# Pruning via Merging: Compressing LLMs via Manifold Alignment Based Layer Merging

**Anonymous ACL submission**

## Abstract

While large language models (LLMs) excel in many domains, their complexity and scale challenge deployment in resource-limited environments. Current compression techniques, such as parameter pruning, often fail to effectively utilize the knowledge from pruned parameters. To address these challenges, we propose Manifold-Based Knowledge Alignment and Layer Merging Compression (MKA), a novel approach that uses manifold learning and the Normalized Pairwise Information Bottleneck (NPIB) measure to merge similar layers, reducing model size while preserving essential performance. We evaluate MKA on multiple benchmark datasets and various LLMs. Our findings show that MKA not only preserves model performance but also achieves substantial compression ratios, outperforming traditional pruning methods. Moreover, when coupled with quantization, MKA delivers even greater compression. Specifically, on the MMLU dataset using the Llama3-8B model, MKA achieves a compression ratio of 43.75% with a minimal performance decrease of only 2.82%. The proposed MKA method offers a resource-efficient and performance-preserving model compression technique for LLMs.

## 1 Introduction

Large Language Models (LLMs), such as GPT-4 (OpenAI et al., 2024), Llama-3[1], Llama-2 (Touvron et al., 2023) and Mistral (Jiang et al., 2024), have demonstrated remarkable proficiency in language understanding and generation. These models, with billions of parameters trained on trillions of tokens, can handle complex tasks and exhibit emergent abilities (Brown et al., 2020; Chowdhery et al., 2023). While these models have achieved unprecedented success, their growing complexity and scale have brought to the fore significant challenges in terms of computational resources, memory requirements, and energy consumption (Bender et al., 2021; Bommasani et al., 2021), raising concerns about their sustainability.

To mitigate these challenges, researchers have developed various model compression techniques in LLM to reduce its parameter size while preserving performance (Cheng et al., 2017; Deng et al., 2020; Ganesh et al., 2021; Zhu et al., 2023). These techniques can be roughly categorized into two main mainstreams (Men et al., 2024): quantization (Gholami et al., 2021; Li et al., 2024; Dettmers et al., 2022; Gong et al., 2024; Li et al., 2024) and pruning (LeCun et al., 1989; Han et al., 2016; Gupta and Agrawal, 2022; Ma et al., 2023a). Quantization based methods aid in the reduction of the memory consumption of weights, activations, and KV caches by using the low-precision values with fewer bits instead of the high-precision values. However, the acceleration benefits of quantization are seriously dependent on hardware support (Tao et al., 2023) and sometimes require additional fine-tuning to maintain performance (Dettmers et al., 2023; Men et al., 2024). Compared to quantization, pruning, especially structural pruning (Li et al., 2017), eliminates redundant LLM's parameters to decrease the overall parameter count, and can be applied directly to a trained LLM without retraining and is generally more hardware-friendly than quantization approaches. While effective, pruning usually risks losing valuable model structures and determining how to prune the LLM with minimal disruption to the origin remains an unsolved problem (Ma et al., 2023b).

To tackle this issue head-on, we delve into the realm of model merging (Wortsman et al., 2022), a powerful technique that seamlessly weaves together the strengths and knowledge of multiple models, creating a robust and efficient aggregation. This technique, through averaging the weights of multiple models with the same architecture, can

[1] https://github.com/meta-llama/llama3

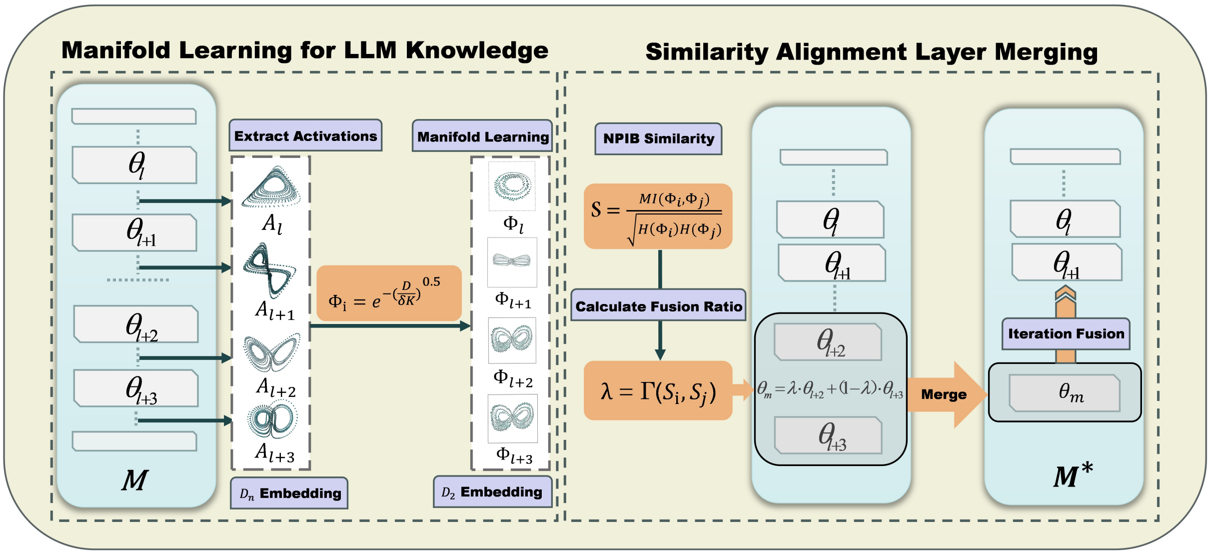


Figure 1: Manifold-Based Knowledge Alignment and Layer Merging (MKA) framework consists of two main components: (1) The left side illustrates manifold learning for LLM knowledge extraction, where layer activations are transformed into low-dimensional manifolds using the Diffusion Kernel algorithm. (2) The right side depicts the similarity-based layer merging process, employing the NPIB metric to identify layers with aligned knowledge.

