Self-Data Distillation for Recovering Quality in Pruned Large Language Models

Vithursan Thangarasa, Ganesh Venkatesh, Nish Sinnadurai, Sean Lie
Cerebras Systems
{vithu, ganesh.venkatesh, nish}@cerebras.net

Abstract

Large language models have driven significant progress in natural language processing, but their deployment requires substantial compute and memory resources. As models scale, compression techniques become essential for balancing model quality with computational efficiency. Structured pruning, which removes less critical components of the model, is a promising strategy for reducing complexity. However, one-shot pruning often results in significant quality degradation, particularly in tasks requiring multi-step reasoning. To recover lost quality, supervised fine-tuning (SFT) is commonly applied, but it can lead to catastrophic forgetting by shifting the model's learned data distribution. Therefore, addressing the degradation from both pruning and SFT is essential to preserve the original model's quality. In this work, we propose *self-data distilled fine-tuning* to address these challenges. Our approach leverages the original, unpruned model to generate a distilled dataset that preserves semantic richness and mitigates catastrophic forgetting by maintaining alignment with the base model's knowledge. Empirically, we demonstrate that self-data distillation consistently outperforms standard SFT, improving average accuracy by up to 8% on the HuggingFace OpenLLM Leaderboard v1. Specifically, when pruning 6 decoder blocks on Llama3.1-8B Instruct (i.e., 32 to 26 layers, reducing the model size from 8.03B to 6.72B parameters), our method retains 91.2% of the original model's accuracy compared to 81.7% with SFT, while reducing real-world FLOPs by 16.30%. Furthermore, our approach scales effectively across datasets, with the quality improving as the dataset size increases.

1 Introduction

The advent of large language models (LLMs) such as GPT-4 (OpenAI et al., 2024), Gemini (Gemini et al., 2024), and Llama 3 (Dubey et al., 2024) has revolutionized natural language processing (NLP), driving significant advancements across various tasks through extensive pre-training on textual data. These models, enhanced by supervised fine-tuning (SFT), demonstrate impressive instruction-following abilities (Ouyang et al., 2022; Touvron et al., 2023a), but come with high computational costs for both training and inference (Hoffmann et al., 2022; Kaplan et al., 2020). To address diverse deployment requirements across varying model scales, sizes, and compute budgets, compressing models for efficient inference is essential, particularly given the significant time, data, and resource constraints associated with training multiple multi-billion parameter models from scratch.

Most model compression techniques can be grouped into four main categories: distillation (Hinton et al., 2015), low-rank factorization (Hu et al., 2022), pruning (LeCun et al., 1989), and quantization (Han et al., 2015). In our work, we focus on pruning, though we aim for our methodology to inspire further developments across these other compression methods. Structured pruning, which selectively removes less critical components of a neural network, has emerged as a promising method for improving LLM efficiency (Ma et al., 2023). This method has gained attention for its ability to reduce memory and computation requirements, making inference more efficient. Recent studies have

Machine Learning and Compression Workshop (NeurIPS 2024).

shown that LLMs exhibit significant redundancy, particularly in the middle layers, where removing these layers has a minimal impact on overall model quality (Gromov et al., 2024; Men et al., 2024). The Transformer (Vaswani et al., 2017) architecture's residual connections allow for the final output to be a summation of earlier layers, enabling the removal of non-essential layers without drastically harming model quality.

Despite its potential advantages, depth-wise structured pruning presents inherent challenges. It often leads to accuracy degradation, especially on tasks requiring multi-step reasoning, such as ARC-C (Clark et al., 2018) or GSM8k (Cobbe et al., 2021), where the structured order of layer outputs plays a crucial role. In these cases, pruning disrupts the flow of information between layers, resulting in poor model quality even after supervised fine-tuning (SFT) (Sun et al., 2024). While SFT can help recover some of the lost quality, it is generally insufficient for tasks with high reasoning complexity, where the structured sequence of layer outputs is essential. In addition, fine-tuning can amplify catastrophic forgetting (McCloskey and Cohen, 1989), where the model loses previously learned information, particularly on tasks not represented in the fine-tuning data. Standard mitigation strategies, such as data replay (Ostapenko et al., 2022) or parameter importance-based methods (Kirkpatrick et al., 2017), often become impractical for LLMs due to their scale. Moreover, fine-tuning often leads to distribution shifts, further degrading model quality (Yang et al., 2024). As LLMs continue to grow in size and complexity, developing more effective strategies to mitigate these challenges during pruning is critical to unlocking its full potential.

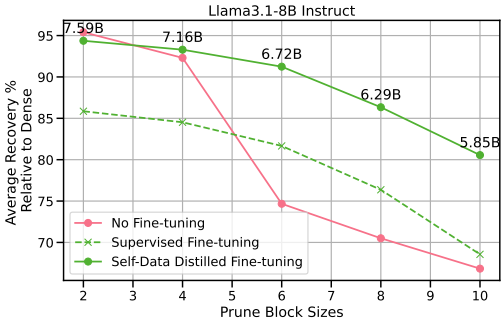

Figure 1: **Average quality recovery (%) of pruned Llama3.1-8B Instruct models relative to the unpruned baseline, across varying prune block sizes on the HuggingFace OpenLLM Leaderboard v1.** The plot compares no fine-tuning, supervised fine-tuning, and self-data distilled fine-tuning using the OpenMathInstruct dataset. While model quality declines with prune block sizes, self-data distillation consistently achieves superior recovery.

In our work, we propose a novel approach to mitigate the adverse effects of structured pruning by employing *self-data distilled fine-tuning*. Our method leverages the original, unpruned model as a seed language model to generate a distilled dataset that upholds semantic equivalence with the original task dataset. This approach not only preserves the semantic richness of the data but also mitigates catastrophic forgetting, a phenomenon where fine-tuned models lose their general instruction-following abilities due to the distribution shift introduced during standard SFT. As seen in Figure 1, we show that self-data distillation improves average accuracy by up to 8% over SFT on the HuggingFace OpenLLM Leaderboard v1 (Beeching et al., 2023). Specifically, when pruning 6 blocks from Llama3.1-8B Instruct, our approach retains 91.2% of the original model's accuracy compared to 81.7% with SFT, while also reducing FLOPs by 16.30%. The main contributions of our work are:

- To our knowledge, we are the first to introduce self-data distillation as a fine-tuning method for recovering the model quality of pruned models. Empirically, we show that self-data distillation on Llama3.1-8B Instruct consistently outperforms SFT across all pruned models.
- We demonstrate that self-data distillation scales effectively across a wide range of open-source fine-tuning datasets for LLMs, covering open-domain conversation, reasoning, and instruction following, with quality recovery significantly improving as the dataset size increases.

2 Methodology

In this section, we present our approach to enhancing the efficiency of LLMs through *structured layer pruning* combined with *self-data distillation*. Our strategy involves systematically identifying and removing redundant layers to optimize model efficiency while preserving task-specific accuracy. Post-pruning, we employ self-data distillation to mitigate the effects of catastrophic forgetting during the fine-tuning phase, thereby ensuring that the pruned model improves its quality over standard SFT.

