S2D: Sorted Speculative Decoding For More Efficient Deployment of Large Language Models

Parsa Kavehzadeh
Huawei Noah's Ark Lab
parsa.kavehzadeh@huawei.com

Mohammadreza Pourreza
Huawei Noah's Ark Lab

Mojtaba Valipour
University of Waterloo

Tianshu Zhu
Huawei Noah's Ark Lab

Haoli Bai
Huawei Noah's Ark Lab

Ali Ghodsi
University of Waterloo

Boxing Chen
Huawei Noah's Ark Lab

Mehdi Rezagholizadeh
Huawei Noah's Ark Lab

Abstract

Deployment of autoregressive large language models (LLMs) is costly, and as these models increase in size, the associated costs will become even more considerable. Consequently, different methods have been proposed to accelerate the token generation process and reduce costs. Speculative decoding (SD) is among the most promising approaches to speed up the LLM decoding process by verifying multiple tokens in parallel and using an auxiliary smaller draft model to generate the possible tokens. In SD, usually one draft model is used to serve a specific target model; however, in practice, LLMs are diverse, and we might need to deal with many target models or more than one target model simultaneously. In this scenario, it is not clear which draft model should be used for which target model, and searching among different draft models, or training customized draft models, can further increase deployment costs. In this paper, we first introduce a novel multi-target scenario for deployment of draft models for faster inference. Then, we present a novel more efficient sorted speculative decoding mechanism that outperforms regular baselines in multi-target setting. We evaluated our method on Spec-Bench in different settings including base models such as Vicuna 7B, 13B, and LLama Chat 70B. Our results suggest that our draft models perform better than baselines for multiple target models at the same time.

1 Introduction

Large language models (LLMs) have advanced very quickly and become popular in different academic and industrial domains Brown et al. [2020]. As the size of these models increases Narayanan et al. [2021], accelerated inference is becoming more popular to reduce the overhead costs of their deployment. There are an increasing number of publications in the literature trying to achieve faster inference Stern et al. [2018], Chen et al. [2023a], Leviathan et al. [2023a], Chen et al. [2023b]. These different approaches include, but are not limited to, reducing redundant layers Men et al. [2024], quantization , early exiting Varshney et al. [2023], optimizing the KV-caching of transformers Zhang et al. [2023a], and speculative decoding Leviathan et al. [2023b], Chen et al. [2023a]. In this paper, we focus on speculative decoding (SD) as one of the most prominent solutions (due to its simplicity and widespread usage) for improving the decoding speed of LLMs.
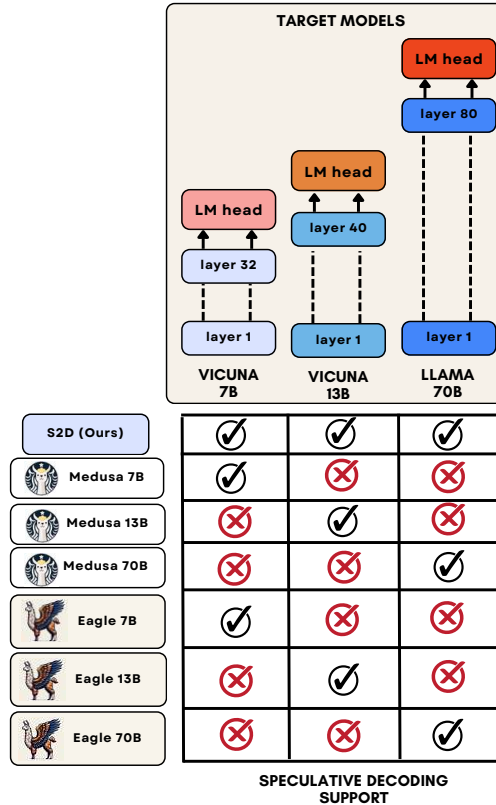
Figure 1: This figure illustrates which target models are compatible with which draft models. Our proposed solution is the only one capable of supporting multiple target models using a single, nested draft model.

While SD is pretty popular in the literature and we have many of its variants available, there are a few bottlenecks in SD, which we will focus on in our paper: **1-"search problem"** we can have target models with different sizes and it is not clear how to obtain the proper draft model for each target model. Moreover, target models can be trained on different downstream tasks, and using a single draft model to serve all the tasks might not yield the best results. This might lead to a distribution mismatch between the target and the draft model unless both the target and the draft model are updated. **2-"minimal training"** We prefer not to train or modify the target model received from the users. This means that most of the solutions in the category of self-speculative solutions are not within our scope.

To address the mentioned problems, we propose our solution called *sorted speculative decoding* (S2D). Sorted refers to the *sorted-training* Valipour et al. [2023] approach in which a model and its selected sub-models can be trained on single or multiple tasks at the same time. Inspired by sorted training, our S2D trains multiple draft models in one model to be able to serve more than just one target model at a time (without needing to maintain multiple draft models) to take care of the *search problem*. In this regard, the initial draft model is extracted from the target model and after designing the sub-models, they are trained together. Moreover, in contrast to the self-speculative solutions, our approach is just applied to the draft side and we do not need to train the target model. Finally, to make an efficient use of the trained sorted draft models, we use an adaptive draft selection mechanism.

