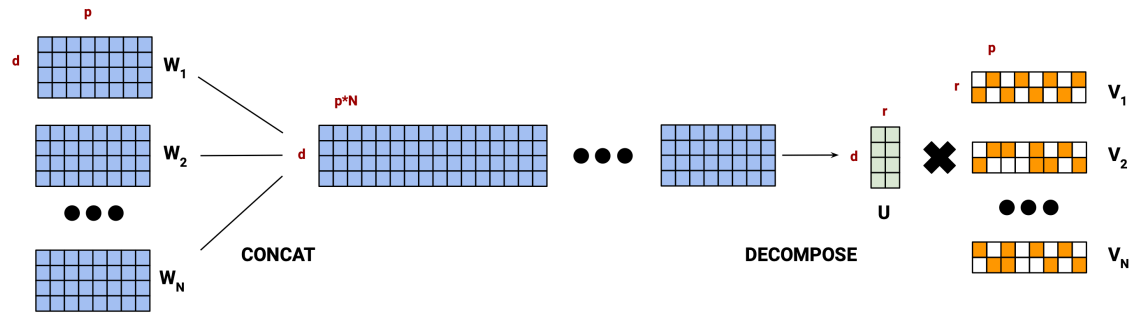


Figure 6: **Parameter Sharing Through Sparse Tensor Decomposition.** A group of FC layers are concatenated along the larger dimension, p, and decomposed into two matrices: a shared basis, $\mathbf{U}$, and a sparse projection matrix, which is then sliced up respectively for each layer.

II. **Module-wise long + inter-module short:**

$$\mathbf{W}_A = [\mathbf{W}_{11}\mathbf{W}_{12}] \in \mathbb{R}^{d\times 8d}$$

$$\mathbf{W}_B = [\mathbf{W}_{21}\mathbf{W}_{22}] \in \mathbb{R}^{d\times 8d}$$

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_A \\ \mathbf{W}_B\end{bmatrix} \in \mathbb{R}^{2d\times 8d}.$$

III. **Module-wise short + inter-module long:**

$$\mathbf{W}_C = [\mathbf{W}_{11}\mathbf{W}_{21}] \in \mathbb{R}^{2d\times 4d}$$

$$\mathbf{W}_D = [\mathbf{W}_{12}\mathbf{W}_{22}] \in \mathbb{R}^{2d\times 4d}$$

,

$$\mathbf{W} = [\,\mathbf{W}_C\,\mathbf{W}_D\,] \in \mathbb{R}^{2d\times 8d}.$$

IV. **Full short-axis concatenation**, yielding

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_{11} \\ \mathbf{W}_{12} \\ \mathbf{W}_{21} \\ \mathbf{W}_{22}\end{bmatrix} \in \mathbb{R}^{4d\times 4d}.$$

These configurations are evaluated in Figure 2b and discussed in § 2.2.

The overall method is depicted in Figure 6, and further details are explained below.

A.1.1 MLP Concatenation Strategies

A.1.2 Growing Neurons in Shared Bases and Sparse Factors

As discussed in § 3, high parameter budgets and sparsity levels (e.g., 26.5% parameter budget, 75% sparsity, and groups of four blocks in DeiT-B) often result in the rank r exceeding the model dimension d. Since SVD yields only d initialization directions, we investigate three methods to initialize the remaining $k = r - d$ dimensions:

1. **Random Growth:** Initialize new neurons in $\mathbf{U}$ to zero and in $\mathbf{V}$ randomly using He et al. (2015);

2. **Neuron Splitting:** Duplicate the top k neurons of $\mathbf{U}$ and halve the top k neurons of $\mathbf{V}$, following Chen et al. (2016);

3. **Hybrid Initialization:** Initialize new neurons in $\mathbf{U}$ to zero and derive those in $\mathbf{V}$ from the top k neurons, scaled by τ. This minimizes the immediate impact of new neurons in $\mathbf{V}$, allowing their gradual reactivation, as proposed by Evci et al. (2022).

After performing a hyperparameter sweep for τ, hybrid initialization outperformed the alternatives, achieving 1% and 2% higher accuracy than methods (1) and (2), respectively.