2.1 Layer-Pruning Algorithm for Language Models

Transformer networks (Vaswani et al., 2017) have become a foundational architecture in deep learning, particularly for tasks involving natural language processing and sequence modeling. A standard Transformer consists of L layers, each of which includes a multi-head self-attention mechanism followed by a feedforward network. These layers sequentially transform an input sequence into increasingly abstract representations, with each layer's output serving as the input to the subsequent layer. Let $x^{(\ell)}$ denote the input to the ℓ^{th} layer, and $h^{(\ell)} = f(x^{(\ell)})$ denote the output of the ℓ^{th} layer after applying the transformation function $f(\cdot)$. The output of the final layer, $h^{(L)}$, is typically used for downstream tasks, such as classification or natural language generation.

Block Importance Metric Recent literature has introduced various metrics to evaluate the importance of layers within Transformer-based vision and language models. For instance, Samragh et al. (2023) proposed a metric based on the relative magnitude, $\left\| \frac{f(x^{(\ell)})}{x^{(\ell)}+f(x^{(\ell)})} \right\|$ to measure the importance of a layer ℓ by characterizing its influence on the network's output. Here, $x^{(\ell)}$ represents the input to layer ℓ, and $f(x^{(\ell)})$ denotes the transformation applied by the layer. Additionally, Men et al. (2024) introduced the Block Influence (BI) score, which assumes that the degree to which a Transformer block alters hidden states correlates with its importance. The BI score for the ℓ^{th} block is calculated as, $1 - \mathbb{E}_{X,i} \frac{x_i^{(\ell)} \cdot x_i^{(\ell+1)}}{\left\|x_i^{(\ell)}\right\|_2 \left\|x_i^{(\ell+1)}\right\|_2}$ where $x_i^{(\ell)}$ represents the i^{th} hidden state vector at layer ℓ, and $x_i^{(\ell+1)}$ represents the corresponding hidden state vector at the subsequent layer $\ell + 1$. Gromov et al. (2024) proposed using an angular cosine metric to measure the similarity between layer outputs as a criterion for pruning. This metric is based on the premise that layers producing highly similar outputs can be pruned with minimal impact on the model's overall quality. Both the BI and the angular cosine metric fundamentally use cosine distance to assess the importance of layers or blocks of layers within a model. However, based on our ablation studies in Section 3, we found no significant difference in their effectiveness for layer pruning. Consequently, we have opted to use the angular cosine metric from Gromov et al. (2024) in our studies. This metric allows us to effectively quantify and identify redundancy in the model's layers. As described in Algorithm 1, the pruning process begins by selecting a block of consecutive layers, denoted by n, for potential removal. The choice of n directly influences the extent of pruning, which has important implications for both the model's efficiency and overall quality.

Algorithm 1 Layer-Pruning Language Models

Require: Model M with L layers, Number of layers to prune n, Dataset D
Ensure: Pruned model M'

```
Initialize ℓ* ← None, d_min ← ∞
for each layer ℓ from 1 to L − n do
    h^(ℓ)(D) ← activation at ℓ with input D
    h^(ℓ+n)(D) ← activation at ℓ + n with input D
    Compute d(h^(ℓ)(D), h^(ℓ+n)(D)) using Eq. 1
    if d(h^(ℓ)(D), h^(ℓ+n)(D)) < d_min then
        d_min ← d(h^(ℓ)(D), h^(ℓ+n)(D))
        ℓ* ← ℓ
    end if
end for
Prune layers ℓ* to ℓ* + n − 1 from M
Connect output of layer ℓ* to input of layer ℓ* + n
return pruned model M'
```

Determine the Prune Block Size To determine which layers to prune, we calculate the angular distance between the activation outputs of layer ℓ and layer $\ell + n$. For each potential starting layer ℓ, the angular distance $d(h^{(\ell)}(D), h^{(\ell+n)}(D))$ is computed using a representative dataset D, which may be a representative pre-training dataset or one that is tailored to a specific downstream task. In our work, we use RedPajama (Computer, 2023) as the representative dataset to evaluate the sample distances. The angular distance metric is formally defined as,

$$d(h^{(\ell)}(D), h^{(\ell+n)}(D)) \equiv \frac{1}{\pi} \arccos \left(\frac{h_T^{(\ell)}(D) \cdot h_T^{(\ell+n)}(D)}{\left\|h_T^{(\ell)}(D)\right\| \left\|h_T^{(\ell+n)}(D)\right\|} \right), \tag{1}$$

where $h_T^{(\ell)}(D)$ and $h_T^{(\ell+n)}(D)$ denote the activation vectors at the final token position T of the input sequence, corresponding to layers ℓ and $\ell + n$, respectively. The activations are normalized using the L^2-norm $\|\cdot\|$, which ensures a consistent scale when comparing layer outputs. The choice to focus on T is motivated by the autoregressive nature of Transformers, where the representation of the final token encapsulates information from the entire input sequence due to the causal attention mechanism.

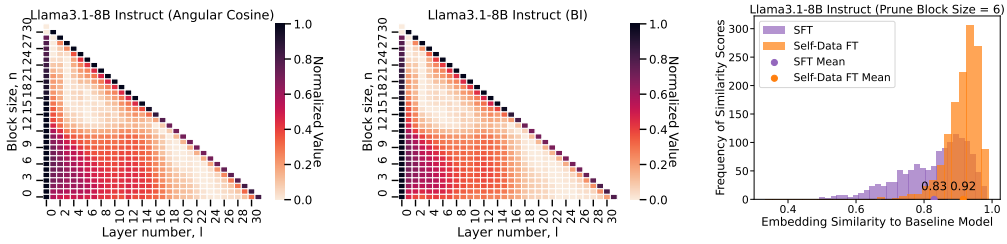

Figure 2: Comparison of Llama3.1-8B Instruct using (left) *angular cosine* and (center) *block influence* (BI) score metrics. (right) Distribution of embedding similarities on GSM8k after fine-tuning on OpenMathInstruct for a pruned Llama3.1-8B Instruct model (i.e., 26 decoder layers, prune block size $n = 6$). Self-data distilled fine-tuning (Self-Data FT) achieves higher similarity to the original baseline, indicating reduced distribution shift compared to standard supervised fine-tuning (SFT).

Identify Optimal Pruning Block We identify the optimal block of layers for pruning by minimizing the angular distance. Specifically, the starting layer $\ell^\star$ of the block is selected as follows,

$$\ell^\star(n) \equiv \arg\min_{\ell} \; d(h^{(\ell)}(D), h^{(\ell+n)}(D)),$$

where $\ell^\star$ corresponds to the layer with the smallest angular distance to its corresponding n-th successor layer. This optimization identifies a block of layers that exhibit high redundancy, as measured by their similar output activations. Pruning such a block is expected to have minimal impact on the model's overall capacity. Once identified, layers from $\ell^\star$ to $\ell^\star + n - 1$ are removed, and the model is restructured by directly connecting the output of layer $\ell^\star$ to the input of layer $\ell^\star + n$. This pruning operation can be efficiently implemented in deep learning frameworks such as PyTorch (Paszke et al., 2019), where the layers are typically encapsulated within a `ModuleList` or similar data structure. The layers within the identified block are excluded when defining the new, pruned model architecture.

2.2 Self-Data Distillation for Pruned Models

After pruning a Transformer, fine-tuning is typically required to adapt the pruned model to specific downstream tasks. However, the fine-tuning process can amplify catastrophic forgetting, especially when the fine-tuning data distribution diverges from the original training distribution. To address this, we propose *self-data distilled fine-tuning*, which aligns the fine-tuning dataset with the original model's learned distribution, helping to mitigate forgetting and maintain model quality across tasks.