The contributions of this paper are listed as follows: 1- **Introduction of Multi-Target Draft Models**: We pioneer the concept of employing a single draft model that can simultaneously accommodate multiple target models, reducing deployment complexity and costs. 2- **Development of a Sorted Speculative Decoding Mechanism**: Our S2D mechanism leverages sorted fine-tuning, enabling the creation of sub-models within a draft model, without the necessity of maintaining separate draft models for each target LLM. 3- **Adaptive Draft Selection Strategy**: We introduce an adaptive draft selection mechanism that optimally chooses sub-models based on confidence thresholds. 4- **Comprehensive Evaluation on Spec-Bench**: We rigorously evaluated our S2D method on the Spec-Bench. We discuss the literature and related work in A.
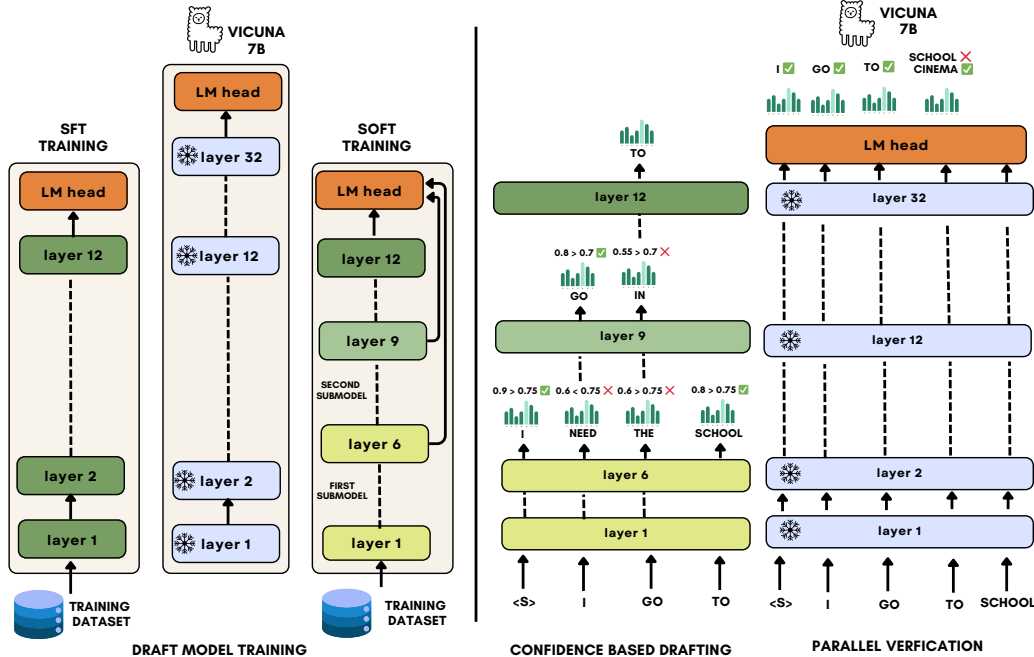
Figure 2: The figure on the left illustrates the draft model training process, comparing supervised fine-tuning (SFT) with Sorted fine-tuning (SoFT) using two sub-models with 6 and 9 layers. The figure on the right demonstrates the confidence-based drafting process, where the SoFT draft model is utilized to generate candidate tokens. The confidence thresholds for the two sub-models are set at 0.75 and 0.7, respectively.

2 Methodology

2.1 Background

Speculative Decoding: Speculative decoding is a two-step process involving drafting and verification. At each decoding step, a draft model efficiently generates multiple potential future tokens, which are then verified in parallel by the target model at inference time. Specifically, during the drafting step, given an input sequence $\{x_1, \ldots, x_n\}$ and the target LLM M_t, a faster drafter model M_d decodes the next K drafted tokens as a speculation of the target LLM's output Xia et al. [2024]:

$$p_1, \ldots, p_K = \text{DS}(x \leq k, M_d),$$
$$\hat{x}_i \sim p_i, \quad i = 1, \ldots, K \tag{1}$$

where $\text{DS}(\cdot)$ represents the drafting strategy, p is the conditional probability distribution calculated by M_d, and $\hat{x}_i$ is the token sampled from the draft model's probability distribution p_i. The tokens generated by the draft model are then verified by the target LLM M_t. Given the input sequence $\{x_1, \ldots, x_n\}$ and the drafted tokens $\{\hat{x}_1, \ldots, \hat{x_K}\}$, the M_t model is used to measure $K+1$ probability distributions simultaneously as follows:

$$q_i = M_t(x \mid x \leq t, x_{<i}), \quad i = 1, \ldots, K+1 \tag{2}$$

Each drafted token $\hat{x}_i$ is then verified using a specific verification criterion using $\hat{x}_i$, q_i, and p_i. Only the tokens that meet this criterion are selected as final outputs.

Sorted Fine-tuning: Sorted Fine-tuning Valipour et al. [2023], Kavehzadeh et al. [2024] is a recently proposed approach for training many-in-one models by forming sub-models from a larger model. In the case of LLMs, sub-models are the sub-layers of the existing LLM. Each sub-model's output is predicted using the shared output prediction head from the last layer (original LLM head). To train the network, we define the loss as the summation of the losses of all the sub-models:

$$L = \frac{\sum_{n \in B} L_n(x; \theta_n)}{|B|} \tag{3}$$

3

Model	GSM8K	
	Auto-regressive Decoding	
	Speedup	**Accuracy**
SFT (Llama2 13B)	1×	48.97
Model	**Self Sorted speculative decoding (Sorted Target)**	
	Speedup	**Accuracy**
Layers 12:40 (SoFT)	1.21×	33.51
Draft Model	**Sorted Speculative Decoding (Sorted Draft)**	
	Speedup	**Accuracy**
Layer 6:12 (SoFT 6,9,12 13B)	**1.53×**	48.97

Table 1: Performance comparison between self sorted speculative decoding (sorted target) and adaptive speculative sampling (sorted draft) proposed in this paper on GSM8K dataset.

where $L_n(x; \theta_n)$ is the loss for the n-th sub-model for input batch x and B denotes the number of sub-models.