A.2 Compression Ratio Computation

The reported compression ratio accounts for all storage overhead, including sparse metadata and uncompressed components:

- **Sparse factors (V):** We store both the nonzero values and their positions. For unstructured sparsity, we use a bitmap mask (1 bit per element). For 2:4 structured sparsity, the fixed pattern eliminates the need for explicit indices.

- **Shared basis (U):** Each group has one shared basis $\mathbf{U}$, counted once per group. For example, with $\beta = [4, 6, 6, 6, 6, 4]$ for Llama-7B (6 groups), we store 6 shared bases.

- **Non-MLP parameters:** Attention projections, embeddings, and layer norms remain uncompressed and are counted at full precision.

- **Quantized setup:** Each parameter is counted at its precision level (e.g., 3 bits for INT3), and masks are counted as 1 bit per element.

Formally, the compression ratio is:

$$\text{Compression Ratio} = \frac{|\theta_{\text{non-MLP}}| \cdot b + G \cdot |\mathbf{U}| \cdot b + \sum_i \left[(1-s) \cdot |\mathbf{V}_i| \cdot b + |\mathbf{V}_i| \cdot 1\right]}{|\theta_{\text{original}}| \cdot b_{\text{original}}}, \tag{4}$$

where G is the number of groups, s is the sparsity level, b is the bits per parameter (e.g., 16 for `bfloat16`), b_{original} is the original precision, and the sum is over all sparse factors $\mathbf{V}_i$ across all layers. For 2:4 structured sparsity, the mask term $|\mathbf{V}_i| \cdot 1$ is omitted as the sparsity pattern is implicit.

A.3 Model Links

- DeiT-B (Touvron et al., 2021): https://huggingface.co/facebook/deit-base-patch16-224

- Swin-L (Liu et al., 2021): https://huggingface.co/microsoft/swin-large-patch4-window7-224

- Gemma-2-2B-IT (Team et al., 2024): https://huggingface.co/google/gemma-2-2b-it

- Gemma-2-9B (Team et al., 2024): https://huggingface.co/google/gemma-2-9b

- Llama-7B (Touvron et al., 2023): https://huggingface.co/huggyllama/llama-7b

- Llama-3.1-8B (Grattafiori et al., 2024): https://huggingface.co/meta-Llama/Llama-3.1-8B

Comp. Ratio	10%	25%	40%	50%	75%
STE	42.89	73.26	78.26	79.36	78.89
SR-STE	45.31	75.53	79.71	80.68	81.24
NMSRigL	44.87	75.71	79.97	80.99	81.40
NMSGMP	52.36	76.88	80.59	81.31	81.51
FiPS (50% Sparsity)	54.00	77.56	80.94	81.63	81.77
FiPS (75% Sparsity)	70.04	80.64	81.69	81.83	81.82

Table 7: **Structured Sparsity Performance.** ImageNet top-1 accuracy (%) of DeiT-B (81.85%) (Touvron et al., 2021) for various structured sparsification methods at 50% and 75% sparsity, compared to unstructured FiPS. Methods include Straight Through Estimator (*STE*), Sparse-Refined STE, N:M Structured RigL (*NMSRigL*), and N:M Structured GMP (*NMSGMP*) at 50% sparsity, corresponding to 2:4 structures (Lee et al., 2023; Zhou et al., 2021; Lasby et al., 2024).

A.4 Further LLM Results

Variant	Comp. Ratio	PPL ↓
Original	0%	7.34
SVD-LLM	20%	15.84
SVD-LLM V2	20%	11.72
FiPS (ours)	20%	**8.10**
SVD-LLM	40%	75.42
FiPS (ours)	40%	**10.57**

Table 6: Perplexity on C4 for Llama7B under different compression ratios applied with FiPS and the baselines.

A.5 Different Sparsification Methods

The sparsification method comparison is presented in Table 5 in the main text (§5). For DeiT-B, *RigL* consistently outperforms both *Dense* and *Static Sparsity* across parameter budgets ranging from 10% to 50%. At higher parameter budgets, all methods converge to similar accuracies approaching the original model's performance. For Swin-L, *RigL* surpasses *Dense* and *Static Sparsity* at 10% and 25% parameter budgets. However, at higher parameter budgets, *Static Sparsity* achieves slightly higher accuracies.