Supervised Fine-tuning Given a pruned model M' with parameters θ', supervised fine-tuning aims to adapt the model to a specific downstream task t using a task-specific dataset. For each example (x^t, y^t) in the dataset, where x^t is the input and y^t is the corresponding target output, the model is fine-tuned by minimizing the negative log-likelihood of producing the correct output y^t given the input x^t and the context c^t associated with the task,

$$L_{\text{SFT}}(\theta') = -\log f_{\theta'}(y^t \mid c^t, x^t),$$

where $f_{\theta'}$ represents the pruned model with parameters θ'. The objective is to align the model's output distribution with the distribution of the task-specific data, thereby improving quality on the target task t. However, traditional supervised fine-tuning (SFT) can lead to catastrophic forgetting (Kotha et al., 2024), particularly in cases where the task-specific data distribution diverges significantly from the original training distribution.

Self-Data Distilled Fine-tuning First, the self-data distillation process begins with generating a distilled dataset that aligns with the distribution of the original, unpruned model M. Specifically, for each example in the fine-tuning dataset, the original seed model M is used to generate a distilled output $\tilde{y}$ by rewriting the original response y^t as, $\tilde{y} \sim f_\theta(y \mid c^t, x^t, y^t)$, where f_θ represents the original model with parameters θ. This distilled output $\tilde{y}$ is designed to stay within the distribution of the original model, thereby minimizing the risk of catastrophic forgetting. Following Yang et al. (2024), to ensure the quality of the distilled output, a conditional selection process is applied,

$$\tilde{y}' = \begin{cases} \tilde{y} & \text{if Extract}(\tilde{y}) = y^t, \\ y^t & \text{otherwise.} \end{cases}$$

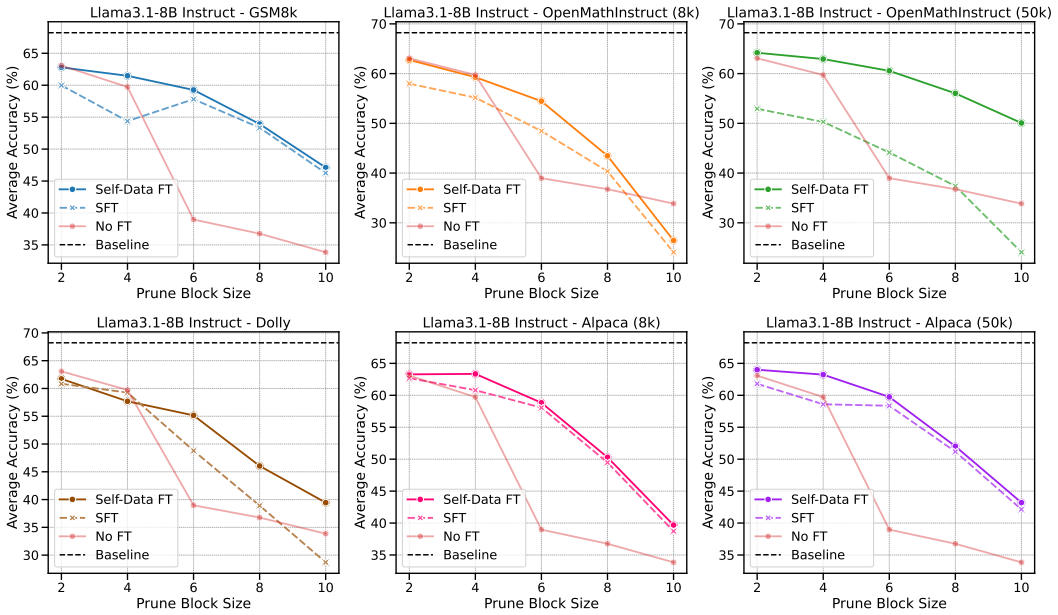

Figure 3: **Quality of pruned Llama3.1-8B Instruct models across various datasets and pruning block sizes.** The plots show average accuracy across MMLU, GSM8k, ARC-C tasks for GSM8k, OpenMathInstruct, Dolly, and Alpaca under three strategies: Self-Data FT, SFT, and No FT. Self-Data FT consistently outperforms SFT and No FT, with the largest gains using OpenMathInstruct (50k).

This ensures that the distilled responses retain essential characteristics, such as correctness in structured tasks (e.g., mathematical reasoning). Once the distilled dataset is prepared, the pruned model M' with parameters θ' undergoes supervised fine-tuning. The fine-tuning process is defined by the following objective,

$$L_{\text{Self-Data FT}}(\theta') = -\log f_{\theta'}(\tilde{y}' \mid c^t, x^t),$$

where $f_{\theta'}$ represents the pruned model fine-tuned on the distilled dataset. This objective helps align the pruned model with the distilled data distribution, thereby reducing the impact of catastrophic forgetting compared to standard supervised fine-tuning. Self-data distillation is key to minimizing quality loss after pruning, significantly improving retention of model capabilities. While it may not fully preserve the original model's quality, it offers an effective balance between efficiency and accuracy in LLMs, as evidenced by the results presented in the ablation studies in Section 3.

3 Ablation Studies on Layer-Pruning and Self-Data Distillation

In this section, we examine key factors affecting the quality of pruned Llama3.1-8B Instruct (Dubey et al., 2024) models. Our experiments assess the impact of layer importance metrics, pruning block sizes, and fine-tuning strategies. We compare BI and angular cosine metrics for determining layer redundancy and analyze how these choices influence pruning outcomes. In addition, we evaluate the effectiveness of self-data distilled fine-tuning across various datasets, showing its ability to recover model quality post-pruning, outperforming standard supervised fine-tuning methods.

Effect of Layer Importance Metric We investigated two layer importance metrics, BI and angular cosine distance, to guide pruning decisions in the Llama3.1-8B Instruct model (see Figure 2). Both metrics assess the redundancy of layers by measuring the cosine distance between their inputs and outputs but differ in their focus. BI quantifies the change in hidden states across layers, while angular cosine distance emphasizes output similarity between consecutive layers. Despite minor differences in middle layers, both metrics produced comparable pruning results across block sizes. We ultimately chose the angular cosine metric due to its computational efficiency, leveraging cosine similarity without the added complexity of BI. This efficiency makes it more scalable for large models, and its direct measure of output similarity aligns with our intuition that layers producing highly similar outputs are likely redundant, making it a more practical tool for structured pruning in this study.

Table 1: **Model quality results for pruned Llama3.1-8B Instruct models across various pruning block sizes and fine-tuning strategies.** Average accuracy is reported on the OpenLLM Leaderboard v1, along with the recovery percentage relative to the baseline model. Self-data distillation consistently outperforms other methods, particularly when combined with model merging (MM).

