LAYERDECOMPOSE: Exploring weight sharing for Large Language Model Compression

Anonymous ACL submission

Abstract

Recent advances in large language model (LLM) compression have predominantly focused on pruning and low-rank factorization, leaving weight sharing—despite its success in classical neural network compression—largely unexplored. We introduce LAYERDECOMPOSE, a novel framework that reduces parameter redundancy by sharing a core weight matrix across transformer layers and augmenting each layer with lightweight, low-rank adapters. Unlike prior SVD- and pruning-based methods, our joint optimization of shared weights and residual adapters achieves a 30% model size reduction while retaining 89% of the original performance on seven standard benchmarks. Experiments on LLaMA-7B and three other 7B-parameter models demonstrate that LAYERDECOMPOSE consistently outperforms state-of-the-art baselines. These results highlight the promise of combining weight sharing with low-rank adaptation for efficient, scalable LLM deployment.

1 Introduction

Transformers underpin virtually every state-of-the-art large language model (LLM) today, delivering remarkable capabilities in tasks ranging from question answering and commonsense reasoning to code generation and dialogue. As model capacities have grown—from millions to hundreds of billions of parameters—the computational and memory demands for both training and inference have skyrocketed. Such scaling presents a formidable barrier to deploying these models in real-world settings, especially on resource-constrained hardware or at low latency. To bridge this gap, a rich body of work has explored post-training compression techniques—quantization, pruning, and low-rank factorization—that reduce model size and accelerate inference while striving to preserve performance.

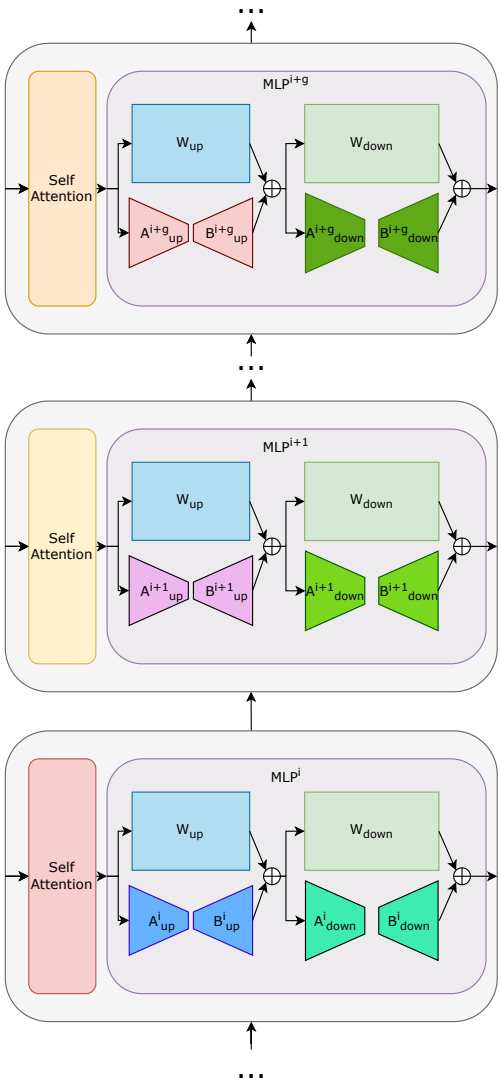

Figure 1: Schematic overview of the proposed approach. Within a group of size g weights of each type (e.g. up and down projections in MLP) are shared between transformer blocks, but have a unique low-rank residuals, which are optimized to match the original weights. This decomposition is also applied to the self attention layer, omitted for brevity.

Quantization methods (Lin et al., 2024; Frantar et al., 2022) map high-precision weights to

Compression	Method	OBQA	PIQA	HellaS.	WinoG.	ARC-e	ARC-c	MathQA	AVG	RP (%)
0 %	Uncompressed	0.44	0.79	0.76	0.70	0.73	0.46	0.27	0.59	100.0
30 %	SVD-LLM	0.20	0.65	0.37	0.59	0.48	0.26	0.22	0.40	66.7
27.2 %	LLM-Pruner	0.39	0.75	0.63	0.61	0.48	0.35	0.23	0.49	82.7
30 %	SliceGPT	0.29	0.68	0.43	0.58	0.56	0.35	0.23	0.44	74.8
30 %	LAYERDECOMPOSE (ours)	**0.39**	**0.75**	**0.67**	**0.64**	**0.62**	**0.37**	**0.24**	**0.53**	**88.9**

Table 1: Accuracy of *LLaMA-7B* after 30 % compression (27.2 % for LLM-Pruner) on seven benchmarks. **AVG** is the mean accuracy; **RP** is the average accuracy expressed as a percentage of the uncompressed baseline. Best compressed results are in bold. LAYERDECOMPOSE retains nearly 89 % of the original model's performance and surpasses all compared methods.

lower-bit representations, offering dramatic memory savings but often requiring hardware support for efficient low-bit arithmetic. Unstructured pruning (Frantar and Alistarh, 2023; Li et al., 2023) discards individual parameters based on some importance criterion, yet its resulting sparsity patterns can be difficult to exploit without specialized sparse-compute kernels. Structured pruning (Zhang et al., 2024; Wei et al., 2024) removes entire neurons or attention heads to maintain dense linear algebra, but aggressiveness can quickly degrade model quality. Low-rank adaptation approaches—exemplified by LoRA and its variants—reparameterize pretrained weights with rank-constrained updates, reducing fine-tuning cost but typically leaving the bulk of the original dense weights intact. Each of these strategies trades off ease of deployment, hardware compatibility, and final model accuracy.