2.2 Why sorted draft instead of sorted target?

In this paper, we introduce a method that involves Sorted fine-tuning (SoFT) of a draft model and using sub-models for sorted speculative decoding to increase the inference speed of multiple target models. An alternative method, is to use SoFT to train the target model, instead of the draft model, similar to the approach proposed in Kavehzadeh et al. [2024]. To evaluate these two methods, we fine-tuned the Llama2 13B Touvron et al. [2023] on the GSM8K dataset using both standard supervised fine-tuning (SFT) and Sorted fine-tuning (SoFT) as described in Kavehzadeh et al. [2024]. The results are provided in Table 1, where we compare our sorted speculative decoding with sorted draft model training with self sorted speculative decoding.

According to Table 1, the sorted target model training method has three significant disadvantages. Firstly, it decreases accuracy by 16% in final task performance and offers lower speed improvements because the sub-models used are larger than those in the sorted draft model training. Secondly, this method is not suitable for scenarios with multiple target models as it requires each target model to undergo SoFT training for self-speculative decoding to be applicable. Lastly, SoFT training of the target model incurs considerably higher costs compared to our method of SoFT training a smaller draft model.

2.3 Sorted speculative decoding

In this section, we introduce our approach which utilizes multiple draft models in the same architecture in an adaptive way to address multi-target inference acceleration problem. To reach this goal, we first introduce a new sorted draft architecture that can incorporate multiple draft sub-models in the same architecture. Then we explain the adaptive draft generation algorithm that we devise in order to use the draft sub-models efficiently in speculative decoding paradigm.

Training SoFT Draft   Supposed we have a pre-trained large language model $f(x; \theta_N)$ with the parameters θ, input x and N number of layers. Also consider $f(\theta_n)$ as the sub-model with the parameters of first n layers of the LLM ($n \leq N$). To reach our draft architecture, we first extract a sub-model with $f(\theta_{N_d})$, where $N_d < N$. Then we also determine three different sub-models in the extracted draft architecture as $f(x; \theta_{N_{ds}})$, $f(x; \theta_{N_{dm}})$ and $f(x; \theta_{N_d})$, where $N_{ds} < N_{dm} < N_d$. We utilize the sorted fine-tuning approach Kavehzadeh et al. [2024] to fine-tune the whole draft on the downstream dataset to reach three draft models with different sizes in the same architecture. In this paper, we use Vicuna 7B as the pre-trained language model with 32 layers. To define our draft sub-models, we set N_{ds} to 6, N_{dm} to 9, and N_d to 12 in our experiments. Figure 2 (Left) shows the two SFT and SoFT methods to train an extracted draft model from the target Vicuna 7B.

Draft Generation   In order to make most out of the speculative decoding algorithm, we need to have both low latency draft models and high acceptance ratio compared to target model. To generate each token, we employ a confidence-based early-exiting approach in $f(x; \theta_{N_d})$ architecture. Supposed the draft sub-model layers $L_D = \{N_{ds}, N_{dm}, N_d\}$, we have the set of confidence thresholds $T_D = \{\tau_{ds}, \tau_{dm}, \tau_d\}$. To generate one draft token given input sequence S, we start iterating over

Method	Greedy (T = 0)						Non-Greedy (T = 1)						Avg Speedup
	Vicuna 7B		Vicuna 13B		LLaMA Chat 70B		Vicuna 7B		Vicuna 13B		LLaMA Chat 70B		
	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	
Eagle Li et al. [2024]	2.62×	3.84	✗	✗	✗	✗	2.05×	3.42	✗	✗	✗	✗	N/A
Eagle (No Attention Tree)	2.04×	3.11	✗	✗	✗	✗	1.72×	2.73	✗	✗	✗	✗	N/A
Medusa Cai et al. [2024]	1.74×	2.51	✗	✗	✗	✗	1.93×	2.80	✗	✗	✗	✗	N/A
Medusa (No Attention Tree)	1.34×	1.76	✗	✗	✗	✗	1.48×	1.92	✗	✗	✗	✗	N/A
Hydra Ankner et al. [2024]	2.14×	2.70	✗	✗	✗	✗	2.36×	4.01	✗	✗	✗	✗	N/A
Hydra (No Attention Tree)	1.78×	2.65	✗	✗	✗	✗	2.03×	3.11	✗	✗	✗	✗	N/A
SFT + SD	1.19×	3.19	1.21×	3.05	1.94×	2.46	1.16×	3.44	1.10×	3.16	1.94×	2.54	1.42
SFT + EarlyExit L6 + SD	0.98×	1.51	1.00×	1.51	1.32×	1.46	0.96×	1.52	1.03×	1.51	1.37×	1.53	1.11
SFT + EarlyExit L9 + SD	1.08×	1.99	1.05×	1.98	1.54×	1.80	0.99×	2.04	1.12×	2.04	1.57×	1.86	1.22
SoFT L6 + SD	**1.38×**	2.43	1.38×	2.40	1.83×	2.05	**1.30×**	2.53	1.35×	2.87	1.87×	2.14	1.51
SoFT L9 + SD	1.16×	3.00	1.31×	2.78	1.92×	2.27	1.26×	3.05	1.32×	2.87	1.94×	2.34	1.49
SoFT L12 + SD	1.17×	3.11	1.23×	3.01	1.91×	2.39	1.07×	3.22	1.20×	3.17	1.96×	2.53	1.42
SoFT + S2D (ours)	1.34×	2.86	**1.38×**	2.76	**1.95×**	2.36	1.27×	3.01	**1.38×**	2.89	**1.98×**	2.44	**1.55**

