SharVeT: Similarity-aware Parameter Sharing with Vector-based Tuning for Efficient LLM Compression

Anonymous ACL submission

Abstract

How can we share parameters within large language models to significantly reduce memory costs while preserving accuracy? While parameter sharing is a promising solution to the memory overhead of large language models, existing methods rely on naive grouping and fail to correct sharing-induced discrepancies. We propose an accurate and efficient parameter sharing framework, SharVeT (Similarity-aware sharing with Vector-based Tuning), which performs similarity-based grouping to ensure accurate sharing, allocates parameters adaptively to preserve diversity within each group, and applies lightweight refinement with knowledge distillation to correct sharing-induced discrepancies. Experiments show that SharVeT outperforms existing sharing methods, achieving up to 32.1% lower perplexity and 23.3% higher few-shot reasoning accuracy.

1 Introduction

How can we share parameters within large language models to significantly reduce memory costs while preserving accuracy? Large language models (LLMs) have achieved remarkable progress in natural language processing and multi-modal tasks (Team et al., 2023; Zhang et al., 2022; Dubey et al., 2024; Research et al., 2024). However, their ever-growing scale imposes a severe memory bottleneck (Chowdhery et al., 2023; Kaplan et al., 2020), making efficient compression essential for training and deployment in real-world environments (Miao et al., 2023; Yang et al., 2024). Among various approaches, parameter sharing is particularly attractive because it reduces redundancy while preserving functional capacity, and it naturally combines with quantization or pruning to achieve high compression ratios with minimal accuracy loss. These properties make parameter sharing a critical direction for sustaining the scalability of LLMs.

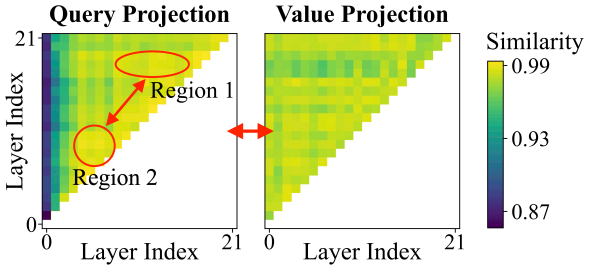

Figure 1: Cross-layer similarity of LLaMA modules. Similarity patterns differ across modules and high-similarity regions within a single module exhibit distinct characteristics.

Despite its promise, parameter sharing presents challenges in both the sharing process and the post-sharing stage. In the sharing process, one must decide what to group and how to handle each group. Previous works group weights based on simple layer-wise patterns (Takase and Kiyono, 2023; Wang et al., 2025; Bae et al., 2025; Lin et al., 2022). However, parameter sharing inherently assigns the same weights to groups, and thus weights need to be shared only among similar components to avoid information loss. Moreover, existing methods apply identical sharing strategies to all groups, which may not be optimal. In fact, our analysis of cross-layer similarity show in Figure 1 reveals otherwise: (1) different module types exhibit distinct similarity patterns, and (2) high-similarity regions within a module vary in both size and characteristics. These findings indicate that effective parameter sharing requires principled decisions about what to group and how to handle each group, guided by different properties of modules and groups. In the post-sharing stage, it is crucial to effectively resolve the inevitable discrepancies introduced by sharing. Yet, existing approaches (Bae et al., 2025; Lin et al., 2022; Dehghani et al., 2019) mainly employ generic LoRA-like adapters, which not only introduce substantial additional parameters but also are not optimal for discrepancy correction.

We propose SharVeT (Similarity-aware sharing with Vector-based Tuning), a parameter sharing framework that achieves efficient compression while maintaining high accuracy in large language models. First, we group modules using both structural information from weights and functional information from activations, ensuring that sharing occurs only among related components. Second, we allocate more parameters to groups with lower intra-group similarity, preserving distinctive module behaviors. Finally, we refine the shared model using lightweight dual-vector adapters and knowledge distillation from the original model to minimize discrepancies.

Our contributions are summarized as follows:

- **Observation.** We identify heterogeneous similarity patterns across modules in LLMs, showing that naive grouping strategies severely degrade sharing effectiveness.
- **Method.** We propose SharVeT, a unified framework for parameter sharing that integrates module similarity clustering, adaptive parameter allocation, and dual-vector refinement with knowledge distillation.
- **Experiments.** Extensive experiments on the LLMs demonstrate that SharVeT consistently outperforms existing sharing approaches at the same compression ratio, yielding up to 32.1% lower perplexity and 23.3% higher few-shot accuracy on common sense reasoning tasks.

In the rest of the paper, we present preliminaries in Section 2, introduce our method in Section 3, evaluate SharVeT against baselines in Section 4, review related works in Section 5, and conclude the paper in Section 6. The codes are available at https://github.com/SharVet/SharVeT.

2 Preliminaries

2.1 Problem Definition

Parameter sharing is an effective approach to address the model compression problem. We formally define the model compression problem as follows.

Problem 1 (LLM Compression)**.** *Given a pre-trained large language model (LLM) $\mathcal{M}_\theta$, a small fine-tuning dataset consisting of text sequences, and a target compression ratio $\alpha \in (0, 1]$, LLM compression is to construct a compressed model $\mathcal{M}_{\theta_c}$ that satisfies*

$$\frac{|\theta_c|}{|\theta|} \leq \alpha, \tag{1}$$

where $|\theta|$ and $|\theta_c|$ are the numbers of parameters in the original $\mathcal{M}_\theta$ and compressed $\mathcal{M}_{\theta_c}$ models, respectively, while preserving the accuracy of the compressed model $\mathcal{M}_{\theta_c}$ on downstream tasks.

2.2 Architecture of Large Language Models

Large language models are typically composed of stacked Transformer decoder layers, each consisting of a multi-head attention sublayer and a feed-forward sublayer. These sublayers are decomposed into modules; the multi-head attention sublayer contains query, key, and value projection modules, while the feed-forward sublayer comprises gate, up, and down projection modules (we follow the design of models such as LLaMA (Touvron et al., 2023)). Previous studies on parameter sharing have treated each Transformer layer as a unit of sharing. In contrast, this paper explores module-wise parameter sharing, motivated by our observation that similarities vary across modules within a layer.