retain essential features without significant additional resources (Liu et al., 2024; Wan et al., 2024). Furthermore, by offsetting the biases and errors of individual models, model merging often leads to greatly improved performance (Li et al., 2023). Additional, the number of models in the merging process can be gradually and naturally reduced. However, such a useful technology are limited to merging between models currently, and few studies pay attention on merging the same internal structures within a model.

This raises the question of whether model compression could be achieved by reducing the total number of layers through the progressive aggregation of knowledge between layers. To answer this question, we introduce Manifold-Based Knowledge Alignment and Layer Merging Compression (MKA) in this paper. MKA combines manifold learning and layer merging to preserve essential information while significantly reducing LLM parameter size. As illustrated in Figure 1, our method mainly comprises two primary components:

***Manifold Learning for LLM Knowledge:*** We employ manifold learning techniques to align knowledge across layers by extracting layer activations from a LLM and applying the Diffusion Kernel algorithm (Tenenbaum et al., 2000) to learn low-dimensional manifold representations. This approach captures the nonlinear structure in the activation and achieves dimensionality reduction while preserving important activation features, enabling more effective comparison of knowledge patterns across different layers.

***Similarity Alignment Layer Merging:*** Following manifold learning, we use the Normalized Pairwise Information Bottleneck (NPIB) measure (Tishby et al., 2000) to construct a similarity matrix that quantifies the similarity between layers by maximizing their mutual information while considering the entropy of each layer. Based on this similarity matrix, we select the most similar layer pairs for merging.

To rigorously validate the effectiveness of MKA, we conduct extensive empirical evaluations on a diverse array of benchmark datasets, like MMLU and PIQA, and a wide range of state-of-the-art large language models, including Llama-3 series with 8B and 70B parameters, Llama-2 series with 7B and 13B parameters, and Mixtral-7B. Our experimental results indicate that MKA can maintain good performance while achieving a significant compression ratio, outperforming existing pruning methods and achieving even greater compression when combined with quantization. For example, on the MMLU dataset with Llama3-8B, MKA can achieve a compression ratio of 43.75% with only a 2.82% performance drop.

In summary, the main contributions of this paper are as follows:

- We introduce MKA, an innovative model compression technique that leverages manifold learning to align and integrate knowledge across layers, achieving significant reductions in model size while preserving performance.

- We develop a manifold-based knowledge alignment approach, utilizing the Diffusion Kernel and Normalized Pairwise Information Bottleneck (NPIB) to effectively capture and align similarities between layers in the parameter space.
- We validate the efficacy of MKA through comprehensive experiments on multiple benchmark datasets and a variety of large language models, demonstrating its capability to achieve substantial compression without compromising model performance.

## 2 Manifold-Based Knowledge Alignment and Layer Merging

Our MKA method relies on the redundancy present in the latter layers of post-training LLMs (Gromov et al., 2024). By merging layers with high input-output similarity from back to front, we maintain the model's performance while reducing its size. In this section, we first describe the extraction and dimensionality reduction processes for the intermediate states, as high-dimensional intermediate states are challenging to analyze. Then, we propose our layer merging method based on similarity alignment, which aims to maintain performance by aligning intermediate states through merging techniques.

### 2.1 Manifold Learning for LLM Knowledge

To effectively align knowledge across LLM's layers, MKA employs manifold learning techniques that can capture the intricate nonlinear dependencies within the LLM's internal structure. This approach allows us to compare and align layer activations in a meaningful way, preserving essential information while reducing model complexity.

The process begins with the extraction of layer activations $H^l$ from a LLM on the dataset $D$. These activations represent the outputs of each layer given a set of input samples, encapsulating the knowledge learned at different stages. To transform these high-dimensional activations into a lower-dimensional space that preserves their essential features and geometric structure, we apply the Diffusion Kernel algorithm (Coifman and Lafon, 2006). Here are the key steps involved in this process:

**Extracting Layer Activations:** For each layer $l$, we extract the activations $H^l$ given input samples. These activations $\mathbf{H}^l$ are computed using the following equation:

$$\mathbf{H}^l = \text{LayerNorm}\left(\mathbf{H}^{l-1} + \text{MultiHead}\left(\mathbf{H}^{l-1}\right)\right) + \text{FeedForward}\left(\mathbf{H}^{l-1}\right) \tag{1}$$

**Constructing the Pairwise Distance Matrix:** Next, we calculate the pairwise Euclidean distance matrix $D$ for the activations $H^l$. This matrix captures the distances between all pairs of activations, serving as the basis for the manifold learning process.