Prune Block Size	Model Savings	Fine-tuning Method	Dataset	ARC-C (25-shot)	HellaSwag (10-shot)	TruthfulQA (0-shot)	MMLU (5-shot)	Winogrande (5-shot)	GSM8k (5-shot)	Avg. Score	Avg. Recovery
Baseline	-	No FT		60.92	80.16	54.02	68.15	77.58	75.59	69.40	-
4	10.86% (7.16B)	No FT		55.20	75.70	**52.40**	**67.79**	**75.29**	56.18	64.06	92.31%
		SFT	OpenMathInstruct	51.62	71.70	49.40	61.65	73.25	44.35	58.66	84.52%
		Self-Data Distillation	OpenMathInstruct	53.93	74.27	50.61	65.34	74.90	**69.44**	64.75	93.29%
		Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**56.22**	**76.48**	51.60	66.93	74.43	69.37	**65.84**	**94.86%**
6	16.30% (6.72B)	No FT		49.49	68.72	53.63	67.42	70.40	1.29	51.82	74.67%
		SFT	OpenMathInstruct	46.93	68.51	50.81	59.98	70.01	43.82	56.68	81.66%
		Self-Data Distillation	OpenMathInstruct	50.00	71.57	53.14	64.96	73.64	**66.64**	63.33	91.24%
		Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**54.52**	**73.58**	**53.57**	**66.35**	**74.82**	65.67	**64.75**	**93.30%**
8	21.73% (6.29B)	No FT		44.71	61.22	56.11	65.57	65.98	0.00	48.93	70.50%
		SFT	OpenMathInstruct	42.15	64.49	54.69	60.65	66.38	29.64	53.00	76.37%
		Self-Data Distillation	OpenMathInstruct	46.67	65.70	53.32	64.87	71.27	**57.70**	59.92	86.38%
		Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**46.93**	**67.85**	**53.33**	**65.81**	**72.30**	56.41	**61.24**	**88.24%**
10	27.16% (5.85B)	No FT		37.46	54.45	**55.06**	64.09	67.25	0.00	46.39	66.83%
		SFT	OpenMathInstruct	39.33	57.38	51.47	57.44	64.48	15.39	47.58	68.56%
		Self-Data Distillation	OpenMathInstruct	40.88	61.11	53.46	64.54	**70.88**	**44.58**	55.91	80.56%
		Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**41.72**	**63.30**	52.87	**65.57**	70.33	42.26	**56.08**	**80.70%**

Analysis on Self-Data Distilled Datasets We assess the role of fine-tuning datasets in the self-data distillation process for recovering quality in pruned Llama3.1-8B Instruct models, comparing LoRA (Hu et al., 2022) fine-tuning on standard versus self-distilled datasets. We fine-tuned the pruned models on a range of open-source datasets, including GSM8k (math word problems), Dolly (Conover et al., 2023) (open-domain conversation), OpenMathInstruct (Toshniwal et al., 2024) (math and reasoning), and Alpaca (Taori et al., 2023) (instruction-following), with a primary focus on reasoning-heavy tasks. Therefore, we evaluated the orginal baseline and pruned models' accuracy on ARC-C (25-shot), GSM8k (5-shot), and MMLU (Hendrycks et al., 2021a) (5-shot) tasks using the LM-eval-harness (Gao et al., 2024). We provide additional details on the experimental setup in Appendix A, and extended results from our fine-tuning ablation studies can be found in Appendix B.

The results, presented in Figure 3, show that self-data distillation offers significant quality improvements over SFT and one-shot pruned models, particularly with larger datasets like the 50k-sample OpenMathInstruct. Self-data distilled models achieved 5-10% higher accuracy across all pruning block sizes, demonstrating the effectiveness of self-data distillation in recovering a substantial portion of the model's quality post-pruning. Our ablations revealed a strong correlation between dataset size and quality recovery, with larger datasets consistently outperforming smaller ones. The improvements being most pronounced with medium to large block sizes, where self-data distillation notably enhanced generalization, especially in reasoning tasks. These findings highlight the importance of dataset scale and the self-data distillation process, with the 50k-sample OpenMathInstruct dataset delivering the best overall results, and will be the focus of subsequent experiments.

Mitigating Catastrophic Forgetting The right plot in Figure 2 clearly illustrates the advantages of self-data distilled fine-tuning (Self-Data FT) over SFT in mitigating catastrophic forgetting after pruning. We study the similarities between Llama3.1-8B Instruct models fine-tuned on a supervised 50k-sample OpenMathInstruct dataset and a self-data distilled version of the same dataset. The plot compares the sentence embeddings of the two models' generated responses on the GSM8k task to the baseline unpruned Llama3.1-8B Instruct model. Self-Data FT maintains a narrower distribution of embedding similarities, with a higher mean score of 0.92, indicating better preservation of the original model's learned representations. In contrast, SFT exhibits a wider spread of similarity scores and a lower mean of 0.83, reflecting a more significant distribution shift and a greater risk of quality degradation. The tighter distribution in Self-Data FT highlights its ability to retain model quality across tasks, while the wider distribution in SFT suggests a greater distribution shift, leading to higher risk of catastrophic forgetting (see Appendix C). This highlights Self-Data FT as a more effective method for mitigating model quality degradation post-pruning.

4 Empirical Results

We evaluated the quality of Llama3.1-8B Instruct models pruned at various block sizes under three fine-tuning strategies: no fine-tuning (No FT), supervised fine-tuning (SFT), and our proposed self-data distillation. As shown in Table 1, pruned models without fine-tuning experience substantial accuracy losses, particularly at larger block sizes (e.g., a 46.39% average score at block size 10), highlighting the critical need for post-pruning adaptation. SFT improves quality, with an average

recovery of 81.66% at block size 6, but struggles on reasoning-heavy tasks like GSM8k and ARC-C. In contrast, self-data distillation significantly enhances quality recovery, achieving 91.24% at block size 6, with GSM8k accuracy reaching 66.64% (compared to 43.82% with SFT). Even at block size 10, self-data distillation maintains an 80.56% recovery, outperforming SFT's 68.56%. These findings establish self-data distillation as the most effective method for preserving model quality post-pruning, particularly for reasoning-intensive tasks, making it essential for large-scale model compression.

Improving Self-Data Distillation with Model Merging We extend self-data distillation by introducing *model merging* using Spherical Linear Interpolation (SLERP) (Shoemake, 1985). While self-data distillation has shown significant improvements in maintaining model quality post-pruning, we investigate whether merging models fine-tuned on different datasets can provide further gains. Specifically, we merge Llama3.1-8B Instruct models fine-tuned on OpenMathInstruct and Alpaca, as these datasets delivered the best results in our ablations in Section 3. SLERP enables smooth interpolation between model parameters, preserving the geometric properties of the parameter space (see Appendix D for details). Our results, as shown in Table 1, demonstrate that self-data distillation with model merging via SLERP yields the highest recovery in quality across all pruning block sizes. At block size 6, the merged model achieves a 93.30% recovery, compared to 91.24% for the OpenMathInstruct model alone. These findings suggest that SLERP-based model merging not only mitigates pruning-related quality loss but also improves generalization, particularly on tasks such as GSM8k and ARC-C.

5 Related Work

Pruning for Model Compression Pruning is a well-established method for reducing the complexity of overparameterized models in both computer vision and NLP (Hassibi et al., 1993; LeCun et al., 1989). It is typically classified into *structured* and *unstructured* pruning. Unstructured pruning removes individual weights and can achieve high compression rates in LLMs, particularly when paired with hardware accelerators like the Cerebras CS-3 (Lie, 2022; Thangarasa et al., 2024a) or Neural Magic DeepSparse (Neural Magic, 2021), which exploit sparsity for significant speedups. However, without specialized infrastructure, unstructured pruning can result in inefficient acceleration. Structured pruning, which removes entire channels, layers, or attention heads, is more effective in models with architectural redundancy, but can degrade model quality, especially in complex tasks which require multi-step reasoning (Kurtic et al., 2023; Ma et al., 2023; Sun et al., 2024).