In contrast to the extensive exploration of pruning and low-rank methods, weight sharing—one of the oldest and most general compression ideas in neural networks—has received surprisingly little attention for LLMs. Classic works such as the Universal Transformer (Dehghani et al., 2019) and ALBERT (Lan et al., 2020) have shown that sharing the same parameters across all layers can dramatically cut model size with only a modest hit to accuracy, yet naively tying weights across dozens of transformer blocks often yields unsatisfactory performance. A more nuanced form of weight sharing, combined with layer-specific lightweight adaptations, promises to balance redundancy elimination with expressive power, but has not been systematically studied in the context of large pretrained transformers.

In this paper, we introduce LAYERDECOMPOSE, a novel compression framework that leverages weight sharing across groups of transformer layers together with low-rank residual adapters to reduce parameter redundancy. Our core observation is that key transformer blocks express similar linear transformations up to permutation invariances. By learning a single shared "base" weight matrix for each group of layers and modeling inter-layer differences via trainable low-rank adapters, LAYERDECOMPOSE achieves up to 30% reduction in model size while retaining over 89% of original performance on seven standard benchmarks. Crucially, we jointly optimize both the shared weights and the residual factors in a two-stage procedure—closed-form initialization via truncated SVD followed by gradient-based refinement.

Contributions. Our main contributions are:

- We propose a hybrid weight-sharing and low-rank decomposition that represents a group of m corresponding linear layers with a single shared matrix W plus layer-specific residual factors $\{A_i B_i\}_{i=1}^{m}$, reducing parameters from mn^2 to $n^2 + 2mnr$ with minimal extra compute.

- We characterize and exploit permutation invariances in both MLP and self-attention modules, using assignment solvers to optimally permute and align layer weights before decomposition, thereby lowering reconstruction error.

- We validate LAYERDECOMPOSE on LLaMA-7B and three additional 7B-parameter models, showing that it consistently outperforms state-of-the-art SVD- and pruning-based baselines, retaining nearly 89% of uncompressed performance at 30% size reduction across seven diverse benchmarks.

2 Preliminaries

2.1 Low-Rank Adaptation

LoRA (Hu et al., 2022) replaces the standard linear layer

$$Y = XW + b \tag{1}$$

with

$$Y = X(W + AB) + b = XW + XAB + b, \tag{2}$$

where $\text{rank}(AB) < \text{rank}(W)$. This reparameterization permits fine-tuning only the low-rank matrices A and B, greatly reducing memory usage. Subsequent works have explored modified initializations (Meng et al., 2024), alternative reparameterizations (Liu et al., 2024b; Kopiczko et al., 2024; Lingam et al., 2024; Liu et al., 2024a), and revised optimization strategies (Hayou et al., 2024; Zhang et al., 2023).

2.2 SVD-based Model Compression

Large language models require a significant amount of memory and computational power to operate. To reduce these resource demands, various model compression techniques have been developed. One approach to reducing the parameter count is to factorize the weight matrix $W \in \mathbb{R}^{m \times n}$ into a product of two matrices with fewer total parameters, AB, where $A, B^T \in \mathbb{R}^{m \times \tilde{n}}$ and $\tilde{n} < n$, while striving to retain as much model performance as possible. A substantial body of work applies Singular Value Decomposition (SVD) to address this problem.

An early work (Winata et al., 2019) applies SVD for the LSTM cell and explores the effectiveness on different NLP tasks. FWSVD (Hsu et al., 2022) utilizes Fisher information to assign importance weights to the model parameters. However, computing the Fisher information matrix involves computationally expensive gradient calculations. To mitigate these costs, ASVD (Yuan et al., 2023) proposes an activation-aware decomposition method, which incorporates the distribution of activations into the weight decomposition process. In this approach, the scaling matrix is designed based on the distribution patterns observed across input activation channels. SVD-LLM (Wang et al., 2025) extends this idea further by whitening the input matrix to reduce its impact on SVD truncation, with proven guarantees of achieving an optimal theoretical truncation loss. Unlike previous works, (Gao et al., 2024b) developed an approach to automatically allocate various ranks to different layers using a differential hypernetwork.

2.3 Weight Sharing in Neural Networks

One fundamental application of weight sharing in language models is embedding weight tying, where the input and output embeddings share the same weight matrix (Press and Wolf, 2017; Raffel et al., 2020). Another significant aspect is weight sharing across layers in deep networks. Instead of assigning each layer its own parameters, a common set of weights is employed across multiple layers, thereby reducing redundancy and lowering the overall parameter count.

