AdaSPEC: Selective Knowledge Distillation for Efficient Speculative Decoders

Yuezhou Hu[1*], **Jiaxin Guo**[2*], **Xinyu Feng**[3], **Tuo Zhao**[3†]
[1] University of California, Berkeley [2] Tsinghua University
[3]Georgia Institute of Technology
yuezhouhu@berkeley.edu jx-guo21@mails.tsinghua.edu.cn
{xfeng300,tourzhao}@gatech.edu

Abstract

Speculative Decoding (SD) accelerates large language model inference by employing a small draft model to generate predictions, which are then verified by a larger target model. The effectiveness of SD hinges on the alignment between these models, which is typically enhanced by Knowledge Distillation (KD). However, conventional KD methods aim to minimize the KL divergence between the draft and target models across all tokens, a goal that is misaligned with the true objective of SD, which is to maximize token acceptance rate. Therefore, draft models often struggle to fully assimilate the target model's knowledge due to capacity constraints, leading to suboptimal performance. To address this challenge, we propose AdaSPEC, a novel method that incorporates selective token filtering into the KD process. AdaSPEC utilizes a reference model to identify and filter out difficult-to-fit tokens, enabling the distillation of a draft model that better aligns with the target model on simpler tokens. This approach improves the overall token acceptance rate without compromising generation quality. We evaluate AdaSPEC across diverse tasks, including arithmetic reasoning, instruction-following, coding, and summarization, using model configurations of 31M/1.4B and 350M/2.7B parameters. Our results demonstrate that AdaSPEC consistently outperforms the state-of-the-art DistillSpec method, achieving higher acceptance rates across all tasks (up to 15%). The code is publicly available at https://github.com/yuezhouhu/adaspec.

1 Introduction

Large language models (LLMs) have revolutionized natural language processing, achieving impressive performance across a wide range of tasks. Models like GPT-4 [25] and Llama 3 [11] demonstrate state-of-the-art results in various natural language understanding and generation tasks [2, 27], including highly complex tasks such as summarization [23] and mathematical reasoning [9, 13]. However, as these models grow in size and complexity, their inference becomes increasingly computationally intensive, leading to practical challenges in deployment, including slow generation speeds and significant output latency.

To address these shortcomings, current approaches primarily focus on achieving a trade-off between efficiency and performance through two main strategies. The first involves compressing the model scale to enhance the capability of smaller models, often using techniques like Knowledge Distillation (KD) [14]. The second approach employs methods such as quantization to enable faster computation. However, these strategies inevitably lead to a sacrifice of performance, either due to a loss of representational capacity during compression or reduced accuracy resulting from optimization for

[*]Equal Contribution. Work done during an internship at Georgia Tech. [†]Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

speed. As a result, there is a growing need for methods that can maintain the high performance of LLMs while significantly improving their inference efficiency.

Recently, Speculative Decoding (SD) [16, 7] has emerged as a promising paradigm for accelerating LLM inference without sacrificing performance. Unlike model compression or quantization, which modify the model architecture or parameters, SD accelerates generation by restructuring the decoding process itself. Specifically, it introduces a lightweight draft model that speculatively generates multiple candidate tokens, which are then verified by the larger target model. This paradigm preserves the target model's predictive quality while substantially reducing the number of expensive forward passes, offering a new efficiency–performance trade-off.

The core of SD lies in the design of the draft model. This model is typically much smaller than the target model, even ranging from one-tenth to one-hundredth of the size, enabling faster token generation while maintaining a certain level of capability. Consequently, the actual inference speed-up achieved by SD relies on the draft model closely aligning its predictions with the target model's output distribution. Typically, this alignment is achieved by pre-training and fine-tuning both models on the same datasets, yielding a pair of homogeneous models from the same family, sharing the same architecture but differing in size. However, training two models on the same datasets does not necessarily produce optimal alignment, especially given the significant scale disparity between the draft and target models. This difference in scale makes the draft model prone to prediction errors. To address this challenge, state-of-the-art methods employ KD techniques to refine the draft model, rather than relying solely on direct fine-tuning [32]. However, optimizing fidelity metric (e.g., forward KL divergence) does not necessarily lead to a high acceptance rate. Worse still, it may waste the draft model's limited capacity on tokens that are inherently hard to learn and unlikely to be accepted anyway. Additionally, these methods may encounter issues such as the loss failing to converge. Given these challenges, there is a critical need for SD-specific training regimes that effectively balance model capacity constraints with prediction accuracy requirements.