To address these challenges, several metrics have been developed to guide pruning decisions more effectively. For instance, Shortened Llama (Kim et al., 2024) demonstrated that depth pruning (removing layers) can be as effective as width pruning (removing units within layers), or even a combination of both. The Block Influence (BI) score (Men et al., 2024), applied in Llama-2 (Touvron et al., 2023b), measures block importance by evaluating changes in hidden state magnitudes. Additionally, the angular cosine similarity metric (Gromov et al., 2024) identifies layers with redundant activations, allowing for selective pruning in models such as Llama-2 and Mistral (Jiang et al., 2023). Gromov et al. (2024) also proposed a healing method using low-rank adapters (Hu et al., 2022) to recover lost quality. Despite these advancements, pruning LLMs still results in sharp accuracy degradation (Sun et al., 2024), and traditional recovery methods such as fine-tuning or re-pretraining are resource-intensive (Sreenivas et al., 2024; Xia et al., 2024). Our self-data distillation approach extends this by leveraging the unpruned model to generate a distilled dataset for fine-tuning the pruned model, ensuring semantic alignment and mitigating the quality degradation caused by pruning. While combining this with standard KD techniques could further improve generalization, we leave that exploration for future work.

Distillation Knowledge distillation (KD) (Hinton et al., 2015) is a widely-used model compression technique where a smaller student model learns from a larger teacher model, enabling efficient model quality retention. In NLP, KD has been applied in various contexts to align student models with teacher outputs (Agarwal et al., 2024; Gu et al., 2024; Liang et al., 2021), hidden states (Jiao et al., 2020), and attention mechanisms (Wang et al., 2021). Recent work on Llama3.2 (Llama Team, 2024) extends this by using logits from larger Llama3.1 models (e.g., 8B, 70B) as token-level targets during pre-training, allowing the smaller models (e.g., 1B, 3B) to achieve superior quality compared to training from scratch (Llama Team, 2024). Supervised fine-tuning (SFT) has been widely employed in various self-distillation frameworks to train student models using sequences

generated by teacher LLMs (Sun et al., 2023; Wang et al., 2023; Zelikman et al., 2022). Yang et al. (2024) investigated self-distillation as a way to alleviate distribution shifts, improving model quality during SFT while improving generalization across tasks. Our self-data distillation method builds on these techniques by leveraging the original unpruned model to generate a distilled dataset for fine-tuning the pruned model. This ensures semantic alignment and mitigates the quality degradation seen after pruning. Furthermore, while our approach can be combined with standard KD techniques to enhance generalization and recover quality while lowering computational costs, we leave the exploration of such combinations for future work.

Catastrophic Forgetting One of the major challenges of pruning and distillation techniques in LLMs is catastrophic forgetting, where a model loses its previously learned capabilities during fine-tuning (Korbak et al., 2022; Kotha et al., 2024). Regularization techniques such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) aim to alleviate this by controlling parameter updates, but are task-dependent and require careful tuning (Huang et al., 2021). Architecture-based methods, which allocate separate parameters for each task (Razdaibiedina et al., 2023), preserve task-specific knowledge but add complexity and overhead, reducing the overall efficiency of model compression. Replay-based techniques (Ostapenko et al., 2022; Rolnick et al., 2019; Sun et al., 2019) store data subsets from previous tasks for rehearsal, either through direct storage or synthesis via generative models. However, these methods demand substantial memory to store large datasets and are often impractical due to privacy concerns or lack of access to past data. Our self-data distilled fine-tuning approach avoids these challenges by aligning the fine-tuning dataset with the original model's learned distribution, preserving knowledge across tasks without requiring new parameters or architectural changes. This method offers a robust solution for mitigating catastrophic forgetting while maintaining model quality after pruning.

6 Conclusion

In conclusion, we introduced *self-data distilled fine-tuning* as an effective method to counteract quality degradation in pruned Llama3.1-8B Instruct models, addressing catastrophic forgetting while preserving alignment with the model's original distribution. Our approach consistently outperforms standard supervised fine-tuning, demonstrating superior accuracy recovery post-pruning across various downstream tasks on the HuggingFace OpenLLM Leaderboard v1. Additionally, model merging via SLERP further enhances recovery, achieving significant quality retention. We also show that our method scales with dataset size, where larger self-distilled datasets lead to improved quality recovery. These findings highlight self-data distilled fine-tuning as a critical tool for maintaining high model quality post-pruning, offering an efficient solution for large-scale model compression. Future work may involve integrating self-data distilled fine-tuning with complementary model compression techniques such as sparsity, quantization or teacher distillation, potentially yielding greater efficiency without sacrificing model quality. Furthermore, adopting fine-tuning strategies that leverage dynamically generated datasets or incorporate multi-modal inputs could enhance the retention of critical knowledge, broadening the scope of self-data distillation to a wider array of tasks. Extending these methodologies to next-generation LLM architectures presents a promising avenue for unlocking additional computational efficiency and model robustness.

7 Acknowledgements

We would like to express our gratitude to Valavan Manohararajah, Mike Lasby, Gokul Ramakrishnan, Alex Tsaptsinos, and Eric Sather for their valuable discussions, as well as their insightful comments and feedback, which greatly enhanced the quality of our manuscript.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. `https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard`, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv*, 2021.

Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

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

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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. 07 2024.

Gemini, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, and et al. Gemini: A family of highly capable multimodal models. *arXiv*, 2024.

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

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

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv*, 2015.

Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In *NeurIPS*, 1993.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021b.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv*, 2022.

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

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. Continual learning for text classification with information disentanglement based regularization. In *NAACL*, 2021.

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

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,

Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv*, 2024.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *EMNLP*, 2020.

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

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *arXiv*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. In *PNAS*, 2017.

Tomasz Korbak, Hady Elsahar, German Kruszewski, and Marc Dymetman. Controlling conditional language models without catastrophic forgetting. In *ICML*, 2022.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *ICLR*, 2024.

Eldar Kurtic, Elias Frantar, and Dan Alistarh. ZipLM: Inference-aware structured pruning of language models. In *NeurIPS*, 2023.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *NeurIPS*, 1989.

Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. Mix{kd}: Towards efficient distillation of large-scale language models. In *ICLR*, 2021.

Sean Lie. Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning : Cerebras systems. In *2022 IEEE HCS*, 2022.

Meta AI Llama Team. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation. 1989.

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

Neural Magic. Deepsparse. `https://github.com/neuralmagic/deepsparse`, Feb 2021.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. Gpt-4 technical report. *arXiv*, 2024.

Oleksiy Ostapenko, Timothée Lesort, Pau Rodr'iguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *CoLLAs*, 2022.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *arXiv*, 2019.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *ICLR*, 2023.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *NeurIPS*, 2019.

Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. Weight subcloning: direct initialization of transformers using larger pretrained ones. *arXiv*, 2023.

Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH*, 1985.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Llm pruning and distillation in practice: The minitron approach. *arXiv*, 2024.

Fan-Keng Sun, Cheng-Hao Ho, and Hung yi Lee. Lamol: Language modeling for lifelong language learning. In *ICLR*, 2019.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. *arXiv*, 2024.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Daniel Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *NeurIPS*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. *GitHub repository*, 2023.