**Applying the Diffusion Kernel:** We apply the Diffusion Kernel to transform the distance matrix $D$ into low-dimensional manifold representations $\Phi_i$, capturing the intrinsic geometric structure of the data. The kernel function smooths the data, emphasizing the intrinsic geometric structure:

$$\mathbf{E} = \text{EigVectors}_d\left(\text{Diag}\left(\sum_j e^{-\left(\frac{\|\mathbf{H}_i-\mathbf{H}_j\|^2}{\sigma K}\right)^{0.5}}\right) - e^{-\left(\frac{\|\mathbf{H}_i-\mathbf{H}_j\|^2}{\sigma K}\right)^{0.5}}\right) \tag{2}$$

where $\sigma_K$ is the kernel bandwidth parameter, and $\text{EigVectors}_d$ refers to the eigenvectors corresponding to the $d$ smallest eigenvalues of the Laplacian matrix $L$. This transformation captures the essential features and relationships within the activations, enabling effective comparisons across different layers.

### 2.2 Similarity-based Layer Merging

Building upon the manifold learning representations, MKA employs a similarity-based layer merging approach to identify and fuse layers with highly aligned knowledge. By quantifying the similarity between layers using the Normalized Pairwise Information Bottleneck (NPIB) (Tishby et al., 2000) metric, we can determine which layers are most suitable for merging. This process allows us to reduce model size, improve inference speed, and decrease GPU memory consumption.

The layer merging process involves several key steps. First, we construct a similarity matrix using the NPIB metric to compare the knowledge patterns across layers. Next, we introduce an adaptive weight allocation function to determine the optimal merging ratio for each pair of layers, ensuring that the merged layer retains the most critical features.

**Algorithm 1** Manifold-Based Knowledge Alignment and Layer Merging Compression (MKA)

**Input**: LLM $\mathcal{M}$ with Layers $L_1, L_2, \ldots, L_N$, Layer Parameters $\Theta = \theta_1, \theta_2, \ldots, \theta_N$, Dataset $\mathcal{D}$
**Output**: Compressed Model $\mathcal{M}^*$ with Aligned Knowledge
$\mathcal{H} \leftarrow$ ExtractActivations$(\mathcal{M}, \mathcal{D})$ ▷ Extract activations for each layer on dataset $\mathcal{D}$
$D \leftarrow$ ComputePairwiseDistances$(\mathcal{H})$ ▷ Compute pairwise Euclidean distance matrix of activations
$\mathbf{E} \leftarrow$ DiffusionKernel$(D, \sigma_K)$ ▷ Apply diffusion kernel for manifold learning
$\mathcal{S} \leftarrow$ ComputeNPIB$(\mathbf{E})$ ▷ Compute NPIB similarity matrix
$\Omega \leftarrow$ SortLayersBySimilarity$(\mathcal{S})$ ▷ Sort layers by similarity for merging
**while** $|\Omega| > 1$ **do**
    $(L_i, L_j) \leftarrow$ SelectTopLayerPair$(\Omega)$ ▷ Select top-ranked layer pair based on similarity
    $\lambda_m \leftarrow$ ComputeFusionRatio$(\mathcal{S}, L_i, L_j)$ ▷ Compute adaptive merging ratio
    $\theta_m \leftarrow \lambda_m \cdot \theta_i + (1 - \lambda_m) \cdot \theta_j$ ▷ Fuse layer parameters
    $L_m \leftarrow$ FuseLayer$(L_i, L_j, \theta_m)$ ▷ Create fused layer using the fused parameters
    $\mathcal{M} \leftarrow$ ReplaceLayer$(\mathcal{M}, L_i, L_j, L_m)$ ▷ Update model with fused layer
    $\Omega \leftarrow$ UpdateLayerList$(\Omega, L_i, L_j, L_m)$ ▷ Update layer list with the new fused layer
**end while**
**return** $\mathcal{M}$

Finally, we fuse the parameters of the selected layers using the weighted sum and update the model architecture accordingly.

**Constructing the Similarity Matrix:** To identify layers suitable for merging, we first construct a similarity matrix $S$ using the Normalized Pairwise Information Bottleneck (NPIB) metric. NPIB quantifies the shared information between layers while normalizing their individual entropies, providing an ideal measure for comparing knowledge patterns across layers:

$$S_{ij} = \mathrm{NPIB}(\mathbf{E}_i, \mathbf{E}_j) = \frac{\sum_{x \in \mathbf{E}_i} \sum_{y \in \mathbf{E}_j} p(x, y) \log \frac{p(x,y)}{p(x)p(y)}}{\sqrt{\sum_{x \in \mathbf{E}_i} p(x) \log p(x) \cdot \sum_{y \in \mathbf{E}_j} p(y) \log p(y)}} \tag{3}$$

where $p(x, y)$ denotes the joint probability distribution of $\mathbf{E}_i$ and $\mathbf{E}_j$, and $p(x)$ and $p(y)$ represent the marginal probability distributions of $\mathbf{E}_i$ and $\mathbf{E}_j$, respectively. This similarity matrix helps us determine which layers have the most aligned knowledge representations.