This concept was initially explored in the Universal Transformer (Dehghani et al., 2019), which introduced a recurrent inductive bias into the Transformer architecture by reusing the same layer weights at every depth. ALBERT (Lan et al., 2020) further demonstrated that, with appropriate hyperparameters tuning, full weight sharing in BERT (Devlin et al., 2019) results in only a minor reduction in accuracy while achieving faster training, enhanced memory efficiency, and improved regularization.

More recent work has investigated weight sharing strategies tailored for resource-constrained environments. For example, Subformer (Reid et al., 2021) and MobileLLM (Liu et al., 2024c) explored various methods for sharing transformer blocks to optimize performance on mobile devices. Similarly, Residualformer (Xie et al., 2023) employed LoRA reparameterization with shared base weights for training speech recognition models from scratch, in contrast to our focus on compressing existing pretrained models.

3 Method

3.1 Layer decomposition

Transformers consist of a stack of identical layers, each containing self-attention and MLP submodules. Both submodules are composed of linear transformations whose parameters are stored in weight matrices.

Figure 1 illustrates our approach. Let G be a set of m corresponding linear layers (for example, the "up" projections of the MLP in layers 17 through 23). For each layer $i \in G$, the original computation is

$$Y = XW_i + b_i.$$

We replace this with a shared base weight W plus a low-rank residual for each layer:

$$Y \;=\; XW \;+\; XA_iB_i \;+\; b_i, \quad i \in G.$$

Omitting biases for simplicity, if $W \in \mathbb{R}^{n\times n}$, $|G| = m$, and each $A_i, B_i^T \in \mathbb{R}^{n\times r}$ with $r < n$, then the total parameters drop from $m\,n^2$ to

$$n^2 \;+\; m \cdot 2nr,$$

at the cost of a small extra compute for the adapters.

To initialize $W, \{A_i, B_i\}$, we minimize the Frobenius-norm reconstruction loss

$$\begin{aligned}\mathcal{L}(W, A, B) &= \sum_{i\in G} \big\|W_i - (W + A_iB_i)\big\|_F \quad (3)\\ &= \sum_{i\in G} \big\|(W_i - W) - A_iB_i\big\|_F.\end{aligned}$$

This loss can be viewed as seeking a rank-r approximation of each difference $W_i - W$. Hence, by the Eckart–Young–Mirsky theorem (Eckart and Young, 1936), for a fixed base W the optimal low-rank factors (A_i, B_i) are given by the truncated SVD of $(W_i - W)$. Conversely, when $\{A_i, B_i\}$ are held fixed, the optimal shared weight is simply the element-wise mean

$$W \;=\; \frac{1}{|G|} \sum_{i\in G} \big(W_i - A_iB_i\big).$$

After initializing via these two closed-form updates, we perform a final joint refinement of W and all $\{A_i, B_i\}$ using Adam (Diederik, 2014). The full procedure is outlined in Algorithm 1.

Algorithm 1 Alternating Shared W Optimization

Require: Weight group $\{W_i\}_{i\in G}$, rank r, alternation steps T, Adam steps T_{adam}
Ensure: Optimal shared weight W, low-rank factors $\{A_i, B_i\}_{i\in G}$

```
W ← (1/m) Σ_{i∈G} W_i                          ▷ W initialization
for each i ∈ G do                              ▷ A_i, B_i initialization
    (A_i, B_i) ← TruncSVD(W_i − W, r)
end for
for t = 1 to T do                              ▷ Alternating optimization
    W ← (1/m) Σ_{i∈G} (W_i − A_i B_i)
    for each i ∈ G do
        (A_i, B_i) ← TruncSVD(W_i − W, r)
    end for
end for
for t = 1 to T_adam do                         ▷ Adam Optimization
    Compute L(W, A, B) as in Eq. 3
    Update W, {A_i, B_i} via Adam
end for
```

3.2 How to Choose Groups?

To apply our decomposition, we must partition the L transformer layers into groups that will share the same base weight matrix. A natural baseline is to form consecutive groups of fixed size, but ideally we would group layers whose weights are most alike so as to minimize the reconstruction loss in Eq. 3.

We first measured pairwise Frobenius distances

$$d(L_i, L_j) \;=\; \|W_i - W_j\|_F$$

between corresponding weight matrices across layers. Figure 2 shows both the distance matrix and its histogram for the MLP up-projection weights of LLaMA. The heat-map reveals no clear block structure, and the histogram is tightly centered around its mean, indicating that all layers are roughly equally dissimilar. Clustering on these distances failed to produce consistent, meaningful groups.

Given the limitations of a purely weight-space metric, we next define a functional similarity measure:

$$\rho(L_i, L_j) \;=\; d\big(L_i(X_i),\, L_j(X_i)\big), \qquad (4)$$

where L_i and L_j denote the ith and jth layers, X_i is the actual input to L_i collected during a forward pass, and $d(x, y) = \|x - y\|_2^2$. This quantity captures how closely layer j can mimic layer i on its native inputs. Note that ρ is not symmetric in general.

Figure 3 shows the resulting similarity matrix for the MLP blocks in LLaMA. We observe a banded structure along the diagonal, indicating that adjacent layers produce more similar outputs. Furthermore, the first half of the network exhibits larger approximation errors than the second half. Motivated by these observations and for implementation simplicity, we form groups of consecutive layers for weight sharing in this work.