A.6 Structured Sparsity

We evaluate the generalization performance of FiPS using structured sparsity, with results presented in Table 7. The methods evaluated include the Straight Through Estimator (*STE*), which employs top-k weight magnitude selection, projects parameters into a sparse subspace during training, and applies gradients to dense parameters through a gradual pruning schedule; Sparse-Refined STE (*SR-STE*) (Zhou et al., 2021), which mitigates the adverse effects of approximated gradients; and N:M Structured RigL (*NMSRigL*) and N:M Structured GMP (*NMSGMP*) (Lee et al., 2023; Lasby et al., 2024), where N:M specifies the sparsity pattern of the weight matrix (e.g., 50% sparsity in FC matrices of size $d \times 4d$ corresponds to a 2:4 structure).

A.7 Sensitivity Analysis

The sensitivity analysis plots are presented in Figure 4 in the main text (§5).

Calibration Dataset Size and Training Length. We examine how the number of calibration batches and training epochs affects performance using a fixed batch size of 128. To ensure at least one example from each category, we begin with a minimum of 10 batches and also evaluate 20, 40, and 80 batches. After filtering out configurations more than 0.25% below the highest accuracy, we adopt the most efficient setting of 20 epochs over 20 batches for all reported results, as shown in Figure 4b.

Optimal Sparsity for Sparse Factors. We compressed DeiT-B as described in § 4.1, using sparsity levels ranging from 50% to 96.9% (Figure 4a). The best performance was observed at 75% sparsity. While increasing sparsity to 87% yielded similar accuracy, lowering it to 50% resulted in a notable performance drop, likely due to a significant reduction in rank.

A.8 Hyper-parameters

A.8.1 Ablation on the Block–Grouping Hyper-parameter β

Definition. We define β as an *ordered list* whose i^{th} element gives the number of consecutive decoder blocks whose MLP weights are tied in the ith parameter sharing group:

$$\beta = \big[\beta_1,\ \beta_2,\ \ldots,\ \beta_G\big], \quad \textstyle\sum_{g=1}^{G} \beta_g = L,$$

where L is the total number of decoder blocks. Self-attention parameters remain untied in all experiments. For every architecture we sweep over a small number of plausible β lists (3–5 candidates) and keep the one with the lowest validation perplexity (PPL) after compression.

Default Heuristic. Our analysis in § 2.3 shows that adjacent layers exhibit smaller reconstruction error when sharing, and deeper layers require more capacity. This motivates a "tapered" grouping where group sizes are smaller at the network extremes and larger in the middle. Based on our experiments, we propose the following reproducible default for a decoder with L blocks:

> Use 4–6 groups with β tapered as $[\mathit{small}, \mathit{medium}, \ldots, \mathit{medium}, \mathit{small}]$, where $\mathit{small} \approx L/8$ and $\mathit{medium} \approx L/5$.

For example:

- $L = 32$ (Llama): $\beta = [4, 6, 6, 6, 6, 4]$
- $L = 26$ (Gemma-2-2B): $\beta = [4, 4, 5, 5, 4, 4]$
- $L = 12$ (DeiT-B): $\beta = [4, 4, 4]$

Block Groups of ViTs. Using the list-valued notation for β introduced above, we set

$$\beta_{\text{DeiT-B}} = [4,\ 4,\ 4],$$

i.e. three groups of four consecutive blocks (each block contains one MLP).

The depth pattern of Swin-L is 2+2+18+2 blocks. We tie MLP weights inside every 2-block stage and split the 18-block stage into three groups of six, which gives

$$\beta_{\text{Swin-L}} = [2,\ 2,\ 6,\ 6,\ 6,\ 2].$$

Gemma-2-2B. Table 8 shows the five β lists evaluated for Gemma-2-2B-it at 20% compression. Config. 1—$\beta = [\,4, 4, 5, 5, 4, 4\,]$—yields the lowest PPL and is therefore used in the main paper. Notably, alternative groupings yield PPL within $\sim$2 points of the optimal (21.42 vs. $\approx$23), indicating that FiPS is not overly sensitive to this hyperparameter.

Table 8: **Block–grouping sweep for Gemma-2-2B.** Each row lists the candidate β and the resulting validation perplexity (PPL) on WikiText-2. Baseline (no sharing) PPL is 15.61; lower is better.

