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

Vithursan Thangarasa[1] Ganesh Venkatesh[1] Mike Lasby[2*] Nish Sinnadurai[1] Sean Lie[1]

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 utilize *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 six 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 16.3%. Furthermore, combining self-data distilled models through model merging yields enhanced quality retention. Additionally, leveraging these pruned models in speculative decoding increases token acceptance rates, thereby improving inference efficiency in applied settings.

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 compute costs for both training and inference (Kaplan et al., 2020; Hoffmann et al., 2022). 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: knowledge distillation (KD) (Hinton et al., 2015), 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 method 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 compute requirements, making inference more efficient. Recent works have shown that LLMs exhibit significant redundancy, particularly in the middle layers, where removing these layers has a minimal impact on overall model quality (Men et al., 2024; Gromov et al., 2024). The residual stream of the Transformer (Vaswani et al., 2017) architecture is only slightly modified by the output of non-essential layers, enabling the removal of these 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

[1] Cerebras Systems, Sunnyvale, California [2] University of Calgary, Calgary, Alberta. [*] Work done during an internship at Cerebras. Correspondence to: Vithursan Thangarasa <vithu@cerebras.net>.

Proceedings of the 8^{th} MLSys Conference, Santa Clara, CA, USA, 2025. Copyright 2025 by the author(s).

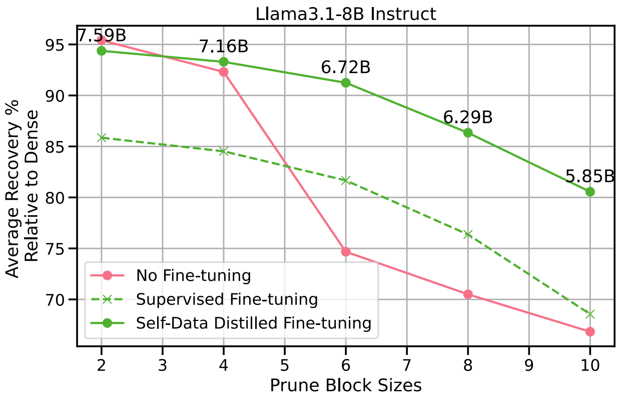

Figure 1. **Average quality recovery (%) of pruned Llama3.1-8B Instruct models relative to the unpruned baseline, across varying prune block sizes on the HuggingFace OpenLLM Leaderboard v1.** The plot compares no fine-tuning, supervised fine-tuning, and self-data distilled fine-tuning using the OpenMathInstruct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

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 & Cohen, 1989; Kotha et al., 2024), 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.

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 accuracy recovery by greater than 10% 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%. In addition to improving the quality of pruned models, our work explores extending self-data distilled fine-tuning to speculative decoding (Leviathan et al., 2023; Chen et al., 2023). This approach allows pruned models to not only recover their accuracy but also achieve efficient inference through parallel token generation. The integration of self-data distillation with speculative decoding is a natural extension, as it leverages the semantic alignment achieved during data distillation to enhance the speculative capabilities of pruned models. By aligning the pruned model's predictions more closely with the target model, speculative decoding can benefit from improved token acceptance rates, reducing latency while preserving model quality. 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 and Mistral-7B-v0.3 Instruct (Jiang et al., 2023) 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.
- We extend self-data distilled fine-tuning to speculative decoding, showing that it enhances token acceptance rates and reduces latency during inference. This extension highlights the broader applicability of our approach, supporting both compression and acceleration of large language models.

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

Transformers (Vaswani et al., 2017) have become a foundational architecture in deep learning, particularly for tasks

Algorithm 1 Layer-Pruning Language Models

```
Input: Model $M$ with $L$ layers, Number of layers to prune $n$, Dataset $D$
Output: Pruned model $M'$
Initialize $\ell^\star \leftarrow$ None, $d_{\min} \leftarrow \infty$
for each layer $\ell$ from 1 to $L - n$ do
    $h^{(\ell)}(D) \leftarrow$ activation at $\ell$ with input $D$
    $h^{(\ell+n)}(D) \leftarrow$ activation at $\ell + n$ with input $D$
    Compute $d(h^{(\ell)}(D), h^{(\ell+n)}(D))$ using Eq. 1
    if $d(h^{(\ell)}(D), h^{(\ell+n)}(D)) < d_{\min}$ then
        $d_{\min} \leftarrow d(h^{(\ell)}(D), h^{(\ell+n)}(D))$
        $\ell^\star \leftarrow \ell$
    end if
end for
Prune layers $\ell^\star$ to $\ell^\star + n - 1$ from $M$
Connect output of layer $\ell^\star$ to input of layer $\ell^\star + n$
return pruned model $M'$
```

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 (e.g., 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.

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 (see more details in Appendix A.3). The angular distance metric, $d(h^{(\ell)}(D), h^{(\ell+n)}(D))$, is formally defined as,

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

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.