**Calculate Weight ratio:** To determine the merging ratio $\lambda_m$ for each pair of layers, we introduce the adaptive weight allocation function $\Psi$. This function dynamically adjusts the merging ratio based on the similarity differences between layers, ensuring that the merged layer retains the most critical features from each original layer:

$$\lambda_m = \Psi(\bar{s}i, \bar{s}j) = \frac{e^{\mathcal{S}ij}}{\sum k \in \Omega e^{\mathcal{S}_k}} \tag{4}$$

The adaptive weight allocation function $\Psi$ adjusts the merging weights based on the similarity difference between layers. When the similarity difference between two layers is large, $\Psi$ assigns a higher weight to the layer with higher similarity, reducing the weight of the layer with lower similarity. This mechanism ensures that the merged layer better preserves the knowledge from the more similar layer.

**Merging Layer Parameters:** Once the merging ratio $\lambda_m$ is determined, we fuse the parameters $\theta_i$ and $\theta_j$ of the selected layers using a weighted sum:

$$\widetilde{\theta}_m = \lambda_m \theta_i + (1 - \lambda_m) \theta_j \tag{5}$$

The merged layer $L_m$ is obtained through the function FuseLayer$(L_i, L_j, \widetilde{\theta}_m)$, which constructs a new layer based on the fused parameters $\widetilde{\theta}_m$. This new layer integrates the aligned knowledge from the original layers, preserving essential information while reducing redundancy.

Finally, we update the model $\mathcal{M}$ by replacing the original layers $L_i$ and $L_j$ with the newly merged layer $L_m$, utilizing the function ReplaceLayer$(\mathcal{M}, L_i, L_j, L_m)$. This step ensures that the model's architecture is updated to reflect the compression process, maintaining performance while significantly reducing model size.

## 3 Experiments

We conduct a comprehensive set of experiments to evaluate the effectiveness and generalizability of our MKA method across various domains. More-

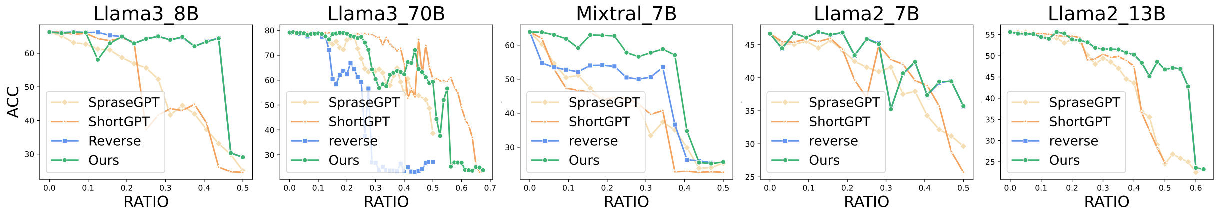


Figure 2: Comparison of Accuracy (ACC) during merging and pruning on the MMLU dataset. MKA achieves higher compression ratios (approximately 43.5% for Llama3-8B, 45% for Llama3-70B, 40% for Mistral-7B, 31.25% for Llama2-7B, and 57.5% for Llama2-13B) while preserving 90% performance. Please see the appendix A for details.

over, we aim to compare our approach with pruning techniques to assess whether it offers improvements and to investigate if it can be combined with quantization methods to achieve even higher compression ratios.

## 3.1 Experimental Setup

### 3.1.1 Datasets

We conduct evaluations using the MKA methods across various benchmark datasets, each specifically designed to test various facets of language comprehension and generation. In detail, **MMLU** (Hendrycks et al., 2020) evaluates broad language understanding across a wide range of domains. **PIQA** (Bisk et al., 2020) is designed to test models on commonsense reasoning in the physical world, aiming to assess NLP models' grasp of everyday physical interactions. **HellaSwag** (Zellers et al., 2019) is a challenge dataset for commonsense natural language inference, consisting of event descriptions with multiple possible continuations, where the task is to select the most plausible one. **RACE-H** (Lai et al., 2017) is a large-scale reading comprehension dataset collected from English exams for Chinese high school students, featuring a high proportion of questions that require reasoning. **BoolQ** (Clark et al., 2019) is a reading comprehension dataset focusing on naturally occurring yes/no questions that often query for complex, non-factoid information and require difficult entailment-like inference to answer correctly.

### 3.1.2 LLMs

In our experiments, we employ the Llama-2 (Touvron et al., 2023), Llama-3, and Mistral-7B (Jiang et al., 2023) models, each distinct in their capabilities and configurations: **Llama-2**: Encompassing models from 7 billion to 70 billion parameters, exhibits superior performance and safety on diverse benchmarks. **Llama-3**: Featuring models with 8 billion and 70 billion parameters, Llama3 offers state-of-the-art performance and advanced reasoning capabilities. **Mistral-7B**: a 7-billion-parameter model that surpasses Llama-2 and Llama-1 in performance and efficiency, leveraging grouped-query and sliding window attention mechanisms for optimal inference across lengthy sequences.