2.3 Cross-Layer Weight Sharing via SVD

Singular value decomposition (SVD) offers an efficient mechanism for weight sharing (Wang et al., 2025). The key idea is to jointly factorize multiple weight matrices such that different layers reuse a common low-rank basis.

Let $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_Z\}$ be the set of Z weight matrices, where the z-th weight $\mathbf{W}_z \in \mathbb{R}^{d_i \times d_o}$ has the input dimension of d_i and the output dimension of d_o. The matrices are concatenated along the output dimension d_o to form a single matrix $\bar{\mathbf{W}}$:

$$\bar{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 & \cdots & \mathbf{W}_L \end{bmatrix} \in \mathbb{R}^{d_i \times (d_o Z)}. \tag{2}$$

Applying truncated SVD to the concatenated matrix $\bar{\mathbf{W}}$ yields $\bar{\mathbf{W}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{d_i \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, $\mathbf{V} \in \mathbb{R}^{(d_o Z) \times r}$, and r is the rank of the truncated SVD. By partitioning $\mathbf{V}$ into Z contiguous blocks $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_Z$ with $\mathbf{V}_z \in \mathbb{R}^{d_o \times r}$, the weight matrix of z-th layer is approximated as

$$\mathbf{W}_z \approx \mathbf{U}\mathbf{\Sigma}(\mathbf{V}_z)^\top, \quad z = 1, \ldots, Z. \tag{3}$$

The matrices $\mathbf{U}$ and $\mathbf{\Sigma}$ are shared across all layers, while the layer-specific variations are captured only in $\mathbf{V}_z$. In this paper, we form groups of weight matrices across different layers for each module type, and exploit the above SVD-based weight sharing within each group.

In this structure, the degree of compression is controlled by the rank r, since the number of de-

composed parameters is $r\big(d_i + 1 + d_o Z\big)$. To satisfy the target number P of parameters, the corresponding rank is obtained as $f\big(P, \phi(\mathcal{W})\big)$, where $\phi(\mathcal{W}) = (d_i, d_o, Z)$ for a set $\mathcal{W}$ of weights:

$$f(P, \phi(\mathcal{W})) = \left\lfloor \frac{P}{(d_i + 1 + d_o Z)} \right\rfloor. \tag{4}$$

3 Proposed Method

We propose SharVeT (Similarity-aware sharing with Vector-based Tuning), a unified parameter sharing framework that achieves substantial compression with minimal accuracy loss.

3.1 Overview

We address the following challenges for parameter sharing:

C1. Ambiguity in grouping diverse components. Large language models contain heterogeneous components, making it unclear which should be grouped together. How can we group components to ensure sharing occurs only among related ones?

C2. Handling heterogeneous group similarity. Different groups exhibit diverse characteristics, including variations in size and sensitivity. How can we account for heterogeneous characteristics when sharing parameters?

C3. Discrepancies introduced by sharing. Parameter sharing inevitably distorts the behaviors of the original model. How can we efficiently correct the discrepancies?

The main ideas are summarized as follows:

I1. Similarity-based grouping of modules (Section 3.2). We perform grouping within the each module type using both weight statistics and activation patterns to ensure sharing occurs only among structurally and functionally related modules.

I2. Adaptive parameter allocation (Section 3.3). We allocate parameters based on intra-group similarity, assigning more capacity to groups with low internal similarity while aggressively compressing those with high similarity.

I3. Lightweight discrepancy refinement (Section 3.4). We refine shared modules with lightweight dual-vector adapters and knowledge distillation to correct discrepancies.

Figure 2 provides an overview of SharVeT. The three components, similarity-based grouping of modules, adaptive parameter allocation, and lightweight discrepancy refinement, are applied sequentially for effective parameter sharing.

3.2 Similarity-based Grouping of Modules

How can we group components appropriately so that sharing occurs only among related modules? Our analysis in Figure 1 shows that different module types exhibit distinct similarity patterns. Motivated by this observation, we perform grouping within each module type based on pairwise similarity of modules, ensuring that sharing occurs only among structurally and functionally related components. Once the groups are formed, we apply SVD-based cross-layer weight sharing within each group, as described in Section 2.3.

Formally, for each module type t, we calculate similarity across all layers and apply spectral clustering (Ng et al., 2001; von Luxburg, 2007) to partition the modules into k groups. We adopt spectral clustering because it captures the global structure of the similarity matrix, thereby enabling distant but related modules to be grouped together. Let $\mathbf{W}_l^{(t)} \in \mathbb{R}^{d_i \times d_o}$ denote the weight of module type t at layer l. The resulting set $\mathcal{G}^{(t)}$ of groups for module type t is

$$\mathcal{G}^{(t)} = \{\mathcal{G}_1^{(t)}, \mathcal{G}_2^{(t)}, \ldots, \mathcal{G}_k^{(t)}\}, \tag{5}$$

where each group $\mathcal{G}_g^{(t)}$ contains weights $\mathbf{W}_l^{(t)}$ of modules that are highly similar.