3.3 Optimal Parameter Budget Allocation

Even with consecutive grouping, our framework has three key hyperparameters under a fixed parameter budget: the span of affected layers, the size of each group, and the adapter rank. With a fixed parameter budget, one can either apply strong

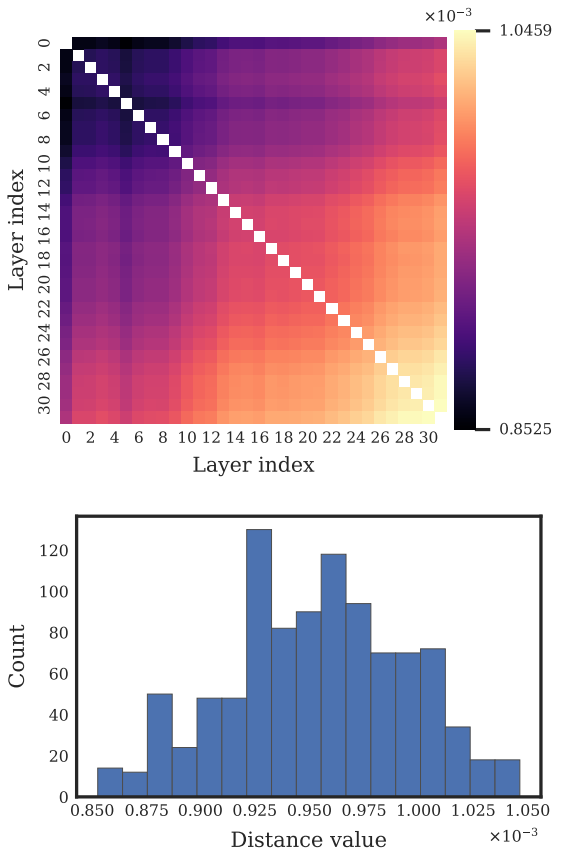

Figure 2: Pairwise Frobenius distance matrix and histogram for the MLP up-projection weights in LLaMA. The lack of visible structure and the narrow distribution of distances suggest that fixed-size consecutive grouping is as reasonable as any clustering based on these metrics.

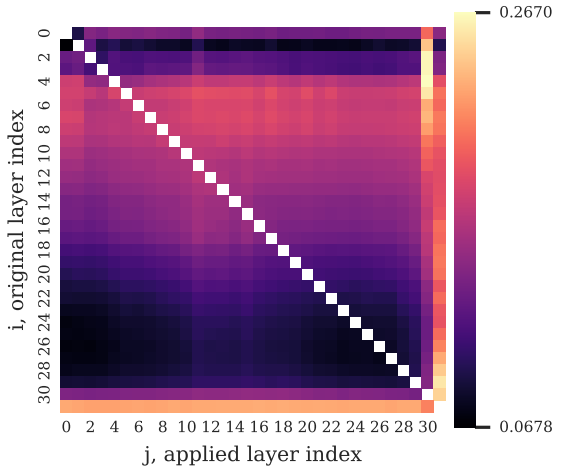

Figure 3: Functional similarity matrix for MLP blocks in the LLaMA model, where each entry (i, j) is given by $\rho(L_i, L_j)$. Lower values along the diagonal indicate that nearby layers are more functionally similar.

compression to a few layers or perform milder compression across a larger number of layers.

To identify which regions of the network tolerate compression best, we first compressed a single block of ten consecutive layers at a time. Figure 4 shows that compressing the earliest or latest layers incurs large perplexity increases, whereas targeting the final third of layers yields the smallest degradation. These findings corroborate prior analyses of layer sensitivity (Gromov et al., 2025; Men et al., 2024; Wang et al., 2024).

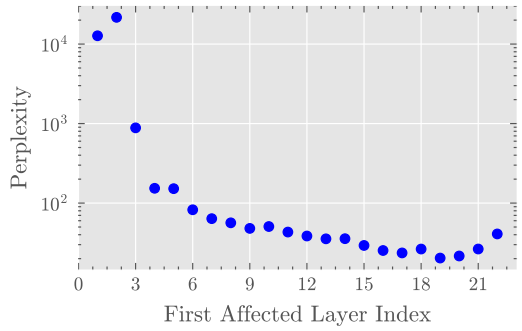

Figure 4: Perplexity after decomposition (before healing) for a single group of 10 layers. Each x-value denotes the index of the first layer in the compressed block (e.g., 19 covers layers 19–29).

We then performed an exhaustive search over hyperparameters under a fixed budget. For a 30% compression of LLaMA-7B (4.7 B parameters), we fixed the final layer at index 30 and varied the starting layer and group size. Given each choice of start and group, we computed the adapter rank to exactly match the remaining parameter budget (see section 3.4 for more details).

Figure 5 presents the results of this exhaustive search. The performance varies substantially across configurations. One clear trend is that applying milder compression over a wider range of layers—using larger residual ranks—yields better perplexity than more aggressive compression on a smaller subset of layers.