### 3.1.3 Baselines

In this study, we assess the effectiveness of our proposed method, MKA, through two distinct comparative analyses. Firstly, we evaluate MKA directly against several well-established pruning techniques to gauge its standalone efficacy in reducing model size while maintaining performance. Secondly, we extend the comparison to include scenarios where both the traditional pruning methods and MKA are further enhanced through quantization. The baseline methods included in our analysis are: **SparseGPT** (Frantar and Alistarh, 2023): An efficient one-shot pruning method that can induce high sparsity levels in large language models with billions of parameters while preserving accuracy, by reducing the pruning problem to a set of large-scale sparse regression instances solved by a novel approximate solver. **ShortGPT** (Men et al., 2024): A pruning method that removes redundant layers from large language models based on a Block Influence metric, which assesses the significance of each layer. **Reverse Pruning**: A heuristic approach where the importance of layers is considered inversely proportional to their order in the model, prioritizing the retention of earlier layers. **SmoothQuant** (Xiao et al., 2023): SmoothQuant is a training-free post-training quantization solution that enables efficient 8-bit weight and activation quantization for large language models, offering up to 1.56× speedup and 2× memory reduction with minimal accuracy loss. **GPTQ** (Frantar et al., 2022): A one-shot weight quantization method that uses approximate second-order information to maintain high accuracy even with severe weight reduction. **AWQ** (Lin et al., 2023): A novel quantization approach that protects salient weights by adjusting per-channel scaling based on activation observations rather than weight Magnitudes.

### 3.2 In what ways does MKA surpass conventional pruning techniques?

We compare the performance of MKA with baseline compression methods on the MMLU dataset using the Llama3-8B, Llama3-70B, Mistral-7B, Llama2-7B, and Llama2-13B models. The evaluation metric is Accuracy (ACC) during merging and pruning. The results are presented in Figure 2.

We compare the performance of MKA with baseline compression methods on the MMLU dataset using the Llama3-8B, Llama3-70B, Mistral-7B, Llama2-7B, and Llama2-13B models. The evaluation metrics include Accuracy (ACC) during merging and pruning. The results are presented in Figure 2. We can observe that, across all models, our method improves the compression ratio while maintaining performance. Specifically, the compression ratio[2] for Llama3-8B reach 43.5%, for Mistral-7B it reaches 40%, and for Llama2-13B it reaches an impressive 57.5%. Additionally, we observe several phenomena: both methods experience a collapse in model performance, but the model merging method can delay the layer collapse to some extent and stabilize the model's performance very well. Since our strategy is based on Reverse Prune, the scores for the Llama3-8B, Llama2-7B, and Llama2-13B models are very close to the Reverse Prune. Our hypothesis is that the pruning or merging of these models is similar, but model merging can adjust the merging ratio to surpass the effect of pruning. Moreover, for the Llama3-70B and Mistral-7B models, we noticed that the results do not closely match the Reverse Prune.

### 3.3 How Does MKA Combined with Quantization Perform Compared to Pruning Combined with Quantization?

We compare the performance of MKA with the baseline pruning method, ShortGPT (Men et al., 2024), on the MMLU dataset using the Llama3-8B, Llama3-70B, Mistral-7B, Llama2-7B, and Llama2-13B models. The results are shown in Table 1.

| Model | Method | Retained layers (Compression Ratio) | Acc |
|---|---|---|---|
| Llama3-8B | Vanilla Model | 32 (0.00%) | 66.29 |
| | ShortGPT+Smooth | 18(85.94%) | 26.54 |
| | ShortGPT+GPTQ | 18(85.94%) | 25.98 |
| | ShortGPT+AWQ | 18(85.94%) | 26.22 |
| | **MKA (Ours) + Smooth** | 18(85.94%) | **64.20** (**+37.66**) |
| | **MKA (Ours) + GPTQ** | 18(85.94%) | **62.98** (**+37.00**) |
| | **MKA (Ours) + AWQ** | 18(85.94%) | **61.66** (**+35.44**) |
| Mistral-7B | Vanilla Model | 32(0.00%) | 63.87 |
| | ShortGPT+Smooth | 20(84.38%) | 24.32 |
| | ShortGPT+GPTQ | 20(84.38%) | 23.16 |
| | ShortGPT+AWQ | 20(84.38%) | 23.96 |
| | **MKA (Ours) + Smooth** | 20(84.38%) | **56.92** (**+32.60**) |
| | **MKA (Ours) + GPTQ** | 20(84.38%) | **56.12** (**+32.96**) |
| | **MKA (Ours) + AWQ** | 20(84.38%) | **55.34** (**+31.38**) |
| Llama2-7B | Vanilla Model | 32(0.00%) | 46.67 |
| | ShortGPT+Smooth | 16(87.50%) | 25.67 |
| | ShortGPT+GPTQ | 16(87.50%) | 25.82 |
| | ShortGPT+AWQ | 16(87.50%) | 26.01 |
| | **MKA (Ours) + Smooth** | 16(87.50%) | **35.66** (**+9.99**) |
| | **MKA (Ours) + GPTQ** | 16(87.50%) | **35.91** (**+10.09**) |
| | **MKA (Ours) + AWQ** | 16(87.50%) | **36.23** (**+10.22**) |
| Llama2-13B | Vanilla Model | 40 (0.00%) | 55.62 |
| | ShortGPT+Smooth | 20 (87.50%) | 25.89 |
| | ShortGPT+GPTQ | 20 (87.50%) | 25.35 |
| | ShortGPT+AWQ | 20 (87.50%) | 23.83 |
| | **MKA (Ours) + Smooth** | 20 (87.50%) | **46.82** (**+20.93**) |
| | **MKA (Ours) + GPTQ** | 20 (87.50%) | **45.44** (**+20.09**) |
| | **MKA (Ours) + AWQ** | 20 (87.50%) | **45.86** (**+22.03**) |

Table 1: Performance comparison of MKA and ShortGPT pruning with quantization (SmoothQuant, GPTQ, AWQ) on MMLU using Llama3-8B, Mistral-7B, Llama2-7B, and Llama2-13B. MKA outperforms ShortGPT in accuracy across all models and quantization methods at similar compression ratios with int4. The calculation of the compression ratio only considers the number of hidden layers in the model without considering the embedding layer.