To quantify the similarity between modules, we consider two complementary perspectives: structural similarity and functional similarity. Structural similarity reflects the alignment of weight parameters, but relying on it alone is insufficient because modules with different weights often exhibit similar behavior depending on the input to the LLM. Conversely, functional similarity reflects the alignment of activation patterns, but it is unstable when used in isolation, since activation-based similarity varies with the input distribution. Thus, we define the overall similarity $S(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)})$ between weights $\mathbf{W}_l^{(t)}$ and $\mathbf{W}_j^{(t)}$ of two modules as a joint measure balancing structural similarity $S_s(\cdot,\cdot)$ and functional similarity $S_f(\cdot,\cdot)$:

$$\begin{aligned} S(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)}) = 1 - \big(1 - S_s(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)})\big) \\ \times \big(1 - S_f(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)})\big). \end{aligned} \tag{6}$$

We next specify the two components of the similarity measure. Structural similarity S_s captures both directional and magnitude alignment of

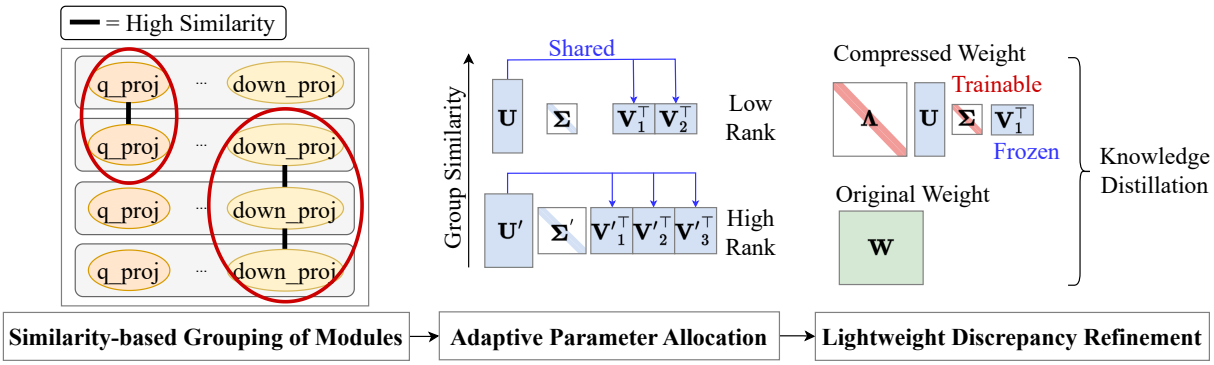

Figure 2: Overview of SharVeT. We group similar modules based on measured similarity, allocate parameters with appropriate sizes to each group, and mitigate sharing-induced errors with a lightweight vector-based adapter. We perform the three stages above for each type of module.

weights. Given the weights $\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)} \in \mathbb{R}^{d_i \times d_o}$ of two modules, we combine cosine similarity and Frobenius distance with normalization:

$$S_s(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)}) = \tfrac{1}{2}\Big(\mathcal{N}\big((\cos(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)}) + 1)\big) + \mathcal{N}\big((||\mathbf{W}_l^{(t)} - \mathbf{W}_j^{(t)}||_F)^{-1}\big)\Big), \tag{7}$$

where $\mathcal{N}(x) = 1 - \exp(-x/\gamma)$ and $\gamma > 0$ controls sensitivity. Smaller γ values make the metric more sensitive to fine-grained differences, while larger values smooth the effect across a wider range.

Functional similarity S_f is computed using centered kernel alignment (CKA) on activations of each module (see Appendix B for details). Given activations $\mathbf{X}_\mathcal{C} \in \mathbb{R}^{n \times d_i}$ from calibration data $\mathcal{C}$, where n denotes the number of tokens, we define the functional similarity $S_f(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)})$ of a pair modules as

$$S_f(\mathbf{W}_l^{(t)}, \mathbf{W}_j^{(t)}) = \mathrm{CKA}\big(\mathbf{X}_\mathcal{C}\mathbf{W}_l^{(t)}, \mathbf{X}_\mathcal{C}\mathbf{W}_j^{(t)}\big). \tag{8}$$

CKA measures how similarly two modules preserve relationships among examples, thus capturing functional equivalence even when their weight structures differ.

3.3 Adaptive Parameter Allocation

How can we account for different characteristics of groups when sharing parameters? Different groups of each module type exhibit heterogeneous internal similarity and sensitivity. To handle these differences, we allocate the parameters adaptively according to group characteristics. Groups with low internal similarity are assigned larger capacity to preserve the diversity of their constituent modules, whereas groups with high similarity are compressed to smaller capacity. Such adaptive allocation is necessary because applying the same capacity to all groups results in over-parameterization for homogeneous groups and under-parameterization for heterogeneous ones, thereby wasting capacity and degrading fidelity.

To this end, we allocate parameter capacity to each group inversely proportionally to its internal similarity, and the corresponding rank is assigned for weight decomposition of each group. Specifically, we guarantee a minimum allocation by reserving a fraction μ of the capacity for every group, and distribute the remaining capacity inversely proportionally to similarity. Formally, the capacity budget $B_g^{(t)}$ for the group $\mathcal{G}_g^{(t)}$ is defined as

$$B_g^{(t)} = \mu \times \alpha \times \psi(|\mathcal{G}_g^{(t)}|, t) + \eta_g^{(t)} \times (1-\mu)\alpha \times \psi(L, t) \tag{9}$$

where α is the compression ratio, $\psi(n, t) = d_i^{(t)} \times d_o^{(t)} \times n$ denotes the number of parameters for n weights of type t, L is the total number of layers in the model, and $\eta_g^{(t)}$ is the inverse-similarity of $\mathcal{G}_g^{(t)}$. The first term $\mu \times \alpha \times \psi(|\mathcal{G}_g^{(t)}|, t)$ corresponds to the fraction μ of the target number $\alpha \times \psi(|\mathcal{G}_g^{(t)}|, t)$ of parameters that we guarantee for each group, ensuring that no group is allocated less than a minimal capacity. The second term $\eta_g^{(t)} \times (1-\mu)\alpha \times \psi(L, t)$ redistributes the residual $(1-\mu)$ fraction of the target parameters of all L layers with the baseline allocation excluded proportionally to the inverse-similarity $\eta_g^{(t)}$ of group $\mathcal{G}_g^{(t)}$.