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 $\ell^\star + n$.

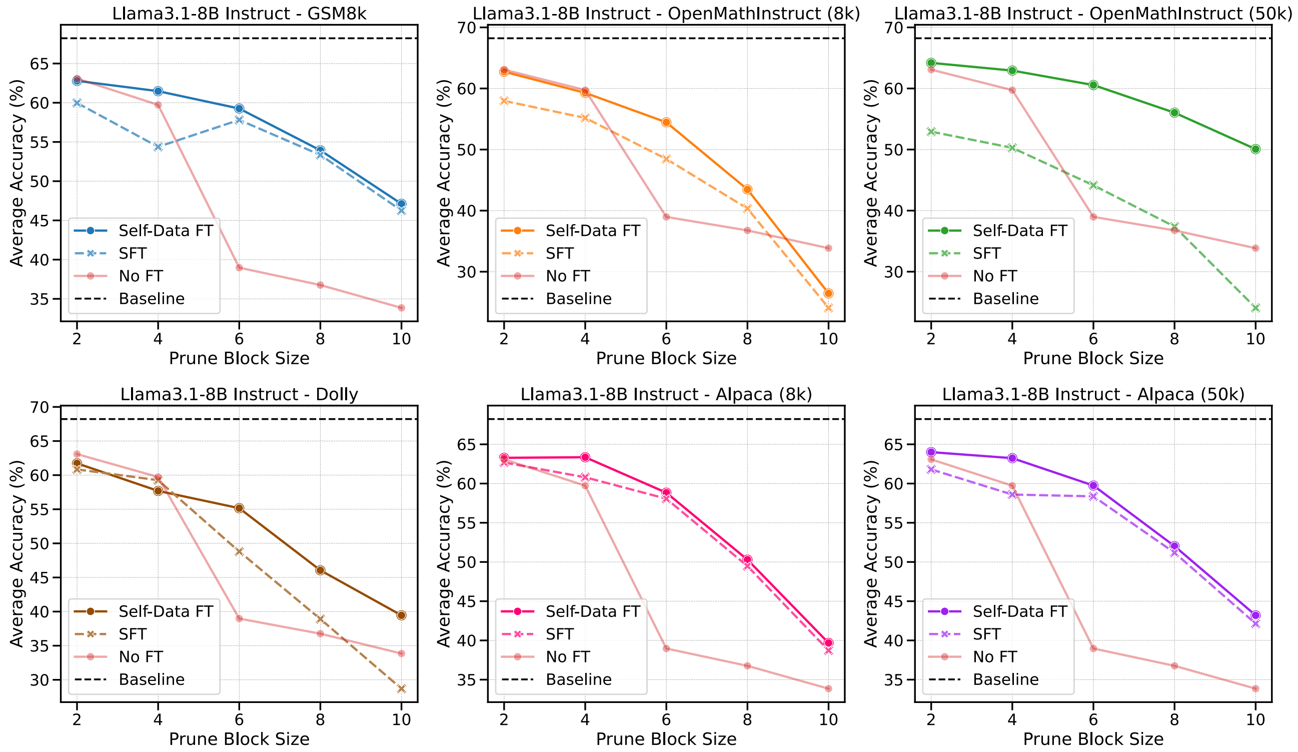

Figure 2. **Quality of pruned Llama3.1-8B Instruct models across various datasets and pruning block sizes.** The plots show average accuracy across MMLU, GSM8k, ARC-C tasks for GSM8k, OpenMathInstruct, Dolly, and Alpaca under three strategies: Self-Data FT, SFT, and No FT. Self-Data FT consistently outperforms SFT and No FT, with the largest gains using OpenMathInstruct (50k).

Aligning with Sreenivas et al. (2024), we adopt a one-shot pruning approach that balances implementation simplicity with computational efficiency. Their method demonstrates that identifying and removing redundant layers in one operation can achieve significant parameter reduction while maintaining model quality when paired with appropriate distillation. While our approach prioritizes this one-shot strategy, we recognize that iterative pruning techniques, which progressively remove smaller network portions with re-training between steps, may offer benefits in certain architectural contexts. This represents a promising direction for future research, particularly for models where more fine-grained control over the accuracy-efficiency tradeoff is desired.

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 utilize *self-data distilled fine-tuning* (Yang et al., 2024), which aligns the fine-tuning dataset with the original model's learned distribution, to mitigate forgetting and maintain model quality across a diverse range of 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

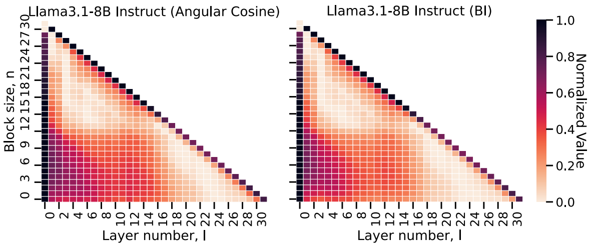

Figure 3. Comparison of Llama3.1-8B Instruct using (left) *angular cosine* and (right) *block influence* (BI) score metrics. Both metrics highlight inherent redundancy in the middle layers of Llama3.1-8B Instruct, suggesting that these layers can be pruned with minimal impact on overall model quality.