3.4 Rank Computation

During the exhaustive search with fixed hyperparameters, the adapter rank could be computed in a single way as follows:

$$r = \frac{P_B - \left(P_{nb} + (L_T - L_A) \cdot P_l + G \cdot P_l\right)}{L_A \cdot (d_I + d_O)}, \tag{5}$$

where:

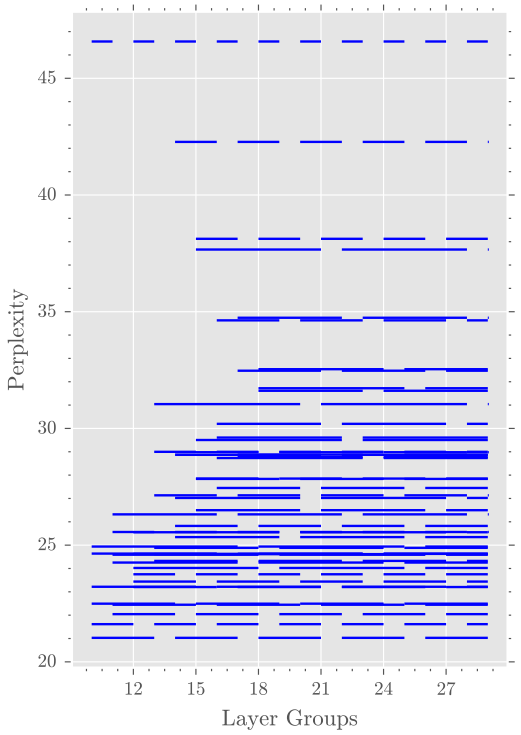

Figure 5: Perplexity after decomposition for various compression configurations. Each line corresponds to a different group of layers. Slight compression over a broader layer span—with higher adapter rank—yields better perplexity than aggressive compression on a smaller subset.

- r is the adapter rank,
- P_B is the fixed total parameter budget,
- P_{nb} denotes the number of parameters that are not subject to compression (e.g. embedding and LM head layers),
- L_T is the total number of layers in the model,
- L_A is the number of affected layers, i.e. selected for compression,
- P_l represents the number of parameters in one layer,
- G is the number of groups within the affected layers,
- d_I and d_O are the sum of input and output dimensions of a layer, respectively.

3.5 Transformer Permutation Invariance

Permutation invariance in transformer modules allows multiple weight configurations to produce identical outputs by appropriately reordering intermediate dimensions.

Multi-Layer Perceptron A gated transformer MLP block computes

$$y \;=\; W_d\,\big(\sigma(W_g\,x)\;\odot\;W_u\,x\big),$$

where σ is applied element-wise. By permuting the intermediate hidden dimensions via an $n \times n$ permutation matrix P (and its inverse P^T), one can rearrange the rows of W_g and W_u without affecting the final output. Concretely, we exploit $P^T P = I$ as follows:

$$\begin{aligned} y &= W_d\,P^T\Big(\sigma\big(P\,W_g\,x\big)\;\odot\;P\,W_u\,x\Big) \\ &= W_d\,P^T\Big(P\,\sigma(W_g\,x)\;\odot\;W_u\,x\Big) \\ &= W_d\,\big(\sigma(W_g\,x)\;\odot\;W_u\,x\big), \end{aligned}$$

since $P^T(P\,\sigma(W_g x)) = \sigma(W_g x)$. Hence one can absorb P into the weights by defining

$$W_u' = P\,W_u, \quad W_g' = P\,W_g, \quad W_d' = W_d\,P^T,$$

yielding the same output y. Because there are $n!$ permutation matrices of size n, this gives $n!$ equivalent MLP configurations.

Query and Key Projections In self-attention, the query and key projections satisfy a similar invariance: permuting their shared intermediate dimensions does not alter the attention scores. Recall

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

and

$$\mathrm{Attn}(Q,K,V) = \mathrm{softmax}\big(QK^T\big)\,V.$$

Inserting a permutation P with $P^T P = I$ into the score computation gives

$$\begin{aligned} \mathrm{softmax}(QK^T) &= \mathrm{softmax}\big((XW_Q)(XW_K)^T\big) \\ &= \mathrm{softmax}\big(XW_Q\,P\,P^T\,W_K^T\,X^T\big). \end{aligned}$$

so that defining

$$W_Q' = W_Q\,P, \quad W_K' = P^T\,W_K$$

leaves $\mathrm{softmax}(QK^T)$ unchanged.

Value and Output projections Previously, we showed that permuting the dimensions of Q and K does not alter the attention score matrix. We now demonstrate a similar invariance for the Value and subsequent Output projections.