To serve as a measure of group heterogeneity, the inverse-similarity $\eta_g^{(t)}$ is defined as the reciprocal of the total pairwise similarity $S(\cdot, \cdot)$ (see equation 6)

within $\mathcal{G}_g^{(t)}$:

$$\eta_g^{(t)} \propto \Big(\sum_{\substack{\mathbf{W},\, \mathbf{W}' \in \mathcal{G}_g^{(t)} \\ \mathbf{W} \neq \mathbf{W}'}} S(\mathbf{W}, \mathbf{W}') \Big)^{-1} \tag{10}$$

Finally, the rank $R_g^{(t)}$ of each group $\mathcal{G}_g^{(t)}$ is determined from the allocated budget $B_g^{(t)}$ using the equation 4 as:

$$R_g^{(t)} = f\big(B_g^{(t)}, \phi(\mathcal{G}_g^{(t)})\big) \tag{11}$$

3.4 Lightweight Discrepancy Refinement

How can we efficiently correct the discrepancies? Although group-wise factorization reduces redundancy effectively, it inevitably introduces misalignment between shared and individual modules, which leads to performance degradation. To mitigate this problem, we propose a refined structure that leverages two lightweight vectors for compression efficiency, and tailored loss objectives for effective discrepancy recover.

The key idea of lightweight discrepancy refinement is to adjust the shared decomposition with minimal additional parameters. As established in equation 3, the weight matrix $\mathbf{W}_l^{(t)}$ of each module in $\mathcal{G}_g^{(t)}$ is approximated by the shared decomposition with group-shared and layer-specific components. We refine this approximation by introducing a layer-specific diagonal scaling matrix applied to the shared component, and by replacing the shared singular values with layer-specific ones. This design allows each module to adjust both the orientation of the shared parameter and the strength of its singular values for the layer-specific component. Formally, the weight matrix of module $\mathbf{W}_l^{(t)} \in \mathcal{G}_g^{(t)}$ is approximated as

$$\mathbf{W}_l^{(t)} \approx \mathbf{\Lambda}_l^{(t)} \mathbf{U}_g^{(t)} \mathbf{\Sigma}_l^{(t)} (\mathbf{V}_l^{(t)})^\top, \tag{12}$$

where $\mathbf{U}_g^{(t)} \in \mathbb{R}^{d_i \times r}$ is the group-shared parameter, $\mathbf{\Lambda}_l^{(t)} \in \mathbb{R}^{d_i \times d_i}$ is a diagonal matrix for layer-specific scaling initialized with an identity matrix, $\mathbf{\Sigma}_l^{(t)} \in \mathbb{R}^{r \times r}$ is a diagonal matrix of layer-specific singular values, and $\mathbf{V}_l^{(t)} \in \mathbb{R}^{d_o \times r}$ is the layer-specific projection. During training, only the diagonal entries $\mathrm{diag}(\mathbf{\Lambda}_l^{(t)})$ and $\mathrm{diag}(\mathbf{\Sigma}_l^{(t)})$ are updated, while all other components remain fixed. Nevertheless, these two vectors effectively rescale all rows and columns of the frozen matrices, enabling flexible control over the entire weights. As a result, each module requires only one additional vector and trains only two vectors, keeping the parameter overhead negligible while enabling effective recovery of inter-module variations.

While the refinement of structure enables efficient parameterization, its effectiveness depends critically on how the refinement vectors are trained. General fine-tuning typically optimizes the compressed models only by the language modeling objective. We augment this with additional alignment terms to better guide the refinement vectors. Specifically, our training objective $\mathcal{L}$ combines the standard language modeling loss $\mathcal{L}_{LM}$ with distillation loss $\mathcal{L}_{KD}$, feature-level consistency loss $\mathcal{L}_f$, and regularization $\mathcal{L}_{L2}$:

$$\mathcal{L}(\theta_c \mid \theta) = \mathcal{L}_{LM} + \lambda_{KD}\, \mathcal{L}_{KD} + \lambda_f\, \mathcal{L}_f + \lambda_{L2}\, \mathcal{L}_{L2}, \tag{13}$$

where θ and θ_c are the parameters of original and compressed model, respectively. $\lambda_{(\cdot)} \geq 0$ is a hyper-parameter that controls the contribution of loss term.

The distillation loss $\mathcal{L}_{KD}$ matches the output distributions of original and compressed model:

$$\mathcal{L}_{KD} = \mathbb{E}_{x \sim \mathcal{D}} \left[\mathrm{KL}\big(p_\theta(\cdot \mid x) | p_{\theta_c}(\cdot \mid x)\big) \right], \tag{14}$$

where $\mathcal{D}$ denotes the data distribution, $p_\theta(\cdot \mid x)$ is the output distribution of model with parameters θ over the vocabulary of the input text x, and $\mathrm{KL}(\cdot|\cdot)$ denotes the KL divergence between two probability distributions (Kullback and Leibler, 1951).

The feature-level consistency loss $\mathcal{L}_f$ enforces alignment between hidden representations of the teacher and the compressed model:

$$\mathcal{L}_f = \mathbb{E}_{x \sim \mathcal{D}} \left[\| h_\theta(x) - h_{\theta_c}(x) \|_2^2 \right], \tag{15}$$

where h_θ denote the hidden representations of input x obtained from the model with parameters θ.

Parameter efficiency. We analyze the parameter efficiency of the proposed refinement scheme. We compare our refinement scheme with LoRA (Hu et al., 2022), a common parameter-efficient fine-tuning method in parameter sharing. Our proposed SharVeT achieves multiple-fold reduction in the number of trainable parameters than LoRA. Table 1 shows the parameter counts of both approaches.

In existing parameter sharing approaches, fine-tuning is typically performed by attaching standard LoRA modules after sharing weights. Let $\tilde{\mathbf{W}}$ denote the decomposed weight matrix obtained by applying singular value decomposition to $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ using r singular values. When

Table 1: Per-layer parameter counts under parameter sharing. Our proposed refinement dramatically reduces the number of trainable parameters compared to LoRA.