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,

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

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

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.

3.1 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 4). 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 the middle layers, both metrics produced comparable pruning results across block sizes. We ultimately chose the angular cosine metric for its computational efficiency, as it uses cosine similarity on only the last token, whereas BI operates over the entire sequence, adding computational overhead. This efficiency makes it more scalable for LLMs, and its direct measure of output similarity aligns with our intuition that layers producing highly similar outputs are likely redundant, making it a practical tool for structured pruning in this study.

3.2 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 2, 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 achieve 5-10% higher accuracy than models without any fine-tuning for pruning block sizes above four. Across all block sizes, they also outperform models fine-tuned with SFT, 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

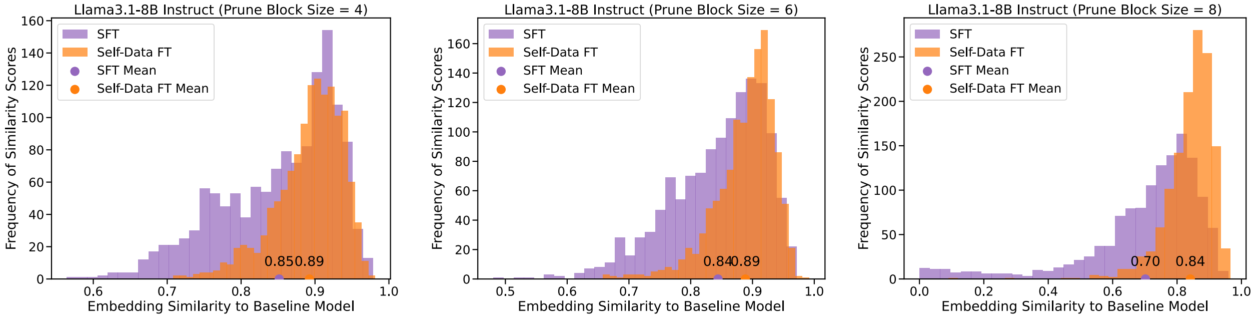

Figure 4. **Distribution of embedding similarities on GSM8k test dataset.** We present the distribution of embedding similarities after fine-tuning a structurally pruned variant of the Llama3.1-8B Instruct model on 50k samples of OpenMathInstruct. We compute the cosine similarity between the sentence embeddings of the pruned models and those generated by the original Llama3.1-8B Instruct model. The plots show: (left) 28 decoder layers (prune block size = 4), (center) 26 decoder layers (prune block size = 6), and (right) 24 decoder layers (prune block size = 8). Self-Data Distilled Fine-Tuning (Self-Data FT) achieves higher similarity to the original baseline model, indicating a reduced distribution shift compared to Supervised Fine-Tuning (SFT).

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 outperforms standard supervised fine-tuning (SFT).

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	-
2	5.23% (7.61B)	No FT		**58.71**	**78.12**	**53.21**	63.15	**76.72**	67.39	**66.22**	**95.41%**
		SFT	OpenMathInstruct	52.91	73.24	51.77	60.91	73.75	44.95	59.59	85.85%
		Self-Data Distillation	OpenMathInstruct	56.41	75.03	50.23	**66.17**	75.18	**69.97**	65.51	94.37%
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%**
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	**50.02**	**71.57**	53.14	64.96	**73.64**	**66.64**	**63.33**	**91.24%**
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%**
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%**

process, with the 50k-sample OpenMathInstruct dataset delivering the best overall results, and will be the focus of subsequent experiments.

3.3 Mitigating Catastrophic Forgetting

Figure 4 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 plots compares the sentence embeddings of the two models' generated responses on the GSM8k task to the baseline unpruned Llama3.1-8B Instruct model at various prune block sizes. Self-Data FT maintains a narrower distribution of embedding similarities, with higher mean scores, indicating better preservation of the original model's learned representations. In contrast, SFT yields a wider spread and reduced mean in similarity scores, indicative of a heavier-tailed distribution. This shift suggests a more pronounced distributio shift, thereby increasing the 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 and Mistral-7B-v0.3 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 with-

Table 2. **Model quality results for pruned Mistral-7B-v0.3 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 standard supervised fine-tuning (SFT).