In multi-head self-attention, for each head $i = 1, \ldots, h$ we define

$$\begin{aligned} V^{(i)} &= X\, W_V^{(i)}, \\ H^{(i)} &= \mathrm{softmax}\big(Q^{(i)}(K^{(i)})^T\big)\, V^{(i)}, \end{aligned} \tag{6}$$

where $W_V^{(i)} \in \mathbb{R}^{d \times d_v}$ is the value-projection for head i. We then concatenate the head outputs and apply the final output projection:

$$Y = \big[\, H^{(1)}, \ldots, H^{(h)} \big]\, W_O, \quad W_O \in \mathbb{R}^{(h\, d_v) \times d}.$$

Any permutation of the h head-blocks and of the d_v channels within each head can be absorbed into the weight matrices $\{W_V^{(i)}\}$ and W_O. Concretely, let

$$\begin{aligned} P_{\text{blocks}} &\in \{0,1\}^{(h\, d_v) \times (h\, d_v)}, \\ P_{\text{intra}}^{(i)} &\in \{0,1\}^{d_v \times d_v} \quad (i = 1, \ldots, h), \end{aligned} \tag{7}$$

and form

$$P = P_{\text{blocks}} \Big(\bigoplus_{i=1}^{h} P_{\text{intra}}^{(i)} \Big),$$

where $\bigoplus$ denotes the block-diagonal direct sum (so the ith diagonal block is $P_{\text{intra}}^{(i)}$). If we collect all the per-head projections into

$$\begin{aligned} W_V &= \big[\, W_V^{(1)}, \ldots, W_V^{(h)} \big] \in \mathbb{R}^{d \times (h\, d_v)}, \\ H &= \big[H^{(1)}, \ldots, H^{(h)} \big], \end{aligned} \tag{8}$$

then one checks

$$\begin{aligned} Y = H\, W_O &= \mathrm{softmax}(QK^T)\, V\, W_O \\ &= \mathrm{softmax}(QK^T)\, (X W_V\, P\, P^T)\, W_O \\ &= \mathrm{softmax}(QK^T)\, (X W_V')\, W_O' \end{aligned} \tag{9}$$

with

$$W_V' := W_V\, P, \qquad W_O' := P^T\, W_O,$$

and $P^T P = I$ guarantees the same output. Since there are $h!$ ways to permute the head-blocks and $(d_v!)^h$ ways to permute channels within each head, the total number of distinct permutations of (W_V, W_O) yielding identical outputs is $h! \times (d_v!)^h$.

3.5.1 Finding Optimal Permutations

We leverage these permutation symmetries to reorder layer weights so that they align more closely within each group. Formally, for two weight matrices W_i and W_j, we seek

$$P \;=\; \arg\min_{P \in S_n} \big\| W_i - P\, W_j \big\|_F,$$

where S_n is the set of $n \times n$ permutation matrices. Here, P minimizes the difference between an anchor weight and another weight in the group.

We perform this procedure separately for three components: the MLP block, the Query–Key (QK) projections, and the Value–Output (VO) projections. Note that for QK we restrict intra-head permutations to the identity ($P_{\text{intra}} = I$)—permuting channels would conflict with RoPE embeddings (Su et al., 2021)—and only reorder entire heads.

MLP block Compute a cost matrix $D \in \mathbb{R}^{n \times n}$ whose (i, j) entry is

$$\begin{aligned} D_{ij} = &\big\| W_u^A[i,:] - W_u^B[j,:] \big\|_2^2 + \\ &\big\| W_g^A[i,:] - W_g^B[j,:] \big\|_2^2 + \\ &\big\| W_d^A[:,i] - W_d^B[:,j] \big\|_2^2. \end{aligned} \tag{10}$$

where W_u, W_g, and W_d denote the up-projection, gate, and down-projection weight matrices. Each D_{ij} aggregates via sum the squared ℓ_2 distances between row i of one layer and row j of another for W_u and W_g, plus the column differences in W_d. We formulate the search for the optimal permutation as a linear sum assignment problem (LSAP) (Burkard and Cela, 1999) and solve it with an efficient solver (Crouse, 2016) to obtain the optimal permutation P.

QK and VO projections Here the permutation must respect the block structure of h attention heads, so channels cannot be exchanged across heads. We use a two-stage approach:

1. *Intra-head alignment:* For each pair of corresponding heads, find the best channel permutation $P_{\text{intra}}^{(i)}$ by solving an LSAP on the per-head weight differences.

2. *Inter-head alignment:* Compute aggregated costs between entire heads using the intra-head-aligned weights, then solve a second LSAP to determine the head-reordering permutation P_{blocks}.

Finally, we combine these into a block-diagonal permutation

$$P \;=\; P_{\text{blocks}} \Big(\bigoplus_{i=1}^{h} P_{\text{intra}}^{(i)} \Big),$$

which aligns both head order and internal channels while preserving the attention outputs.

3.6 Healing with Distillation

Because our weight-sharing and low-rank decomposition substantially alter the original parameters, a dedicated "healing" step is required to recover performance. Following Muralidharan et al. (2024), we apply both logit-level and hidden-state distillation (Hinton et al., 2015; Sanh et al., 2019) to encourage the compressed model to mimic the teacher's behavior while reducing reliance on the specific healing dataset.

Concretely, we augment the standard language modeling loss $\mathcal{L}_{LM}$ with two distillation terms:

$$\mathcal{L} = \mathcal{L}_{LM} + \alpha\,\mathrm{KL}\big(p \,\|\, p_{\text{teacher}}\big) + \beta\,\mathrm{MSE}\big(h,\, h_{\text{teacher}}\big), \tag{11}$$

where - $\mathrm{KL}(p \,\|\, p_{\text{teacher}})$ is the Kullback–Leibler divergence between the student's output distribution p and the teacher's distribution p_{teacher}, - $\mathrm{MSE}(h,\, h_{\text{teacher}})$ is the mean squared error between their hidden-state activations, and α, β weight these distillation terms relative to $\mathcal{L}_{LM}$.