Method	Total	Trainable
$\tilde{\mathbf{W}}$	$r(d_{in}+1+d_{out})$	0
+LoRA	$r(d_{in}+1+d_{out})+r_A(d_{in}+d_{out})$	$r_A(d_{in}+d_{out})$
+SharVeT	$r(d_{in}+1+d_{out})+d_{in}$	$(r+d_{in})$

LoRA is applied, the decomposed weight matrix $\tilde{\mathbf{W}}$ is frozen, and two low-rank matrices $\mathbf{A} \in \mathbb{R}^{r_A \times d_{in}}$ and $\mathbf{B} \in \mathbb{R}^{d_{out} \times r_A}$ are introduced per layer, where r_A denotes the LoRA rank. This results in $r(d_{in}+1+d_{out})+r_A(d_{in}+d_{out})$ total parameters and $r_A(d_{in}+d_{out})$ trainable parameters per layer. Although more efficient than full fine-tuning, this design still requires substantial additional parameters.

In contrast, we unfreeze the singular values of the decomposed weight matrix and introduce an additional trainable scaling vector $\mathrm{diag}(\mathbf{\Lambda}) \in \mathbb{R}^{d_{in}}$ (see equation 12). Formally, this yields $r(d_{in}+1+d_{out})+d_{in}$ total parameters, with only $(r+d_{in})$ trainable parameters per layer. Thus, instead of training two full low-rank matrices, our method requires updating only two lightweight vectors.

Assuming a layer where d_{in} equals d_{out}, the proposed method requires at most $2d_{in}$ trainable parameters. Since $r < d_{in}$, the actual number of parameters is smaller than $2d_{in}$. In comparison, LoRA introduces $2r_A d_{in}$ parameters, indicating that our method uses approximately r_A times fewer trainable parameters. This dramatically reduces parameter overhead, and the resulting model is not only more compact but also converges fast and stably during training.

4 Experiments

We perform experiments to answer the following questions on SharVeT.

Q1. Performance (Section 4.2). How accurate is SharVeT compared to existing methods?

Q2. Ablation study (Section 4.3). How effective is each component of SharVeT for the performance of the compressed model?

Q3. Efficiency of fine-tuning (Section 4.4). How efficient is the proposed fine-tuning in terms of convergence speed?

Q4. Comparison of grouping strategies (Section 4.5). How does the proposed grouping perform relative to alternative strategies in terms of intra-group similarity?

4.1 Experimental Settings

Setup. We use TinyLLaMA-1.1B (Zhang et al., 2024) and LLaMA-3-8B (Dubey et al., 2024) as target models for compression. Each model is fine-tuned on 5,000 text sequences from the SlimPajama (Soboleva et al., 2023) dataset. We evaluate perplexity on C4 (Dodge et al., 2021) and WikiText2 (Merity et al., 2017) datasets, and commonsense reasoning accuracy on PIQA (Bisk et al., 2020), ARC-Easy (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020) ,and LAMBADA (Paperno et al., 2016) datasets, following each task's default few-shot setting. In tables, the best and second-best performances are shown in bold and underlined, respectively. Abbreviations 'Wiki.', 'PQ', 'ARC', 'WG', 'LB', and 'Avg' denote WikiText2, PIQA, ARC-Easy, WinoGrande, LAMBADA, and average accuracy, respectively.

Baselines. We compare SharVeT with representative parameter-sharing approaches. SEQ (Takase and Kiyono, 2023) shares parameters sequentially, and Basis Sharing (Wang et al., 2025) employs the same strategy while decomposing the weights using singular value decomposition. CYCLE and its variant CYCLE-R (Takase and Kiyono, 2023) share parameters in cyclic and reverse-cyclic orders, respectively. For a fair comparison, each baseline is augmented with LoRA (Hu et al., 2022) adapters, enabling adaptation on shared parameters.

4.2 Performance

We assess the effectiveness of SharVeT on LLaMA-3-8B and TinyLlama-1.1B under varying compression ratios, as summarized in Table 2. Two key observations emerge. First, SharVeT consistently outperforms all baselines across all models and compression ratios. SharVeT achieves the lowest perplexity on every dataset and the highest accuracy on all commonsense reasoning benchmarks. In particular, SharVeT reduces perplexity by up to 32.1% and improves the average reasoning accuracy by up to 23.3% compared to the best competitor. Second, the performance improvement of SharVeT in both models over the best competitor becomes larger as the compression ratio increases. This demonstrates that SharVeT remains robust under heavy compression. Additional experiments on Mistral-7B-v0.1 (Jiang et al., 2023) and GPT2-XL (Radford et al., 2019) show consistent improvements, validating the generality of SharVeT across different architectures (see Appendix C).

Table 2: SharVeT consistently outperforms all baselines across metrics and compression ratios.

Model	Ratio	Method	Perplexity (↓)		Commonsense Reasoning Accuracy (↑)				
			C4	Wiki.	PQ	ARC	WG	LB	Avg
LLaMA-3 8B	100%	Original Model	9.9	6.8	0.797	0.801	0.728	0.761	0.772
	70%	SEQ	31.8	23.9	0.639	0.456	0.527	0.375	0.499
		CYCLE	45.4	45.4	0.672	0.551	0.656	0.455	0.584
		CYCLE-R	43.3	43.4	0.672	0.551	0.656	0.455	0.584
		Basis Sharing	23.8	24.3	0.696	0.574	0.623	0.537	0.608
		SharVeT (proposed)	**21.2**	**18.2**	**0.706**	**0.611**	**0.666**	**0.558**	**0.635**
	50%	SEQ	59.7	70.6	0.590	0.395	0.530	0.288	0.451
		CYCLE	121.7	197.0	0.577	0.319	0.560	0.033	0.372
		CYCLE-R	120.8	188.5	0.582	0.325	0.560	0.039	0.377
		Basis Sharing	45.2	56.1	0.607	0.419	0.558	0.308	0.473
		SharVeT (proposed)	**31.9**	**36.9**	**0.672**	**0.555**	**0.619**	**0.487**	**0.583**
TinyLLaMA 1.1B	100%	Original Model	9.7	8.9	0.745	0.543	0.604	0.610	0.626
	70%	SEQ	24.3	22.4	0.602	0.359	0.514	0.324	0.450
		CYCLE	64.8	104.7	0.544	0.303	0.525	0.012	0.346
		CYCLE-R	62.1	98.5	0.546	0.316	0.515	0.017	0.348
		Basis Sharing	22.5	25.6	0.615	0.425	0.527	0.356	0.481
		SharVeT (proposed)	**19.0**	**19.3**	**0.636**	**0.460**	**0.543**	**0.391**	**0.508**
	50%	SEQ	55.6	66.8	0.560	0.309	0.516	0.113	0.375
		CYCLE	55.6	66.8	0.523	0.298	0.507	0.003	0.333
		CYCLE-R	94.3	156.1	0.526	0.284	0.505	0.004	0.330
		Basis Sharing	47.4	63.3	0.558	0.316	0.507	0.173	0.388
		SharVeT (proposed)	**32.6**	**40.5**	**0.585**	**0.360**	**0.517**	**0.234**	**0.424**