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		63.39	84.52	59.65	61.99	78.37	48.75	66.11	-
2	6.02% (6.83B)	No FT		60.24	62.31	58.57	62.11	**78.77**	32.75	59.12	89.43%
		SFT	OpenMathInstruct	54.01	**79.38**	**53.71**	57.61	74.82	34.04	58.93	89.13%
		Self-Data Distillation	OpenMathInstruct	**53.93**	74.27	50.61	**65.34**	74.90	**69.44**	**64.75**	**93.29%**
4	12.03% (6.39B)	No FT		**57.17**	**78.89**	**57.44**	**61.36**	**77.19**	9.33	56.90	86.06%
		SFT	OpenMathInstruct	52.65	75.29	53.16	58.21	74.51	27.22	56.84	85.97%
		Self-Data Distillation	OpenMathInstruct	55.29	75.77	53.33	60.12	75.22	**34.65**	**59.06**	**89.34%**
6	18.06% (5.96B)	No FT		**50.25**	**72.67**	**55.85**	**61.56**	**75.45**	0.83	52.77	79.82%
		SFT	OpenMathInstruct	47.52	70.15	51.86	58.29	74.91	20.47	53.87	81.47%
		Self-Data Distillation	OpenMathInstruct	48.98	70.67	52.28	59.56	72.22	**28.13**	**55.31**	**83.65%**
8	24.08% (5.52B)	No FT		41.04	65.27	**58.10**	**61.05**	71.74	0.00	49.53	74.92%
		SFT	OpenMathInstruct	43.09	66.11	52.74	58.52	**72.21**	9.48	50.35	76.16%
		Self-Data Distillation	OpenMathInstruct	**45.14**	**67.09**	53.61	59.81	71.43	**19.03**	**52.68**	**79.69%**
10	30.10% (5.08B)	No FT		34.30	52.14	**58.67**	35.76	65.11	0.00	41.00	62.01%
		SFT	OpenMathInstruct	39.33	59.07	52.07	57.49	68.75	6.07	47.13	71.29%
		Self-Data Distillation	OpenMathInstruct	**39.76**	**59.25**	51.61	**58.42**	**69.61**	**14.39**	**48.84**	**73.87%**

Table 3. **Model quality results for pruned Llama3.1-8B Instruct models across various pruning block sizes.** Average accuracy is reported on the OpenLLM Leaderboard v1, along with the recovery percentage relative to the baseline model. Self-data distillation with model merging (MM) consistently outperforms self-data distillation with mixed datasets.

Prune Block Size	Fine-tuning Method	Dataset	Avg. Score	Avg. Recovery
Baseline	No FT		69.40	-
4	No FT		64.06	92.31%
	Self-Data Distillation	OpenMathInstruct + Alpaca	65.27	94.04%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**65.84**	**94.86%**
6	No FT		51.82	74.67%
	Self-Data Distillation	OpenMathInstruct + Alpaca	63.77	91.88%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**64.75**	**93.30%**
8	No FT		48.93	70.50%
	Self-Data Distillation	OpenMathInstruct + Alpaca	59.37	85.54%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**61.24**	**88.24%**
10	No FT		46.39	66.83%
	Self-Data Distillation	OpenMathInstruct + Alpaca	55.85	80.47%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**56.08**	**80.70%**

out 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%. Similarly, in Table 2, the Mistral-7B models exhibit improved quality recovery across all pruning block sizes. These findings establish self-data distillation as an effective method for preserving model quality post-pruning, particularly for reasoning-intensive tasks, making it essential for large-scale compression.

Table 4. **Model quality results for pruned Mistral-7B-v0.3 Instruct models across various pruning block sizes.** Average accuracy is reported on the OpenLLM Leaderboard v1, along with the recovery percentage relative to the baseline model. Self-data distillation with model merging (MM) consistently outperforms self-data distillation with mixed datasets.

Prune Block Size	Fine-tuning Method	Dataset	Avg. Score	Avg. Recovery
Baseline	No FT		66.11	-
4	No FT		56.90	86.06%
	Self-Data Distillation	OpenMathInstruct + Alpaca	59.04	89.30%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**59.57**	**90.10%**
6	No FT		52.77	79.82%
	Self-Data Distillation	OpenMathInstruct + Alpaca	56.26	85.09%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**56.41**	**85.32%**
8	No FT		49.53	74.92%
	Self-Data Distillation	OpenMathInstruct + Alpaca	52.78	79.83%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**54.14**	**81.89%**
10	No FT		41.00	62.01%
	Self-Data Distillation	OpenMathInstruct + Alpaca	48.86	73.90%
	Self-Data Distillation + MM	OpenMathInstruct + Alpaca	**50.04**	**75.69%**

4.1 Self-Data Distillation with Model Merging

We extend self-data distillation by introducing *model merging* using Spherical Linear Interpolation (SLERP) (Shoemake, 1985). The model merging technique aims to synthesize complementary knowledge from models fine-tuned on distinct datasets, potentially leading to improved generalization and robustness after pruning. While various model merging techniques, such as TIES (Yadav et al., 2023), DARE-TIES (Yu et al., 2024), and Linear Merge (Nagarajan & Kolter, 2019), have been proposed, prior empirical studies have demonstrated SLERP to be particularly effective (Aakanksha et al., 2024), as it effectively balances the dual objectives of maintaining alignment and preserving overall model quality. SLERP enables smooth interpolation between two model parameters along the shortest path on a high-dimensional sphere, preserving the geometric proper-