Vithursan Thangarasa, Shreyas Saxena, Abhay Gupta, and Sean Lie. Sparse-IFT: Sparse iso-FLOP transformations for maximizing training efficiency. In *ICML*, 2024a.

Vithursan Thangarasa, Alex Tsaptsinos, Ian Milton, Ganesh Venkatesh, Valavan Manohararajah, Nish Sinnadurai, and Sean Lie. Llama3.1 model quality evaluation: Cerebras, groq, together, and fireworks. 2024b.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv*, 2024.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *ACL-IJCNLP*, 2021.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *ACL*, 2023.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *ICLR*, 2024.

Zhaorui Yang, Qian Liu, Tianyu Pang, Han Wang, H. Feng, Minfeng Zhu, and Wei Chen. Self-distillation bridges distribution gap in language model fine-tuning. In *ACL*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *ICLR*, 2023.

A Experimental Setup Details

A.1 Baseline Model

We used Llama3.1-8B Instruct[1](Dubey et al., 2024) as the baseline model for all experiments. This model comprises a total of 32 decoder layers, pretrained on a diverse array of instruction-following datasets. This model was chosen for its strong generalization performance across a wide range of natural language processing (NLP) tasks, making it an ideal candidate for studying the impact of structured pruning and fine-tuning. The 8B model size strikes a balance between computational efficiency and model quality, providing a robust foundation for the experiments in this study. Hence, served as the starting point for our structured layer pruning ablations and experiments in Sections 3 and 4, respectively.

A.2 Structured Layer Pruning

In this study, we focus on structured layer pruning of decoder layers to reduce the computational footprint of the LLM while maintaining its quality. Specifically, we prune in block sizes of {2, 4, 6, 8, 10} layers, corresponding to {30, 28, 26, 24, 22} decoder layers, respectively. Each block size reduction effectively removes a group of layers from the original architecture, creating progressively smaller models. These pruned models allow us to systematically evaluate the trade-offs between computational efficiency (fewer layers) and the accuracy on various downstream tasks. By examining multiple block sizes, we analyze how varying degrees of pruning impact model quality, especially in the context of *self-data distilled fine-tuning*, our proposed methodology.

A.3 Fine-tuning Datasets

The following datasets were used for ablation studies and fine-tuning experiments, representing a range of open-domain conversation, instruction-following, reasoning, and mathematical tasks:

- **Dolly 15k** (Conover et al., 2023) The Dolly dataset is an open-source collection of 15,000 instruction-following records generated by thousands of Databricks employees. It covers a wide range of behavioral categories, as outlined in InstructGPT(Ouyang et al., 2022), including brainstorming, classification, closed question answering (QA), generation, information extraction, open QA, and summarization. Dolly is designed to provide a benchmark for general-purpose instruction-following models, emphasizing diverse task types and behavioral categories.
- **GSM8k** (Cobbe et al., 2021) The GSM8k dataset is a collection of 8,000 high-quality grade-school-level math word problems, developed by OpenAI. Each problem is designed to assess a model's ability to perform multi-step reasoning and problem-solving, making it an essential benchmark for evaluating arithmetic, algebraic, and logical reasoning abilities in large models. Fine-tuning on GSM8k highlights the model's capacity for mathematical reasoning, a key focus of our ablation studies.
- **Alpaca Cleaned**[2] (Taori et al., 2023) The Alpaca Cleaned dataset is a cleaned version of the original Stanford Alpaca dataset, containing 50,000 instruction-following examples. It addresses several issues present in the original release, such as hallucinations, incorrect instructions, and output inconsistencies. This dataset provides high-quality general instruction-following tasks, spanning text generation, summarization, reasoning, and more. The cleaned version offers improved consistency and accuracy, making it ideal for fine-tuning large models in real-world instruction-following tasks.
- **OpenMathInstruct** (Toshniwal et al., 2024)T The OpenMathInstruct-1 dataset is specifically designed for fine-tuning language models on mathematical instruction tasks. It contains 1.8 million problem-solution pairs, generated using Mixtral-8x7B (Jiang et al., 2024). The problem sets are drawn from well-established mathematical benchmarks, including the GSM8K and MATH (Hendrycks et al., 2021b) datasets, ensuring a diverse and challenging range of mathematical reasoning tasks. Solutions are generated synthetically by allowing

[1] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`

[2] `https://huggingface.co/datasets/yahma/alpaca-cleaned`

the Mixtral model to leverage a combination of natural language reasoning and executable Python code, which allows for both symbolic computation and procedural solutions. This combination of text and code execution makes the dataset particularly suited for training models to handle complex reasoning, problem-solving, and algebraic tasks.

A.3.1 Data Sampling and Experimental Consistency

To maintain consistency across ablation studies, we fixed the dataset size at 8,000 samples for GSM8k, Alpaca, and OpenMathInstruct, aligning them with the standard GSM8k dataset size. However, the Dolly dataset retained its default size of 15,000 samples to preserve the integrity of this benchmark. To evaluate the impact of dataset size on self-data distillation, we extended the sample sizes for some experiments, using the full 50,000 samples from Alpaca Cleaned and randomly sampling 50,000 training samples from OpenMathInstruct. This allowed us to control for the effects of larger datasets, providing insights into how dataset size influences generalization and model retention following pruning.

A.4 Fine-tuning Pruned Models

For fine-tuning, we employed Low-Rank Adaptation (LoRA) (Hu et al., 2022), as it provides an efficient approach to training while preserving the pretrained model's capacity. Although full fine-tuning is feasible, we focused on LoRA fine-tuning in this study, leaving full parameter fine-tuning for future work. We conducted a comprehensive grid search on an 8k-sample version of the OpenMathInstruct dataset to identify the most effective hyperparameters for LoRA-based fine-tuning. The search was performed across a range of values to ensure optimal performance. We explored different *rank sizes* $\in \{4, 8, 16, 32\}$, aiming to balance model capacity and parameter efficiency. For the *number of epochs*, we tested values ranging $\in \{3, 5, 7, 10\}$, ensuring that the models were fine-tuned enough to converge without overfitting. The *learning rate* was swept across five values $\{2\times10^{-5}, 4\times10^{-5}, 6\times10^{-5}, 8\times10^{-5}, 1\times10^{-4}\}$. Finally, we tested *batch sizes* $\in \{8, 16, 32, 64, 128\}$ to determine the optimal balance between training stability and computational efficiency.

Through this grid search, the optimal configuration was identified as a *rank size* = 8, *epochs* = 5, a *batch size* = 64, and *learning rate* = 1×10^{-4}. These hyperparameters were used consistently across all fine-tuning experiments (i.e., both standard supervised fine-tuning and self-data distilled fine-tuning) in this study to ensure a fair comparison of the models and their quality post-pruning. We conduct our model training using `LLaMA-Factory v0.8.3`[3], a versatile framework designed for large-scale language model training and fine-tuning. This version offers extensive support for efficient parallelism, optimized memory usage, and integration with popular datasets, making it ideal for large model fine-tuning tasks such as those performed in this study.

A.4.1 Computational Resources

Fine-tuning and evaluations were conducted on Nvidia H100 GPUs. For experiments involving larger self-data distillation datasets, we utilized Cerebras CS-3 Inference (Thangarasa et al., 2024b), which achieves output generation speeds exceeding 1800 tokens per second. The CS-3 system was particularly useful for generating large-scale self-distilled datasets. However, for smaller datasets (e.g., up to 15k samples), the H100 GPUs were sufficient for both fine-tuning and generation.