Table 3: Ablation study showing that each component contributes to performance. Results are reported on LLaMA-3-8B under 50% compression.

Method	Commonsense Reasoning Accuracy(↑)				
	PQ	ARC	WG	LB	Avg
SharVeT$_R$	0.63	0.47	0.58	0.37	0.51
SharVeT$_I$	0.63	0.49	0.59	0.42	0.53
SharVeT$_L$	0.64	0.48	0.58	0.41	0.53
SharVeT$_T$	0.54	0.29	0.50	0.00	0.33
SharVeT	**0.67**	**0.55**	**0.62**	**0.49**	**0.58**

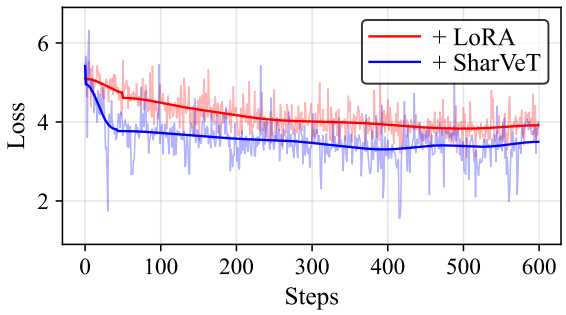

Figure 3: Loss curves of SharVeT and LoRA. SharVeT converges faster to a lower loss, whereas LoRA converges slowly and stabilizes at a higher loss.

4.3 Ablation Study

We conduct an ablation study to evaluate the effect of each component. The results show that all components contribute to the overall performance. Table 3 compares the full model (SharVeT) with four variants: (1) SharVeT$_R$ which replaces similarity-based grouping in Section 3.2 with random grouping, (2) SharVeT$_I$ which assigns identical ranks across all groups instead of adaptively allocating parameters in Section 3.3, (3) SharVeT$_L$ which removes the proposed loss in equation 13 during fine-tuning, and (4) SharVeT$_T$ which skips fine-tuning in Section 3.4. The results show that removing any component degrades model performance, demonstrating that all components are essential for the model's performance.

4.4 Efficiency of Fine-Tuning

We evaluate the efficiency of our fine-tuning method, focusing on the convergence speed in terms of language modeling loss. As shown in Figure 3, SharVeT converges faster and stabilizes at lower loss values than the existing fine-tuning method LoRA, which converges more slowly with higher loss. This demonstrates that SharVeT not only accelerates optimization but also ensures stable training. The fast and stable convergence is attributed to the substantially fewer trainable parameters required by our approach, as supported by Table 1.

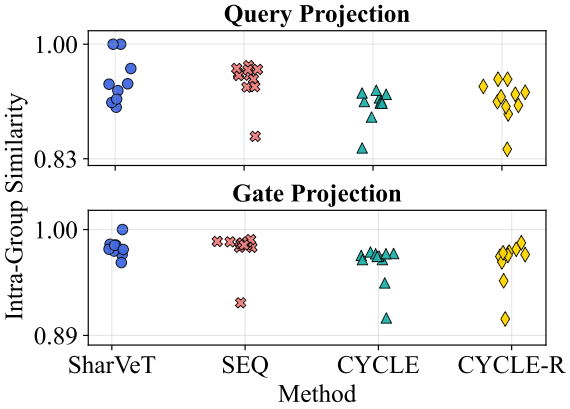

Figure 4: Similarity distributions of groups formed by different strategies. SharVeT ensures high intra-group similarity while effectively excluding outliers.

4.5 Comparison of Grouping Strategies

We compare different grouping strategies to assess how effectively they cluster modules with respect to intrinsic similarity. To evaluate their effectiveness, we use the similarity metric $S(\cdot, \cdot)$ proposed in Section 3.2, which combines weight and activation similarities. Figure 4 shows the distribution of intra-group similarity across different strategies. We observe two key findings. First, the proposed grouping strategy achieves higher overall intra-group similarity than CYCLE and REV, while remaining comparable to SEQ. Second, whereas the other methods yield outliers in which dissimilar layers are grouped together, SharVeT does not exhibit such cases and consistently produces coherent groups. These results highlight the necessity of principled similarity-based grouping to ensure meaningful partitioning of modules.

5 Related Works

5.1 Compression of Large Language Models

The rapid growth of large language models has intensified memory and latency challenges (Wei et al., 2022), spurring researches on model compression (Zhu et al., 2024; Xu and McAuley, 2023). Existing approaches include distillation, pruning, and quantization. Distillation (Gu et al., 2024; Magister et al., 2023; Jeon et al., 2023) transfers knowledge from a large teacher to a smaller student but requires costly retraining. Pruning (Park et al., 2025b, 2024; Song et al., 2024; Sun et al., 2024) removes unimportant weights, reducing capacity. Quantization (Park et al., 2025a; Kim et al., 2025; Liu et al., 2025; Shao et al., 2024; Ashkboos et al., 2024) lowers parameter precision, cutting memory (Lin et al., 2024; Tseng et al., 2024; Egiazarian et al., 2024) but not parameter count. While extensively studied, these methods underexplore parameter sharing, which can complement pruning and quantization to yield higher compression.