Fortunately, we observe substantial variation in the difficulty of learning individual tokens during KD, which has critical implications for transferring knowledge from the teacher (target) model to the student (draft) model. Instead of mimicking the full output distribution of the target model, the draft model only needs to produce correct predictions on the subset of tokens that is easy enough to propose. During the process of distillation, we identify a subset of "hard" tokens that pose particular challenges for the student model to learn and to predict accurately, regardless of training efforts. Conversely, other tokens are relatively easy to assimilate. We argue that uniformly emphasizing the loss on both "easy" and "hard" tokens may be counterproductive. Attempting to reduce the loss on difficult tokens often comes at the expense of increasing the loss on easy tokens, resulting in suboptimal learning across both categories. To address this issue, we propose a novel approach: deliberately excluding "hard" tokens from the training process. By focusing the loss function exclusively on "easy" tokens, we can more effectively utilize the limited capacity of the student model, thus achieving better alignment with the teacher model on these tokens. This strategic exclusion of hard-to-learn tokens allows the student model to concentrate its resources on mastering the more accessible aspect of the teacher's knowledge, potentially leading to improved overall performance in SD tasks. Our approach thus maximizes the alignment between the draft and target models within the constraints of the draft model's capacity.

In this study, we propose **AdaSPEC**, a novel Knowledge Distillation method designed to bridge the capacity gap between the draft and target models in SD. AdaSPEC operates in two phases:

1: Reference Model Distillation and Token Filtering: A reference model, initialized as a copy of the draft model, is distilled using the target model as its teacher. For simplicity, we assume that the target model has been well fine-tuned to downstream tasks of our interests. Here the reference model serves a crucial role as a token filter. It identifies "hard" tokens—those that are difficult for smaller models to predict accurately—by comparing the perplexity differences between the reference and draft models on the training data.

2: Selective Draft Model Distillation: Finally, the draft model undergoes distillation using a filtered dataset. The reference model removes the previously identified "hard" tokens, allowing the draft model to focus its limited capacity on learning to predict the remaining, more manageable tokens accurately.

We conduct extensive experiments on a wide range of models and downstream tasks, where we benchmark AdaSPEC against DistillSpec and find that AdaSPEC sucessfully pushes the limit of SD—across all tasks and model setups, AdaSPEC consistently achieves higher acceptance rates (up to 15%; see Table 1).

2 Preliminaries

In this section, we provide a formal overview of the foundational concepts. We begin with the mathematical framework of SD, followed by a description of various evaluation metrics. Finally, we explore language model families and their significance in enabling techniques like SD to bridge performance gaps between models of different sizes.

Speculative Decoding. Speculative Decoding [7, 16, 29, 20, 32, 6, 17, 18, 30, 6, 21, 26, 32, 20] is originally proposed to accelerate LLM inference by employing a compact draft model to predict potential output sequences in advance and then verified by a larger target model. The typical framework of SD is formulated as follows. Let M_p and M_q denote the large target model and the compact draft model, respectively. SD leverages the draft model to autoregressively generate γ tokens $\boldsymbol{z} \triangleq \{z_i\}_{i=1}^{\gamma} \sim q_\theta(\cdot \mid \boldsymbol{x})$ based on the input $\boldsymbol{x} = [x_1, x_2, \ldots, x_t]$, which includes the prompt and previously generated tokens. The target model then verifies these proposed tokens by evaluating their probabilities $\{p(z_i \mid \boldsymbol{x}, \boldsymbol{z}_{<i})\}_{i=1}^{\gamma}$ in parallel.

Both models generate probability distributions $p(z_{i+1} \mid \boldsymbol{x}, \boldsymbol{z}_{<i})$ and $q(z_{i+1} \mid \boldsymbol{x}, \boldsymbol{z}_{<i})$ for each token $i = 1, \ldots, \gamma$ in a single forward pass. Using a greedy decoding strategy, only the tokens with the highest probabilities are selected for generation or verification. The sampling functions are:

$$S_p(\boldsymbol{z}_{<i}) = \arg\max_{z_{i+1}} p(z_{i+1} \mid \boldsymbol{x}, \boldsymbol{z}_{<i}), \tag{1}$$

$$S_q(\boldsymbol{z}_{<i}) = \arg\max_{z_{i+1}} q(z_{i+1} \mid \boldsymbol{x}, \boldsymbol{z}_{<i}), \tag{2}$$

for each $i = 1, \ldots, \gamma$. The complete sampling and verification process is detailed in Appendix A.1

Acceptance Rate. The acceptance rate, α, measures the accuracy of the draft model M_q compared to the target model M_p. It is calculated as:

$$\alpha = \frac{\text{accept}}{\text{accept} + \text{reject}}. \tag{3}$$

Here, accept and reject are the count of tokens accepted and rejected by M_p, respectively. A higher α indicates greater alignment between M_p and M_q, facilitating faster inference in practical scenarios.