4 Experiments

We now evaluate the effectiveness of LayerDecompose. Our first set of experiments compares it with three state-of-the-art compression baselines applied to the *LLaMA-7B* (Touvron et al., 2023) model at a fixed compression ratio of 30 %. **SVD-LLM** Wang et al. (2025) compresses weights by applying singular-value decomposition with an injected whitening transformation matrix and has so far yielded the strongest results among SVD-based schemes. **LLM-Pruner** (Ma et al., 2023) performs structural pruning, discarding non-critical coupled components on the basis of gradient importance. **SliceGPT** (Ashkboos et al., 2024) multiplies weight matrices by orthogonal projection matrices before eliminating less important rows and columns; the authors exploit a form of transformer invariance that differs from the permutation invariance introduced in our work. All baselines, like our method, perform a post-compression healing step to recover accuracy. Hyperparameters and other healing details for LayerDecompose are provided in Appendix A.

We retain the original evaluation protocol of *LM-Evaluation-Harness* (Gao et al., 2024a) and report accuracy on seven benchmarks covering question-answering and commonsense reasoning: OpenBookQA (OBQA) (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and MathQA (Valentino et al., 2024). In addition to absolute accuracy, we compute *Relative Performance* (RP), the ratio of a compressed model's average accuracy to that of the uncompressed model.

Table 1 demonstrates that LayerDecompose achieves the highest average accuracy and relative performance, retaining approximately 89% of the uncompressed model's quality while matching or surpassing each baseline on all individual benchmarks. These findings underscore the effectiveness of weight sharing in compressing large language models.

To verify the generality of our method, we also applied LayerDecompose to three other 7B-parameter models—*Qwen-7B* (Bai et al., 2023), *DeepSeek-7B* (DeepSeek-AI et al., 2024), and *OLMo-7B* (Groeneveld et al., 2024). The resulting average accuracies and relative performance scores appear in Table 2.

Model	AVG	RP (%)
Qwen-7B	0.50	83.0
DeepSeek-7B	0.52	88.2
OLMo-7B	0.48	84.0

Table 2: Average accuracy (AVG) and relative performance (RP) of LayerDecompose on additional 7B models.

5 Conclusion

We introduced LayerDecompose, a compression framework that represents blocks of consecutive transformer weights with a single shared matrix plus lightweight, layer-specific adapters. By formalizing permutation invariances in both MLP and self-attention components, we revealed a vast family of equivalent weight configurations and leveraged these symmetries to further reduce redundancy. Empirical results on LLaMA-7B and three additional 7B-parameter models show that our weight-sharing approach matches or exceeds state-of-the-art compression baselines across diverse benchmarks. We believe that our findings will inspire further exploration of weight sharing as a systematic strategy for efficient LLM compression and scaling.

Limitations

Although we focus on language models, LAYERDECOMPOSE is not intrinsically tied to the text modality. Transformer architectures now underpin models for images, audio, and graphs; extending our weight-sharing scheme to those domains remains an open direction. Because any linear layer can, in principle, share parameters and have low-rank residuals, similar decompositions may prove useful beyond transformers as well.

In the present work we distribute the adapter parameter budget uniformly across all layer types and blocks. This heuristic simplifies implementation but is unlikely to be optimal. A data-driven or sensitivity-based allocation strategy could improve the accuracy–compression trade-off.

Our experiments group adjacent layers. While early results are encouraging, a more principled grouping criterion might unlock further gains.

Potential Risks While LAYERDECOMPOSE boosts efficiency, it could also lower the barrier to misuse—enabling more convincing disinformation or fake profiles—and may amplify existing biases, further marginalizing underrepresented languages or groups. Weight-sharing might also expose new vectors for model-stealing or adversarial attacks, and its upfront symmetry analysis carries non-trivial compute costs, highlighting the need for energy-efficient methods and gated releases to guard against dual-use harms.

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *arXiv preprint arXiv: 2309.16609*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Rainer E Burkard and Eranda Cela. 1999. Linear assignment problems and extensions. In *Handbook of combinatorial optimization: Supplement volume A*, pages 75–149. Springer.

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:1803.05457v1*.

David F. Crouse. 2016. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696.

DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, and 69 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv: 2401.02954*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kingma Diederik. 2014. Adam: A method for stochastic optimization. *(No Title)*.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv: 2210.17323*.

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. 2024a. The language model evaluation harness.

Shangqian Gao, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2024b. Adaptive rank selections for low-rank approximation of language models. In *Proceedings of the 2024 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 227–241, Mexico City, Mexico. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2025. The unreasonable ineffectiveness of the deeper layers. *Preprint*, arXiv:2403.17887.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv: 2402.12354*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv: 1503.02531*.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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 *International Conference on Learning Representations*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, and Sanjiv Kumar. 2023. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. Svft: Parameter-efficient fine-tuning with singular vectors. *arXiv preprint arXiv: 2405.19597*.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. 2024a. Eora: Training-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv preprint arXiv: 2410.21271*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Zechun Liu, Changsheng Zhao, Forrest N. Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024c. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *International Conference on Machine Learning*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv: 2403.03853*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz,

and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Neurips*.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *NEUROCOMPUTING*.