Table 5. **Evaluation of speculative decoding performance using self-data distilled Llama3.1-8B Instruct draft models for the Llama3.1-70B Instruct target model.** The draft models generate 6 speculative tokens plus 1 base token from the target model, with accepted lengths measured across all tasks from Spec-Bench. Results are shown for various pruning block sizes, where self-data distillation with MM consistently improves over the no fine-tuning (No FT) baseline, with higher average accepted lengths indicating better speculative capabilities and better alignment with the target model. At a prune block size of 10, we achieve 27.16% FLOP savings, and the average accepted length across tasks (final column) improves by 1.70.

				Average Accepted Length on Spec-Bench						
Target Model	Prune Block Size	Model Savings	Fine-tuning Method	MT-Bench	Translation	Summarization	QA	Math Reasoning	RAG	Average
Llama3.1-70B Instruct (70B)	Baseline	-	No FT	5.03	4.25	4.72	4.86	5.86	4.73	5.01
	4	10.86%	No FT	3.50	3.18	3.62	3.16	4.26	3.56	3.55
		(7.16B)	Self-Data Distillation + MM	4.26	3.71	4.13	3.53	5.56	4.13	**4.22**$_{\uparrow 0.67}$
	6	16.30%	No FT	2.77	2.40	2.72	2.35	3.13	2.65	2.67
		(6.72B)	Self-Data Distillation + MM	3.93	3.40	3.91	3.14	5.30	3.83	**3.92**$_{\uparrow 1.25}$
	8	21.73%	No FT	2.38	2.26	2.70	2.11	2.89	2.55	2.40
		(6.29B)	Self-Data Distillation + MM	3.63	3.17	3.77	2.91	4.99	3.41	**3.65**$_{\uparrow 1.25}$
	10	27.16%	No FT	1.58	1.48	1.50	1.48	1.63	1.49	1.55
		(5.85B)	Self-Data Distillation + MM	3.17	2.76	3.21	2.49	4.59	3.30	**3.25**$_{\uparrow 1.70}$

Table 6. **Evaluation of speculative decoding performance using self-data distilled Mistral-7B-v0.3 Instruct draft models for the Mistral Large 2 (123B) target model.** Results on Spec-Bench are shown for various pruning block sizes, highlighting the effectiveness of self-data distillation paired with model merging (MM). Self-data distillation with MM consistently improves over the no fine-tuning baseline, with higher average accepted lengths indicating better speculative capabilities and better alignment with the target model. At a prune block size of 10, we achieve 30.10% FLOP savings, and the average accepted length across tasks (final column) improves by 1.62.

				Average Accepted Length on Spec-Bench						
Target Model	Prune Block Size	Model Savings	Fine-tuning Method	MT-Bench	Translation	Summarization	QA	Math Reasoning	RAG	Average
Mistral Large 2 (123B)	Baseline	-	No FT	3.98	4.36	3.70	3.50	4.85	3.85	4.04
	4	12.03%	No FT	3.40	3.76	3.28	2.77	4.28	3.22	3.45
		(6.83B)	Self-Data Distillation + MM	3.70	4.06	3.53	2.99	4.81	3.44	**3.75**$_{\uparrow 0.30}$
	6	18.06%	No FT	2.97	3.30	2.93	2.36	3.78	2.82	3.03
		(5.96B)	Self-Data Distillation + MM	3.53	3.86	3.41	2.79	4.72	3.30	**3.60**$_{\uparrow 0.58}$
	8	24.08%	No FT	1.98	2.16	2.03	1.76	2.66	1.98	2.10
		(5.52B)	Self-Data Distillation + MM	3.37	3.68	3.25	2.58	4.62	3.10	**3.43**$_{\uparrow 1.34}$
	10	30.10%	No FT	1.50	1.55	1.54	1.45	1.73	1.59	1.56
		(5.06B)	Self-Data Distillation + MM	3.15	3.25	2.97	2.38	4.48	2.87	**3.18**$_{\uparrow 1.62}$

ties of the parameter space (see Appendix D for details).

Building on the gains achieved by self-data distillation in recovering quality post-pruning, we investigate whether merging models fine-tuned on diverse datasets can provide further improvements. We perform model merging within each model family by merging Llama3.1-8B Instruct models fine-tuned on OpenMathInstruct and Alpaca, and similarly merging Mistral-7B models fine-tuned on the same datasets, as these configurations delivered the best results in our ablations in Section 3. To establish a robust baseline for comparison against model merging, we combined the OpenMathInstruct and Alpaca datasets, fine-tuning the pruned models on this interleaved dataset. The total number of training samples was kept consistent across both approaches to ensure a fair evaluation.