Table 2: Overall Speedup and Mean Accepted Tokens lenght (MAT) on MT-Bench dataset in Multi-target inference acceleration setup. The speedups reported for Eagle, Medusa and Hydra are based on the publicly available checkpoints of these methods for Vicuna 7b target model and since these checkpoints cannot be applied to other targets, we denoted that by ✗ in the table.

draft sub-models, starting from N_{ds}. For each sub-model $N_i \in L_D$, we have:

$$t, c \sim f(S; \theta_{N_i}) \tag{4}$$

Where t and c are the draft token and its confidence sampled from draft sub-model N_i. We accept the token t as the final draft token if $c \geq \tau_i$. Algorithm 1 explains the draft generation mechanism of S2D algorithm in more details. Figure 2 (Right) also shows how draft generation works in S2D algorithm.

3 Experiments

3.1 Experimental Setup

We selected the first 12 layers of Vicuna 7b checkpoint to build the architecture of our draft model. Then, we trained the draft model in both SFT and SoFT paradigms on the ShareGPT [1] dataset for 3 epochs. We used the Spec-bench Xia et al. [2024], which is a benchmark for speculative decoding-based methods, to evaluate our draft models and the S2D algorithm. More details about the experimental setup and memory consumption can be found in B.1 and B.2.

3.2 Baselines

We categorize the baselines based on the dependency of draft training procedure on target models:

Target-Dependent Baselines

- **Eagle Li et al. [2024]**: An approach proposing a single layer draft model trained with two feature alignment and cross-entropy losses based on target output.
- **Medusa Cai et al. [2024]**: A method for generating multiple draft candidates for future tokens by training multiple language model heads for each future token position.
- **Hydra Ankner et al. [2024]** A draft model based on recurrent neural architectures on top of the target model, which generates multiple draft candidates.

Target-Independent Baselines   We have different scenarios for our draft generation as baselines:

- **SFT Checkpoint + Speculative Decoding**: We use the SFT checkpoint of first 12 Vicuna 7b layers as draft model in the speculative sampling algorithm.
- **EarlyExit of small sub-model of SFT Checkpoint (6 Layers) + Speculative Decoding**: We fine-tune only the classifier head of draft SFT checkpoint for the smallest sub-model (Layer 6) and use the first 6 layers and classifier head as draft in speculative decoding.

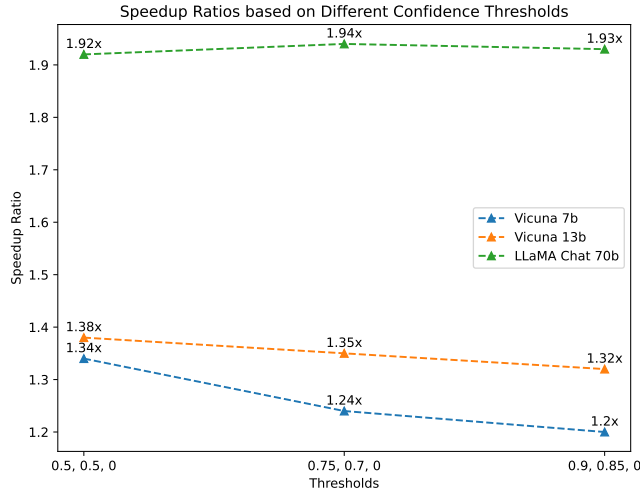[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

Figure 3: Speedup ratios based on difference confidence thresholds in multiple targets on MT-Bench. In the thresholds axis labels, the first, second and third numbers represent the thresholds for the first draft sub-model (layer 6), the second draft sub-model (layer 9) and the last one (layer 12). The temperature was set to 0.0 in these experiments.

- **EarlyExit of Medium sub-model of SFT Checkpoint (9 Layers) + Speculative Decoding**: We fine-tune only the classifier head of draft SFT checkpoint for the medium sub-model (Layer 9) and use the first 9 layers and classifier head as draft in speculative decoding.

- **Small sub-model of SoFT Checkpoint (6 Layers) + Speculative Decoding**: We use the smallest sub-model (Layer 6) of the SoFT checkpoint of first 12 Vicuna 7b layers as draft model in the speculative decoding algorithm.

- **Medium sub-model of SoFT Checkpoint (9 Layers) + Speculative Decoding**: We use the medium sub-model (Layer 9) of the SoFT checkpoint of first 12 Vicuna 7b layers as draft model in the speculative decoding algorithm.

- **Full SoFT Checkpoint (12 Layers) + Speculative Decoding**: We use the full SoFT checkpoint of first 12 Vicuna 7b layers as draft model in the speculative decoding algorithm.

- **Full SoFT Checkpoint (12 Layers) + S2D**: We use the full SoFT checkpoint of first 12 Vicuna 7b layers as draft model in the S2D algorithm. We set the thresholds of the intermediate sub-models from the ablations in section 3.4.1.

3.3 Results

In this section, we will discuss the results of the experiments we conducted to evaluate our SoFT draft model and S2D approach compared to other baselines.