B Extended Results on Fine-tuning Ablations

In this section, we provide extended results from our fine-tuning ablation study to further clarify the impact of dataset choice on self-data distillation efficacy in pruned Llama3.1-8B Instruct models. As detailed in the Section 3, we observed that self-data distillation consistently outperformed SFT across various datasets. Table 2 shows that the largest gains were achieved using the 50k-sample OpenMathInstruct dataset, particularly at medium and large pruning block sizes (e.g., block size 6). At this configuration, self-data distillation was able to recover 95.96% of the baseline model quality, which is a significant improvement compared to other datasets and fine-tuning methods. This result highlights the robustness of the self-data distillation process, especially in recovering quality post-pruning on reasoning-heavy tasks like those in GSM8k, ARC-C, and MMLU.

[3] `https://github.com/hiyouga/LLaMA-Factory/releases/tag/v0.8.3`

Moreover, the recovery rates exhibited a clear trend where, larger datasets such as the 50k OpenMath-Instruct consistently led to higher quality retention, especially when combined with more aggressive pruning. This suggests that the dataset's ability to approximate the model's original data distribution is critical for maintaining generalization capabilities after pruning. In contrast, smaller datasets like Alpaca or Dolly showed comparatively lower recovery rates, which further confirms the importance of dataset scale in the distillation process. Our results suggest that larger datasets are crucial for mitigating quality degradation in pruned models, with the 50k OpenMathInstruct dataset emerging as the most effective in retaining and enhancing model quality across block sizes, particularly in challenging reasoning tasks.

C Experimental Setup for Understanding Catastrophic Forgetting

To understand the impact of distribution shift on catastrophic forgetting, we conducted experiments using the baseline model (i.e., Llama3.1-8B Instruct) and its pruned variants fine-tuned with both supervised fine-tuning (SFT) and self-data distilled fine-tuning (Self-Data FT). Specifically, we pruned 6 decoder layers, reducing the model from 32 to 26 layers, and evaluated the models on the GSM8k dataset. For these experiments, we generated model responses using the baseline and pruned variants on the GSM8k dataset to capture how the distribution shift affects reasoning tasks post-pruning. Following Yang et al. (2024), to quantify the distribution shift, we employed Sentence-BERT (Reimers and Gurevych, 2019) to derive sentence embeddings from the model-generated responses. Then, similar to the method proposed by Zhang et al. (2023), we calculated the cosine similarity between the sentence embeddings of the pruned models and those generated by the original Llama3.1-8B Instruct model.

A lower cosine similarity score indicates a greater distribution shift, suggesting a higher risk of catastrophic forgetting. Conversely, higher similarity scores indicate better preservation of the original model's knowledge and a lower risk of forgetting. These metrics allowed us to assess the extent to which SFT and Self-Data FT preserved the learned distribution of the base model, with the latter showing superior performance in mitigating forgetting, as detailed in our ablations in Section 3.

D Model Merging Self-Data Distilled Models

We employ the Spherical Linear Interpolation (SLERP) method for merging pruned models, which ensures smooth, geometrically consistent interpolation between two pruned model parameter vectors. SLERP operates within the unit sphere's geometry, contrasting with traditional linear interpolation that may destabilize or yield suboptimal parameter combinations by ignoring the geometric properties of the high-dimensional parameter space. SLERP preserves model integrity during interpolation, leading to more stable and consistent outcomes.

Given two pruned model parameter vectors, $\boldsymbol{\theta}'_0$ and $\boldsymbol{\theta}'_1$, corresponding to pruned models M'_0 (fine-tuned on OpenMathInstruct) and M'_1 (fine-tuned on Alpaca), SLERP generates an interpolated parameter vector $\boldsymbol{\theta}'_t$ for any interpolation factor $t \in [0, 1]$. When $t = 0$, the parameters of the OpenMathInstruct fine-tuned model $\boldsymbol{\theta}'_0$ are retrieved, and when $t = 1$, the parameters of the Alpaca fine-tuned model $\boldsymbol{\theta}'_1$ are retrieved.

Normalization to Unit Sphere The first step in SLERP is to normalize both pruned model parameter vectors to lie on the unit sphere,

$$\hat{\boldsymbol{\theta}}'_0 = \frac{\boldsymbol{\theta}'_0}{\|\boldsymbol{\theta}'_0\|}, \quad \hat{\boldsymbol{\theta}}'_1 = \frac{\boldsymbol{\theta}'_1}{\|\boldsymbol{\theta}'_1\|}.$$

This normalization ensures that both parameter vectors have unit norms, placing them on the surface of the unit sphere in the parameter space. Next, we compute the angle θ_{angle} between the normalized pruned model vectors $\hat{\boldsymbol{\theta}}'_0$ and $\hat{\boldsymbol{\theta}}'_1$. This angle is computed using the dot product, $\cos(\theta_{\text{angle}}) = \hat{\boldsymbol{\theta}}'_0 \cdot \hat{\boldsymbol{\theta}}'_1$, and the actual angle is given by $\theta_{\text{angle}} = \arccos(\cos(\theta_{\text{angle}}))$. This angle represents the angular separation between the two pruned models' parameter vectors on the unit sphere.

Spherical Interpolation With the angle θ_{angle} determined, SLERP performs spherical interpolation along the great circle connecting $\hat{\boldsymbol{\theta}}'_0$ and $\hat{\boldsymbol{\theta}}'_1$. The interpolated parameter vector $\boldsymbol{\theta}'_t$ is computed as,

$$\boldsymbol{\theta}'_t = \frac{\sin((1-t)\theta_{\text{angle}})}{\sin(\theta_{\text{angle}})} \cdot \hat{\boldsymbol{\theta}}'_0 + \frac{\sin(t\theta_{\text{angle}})}{\sin(\theta_{\text{angle}})} \cdot \hat{\boldsymbol{\theta}}'_1.$$

This formula ensures that the interpolation remains on the surface of the unit sphere, respecting the geometric structure of the parameter space. The interpolation factor t controls the contribution from each pruned model, when $t = 0$, $\boldsymbol{\theta}'_t = \hat{\boldsymbol{\theta}}'_0$ (i.e., OpenMathInstruct fine-tuned model), and when $t = 1$, $\boldsymbol{\theta}'_t = \hat{\boldsymbol{\theta}}'_1$ (i.e., Alpaca fine-tuned model). The intermediate values of t produce a smooth, spherical blend of the two pruned models.

D.1 Geometric Consistency and Application

By operating within the unit sphere, SLERP respects the *Riemannian geometry* of the high-dimensional parameter space, ensuring a smooth transition between the two pruned models. Traditional linear interpolation in such spaces can distort the relationships between parameters, leading to suboptimal combinations and degraded model performance. In contrast, SLERP maintains geometric consistency, ensuring that the interpolation follows a natural path on the unit sphere.

Merging the pruned OpenMathInstruct and Alpaca models using SLERP combines the unique strengths of both models. For instance, OpenMathInstruct's emphasis on mathematical reasoning and logical structure complements Alpaca's broader instruction-following capabilities. By adjusting the interpolation factor t, the merged model can balance these capabilities, resulting in a versatile and robust model for a range of downstream tasks. We use Arcee.ai's `mergekit`[4] for efficiently merging model checkpoints.