Config.	β **list**	PPL ↓
1	$[4, 4, 5, 5, 4, 4]$	21.42
2	$[1, 4, 4, 4, 4, 4, 4, 1]$	≈23
3	$[2, 3, 4, 4, 4, 4, 3, 2]$	≈22
4	$[2, 6, 6, 6, 6]$	21.81
5	$[3, 5, 5, 5, 5, 3]$	21.86

Llama-2-7B. With $L = 32$ decoder blocks we compared three β lists:

Candidate β	**Block groups (sizes)**	**Observation**
$[4, 4, 4, 4, 4, 4, 4, 4]$	8 groups × 4 blocks	Highest PPL
$\mathbf{[4, 6, 6, 6, 6, 4]}$	**6 groups with sizes** $(4, 6, 6, 6, 6, 4)$	**Best PPL; used in §4.1**
$[8, 8, 8, 8]$	4 groups × 8 blocks	Slightly worse than above

A gently varying list—with smaller groups at the extremes and larger groups in the middle—provides the best compression/accuracy trade-off.

Llama-3.1-8B. The 8B model shares the same 32-layer decoder. We reused the winning β from the 7B sweep,

$$\beta_{\text{opt}} = [4, 6, 6, 6, 6, 4],$$

because (i) it keeps the total tied-parameter ratio identical and (ii) in a spot-check it preserved PPL within +3.5 of the uncompressed baseline—better than the uniform alternatives $[8, 8, 8, 8]$ or $[4, 4, 4, 4, 4, 4, 4, 4]$. This demonstrates that the optimal β transfers across models of similar architecture.

Key Insights.

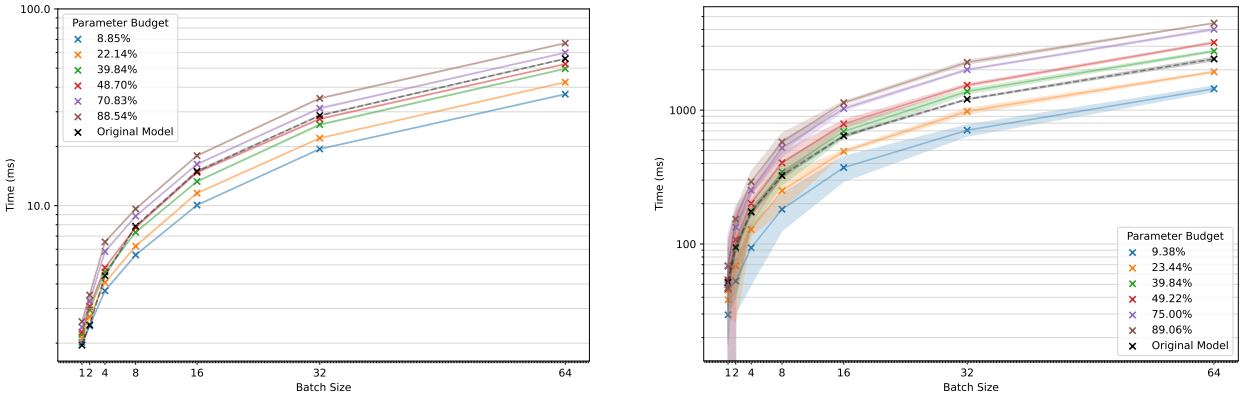
- Optimal β often starts and ends with smaller groups, reflecting the intuition that early and late layers contain more specialized features.
- Extremely fine-grained sharing (e.g., many 4-block groups) hurts accuracy, while overly coarse sharing (uniform 8-block groups) sacrifices capacity.
- For Llama models, a tapered list such as $[4, 6, 6, 6, 6, 4]$ ties roughly 22–25% of MLP parameters yet adds only ∼3–4 perplexity points.
- The method is robust to the choice of β: alternative groupings typically yield performance within 2 PPL points of the optimal.

These ablations inform all main-text compression results.

A.8.2 Optimizer

ViT Compression. To minimize local error, we employ a logarithmic grid for hyper-parameter tuning. The learning rates for Dense, Static Sparsity, GMP, and RigL are set as follows for both DeiT-B and Swin-L:

1. Dense: 1.25×10^{-4},
2. Static Sparsity: 2.5×10^{-4},
3. GMP: 1×10^{-3},
4. RigL: 1×10^{-3}.



(a) GPU latency (ms) vs. parameter budget (%).