Multi-Target Draft and S2D Performance: Table 2 and Figure 4 illustrate the performance of our S2D method and other baselines on multiple targets, including the MT-Bench dataset in Spec-Bench. Fine-tuning an extracted network from Vicuna 7b enables speedup across multiple target models of varying sizes without requiring pre-training. Our SoFT draft sub-models consistently outperform SFT early-exiting baselines across all target models. In smaller targets like Vicuna 7b, where latency is critical, S2D adjusts the draft generation process by favoring intermediate layers, optimizing the balance between speed and performance. For larger targets like LLaMA Chat 70b, the mean accepted tokens length (capacity) becomes more crucial than draft latency, as shown by the higher speedups gained by the 12-layer draft compared to the 6-layer SoFT draft checkpoint. Even in these scenarios, S2D optimally adjusts the exiting layers to maintain top performance.

S2D consistently outperforms or matches the performance of normal speculative decoding (SD) across almost all target sizes. On targets like Vicuna 7b and 13b, S2D achieves significantly better speedup compared to SD with SFT and SoFT full draft models (12 layers). It also shows better

6

Model / Metric		GSM8K EM	GSM8K Speedup Ratio
Autoregressive		48.90	1.00x
EAGLE	w_cls=0.1, w_reg=1.0	48.67	2.73x
	w_cls=0.0, w_reg=1.0	48.90	2.70x
	w_cls=1.0, w_reg =0.0	48.82	1.61x

Table 3: Comparison of GSM8K exact match (EM) and speedup ratios for different decoding configurations. With Llama2 13B target model and 1-layer Eagle draft model.

speedup than SD with medium-size SoFT drafts (Layer 9) in tasks such as MT-Bench and GSM8K. However, SD with the smallest SoFT draft (Layer 6) outperforms S2D in most tasks due to its substantially lower latency, which contributes to the overall speedup in speculative decoding. On larger targets like LLaMA Chat 70b, S2D maintains similar speedup to SD with SFT and SoFT full model architectures. Using SD with smaller draft models, particularly the 6-layer SoFT sub-model, results in a notable speedup drop, highlighting the importance of draft model capacity and mean accepted tokens length in accelerating inference for large targets.

Overall, depending on the necessity of lower draft latency or higher accepted token ratio, S2D performance demonstrates that our proposed approach can choose the sub-models accordingly to have the optimum speedup compared to other draft options in the architecture. The more details about the baselines performance on Spec-Bench can be found in Appendix. Also the details about how SoFT draft is more efficient in training compared to a target-dependent approach, Medusa, can be found in Appendix.

3.4 Ablation Studies

3.4.1 Thresholds

To find the optimum confidence thresholds for SoFT draft sub-models in S2D algorithm, we assess different thresholds sets to see the algorithm's performance in each scenario. Figure 3 shows the performance comparison of different confidence thresholds. While in smaller target models, there is a tendency to choose the smaller draft sub-models by lowering their corresponding confidence thresholds to increase speedup, in the largest target model there is a need for higher draft model capacity therefore higher confidence thresholds result in optimum speedup. We fixed the best thresholds of each target model in all other experiments conducted with S2D algorithm in this paper.

3.4.2 Impact of Attention Tree

Unlike speculative decoding and sorted speculative decoding methods, recent approaches such as EAGLE Li et al. [2024], Medusa Cai et al. [2024], and Hydra Ankner et al. [2024] employ tree attention to simultaneously verify multiple candidate tokens. This addition complicates direct comparisons with other methods. Therefore, we also evaluated these methods without tree attention and provided the results in Table 2. As expected, the speedup ratio significantly decreased for all. Interestingly, when tree attention is removed, Medusa's performance even falls below that of speculative decoding, specifically using layer 6 of the SoFT-trained sub-models.

3.4.3 Impact of Feature Alignment

Using the last layer hidden state representation of the target model to train a draft model has become a common approach in recent works Cai et al. [2024], Ankner et al. [2024]. More specifically, EAGLE Li et al. [2024] employs both features and tokens generated by the target model to train an one-layer draft. Supposed t_i and f_i are the token and hidden state feature generated by target LLM and $\hat{t}_i$ and $\hat{f}_i$ are the token and hidden state features generated by draft at position i, Eagle aligns hidden state features of draft and target by using L1 regression loss:

$$L_{\text{reg}} = \text{SmoothL1}(f_{i+1}, \text{Draft\_Model}(t_{2:i+1}, f_{1:i})).$$

They also employ classification loss to directly optimize towards alignment of tokens:

$$p_{i+2} = \text{Softmax}(\text{LM\_Head}(f_{i+1})),$$

$$\hat{p}_{i+2} = \text{Softmax}(\text{LM\_Head}(\hat{f}_{i+1})),$$

7

| Method | Greedy (T = 0) | | | | | |
| | Vicuna 7B | | Vicuna 13B | | LLaMA Chat 70B | |
	Speedup	MAT	Speedup	MAT	Speedup	MAT
Vicuna 160m + SD	1.05×	2.75	1.13×	2.66	1.93×	2.24
SoFT L6 + SD	1.20×	2.10	1.26×	2.06	1.68×	1.76
SoFT L9 + SD	1.10×	2.33	1.21×	2.29	1.79×	1.96
SoFT L12 + SD	1.00×	2.67	1.07×	2.58	1.90×	2.19
SoFT + S2D	1.17×	2.54	1.28×	2.44	1.91×	2.16