[4] `https://github.com/arcee-ai/mergekit`

Table 2: **Model quality results for pruned Llama3.1-8B Instruct models across various pruning block sizes and fine-tuning strategies.** This table reports the quality of different fine-tuning methods (No Fine-tuning, Standard Fine-tuning (SFT), and Self-Data Distillation) on various datasets, with average accuracy across ARC-C, GSM8k, and MMLU tasks. The "Avg. Recovery" column shows the percentage of model quality recovered relative to the unpruned baseline. The table highlights that the self-data distillation strategy consistently yields superior recovery rates, particularly with the 50k-sample OpenMathInstruct dataset. For instance, at a pruning block size of 6, the self-data distilled OpenMathInstruct model retains 95.96% of the original unpruned Llama3.1-8B Instruct (i.e., 32 layers) model's quality, the highest recovery observed among all datasets and fine-tuning methods.

Prune Block Size	Model Savings	Fine-tuning Method	Dataset	ARC-C (25-shot)	GSM8k (5-shot)	MMLU (5-shot)	Avg. Score	Avg. Recovery
Baseline	-	No FT		58.70	63.15	67.40	63.08	100.00%
2	5.43% (7.59B)	No FT		55.20	67.79	56.18	59.72	94.67%
		SFT	GSM8k	**58.45**	56.25	65.22	59.97	95.07%
		Self-Data Distillation	GSM8k	57.34	64.44	66.60	62.79	99.54%
		SFT	Dolly	55.67	61.64	65.71	61.01	96.71%
		Self-Data Distillation	Dolly	56.48	62.24	66.46	61.73	97.87%
		SFT	Alpaca (50k)	56.61	63.19	65.60	61.80	97.98%
		Self-Data Distillation	Alpaca (50k)	56.91	68.60	**66.50**	63.34	100.41%
		SFT	OpenMathInstruct (50k)	52.91	44.95	60.93	52.93	83.88%
		Self-Data Distillation	OpenMathInstruct (50k)	56.43	**69.97**	66.17	**64.19**	**101.76%**
4	10.86% (7.16B)	No FT		55.20	56.18	**67.79**	59.72	94.67%
		SFT	GSM8k	54.27	43.47	65.40	54.38	86.22%
		Self-Data Distillation	GSM8k	55.20	62.55	66.68	61.48	97.49%
		SFT	Dolly	54.78	59.36	63.65	59.26	93.95%
		Self-Data Distillation	Dolly	51.71	55.96	65.40	57.69	91.44%
		SFT	Alpaca (50k)	56.05	54.40	65.34	58.60	92.89%
		Self-Data Distillation	Alpaca (50k)	**57.27**	66.20	66.24	**63.24**	**100.26%**
		SFT	OpenMathInstruct (50k)	51.62	44.35	61.65	52.54	83.30%
		Self-Data Distillation	OpenMathInstruct (50k)	53.93	**69.44**	65.34	62.24	98.66%
6	16.30% (6.72B)	No FT		49.49	0.00	**67.42**	48.93	70.50%
		SFT	GSM8k	46.67	62.09	64.67	57.81	91.63%
		Self-Data Distillation	GSM8k	51.45	60.05	66.28	59.93	95.02%
		SFT	Dolly	47.18	33.89	65.33	48.13	76.31%
		Self-Data Distillation	Dolly	51.96	50.95	62.56	55.82	88.47%
		SFT	Alpaca (50k)	**54.62**	56.11	64.39	58.37	92.53%
		Self-Data Distillation	Alpaca (50k)	53.80	59.15	66.29	59.75	94.71%
		SFT	OpenMathInstruct (50k)	46.93	43.82	59.98	50.91	80.71%
		Self-Data Distillation	OpenMathInstruct (50k)	50.00	**66.64**	64.96	**60.53**	**95.96%**
8	21.73% (6.29B)	No FT		44.71	0.00	**65.57**	36.76	58.27%
		SFT	GSM8k	44.79	50.86	64.38	53.34	84.56%
		Self-Data Distillation	GSM8k	46.16	50.11	65.53	53.93	85.50%
		SFT	Dolly	39.33	15.39	57.44	37.39	59.27%
		Self-Data Distillation	Dolly	46.50	28.73	62.93	46.05	73.00%
		SFT	Alpaca (50k)	**49.15**	39.59	64.81	51.85	82.19%
		Self-Data Distillation	Alpaca (50k)	48.09	42.77	65.20	52.69	83.51%
		SFT	OpenMathInstruct (50k)	42.15	29.64	60.65	44.81	71.05%
		Self-Data Distillation	OpenMathInstruct (50k)	46.67	**57.70**	64.87	**56.41**	**89.44%**
10	27.16% (5.85B)	No FT		37.46	0.00	64.09	33.85	53.65%
		SFT	GSM8k	39.85	37.45	61.47	46.92	74.36%
		Self-Data Distillation	GSM8k	41.55	37.47	62.33	47.12	74.70%
		SFT	Dolly	38.40	0.76	46.94	28.70	45.51%
		Self-Data Distillation	Dolly	43.60	12.28	62.47	39.45	62.53%
		SFT	Alpaca (50k)	**45.22**	17.51	63.72	42.82	67.88%
		Self-Data Distillation	Alpaca (50k)	44.91	20.05	**64.73**	43.90	69.56%
		SFT	OpenMathInstruct (50k)	39.33	15.39	57.44	37.39	59.25%
		Self-Data Distillation	OpenMathInstruct (50k)	40.88	**44.58**	64.54	**50.33**	**79.79%**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: **The paper's abstract clearly states the main contribution of proposing self-data distilled fine-tuning to mitigate the performance loss from structured pruning. The experimental results and scope align with the claims regarding quality recovery and generalization across datasets.**

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: **The paper acknowledges that while self-data distillation improves quality recovery, it does not fully preserve the original model's performance and may not generalize equally across all tasks. It also briefly mentions the computational trade-offs for dataset size.**

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: **The paper does not focus on theoretical proofs but instead relies on empirical evaluations and methods without formal theoretical results.**

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: **The paper provides sufficient details on the datasets, model configurations, fine-tuning methods, and pruning strategies to allow reproduction of the experiments**

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: **The paper does not mention whether the code or datasets are publicly available, although it uses open datasets like OpenMathInstruct.**

   Guidelines:

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: **The paper provides detailed descriptions of dataset splits, hyperparameters, and fine-tuning methods, which are necessary for understanding the results.**

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: **Standard error is included in the Eleuther eval harness but is often omitted in LLM literature as variance across runs is small, especially with large datasets. Many papers prioritize reporting mean performance for quick comparisons in high-level benchmarks (e.g., HuggingFace Leaderboard), where standard error is seen as redundant.**

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: **The paper provides information about the use of Nvidia H100 GPUs and Cerebras CS-3 systems for larger datasets.**

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: **There is no indication of ethical violations in the research, and the methods seem to adhere to standard practices in NLP research.**

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: **The paper does not discuss broader societal impacts, though the application of model compression has implications for resource efficiency and accessibility.**

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: **The paper does not include high-risk models, and thus safeguards are not required.**

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: **The paper cites and credits datasets like OpenMathInstruct and Dolly, respecting their terms of use.**

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: **No new assets are introduced; the focus is on existing models and datasets.**

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: **The paper does not involve research with human subjects.**

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: **No human subjects are involved.**