As presented in Table 3 for Llama3.1 models and Table 4 for Mistral-7B models, our results indicate that self-data distillation combined with model merging via SLERP achieves the highest recovery of model quality across all pruning block sizes. Specifically, at a pruning block size of 6, the merged Llama3.1 model demonstrates a 93.30% recovery of its original quality, surpassing the 91.88% recovery observed with models trained on a mixed dataset. Similarly, the Mistral-7B model achieves an 85.32% recovery at block size 6 through merging, slightly exceeding the 85.09% recovery obtained with the mixed dataset approach. These results highlight the consistency and effectiveness of self-data distillation with model merging in retaining model quality across varying pruning configurations. 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.

Based on these empirical results, we derive several important insights for effective model pruning and quality recovery. First, using robust, instruction-tuned base models is critical as they provide a stable foundation for pruning and subsequent fine-tuning operations. We observe that both

Llama3.1-8B Instruct and Mistral-7B-v0.3 Instruct models benefit significantly from our approach precisely because they start from a well-calibrated instruction-following state. Second, incorporating diverse, representative datasets during fine-tuning and distillation—as demonstrated by our OpenMathInstruct and Alpaca combination—minimizes distribution mismatches and improves generalization across varied tasks. This diversity is particularly important for maintaining performance on reasoning-intensive benchmarks after aggressive pruning. Finally, our method highlights the flexibility enabled by decoupling pre-training and fine-tuning phases, allowing self-data distillation to effectively transfer knowledge across various contexts while maintaining alignment with the original model's capabilities. This architectural separation ensures that pruned models can be effectively adapted to specific deployment scenarios without compromising their fundamental abilities.

4.2 Speculative Decoding and Self-Data Distillation

Speculative decoding is founded on two key insights into the inference dynamics of LLMs. First, generating tokens with a smaller draft model reduces computational cost by offloading initial token prediction from the larger target model. Second, LLM inference is primarily constrained by memory bandwidth, with latency bottlenecks arising from parameter memory access rather than arithmetic operations (Patterson, 2004; Shazeer, 2019). By adapting principles from speculative execution, speculative decoding aims to prioritize the verification of pre-drafted tokens, thereby minimizing frequent memory operations and enhancing overall inference efficiency. Despite its potential, speculative decoding raises several challenges that require deeper exploration. Central among these is the need to design an effective drafting mechanism that optimally balances speculative accuracy with drafting efficiency (Xia et al., 2023; Li et al., 2024). The effectiveness of this approach depends on two critical factors: the accuracy of the draft model, measured by the average number of accepted tokens per decoding step, and the drafting latency itself (Stern et al., 2018; Xia et al., 2023). Achieving a trade-off between high speculative accuracy and low latency remains a significant challenge, as both elements are essential for maximizing the overall speedup.

While not the main focus of this work, speculative decoding is a natural application of our method. In this context, integrating self-data distillation with speculative decoding presents a viable solution to improve inference performance. Self-data distillation aligns the draft model more closely with the target model by generating a distilled dataset that preserves semantic consistency even after pruning. This integration not only enhances the draft model's speculative capabilities but also mitigates quality degradation, allowing for more efficient and accurate token validation during inference. Consequently, the combination of speculative decoding and self-data distillation emerges as a promising approach for improving both inference speed and model quality retention across diverse tasks.

The draft-and-verify setup in speculative decoding involves two phases: *drafting* and *verification*. In the drafting phase, a smaller draft model M_p speculates multiple future tokens. Given an input sequence $\{x_1, \ldots, x_n\}$, it generates K speculative tokens $\tilde{x}_j$ by sampling from probability distributions p_j, where $\{p_1, \ldots, p_K\} = \text{Draft}(x_{\leq n}, M_p)$ and $\tilde{x}_j \sim p_j$ for $j = \{1, \ldots, K\}$. In the verification phase, the larger target model M_q verifies these tokens by computing probabilities for the drafted tokens as $q_j = M_q(x \mid x_{\leq n}, \tilde{x}_{<j})$ for $j = \{1, \ldots, K+1\}$. Each token $\tilde{x}_j$ is then checked against a criterion $\text{Verify}(\tilde{x}_j, p_j, q_j)$, where in our experimental setup, verification is performed using $\tilde{x}_j = \arg\max q_j$. If a token fails verification, the first failing token $\tilde{x}_c$ is corrected via $\text{Correct}(p_c, q_c)$, specifically as $x_{n+c} \leftarrow \arg\max q_c$, and subsequent tokens are discarded. If all tokens pass verification, an additional token x_{n+K+1} is sampled from the target model. This process repeats until the [EOS] token is generated or a maximum sequence length is reached.