Table 4: Overall Speedup and Mean Accepted Tokens lenght (MAT) on MT-Bench dataset. The SoFT training in this experiment was initialized with Vicuna 160m checkpoint.

| Model | GSM8K | | |
| | Auto-regressive Decoding | | |
	Trained Target	Speedup	Accuracy
Llama2 13B	✓	1×	48.97
Llama2 13B	✗	1×	28.7
Draft Model	**Sorted Speculative Decoding**		
	Trained Target	Speedup	Accuracy
S2D - SoFT	✓	**1.53×**	48.97
S2D - SoFT	✗	1.38×	28.7

Table 5: Comparison of speedup between two different settings: 1) training the target model on the downstream task 2) using the vanilla pre-trained model

$$L_{\text{cls}} = \text{CrossEntropy}(p_{i+2}, \hat{p}_{i+2}).$$

By integrating feature alignment (regression loss) and token alignments (classification loss), EAGLE's autoregression head is trained using the combined loss function: $L = w_{\text{reg}}L_{\text{reg}} + w_{\text{cls}}L_{\text{cls}}$.

We conducted experiments studying the affects each feature and token alignment cause by setting different combination of w_{reg} and w_{cls} (Table 3). We train eagle draft model based on a LLaMA2 13b target model fine-tuned on GSM8K train data. We found out that canceling the token alignment loss ($w_{\text{cls}} = 0$) would not have a significant impact on the draft performance compared to the original setup used in Eagle paper ($w_{\text{reg}} = 1$ and $w_{\text{cls}} = 0.1$). On the other hand, setting w_{reg} to 0 would cause a noticeable impact on the draft performance, dropping speedup from 2.73x to 1.61x. As we can see, feature alignment plays the main role in improving the Eagle draft performance while this is impractical in multi-target setting where the draft model needs to serve multiple target models anytime.

3.4.4 Impact of Pre-training

Using pre-training draft models can be a possible direction to increase the acceptance ratio of draft tokens in speculative decoding algorithm. In this way, we repeated our experiments in a new setup where we replaced the first 12 layers of Vicuna 7b with Vicuna 160m, which is a LLaMA 160m checkpoint fine-tuned on ShareGPT dataset. LLaMA 160m is a small 12 decoder layer architecture pre-trained on C4 corpus. We also sorted fine-tuned the Vicuna 160m on ShareGPT with the same sub-models (Layer 6, 9 and 12). Table 4 shows the benefit of using S2D instead of regular SD for smaller target models (Vicuna 7b and 13b). Based on the results in Table 2, fine-tuning an extracted 12-layer draft model can result in higher speedup compared to employing a similar pre-trained architecture, which can demonstrate the efficiency of our approach in terms of training resources.

3.4.5 Target Model Training

Foundation models like GPT-4, Gemini [Team et al., 2023], and the Claude family [Anthropic, 2024] are trained on vast datasets, enabling them to perform well across various tasks. Nonetheless, for specialized tasks where the model has limited exposure during pre-training, domain adaptation through fine-tuning leads to superior performance in downstream tasks [Liu et al., 2024]. This section evaluates the impact of fine-tuning both the target and draft models for specific tasks on the speed gains achievable using our proposed sorted speculative decoding method. In this way, we fine-tune a Llama 13B model and also sorted fine-tuned the extracted 12-layer draft on GSM8k, a mathematical

reasoning dataset. Results shown in Table 5 demonstrate that fine-tuning both target and draft on the same dataset, due to their improved alignment, results in a $1.14\times$ speed increase at inference time.

4 Conclusion

In this paper, we present a method based on the SoFT training of a draft model to overcome a significant limitation of traditional speculative decoding methods, where each target model necessitates a uniquely trained draft model. Through comprehensive experimentation, we demonstrate that by using a same SoFT-trained draft model with varying thresholds for sub-models, we achieve an average speedup ratio of **1.55** for target models with parameters ranging from 7B to 70B. Moreover, our method surpasses vanilla speculative decoding across all target models, highlighting its effectiveness.

References

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

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023a.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Jie Huang, and Kevin Chen-Chuan Chang. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*, 2023b.

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 preprint arXiv:2403.03853*, 2024.

Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Accelerating llama inference by enabling intermediate layer decoding via instruction tuning with lite, 2023.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023a.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding, May 2023b. URL http://arxiv.org/abs/2211.17192. arXiv:2211.17192 [cs].

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *arXiv preprint arXiv:2309.00255*, 2023.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

Parsa Kavehzadeh, Mojtaba Valipour, Marzieh Tahaei, Ali Ghodsi, Boxing Chen, and Mehdi Reza-gholizadeh. Sorted llama: Unlocking the potential of intermediate layers of large language models for dynamic inference. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2129–2145, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

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

Anthropic. Model card claude 3. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024.

An Liu, Zonghan Yang, Zhenhe Zhang, Qingyuan Hu, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. Panda: Preference adaptation for enhancing domain-specific abilities of llms. *arXiv preprint arXiv:2402.12835*, 2024.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*, 2021.