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. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv: 2302.13971*.

Marco Valentino, Deborah Ferreira, Mokanarangan Thayaparan, and Andre Freitas, editors. 2024. *Proceedings of the 2nd Workshop on Mathematical Natural Language Processing @ LREC-COLING 2024*. ELRA and ICCL, Torino, Italia.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv: 2402.09748*.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *Preprint*, arXiv:2403.07378.

Jiateng Wei, Quan Lu, Ning Jiang, Siqi Li, Jingyang Xiang, Jun Chen, and Yong Liu. 2024. Structured optimal brain pruning for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13991–14007, Miami, Florida, USA. Association for Computational Linguistics.

Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J Barezi, and Pascale Fung. 2019. On the effectiveness of low-rank matrix factorization for lstm model compression. *arXiv preprint arXiv:1908.09982*.

Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. 2023. Residual: Transformer with dual residual connections. *arXiv preprint arXiv: 2304.14802*.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Honghe Zhang, XiaolongShi XiaolongShi, Jingwei Sun, and Guangzhong Sun. 2024. Structured pruning for large language models using coupled components elimination and minor fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1–12, Mexico City, Mexico. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv: 2303.10512*.

A Healing Details and Hyperparameters

An exhaustive search over compression configurations for the LLaMA-7B model on a subset of the development dataset revealed that the optimal perplexity is achieved using the groups `[[10, 11, 12, 13], [14, 15, 16, 17], [18, 19, 20, 21], [22, 23, 24, 25], [26, 27, 28, 29]]` with a residual rank of $r = 649$.

We apply healing on the C4 train corpus (Raffel et al., 2020) for 100,000 iterations with an effective batch size of 8, truncating all sequences to a maximum length of 1,024 tokens. The weights for the distillation loss (Eq. 11) are set to $\alpha = 0.05$ and $\beta = 0.2$.

For optimisation we use Adam (Diederik, 2014) with learning rate `5e-5`, cosine annealing schedule and weight decay 0.01.

Experiments were conducted using 2 NVIDIA A100 GPUs and took approximately 14 hours including evaluation.

B Extra Weight Distances

Pairwise Frobenius distances for all layer types in a transformer are depicted on the Fig. 6. Later layers tend to be less similar.

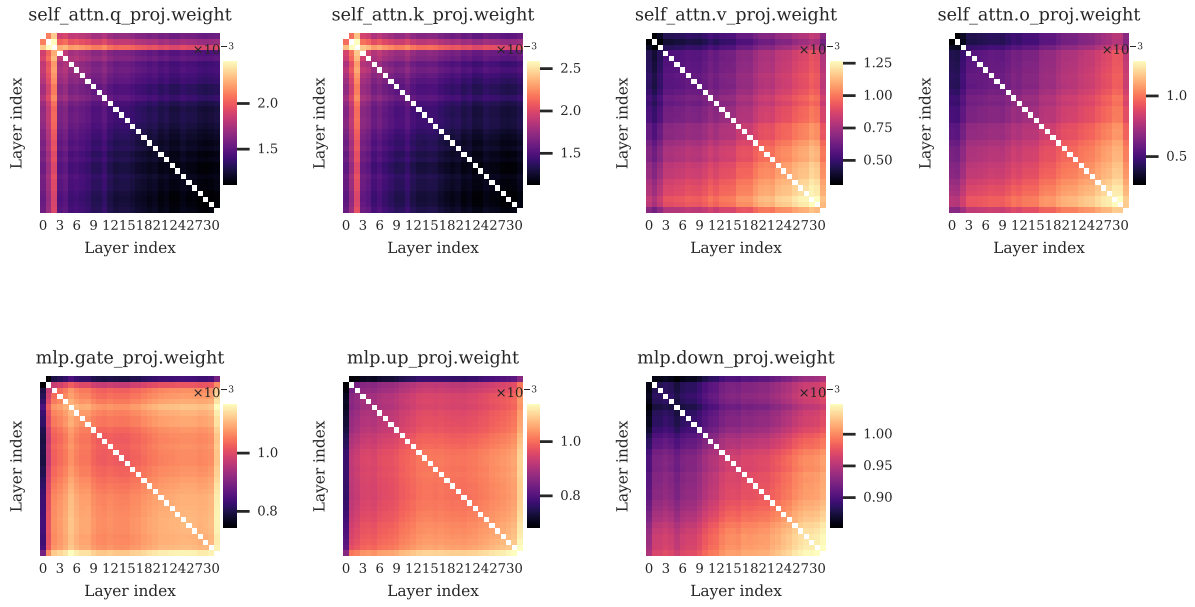

Figure 6: Heat-maps of pair-wise distances for each of the 7 layer groups. Each subplot shows the distance matrix for one layer type.