5.2 Parameter Sharing

Parameter sharing reduces redundancy in LLMs by reusing weights across modules, easing memory and computation costs, yet existing studies address only parts of the problem. The Universal Transformer (Dehghani et al., 2019) repeats a single shared layer recurrently, which requires training from scratch and severely restricts model capacity. Takase and Kiyono (2023) address the sharing process only naively, arranging layers sequentially or cyclically without considering similarity. Wang et al. (2025) group layers based on reconstruction error rather than precise structural or functional similarity. Bae et al. (2025) focus on the post-sharing stage, using LoRA-based adapters to mitigate discrepancies, but their approach incurs substantial parameter overhead and lacks well-defined training objectives for discrepancy correction. Despite their contributions, existing studies do not address the challenges of both stages—how to form accurate sharing groups and how to correct discrepancies after sharing—in a unified manner. In contrast, our work measures similarity to form accurate groups and introduces a lightweight refinement strategy to correct post-sharing discrepancies.

6 Conclusion

We propose SharVeT (Similarity-aware sharing with Vector-based Tuning), an efficient and effective framework for parameter sharing in large language models. First, we cluster modules using structural and functional similarity to overcome the weakness of naive grouping. Second, we allocate parameters adaptively to preserve distinctive behaviors across groups. Finally, we refine shared representations with lightweight adapters and knowledge distillation to correct discrepancies. Experiments show that SharVeT surpasses existing methods at the same compression ratio, achieving up to 32.1% lower perplexity and up to 23.3% higher few-shot reasoning accuracy.

Limitations

While our method demonstrates strong performance on the 7B-scale model, we have not yet evaluated it on larger models owing to limited GPU resources. Our experiments are confined to the LLaMA which serves as a representative architecture, but investigating other architectures would further validate the generality of our approach.

Ethics Statement

We use only publicly available datasets, including SlimPajama, C4, WikiText2, PIQA, ARC-Easy, WinoGrande, and LAMBADA, following their respective licenses. SlimPajama consists primarily of English text but contains a small portion of non-English data, which we retain as is without separate analysis. All models are fine-tuned and evaluated under the official licenses of TinyLLaMA and LLaMA-3, and no private or user-generated data are used. We did not observe or introduce any additional ethical or societal risks beyond those already acknowledged in the datasets and base models.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. In *NeurIPS*.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. 2025. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. In *ICLR*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *ICLR*.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *EMNLP*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. In *ICML*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *ICLR*.

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

Hyojin Jeon, Seungcheol Park, Jin-Gee Kim, and U Kang. 2023. Pet: Parameter-efficient knowledge distillation on transformer. *PLOS ONE*, 18(7):1–21.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Minjun Kim, Jongjin Kim, and U Kang. 2025. SynQ: Accurate zero-shot quantization by synthesis-aware fine-tuning. In *ICLR*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. 2019. Similarity of neural network representations revisited. In *ICML*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Chien-Yu Lin, Anish Prabhu, Thomas Merth, Sachin Mehta, Anurag Ranjan, Maxwell Horton, and Mohammad Rastegari. 2022. Spin: an empirical evaluation on sharing parameters of isotropic networks. In *ECCV*.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4A8KV4 quantization and system co-design for efficient LLM serving. *CoRR*, abs/2405.04532.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025. Spinquant: LLM quantization with learned rotations. In *ICLR*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adámek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *ACL*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *CoRR*, abs/2312.15234.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL*.

Seungcheol Park, Jeongin Bae, Beomseok Kwon, Minjun Kim, Byeongwook Kim, Se Jung Kwon, U Kang, and Dongsoo Lee. 2025a. Unifying uniform and binary-coding quantization for accurate compression of large language models. In *ACL*.

Seungcheol Park, Hojun Choi, and U Kang. 2024. Accurate retraining-free pruning for pretrained encoder-based language models. In *ICLR*.

Seungcheol Park, Sojin Lee, Jongjin Kim, Jinsik Lee, Hyunjik Jo, and U Kang. 2025b. Accurate sublayer pruning for large language models by exploiting latency and tunability information. In *IJCAI*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

LG Research, Soyoung An, Kyunghoon Bae, Eunbi Choi, Stanley Jungkyu Choi, Yemuk Choi, Seokhee Hong, Yeonjung Hong, Junwon Hwang, Hyojin Jeon, and 1 others. 2024. Exaone 3.0 7.8 b instruction tuned language model. *arXiv preprint arXiv:2408.03541*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *ICLR*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. 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.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. SLEB: streamlining llms through redundancy verification and elimination of transformer blocks. In *ICML*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *ICLR*.

Sho Takase and Shun Kiyono. 2023. Lessons on parameter sharing across layers in transformers. In *SustaiNLP*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *ICML*.

Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. 2025. Basis sharing: Cross-layer parameter sharing for large language model compression. In *ICLR*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Canwen Xu and Julian J. McAuley. 2023. A survey on model compression and acceleration for pretrained language models. In *AAAI*.

Ge Yang, Changyi He, Jinyang Guo, Jianyu Wu, Yifu Ding, Aishan Liu, Haotong Qin, Pengliang Ji, and Xianglong Liu. 2024. Llmcbench: Benchmarking large language model compression for efficient deployment. In *NeurIPS*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *CoRR*, abs/2401.02385.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Trans. Assoc. Comput. Linguistics*, 12:1556–1577.

A Notation

We summarize the frequently used notations in Table 4.

Table 4: Frequently used notations.