Experimental Setup Our speculative decoding experiments use pruned, self-data distilled models with different pruning block sizes $n \in \{4, 6, 8, 10\}$. These models are evaluated on the Spec-Bench dataset (Xia et al., 2024a), a comprehensive benchmark that covers categories including multi-turn chat, translation, summarization, question answering (QA), mathematical reasoning, and retrieval-augmented generation (RAG). There are a total of 480 samples in Spec-Bench, where each category in Spec-Bench consists of 80 randomly selected instances drawn from six well-established datasets: MT-Bench (Zheng et al., 2024), WMT14 DE-EN (Nallapati et al., 2016), CNN/Daily Mail (Nallapati et al., 2016), Natural Questions (Kwiatkowski et al., 2019), GSM8K (Cobbe et al., 2021), and DPR (Karpukhin et al., 2020).

We report the overall average accepted token lengths to measure the effectiveness of speculative decoding. Our speculative decoding configuration generates $K = 6$ speculative tokens $\{\tilde{x}_1, \ldots, \tilde{x}_6\}$ using the draft model M_p, while the target model M_q concurrently verifies these tokens and generates an additional token x_{n+1}. We explore two setups, 1) pruned self-data distilled Llama3.1 models speculating for the Llama3.1-70B Instruct (Dubey et al., 2024) target model, and 2) pruned self-data distilled Mistral-7B Instruct models speculating for the Mistral Large 2 (123B) (Mistral, 2024) target model.

Results Our evaluations indicate that self-data distillation enhances speculative capabilities by improving alignment with the target model. Notably, with larger prune block sizes (e.g., 10), self-data distillation paired with model merging achieves higher average accepted lengths across all categories, demonstrating better generalization and more efficient speculative decoding. For instance, in Table 5 pruned Llama3.1 models at block size 10 (i.e., 5.85B resulting in 27.16% FLOPS savings) show an average accepted length improvement of 1.70 tokens. Similarly, in Table 6, we observe Mistral models at block size 10 (i.e., 5.06B resulting in 30.10% FLOPs savings) achieve a gain of 1.62 tokens. While the improvements are evident across general tasks, substantial gains are particularly observed in mathematical reasoning. Pruned Llama3.1 and Mistral-7B models at block size 10 demonstrate an average accepted length increase of 2.96 and 2.75 tokens, respectively, on mathematical reasoning benchmarks. These results show the effectiveness of self-data distillation in preserving complex reasoning capabilities, even under aggressive pruning. Such findings highlight the interplay between speculative decoding and self-data distillation, where the combination optimizes inference efficiency and model quality retention across a broad range of tasks, with notable robustness in challenging reasoning domains.

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 (LeCun et al., 1989; Hassibi et al., 1993). 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 (Xia et al., 2024b; Sreenivas 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, improving semantic alignment and mitigating the quality degradation caused by pruning. While combining this with standard KD techniques could further improve generalization, we leave that for future work.

Distillation Knowledge distillation (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 (Liang et al., 2021; Gu et al., 2024; Agarwal et al., 2024), 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 enhances semantic alignment and mitigates the quality degradation seen after pruning. Furthermore, while our approach can be combined with KD methods 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 (Kotha et al., 2024; Korbak et al., 2022). Regularization techniques such as Elastic Weight Consolidation (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 introduce *self-data distilled fine-tuning* as an effective method to mitigate quality degradation in pruned LLMs, addressing catastrophic forgetting while preserving alignment with the model's original data distribution. Our approach consistently outperforms standard supervised fine-tuning, demonstrating superior accuracy recovery post-pruning across various downstream tasks on the 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. Moreover, integrating self-data distillation with speculative decoding not only enhances token acceptance rates but also reduces inference latency, offering an effective strategy for deploying pruned LLMs efficiently. These findings highlight self-data distilled fine-tuning as a critical tool for maintaining high model quality post-pruning, offering an efficient solution for 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. Extending these methodologies to next-generation LLM architectures presents a promising avenue for unlocking additional computational efficiency and model robustness.

References

Aakanksha, Ahmadian, A., Goldfarb-Tarrant, S., Ermis, B., Fadaee, M., and Hooker, S. Mix data or merge models? optimizing for diverse multi-task learning. *arXiv*, 2024.

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard, 2023.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv*, 2023.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv*, 2021.

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

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., and et al. The llama 3 herd of models. *arXiv*, 2024.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation. 07 2024.

Gemini, Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., and et al. Gemini: A family of highly capable multimodal models. *arXiv*, 2024.

Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P., and Roberts, D. A. The unreasonable ineffectiveness of the deeper layers. *arXiv*, 2024.

Gu, Y., Dong, L., Wei, F., and Huang, M. MiniLLM: Knowledge distillation of large language models. In *ICLR*, 2024.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv*, 2015.

Hassibi, B., Stork, D., and Wolff, G. Optimal brain surgeon: Extensions and performance comparisons. In *NeurIPS*, 1993.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *ICLR*, 2021a.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021b.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv*, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. *arXiv*, 2022.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Huang, Y., Zhang, Y., Chen, J., Wang, X., and Yang, D. Continual learning for text classification with information disentanglement based regularization. In *NAACL*, 2021.