Heming Xia, Tao Ge, Furu Wei, and Zhifang Sui. Lossless speedup of autoregressive translation with generalized aggressive decoding. *arXiv preprint arXiv:2203.16487*, 2022.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, 2023.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023b.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Joud Chataoui, Mark Coates, et al. Jointly-learned exit and inference for a dynamic neural network. In *The Twelfth International Conference on Learning Representations*, 2023.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Ion Stoica, Zhijie Deng, Alvin Cheung, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 30222–30242. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6034a661584af6c28fd97a6f23e56c0a-Paper-Conference.pdf.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 1(2):4, 2023.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Hanling Yi, Feng Lin, Hongbin Li, Peiyang Ning, Xiaotian Yu, and Rong Xiao. Generation meets verification: Accelerating large language model inference with smart parallel auto-correct decoding. *arXiv preprint arXiv:2402.11809*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Algorithm 1 Sorted Speculative Decoding

Require: Sorted draft layers L, Target model $f(\theta_N)$, Input context C, Draft thresholds T, Draft candidates verification function VerifyTokens

Ensure: Generated sequence S

0: **function** GENERATECANDIDATES(S)
0:     $candidates \leftarrow [\,]$
0:     **while** not end of draft generation **do**
0:         {Adaptively generate draft candidates}
0:         **for** $n, threshold$ in zip(L, T) **do**
0:             $p_S \leftarrow f(S; \theta_n)$
0:             {Sample from draft sub-model distribution}
0:             $x, c \sim p_S$
0:             **if** $threshold \leq c$ **then**
0:                 **append** (x, c) **to** $candidates$
0:                 **break** {Exit from intermediate draft}
0:             **end if**
0:         **end for**
0:     **end while**
0:     **return** $candidates$
0: **end function**
0: Initialize $S \leftarrow C$
0: **while** not end of sequence **do** {Initialize generation}
0:     {Draft generation}
0:     $Cands \leftarrow$ GENERATECANDIDATES(S)
0:     {Verify draft tokens}
0:     $Matches \leftarrow$ VerifyTokens($f(\theta_N), S, Cands$)
0:     **append** $Matches$ **to** S
0: **end while**
0: **return** S
    =0

A Related Works

For LLMs to perform more efficiently, efficient decoding/sampling methods Leviathan et al. [2023a], Li et al. [2024] are essential. As LLMs grow in complexity and size, the need for innovative techniques to enhance their speed and accuracy becomes even more pressing. Various methodologies are discussed in this literature review, including parallel sampling, speculative decoding, and early exit strategies, alongside their contributions and advancements.

Parallel Decoding The first mechanism aiming to accelerate the inference process of large language models was presented in Stern et al. [2018]. This paper introduced a blockwise parallel decoding strategy, aiming to generate the next k tokens simultaneously in a single forward pass using a set of auxiliary models. Then they proposed to use the same language model to verify the generated tokens in parallel. This simple draft-then-verify mechanism, as discussed in the paper, can potentially reduce the number of forward passes from m to m/k+1 Stern et al. [2018].

Speculative Decoding Inspired by Sun et al. [2021], Xia et al. [2022] proposed a draft-then-verify mechanism to aggressively generate a fixed number of tokens in parallel without the new tokens depending on the previous ones, and then verify the generated tokens in one forward pass.

Later, they Xia et al. [2023] proposed a more advanced attention mechanism to generate independent tokens in parallel by using distinct attention queries instead of using a shared attention query, or simply adding more language model heads as done in the past. In addition, for the verification process, they also relaxed the top-1 greedy decoding. Instead, they proposed to accept any token from the top-β candidates as long as their score gap is not far from the most likely token Xia et al. [2023].

Speculative Sampling Other methods, such as Chen et al. [2023a], Leviathan et al. [2023a], generalized speculative decoding to the stochastic non-greedy setting. As these methods are just a

Method/Target Model	Vicuna 7b	Vicuna 13b	LLaMA2 Chat 70b
SoFT Draft (12 Layers)	92,744 MB	92,744 MB	92,744 MB
Medusa	113,416 MB	152,616 MB	OOM

Table 6: Comparison of speedup between two different settings: 1) training the target model on the downstream task 2) using the vanilla pre-trained model

variant of the draft-then-verify mechanism with a modified rejection sampling algorithm for ensuring sampling quality, we will leave the integration of these methods with our proposed method as future work.

Self-Speculative Decoding   Other methods, such as Zhang et al. [2023b], introduced self-speculating, which tries to get rid of the auxiliary models by selectively skipping certain intermediate layers during the drafting phase. As we can use the full LLM to validate the generated tokens, without any additional model we can enjoy accelerated inference. This is also aligned with approaches like Elhoushi et al. [2024], Chataoui et al. [2023], Kavehzadeh et al. [2024], and Valipour et al. [2023].

Other Methods   More recently, new techniques Liu et al. [2023], Chen et al. [2023b], Li et al. [2024], Cai et al. [2024], Fu et al. [2024], Sun et al. [2023], Miao et al. [2023], Varshney et al. [2023], Ankner et al. [2024], He et al. [2023], Yi et al. [2024] have emerged that trying to incorporate sophisticated mechanisms to further improve the speculative sampling speedup gain.

For simplicity, this paper will focus on the Self-Speculative Decoding setting, but our method is also applicable to other speculative sampling methods with minor adjustments, without loss of generality.

Benchmarks   In addition, to evaluate these different algorithms, several benchmarks Zheng et al. [2024], Taori et al. [2023], Chen et al. [2021] can be used to measure performance and speed up gains. One of the most comprehensive benchmarks, however, is Spec-Bench, specifically designed to evaluate speculative decoding methods Xia et al. [2024]. Spec-Bench comprises 6 sub-tasks: translation, multi-turn conversation (MT-Bench), retrieval-augmented generation, mathematical reasoning, question answering, and summarization, each with 80 instances. In this paper, we will focus mainly on Spec-Bench.