Symbol	Description
$\mathcal{M}_\theta, \mathcal{M}_{\theta_c}$	Pre-trained and compressed models, resp.
$\alpha \in (0, 1]$	Compression ratio
$\mathbf{W}_l \in \mathbb{R}^{d_i \times d_o}$	Weight matrix of the l-th layer
$\mathbf{X}_\mathcal{C}$	Activations from calibration dataset $\mathcal{C}$
$\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$	SVD outputs
$f(P, \phi(\mathcal{W}))$	Rank given target number P of parameter and weight set $\mathcal{W}$
$\mathcal{D}$	Data distribution
$p_\theta(\cdot \mid x)$	Output distribution of the model with parameters θ
$h_\theta(x)$	Hidden representations
$\mathbf{W}_l^{(t)}$	Weight of module type t at layer l
$\mathcal{G}^{(t)}$	Set of partitioned groups
$S_s(\cdot,\cdot), S_f(\cdot,\cdot)$	Structural and functional similarities, resp.
$\mathrm{CKA}(\cdot,\cdot)$	Centered Kernel Alignment (CKA)
$\psi(n, t)$	Number of parameters for n weights of type t
$B_g^{(t)}$	Capacity budget of parameters for group $\mathcal{G}_g^{(t)}$
$R_g^{(t)}$	Adaptive rank for group $\mathcal{G}_g^{(t)}$
$\eta_g^{(t)}$	Inverse-similarity of weights in group $\mathcal{G}_g^{(t)}$
$\mathbf{\Lambda}_l^{(t)}$	Diagonal matrix for layer-specific scaling
$\mathcal{L}(\theta_c \mid \theta)$	Training objective of SharVeT
$\mathcal{L}_{LM}, \mathcal{L}_{KD}, \mathcal{L}_f, \mathcal{L}_{L2}$	Language modeling, distillation, feature-level consistency, and regularization losses

B Centered Kernel Alignment

Centered Kernel Alignment (CKA) (Kornblith et al., 2019) is a widely used similarity metric for comparing neural network representations. Given two activation matrices, CKA evaluates how similarly the two representations capture pairwise relationships among input examples. Formally, linear CKA is defined as

$$\mathrm{CKA}(\boldsymbol{U}_m, \boldsymbol{U}_{m'}) = \frac{\|\boldsymbol{U}_{m'}{}^\top \boldsymbol{U}_m\|_F^2}{\|\boldsymbol{U}_m{}^\top \boldsymbol{U}_m\|_F \, \|\boldsymbol{U}_{m'}{}^\top \boldsymbol{U}_{m'}\|_F}, \tag{16}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\boldsymbol{U}_m \in \mathbb{R}^{n \times p_1}$ and $\boldsymbol{U}_{m'} \in \mathbb{R}^{n \times p_2}$ are activation matrices obtained from different networks or layers given the same set of n inputs, and p_1 and p_2 denote their respective hidden dimensions. In this paper, we exploit CKA to quantify the functional similarity between modules of large language models. Details of how we incorporate CKA into our method are provided in Section 3.2.

C Performance on Other Models

We present additional performance analysis in addition to that in Table 2. We report results on the Mistral-7B-v0.1 (Jiang et al., 2023) and GPT2-XL (Radford et al., 2019) models under 50% compression ratio in Table 5. SharVeT achieves the lowest perplexity and the highest average common-

sense reasoning accuracy among all baselines. On Mistral-7B-v0.1, SharVeT reduces perplexity by up to 27.8% and improves average reasoning accuracy by up to 12.9%. On GPT2-XL, it further reduces perplexity by up to 3.2% and raises the average accuracy by up to 2.9%. The gains are larger for the bigger model, showing that SharVeT remains reliable under compression and benefits more from larger model capacity.

D Details of the Experimental Settings

All experiments are conducted on a single NVIDIA H200 GPU. The models are implemented in PyTorch (Paszke et al., 2019) and evaluated using the Language Model Evaluation Harness (Gao et al., 2023). We train all models for 2 epochs, searching the learning rate within the range of 1×10^{-4} to 1×10^{-3}. The maximum sequence length is fixed to 1024 tokens, and all experiments use the Adam optimizer. To ensure a fair comparison, all baseline models are trained under identical conditions, using the same number of epochs and the same learning rate search range. When official implementations are publicly available, we follow their default configurations for all remaining hyperparameters. All reported results are based on single-run experiments under fixed random seeds for fair and reproducible comparison. For our proposed method, we empirically search the number of clusters between one-third and one-half of the total number of layers, which yields stable convergence and strong generalization performance.

Table 5: Comparison of the proposed method with baselines on Mistral-7B-v0.1 (Jiang et al., 2023) and GPT2-XL (Radford et al., 2019) models at 50% compression ratio. The proposed method consistently achieves the lowest perplexity and the highest average commonsense reasoning accuracy across all datasets, outperforming all baselines on both models.

Model	**Method**	**Perplexity (↓)**		**Commonsense Reasoning Accuracy (↑)**				
		C4	Wiki.	PQ	ARC	WG	LB	Avg
Mistral-7B-v0.1	SEQ	28.0	25.7	0.621	0.412	0.514	0.286	0.458
	CYCLE	43.6	51.7	0.612	0.406	0.599	0.177	0.448
	CYCLE-R	48.5	62.7	0.595	0.378	0.569	0.100	0.411
	Basis Sharing	22.8	20.1	0.619	0.463	0.588	0.387	0.514
	SharVeT (proposed)	**18.0**	**14.5**	**0.656**	**0.549**	**0.606**	**0.510**	**0.580**
GPT2-XL	SEQ	34.1	33.0	0.629	0.414	0.496	**0.357**	0.474
	CYCLE	69.7	86.5	0.569	0.356	0.507	0.103	0.384
	CYCLE-R	67.7	82.7	0.584	0.358	**0.533**	0.126	0.400
	Basis Sharing	33.0	33.3	0.618	0.444	0.504	0.331	0.474
	SharVeT (proposed)	**31.4**	**32.2**	**0.633**	**0.452**	0.525	0.342	**0.488**