We can see that the pruned models are able to be further quantized and maintain performance with a higher compression ratio. Notably, at a high compression ratio of around 50%, MKA significantly outperforms ShortGPT. Additionally, we achieve excellent results with various quantization methods. For example, on Llama3-8B, at a compression ratio of 43.75%, MKA with SmoothQuant achieves 64.20%, far exceeding ShortGPT with SmoothQuant at 37.66%. Similarly, with the GPTQ quantization method, we achieve 62.98%, surpassing ShortGPT's 37.00%, and with AWQ, we achieve 61.66%, exceeding ShortGPT's 35.44%.

### 3.4 MKA vs. Other Pruning Methods on varies benchmarks

We compared the performance of MKA and several other pruning methods on the LLama3-8B model using multiple benchmark datasets at compression ratios of 21.875% and 45.75%. The results are shown in Table 2. From the results, we can ob-

[2] Note that, the compression ratio is calculated as: $\left(L_{\text{total}} - \left(\frac{L_{\text{retained}}}{Q}\right)\right) / L_{\text{total}}$, where $L_{total}$ is the total number of layers before compression, $L_{retained}$ is the number of retained layers, and $Q$ is the quantization factor.

| Method | Compression Ratio = 34.375% | | | | | Compression Ratio = 37.5% | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MMLU | PIQA | HellaSwag | RACE-H | BoolQ | MMLU | PIQA | HellaSwag | RACE-H | BoolQ |
| Vanilla Model | 66.29 | 81.12 | 74.54 | 66.07 | 66.79 | 66.29 | 81.12 | 74.54 | 66.07 | 66.79 |
| SparseGPT | 44.45 | 58.77 | 32.14 | 35.06 | 48.29 | 41.95 | 56.23 | 28.63 | 37.84 | 52.40 |
| ShortGPT | 42.95 | 60.99 | 33.00 | 41.68 | 51.96 | 44.80 | 61.70 | 38.69 | 40.05 | 57.09 |
| **MKA (Ours)** | **64.87** | **67.79** | **51.32** | **55.20** | **63.36** | **62.05** | **66.26** | **50.16** | **49.49** | **63.46** |

Table 2: Comparison of MKA and pruning methods across MMLU, PIQA, HellaSwag, RACE-H, and BoolQ datasets and on different compression ratios.

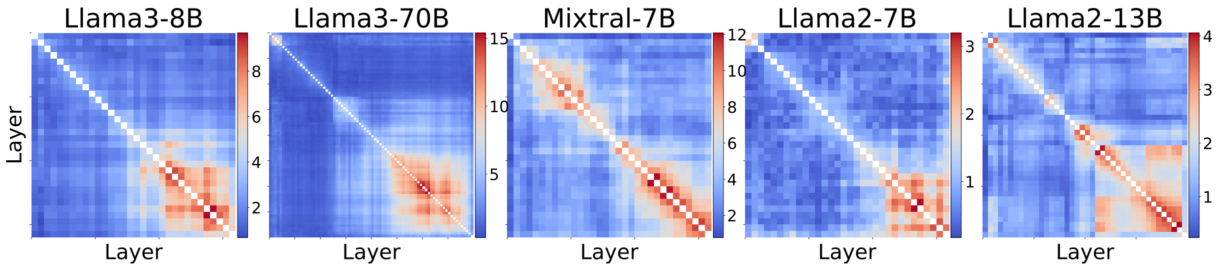


Figure 3: Similarity matrices for Llama-3-8B, Llama-3-70B, Mistral-7B, Llama-2-7B, and Llama-2-13B before and after MKA. Later layers show high similarity, supporting layer merging.

serve that, similar to the previous section, the performance of Reverse Pruning and our method are quite similar. However, model fusion can retain performance better compared to pruning. Relative to SparseGPT and ShortGPT, our method can achieve better performance retention, with significant improvements across all datasets. For example, at a compression ratio of 34.375% on the MMLU dataset, our method can outperform ShortGPT by 21.92% and SparseGPT by 20.42%. Similarly, on the HellaSwag dataset, our proposed method can surpass ShortGPT by 18.32% and SparseGPT by 18.32%.

### 3.5 Are Inter-Layer Knowledge Alignment Similarity Matrices Consistent Across Different Large Models?

We generate layer similarity heatmaps for different models before and after applying MKA. These heatmaps visualize the knowledge alignment and layer merging effects of MKA on various models. Figure 3 presents the similarity heatmaps for Llama-3-8B, Llama-3-70B, Mistral-7B, Llama-2-7B, and Llama-2-13B. We observe that the heatmaps for the later layers of each model exhibit high similarity values, indicating that inter-layer similarity is consistently high in the later layers across different models. This observation supports our layer merging approach. Additionally, when merging the earlier layers, we notice a collapse of the matrix in the final figure, suggesting that earlier layers have a significant influence on later layers. Thus, simple merging operations on the earlier layers of the model are not feasible.