B   Experiments

B.1   Hyperparameters

The batch size during the ShareGPT training experiments (SFT and SoFT) was 8 and gradient accumulation steps was set to 16. We used 4 NVIDIA V100 GPUs for training experiments. Each inference experiment on Spec-bench was done on 2 NVIDIA V100 GPUs, except the experiments for LLaMA Chat 70b where we used 8 GPUs to avoid memory issues.

B.2   Memory Consumption

We conducted an experiment to measure the memory footprint of training our target-independent S2D draft model compared to Medusa, a target-dependent inference acceleration method. The results are presented in Table 6. As indicated in the table, our nested draft model requires only a single training session to apply to multiple target models, maintaining a consistent memory footprint of 92,744 MB across all three target models. In contrast, Medusa necessitates loading the target model each time to train the attached classifier heads as drafters, resulting in a significantly larger and variable memory footprint depending on the target model. This underscores the efficiency of our approach in terms of memory usage.

B.3   Spec-bench Results

Figure 4 shows the speedup ratios of S2D versus speculative decoding with different draft options, including Vicuna 160m pre-trained. As S2D demonstrate superior performance than speculative
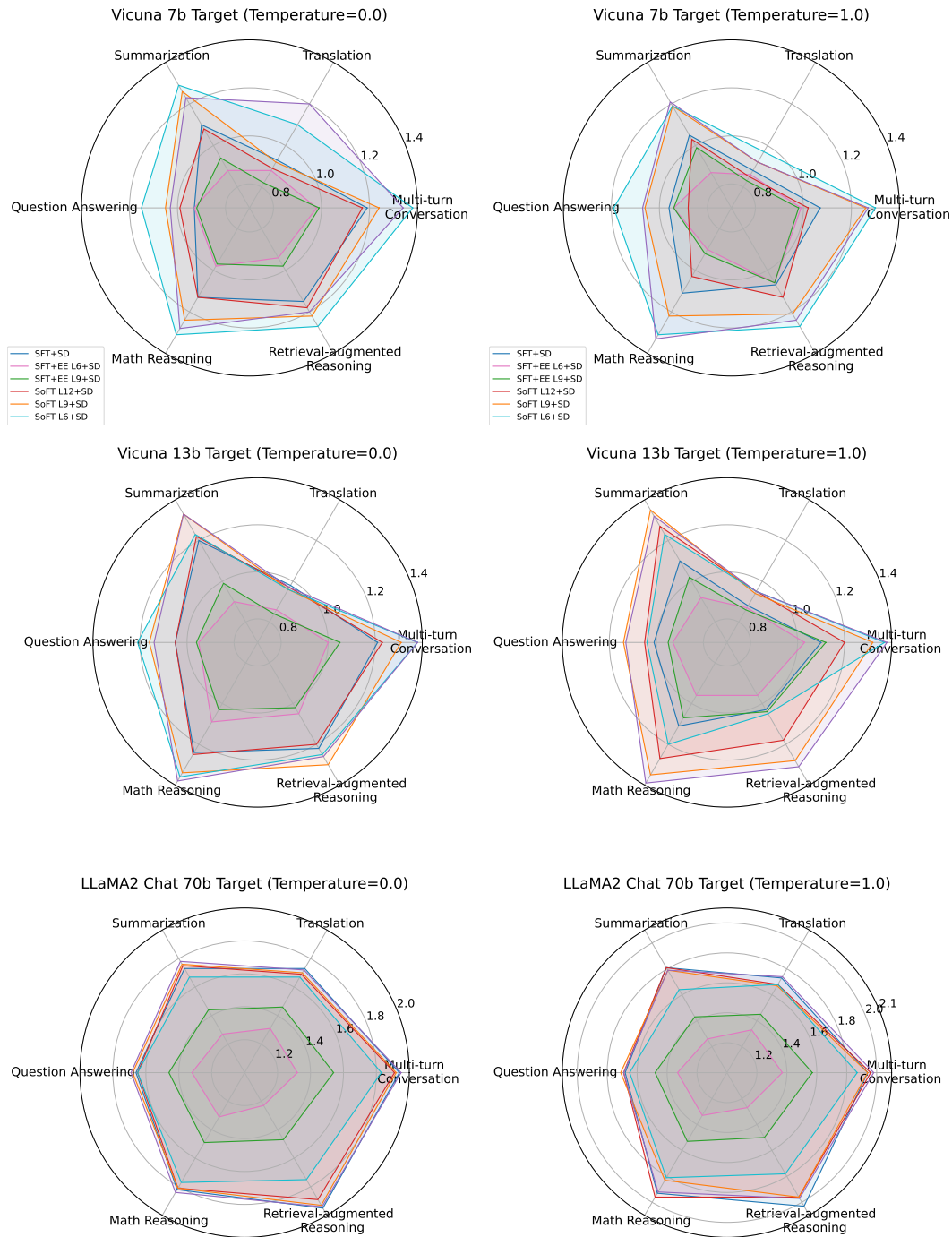
Figure 4: Comparison among Speedup ratios of speculative and S2D methods on different domains on multiple targets.

decoding with the same size draft models (12 layers) in smaller target sizes (Vicuna 7B and 13B), once we get to larger target sizes, larger drafts outperform smaller ones (Layer 6 and 9) in speculative decoding algorithm. This can demonstrate the importance of capacity and accepted tokens length in larger target sizes. However, S2D can maintain a similar performance to speculative decoding with the same draft size even in case of using a target with 70B size.