Ji, Y., Xiang, Y., Li, J., Xia, Q., Li, P., Duan, X., Wang, Z., and Zhang, M. Beware of calibration data for pruning large language models. *arXiv*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *arXiv*, 2023.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *arXiv*, 2024.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. TinyBERT: Distilling BERT for natural language understanding. In *EMNLP*, 2020.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv*, 2020.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Kim, B.-K., Kim, G., Kim, T.-H., Castells, T., Choi, S., Shin, J., and Song, H.-K. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *arXiv*, 2024.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. In *PNAS*, 2017.

Korbak, T., Elsahar, H., Kruszewski, G., and Dymetman, M. Controlling conditional language models without catastrophic forgetting. In *ICML*, 2022.

Kotha, S., Springer, J. M., and Raghunathan, A. Understanding catastrophic forgetting in language models via implicit inference. In *ICLR*, 2024.

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

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *ACL*, 2019.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *NeurIPS*, 1989.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty, 2024.

Liang, K. J., Hao, W., Shen, D., Zhou, Y., Chen, W., Chen, C., and Carin, L. Mix{kd}: Towards efficient distillation of large-scale language models. In *ICLR*, 2021.

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

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

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

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

Men, X., Xu, M., Zhang, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv*, 2024.

Mistral. Large enough. July 2024.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. In *NeurIPS*, 2019.

Nallapati, R., Zhou, B., dos Santos, C., Guùlçehre, Ç., and Xiang, B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *CoNLL*, 2016.

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

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., and et al. Gpt-4 technical report. *arXiv*, 2024.

Ostapenko, O., Lesort, T., Rodr'iguez, P., Arefin, M. R., Douillard, A., Rish, I., and Charlin, L. Continual learning with foundation models: An empirical study of latent replay. In *CoLLAs*, 2022.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. *arXiv*, 2022.

Patterson, D. A. Latency lags bandwith. *Commun. ACM*, 2004.

Razdaibiedina, A., Mao, Y., Hou, R., Khabsa, M., Lewis, M., and Almahairi, A. Progressive prompts: Continual learning for language models. In *ICLR*, 2023.

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

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. In *NeurIPS*, 2019.

Samragh, M., Farajtabar, M., Mehta, S., Vemulapalli, R., Faghri, F., Naik, D., Tuzel, O., and Rastegari, M. Weight subcloning: direct initialization of transformers using larger pretrained ones. *arXiv*, 2023.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv*, 2019.

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

Sreenivas, S. T., Muralidharan, S., Joshi, R., Chochowski, M., Mahabaleshwarkar, A. S., Shen, G., Zeng, J., Chen, Z., Suhara, Y., Diao, S., Yu, C., Chen, W.-C., Ross, H., Olabiyi, O., Aithal, A., Kuchaiev, O., Korzekwa, D., Molchanov, P., Patwary, M., Shoeybi, M., Kautz, J., and Catanzaro, B. Llm pruning and distillation in practice: The minitron approach. *arXiv*, 2024.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. In *NeurIPS*, 2018.

Sun, F.-K., Ho, C.-H., and yi Lee, H. Lamol: Language modeling for lifelong language learning. In *ICLR*, 2019.

Sun, Q., Pickett, M., Nain, A. K., and Jones, L. Transformer layers as painters. *arXiv*, 2024.

Sun, Z., Shen, Y., Zhou, Q., Zhang, H., Chen, Z., Cox, D. D., Yang, Y., and Gan, C. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *NeurIPS*, 2023.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. 2023.

Thangarasa, V., Saxena, S., Gupta, A., and Lie, S. Sparse-IFT: Sparse iso-FLOP transformations for maximizing training efficiency. In *ICML*, 2024a.

Thangarasa, V., Tsaptsinos, A., Milton, I., Venkatesh, G., Manohararajah, V., Sinnadurai, N., and Lie, S. Llama3.1 model quality evaluation: Cerebras, groq, together, and fireworks. 2024b.

Toshniwal, S., Moshkov, I., Narenthiran, S., Gitman, D., Jia, F., and Gitman, I. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *arXiv*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 2023b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, W., Bao, H., Huang, S., Dong, L., and Wei, F. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *ACL-IJCNLP*, 2021.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In *ACL*, 2023.

Xia, H., Ge, T., Wang, P., Chen, S.-Q., Wei, F., and Sui, Z. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *EMNLP*, 2023.

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *ACL*, 2024a.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *ICLR*, 2024b.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Yang, Z., Liu, Q., Pang, T., Wang, H., Feng, H., Zhu, M., and Chen, W. Self-distillation bridges distribution gap in language model fine-tuning. In *ACL*, 2024.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STar: Bootstrapping reasoning with reasoning. In *NeurIPS*, 2022.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. In *ICLR*, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2024.