## 4 Discussion

### 4.1 Extension to Multimodal and Specialized Models

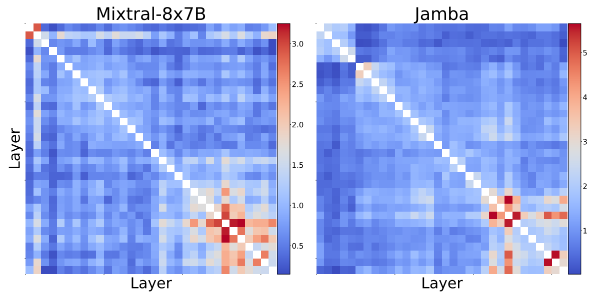


Figure 4: The similarity matrix of Mixtral-8x7B and Jamba model.

In addition to its application to large language models, the MKA method shows promising potential for broader adoption across a variety of deep learning architectures. This includes Mixture-of-Experts (MoE) (Jiang et al., 2024), and Mamba (Gu and Dao, 2023; Lieber et al., 2024) models, which can exhibit similar redundancies in their processing layers.The results show in Figure 4. Initial experiments conducted on these diverse architectures have reinforced the viability of our approach. For instance, the similarity matrices generated on jamba (Lieber et al., 2024) and Mixtral-8x7B (Jiang

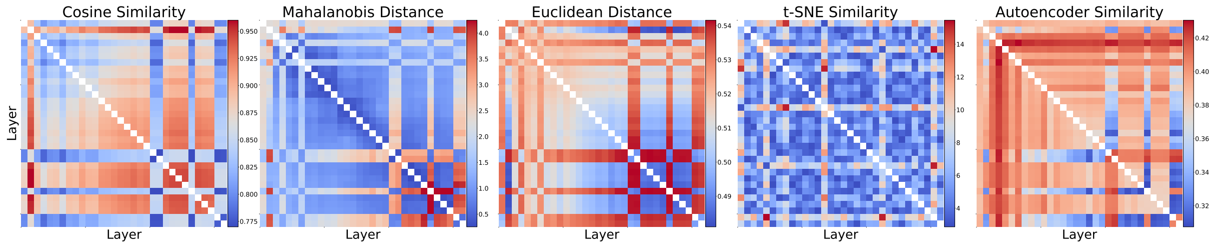


Figure 5: Similarity matrices for various measures in the Llama3-8B model, showing different patterns and effectiveness in capturing layer relationships, with none fully matching the expected merging patterns.

et al., 2024) applying MKA have shown that Our method can also be generalized to other similar models, but the similarity distributions of jamba and Mixtral-8x7B are slightly different from LLM, and we do not yet know the reason. These experiments further validates the effectiveness of our method across different model types.

### 4.2 Analysis of Similarity Measures

In our evaluation of the Llama3-8B model, we explored several similarity measures: Cosine Similarity, Mahalanobis Distance, Euclidean Distance, t-SNE Similarity, and Autoencoder Similarity. The similarity matrices are shown in Figure 5. From the results, we observe that Cosine Similarity, Mahalanobis Distance, and Euclidean Distance display similar distribution patterns with vertical stripes and varied heat values. However, Mahalanobis Distance shows irregular heat values within these stripes, indicating a misalignment with the fused layer data structure. t-SNE Similarity appears random and lacks consistent patterns. For Autoencoder Similarity, the high heat values do not correspond to suitable merging areas or expected high-similarity regions.

### 4.3 Variations in Accuracy Across Different MMLU Subjects During Layer Merging

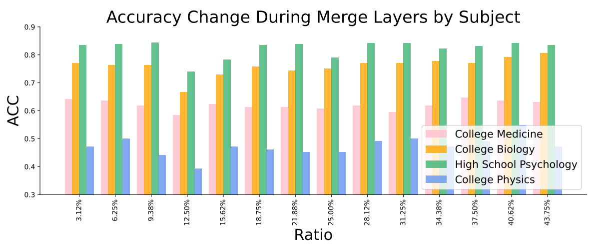


Figure 6: Different MMLU dataset subjects ACC change during merging.

We examine the impact of model merging on performance across various academic subjects in the MMLU benchmark. Figure 6 shows the accuracy changes across subjects such as College Medicine, College Biology, High School Psychology, and College Physics during different stages of merging model layers. From our results, we observe that High School Psychology maintained a stable accuracy with only minor fluctuations, suggesting a consistent performance and low sensitivity to the merging process. In contrast, College Biology experiences a significant drop in accuracy at the 12.5% merging ratio, followed by a recovery. College Physics exhibits frequent fluctuations in accuracy, pointing to a high sensitivity to layer merging. Conversely, College Medicine experiences a steady increase in performance with only minor variations.