(b) CPU latency (ms) vs. parameter budget (%).

Figure 7: **DeiT-B inference latency benchmarks.** (a) End-to-end latency of 2:4 sparse FiPS on an NVIDIA A4000 for batch sizes 1–64; a 22 % parameter budget yields a 25 % speed-up once the batch size exceeds 8. (b) Latency of 75 % unstructured sparse FiPS accelerated by DeepSparse on an Intel Xeon W-2145 CPU, outperforming the dense model at every tested batch size. Shaded regions denote $\pm 1\sigma$ over runs collected by `torch.benchmark.Timer.blocked_autorange`; GPU timings exclude `torch.compile` warmup iterations.

ViT Transfer Learning. We use a linear grid, as some hyper-parameters are derived from the codebase of DeiT. The optimal learning rates for FiPS are:

1. CIFAR-100: 2.5×10^{-5};
2. Flowers102: 1×10^{-4};
3. Oxford-III-Pets: 7.5×10^{-6};
4. iNaturalist 2019: 1×10^{-4}.

LLMs. Eight logarithmically spaced values were swept. The final values for FiPS are:

1. Gemma-2-2B: 4.0×10^{-4},
2. Llama-7B: 1.0×10^{-6},
3. Llama-3.1-8B: 1.0×10^{-5}.

Sparsifier

Gradual Magnitude Pruning (GMP). GMP begins with an initial sparsity level of 25%. During training, sparsity is gradually increased to 50% at the 25% training mark and ultimately reaches 75% by training completion. We use $\Delta T = 50$ for update steps.

RigL. RigL employs an initialization phase that combines pruning with a growth ratio of 0.1 for block-wise error minimization and a growth ratio of 0.05 for transfer learning tasks, with $\Delta T = 50$ for growth and pruning steps. This conservative growth ratio in transfer learning helps preserve the mask obtained during initial training, ensuring that learned masks are retained.

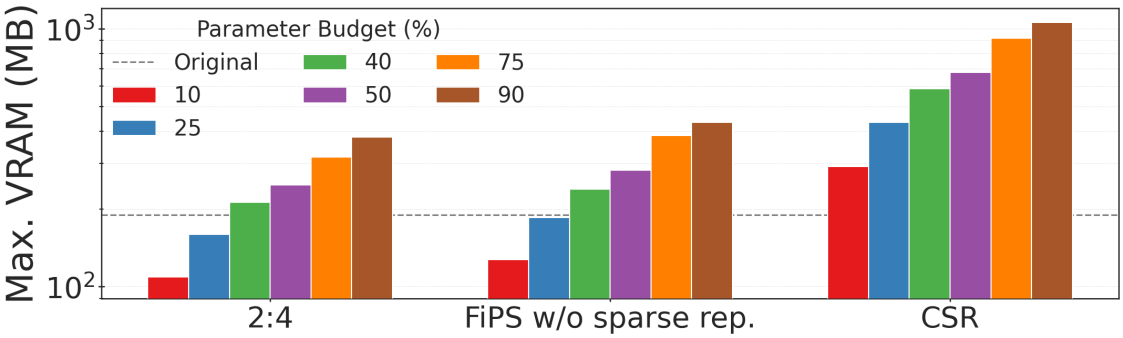


Figure 8: **DeiT-B Inference Memory Profile.** Maximum VRAM allocation at batch size 64 for 50% sparse FiPS using 2:4, strided (dense), and CSR tensor formats. At 10% and 25% parameter budgets, 2:4 sparsity reduces peak memory by 44% and 18%, respectively; CSR incurs higher overhead at modest sparsity due to index storage.

A.9 Latency and Memory Profiling

As discussed in § 3, high levels of sparsity and parameter budgets can result in the SVD rank exceeding a model's hidden dimension. For instance, in DeiT-B, achieving 75% sparsity under parameter budget constraints exceeding 26.5% with four block groups increases the rank of the shared singular vectors beyond the original model's embedding dimension. Efficient sparse operations and representations are crucial for minimizing the latency and memory overhead introduced by FiPS. Figure 7 and Figure 8 summarize the latency and memory results for DeiT-B compressed with FiPS using 2:4 structured GMP, highlighting the resulting speedups and memory savings on both CPU and GPU platforms.