## 5 Conclusion

In this paper, we have proposed Manifold-Based Knowledge Alignment and Layer Merging Compression (MKA), a novel model compression technique specifically designed to efficiently reduce the size of large language models (LLMs) while maintaining their performance. MKA leverage manifold learning techniques to align knowledge across layers and utilizes the Normalized Pairwise Information Bottleneck (NPIB) measure to identify the most similar layers for merging. By capturing the intricate nonlinear dependencies within LLMs and integrating knowledge from similar layers, MKA achieves remarkable compression ratios without sacrificing model accuracy. We have conducted extensive experiments on a diverse set of benchmark datasets and various state-of-the-art LLMs to rigorously evaluate the effectiveness of MKA in preserving model performance while significantly reducing model size. Our empirical results demonstrate that MKA consistently outperforms existing pruning methods and can achieve even higher compression ratios when combined with quantization techniques.

## Limitations

The quality of the manifold learning process in MKA heavily depends on the diversity and representativeness of the layer activations extracted from the input dataset. In our experiments, we used $\sigma$ value of 8 and selected the first question from the 57-question MMLU dataset to extract activations. We observed that the number of questions sampled can significantly impact the manifold learning results. Ensuring the Condition Number remains below 2000 is crucial for maintaining the integrity of the learned manifold representations. If the dataset used for extracting activations does not adequately cover the model's operational range, the learned manifold representations might fail to capture the true geometric structure of the data.

The current implementation of MKA has been primarily tested on transformer-based architectures. Although we believe that deep neural networks inherently contain redundancies, the applicability and effectiveness of MKA on other neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), have not been thoroughly explored. Future research can investigate these architectures to confirm whether MKA can achieve similar compression benefits across different types of neural networks.

## References


Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Ronald R Coifman and Stéphane Lafon. 2006. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Preprint*, arXiv:2208.07339.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

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*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *Preprint*, arXiv:2103.13630.

Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. 2024. What makes quantization for large language models hard? an empirical study from the lens of perturbation. *Preprint*, arXiv:2403.06408.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Manish Gupta and Puneet Agrawal. 2022. Compression of deep learning models for text: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–55.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Preprint*, arXiv:1510.00149.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. *Preprint*, arXiv:1608.08710.

Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Evaluating quantized large language models. *Preprint*, arXiv:2402.18158.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Deyuan Liu, Zecheng Wang, Bingning Wang, Weipeng Chen, Chunshan Li, Zhiying Tu, Dianhui Chu, Bo Li, and Dianbo Sui. 2024. Checkpoint merging via bayesian optimization in llm pretraining. *arXiv preprint arXiv:2403.19390*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023a. Llm-pruner: On the structural pruning of large language models. *Preprint*, arXiv:2305.11627.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023b. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal

Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10880–10895, Toronto, Canada. Association for Computational Linguistics.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

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. *Preprint*, arXiv:2302.13971.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *Preprint*, arXiv:2401.10491.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

| **Model** | **Methods** | 0 | 0.03125 | 0.0625 | 0.09375 | 0.125 | 0.15625 | 0.1875 | 0.21875 | 0.25 | 0.28125 | 0.3125 | 0.34375 | 0.375 | 0.40625 | 0.4375 | 0.46875 | 0.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama3_8b | ACC (Reverse) | 66.29 | 66.12 | 66.33 | 66.15 | 66.21 | 65.31 | 64.96 | 62.91 | 64.28 | 65.00 | 63.99 | 64.71 | 62.04 | 63.52 | 64.51 | 30.31 | 29.07 |
| | ACC (Ours) | 66.29 | 65.96 | 66.26 | 66.15 | 58.08 | 62.94 | 64.96 | 62.92 | 64.28 | 65.01 | 63.99 | 64.87 | 62.05 | 63.42 | 64.42 | 30.29 | 29.05 |
| Llama2_7b | ACC (Reverse) | 46.67 | 44.37 | 46.71 | 46.09 | 46.89 | 46.51 | 46.79 | 43.33 | 45.90 | 45.22 | 35.33 | 40.58 | 42.33 | 37.34 | 39.26 | 39.53 | 35.65 |
| | ACC (Ours) | 46.67 | 44.45 | 46.74 | 46.07 | 46.93 | 46.52 | 46.84 | 43.41 | 45.85 | 45.09 | 35.25 | 40.67 | 42.40 | 37.38 | 39.41 | 39.45 | 35.71 |

Table 3: ACC during the compression process of Ours and Reverse Prune on Llama3-8b and Llama2-7b models.

| Methods | 0 | 0.025 | 0.05 | 0.075 | 0.1 | 0.125 | 0.15 | 0.175 | 0.2 | 0.225 | 0.25 | 0.275 | 0.3 | 0.325 | 0.35 | 0.375 | 0.4 | 0.425 | 0.45 | 0.475 | 0.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC (Reverse) | 55.62 | 55.24 | 55.21 | 55.12 | 54.44 | 54.02 | 55.63 | 55.27 | 53.87 | 53.66 | 53.17 | 51.89 | 51.56 | 51.56 | 51.48 | 50.75 | 50.28 | 48.37 | 45.18 | 48.59 | 46.78 |
| ACC (Ours) | 55.62 | 55.24 | 55.21 | 55.12 | 54.44 | 54.02 | 55.63 | 55.27 | 53.87 | 53.66 | 53.17 | 51.89 | 51.56 | 51.56 | 51.49 | 50.75 | 50.28 | 48.37 | 45.18 | 48.59 | 46.78 |

Table 4: ACC during the compression process of Ours and Reverse Prune on Llama2-13b model.

# A More Results