Block Efficiency and Wall-time Improvement. Block efficiency [7, 16], τ, quantifies the average number of tokens generated per iteration. It is defined as the expected number of accepted tokens per block, with a maximum value of $\gamma + 1$ for a block size of γ. The block efficiency can also be expressed in terms of the acceptance rate α [16]:

$$\tau(x) = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}. \tag{4}$$

This metric evaluates how effectively M_q approximates M_p. The speed-up factor for the total wall-time is given by:

$$\text{Speed-up} = \frac{\tau(x)}{\gamma c + 1}, \tag{5}$$

where c is the cost coefficient, representing the ratio of the time taken by a single execution of M_q to that of M_p.

Language Model Families. Modern language models are often developed as part of a family of models that share the same core architecture but differ in scale, typically measured by the number of parameters or the size of the training dataset. These families, such as Llama 3 [11], BERT [10], and Pythia [5], are designed to enable researchers and practitioners to balance computational efficiency and performance based on specific use cases. Within a family, smaller models are generally used for tasks requiring faster inference or lower computational cost, while larger models are leveraged for tasks demanding higher accuracy and richer representations. This structural consistency within a family allows for techniques like Knowledge Distillation [14] and Speculative Decoding [7, 16] to transfer knowledge or align predictions effectively between models of varying sizes.

3 Method

We introduce AdaSPEC, an adaptive distillation framework for SD that enhances the alignment between a target model and a smaller draft model through selective Knowledge Distillation. Given a target model M_p fine-tuned for a specific downstream task, AdaSPEC consists of two key steps: (1) constructing a reference model M_{ref} and (2) selectively distilling knowledge from M_p and M_{ref} to the draft model M_q.

Step 1: Constructing the Reference Model. The reference model M_{ref} is constructed by distilling M_p on a downstream task dataset $\mathcal{D}$ using the DistillSpec framework [32]. The objective is to minimize the forward KL divergence between the target model and the reference model:

$$\mathcal{L}_{\text{KD}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{y}\sim P(\boldsymbol{y}|\boldsymbol{x})}\left[\mathcal{K}\left(P(\boldsymbol{y}|\boldsymbol{x})||R(\boldsymbol{y}|\boldsymbol{x})\right)\right], \tag{6}$$

where $\boldsymbol{x}$ represents the input prefix, $\boldsymbol{y}$ denotes the generated context, $\mathcal{K}$ denotes the forward KL divergence, $P(\boldsymbol{y}|\boldsymbol{x})$ represents the probability distribution of the target model, and $R(\boldsymbol{y}|\boldsymbol{x})$ corresponds to the probability distribution of the reference model.

Step 2: Selective Knowledge Distillation for the Draft Model. To identify learnable tokens for the draft model M_q, we compute token-wise losses based on the predicted distributions of M_{ref} and M_q. Specifically, for each token w, the token-wise KL divergence losses are computed as:

$$\mathcal{L}_{\text{ref}}(w) = \mathcal{K}\left(P(w \mid \text{context})||R(w \mid \text{context})\right), \tag{7}$$

$$\mathcal{L}_{\text{draft}}(w) = \mathcal{K}\left(P(w \mid \text{context})||Q(w \mid \text{context})\right), \tag{8}$$

where $Q(w \mid \text{context})$ is the probability predicted for token w by the draft model, given the context.

Next, we calculate the difference in token-wise losses:

$$\Delta_{\mathcal{L}}(w) = \mathcal{L}_{\text{draft}}(w) - \mathcal{L}_{\text{ref}}(w). \tag{9}$$

Tokens with higher $\Delta_{\mathcal{L}}(w)$ represent a larger performance gap between M_q and M_{ref} relative to M_p, suggesting that these tokens are **not yet well aligned** but are **highly learnable** for the draft model. Accordingly, we select the subset of tokens with larger $\Delta_{\mathcal{L}}(w)$ values, as they are most promising for improving the alignment between the draft and target models. Specifically, we denote

$$\mathbb{S} = \{\, w \mid \Delta_{\mathcal{L}}(w) \text{ is among the top } k\times 100\% \text{ of all tokens} \,\}, \quad k \in [0, 1].$$

Therefore, the overall loss for training the draft model M_q is:

$$\mathcal{L}_{\text{distill}} = \frac{1}{k \cdot |\boldsymbol{y}|} \sum_{i=1}^{|\boldsymbol{y}|} \mathbb{I}\left[y_i \in \mathbb{S}\right] \cdot \mathcal{L}_{\text{draft}}(y_i), \tag{10}$$

where $\mathbb{I}[\cdot]$ is the indicator function that equals 1 if the condition inside the brackets is satisfied, and 0 otherwise. It ensures that only the selected learnable tokens contribute to the loss calculation. The whole filtering process is shown in figure 1.

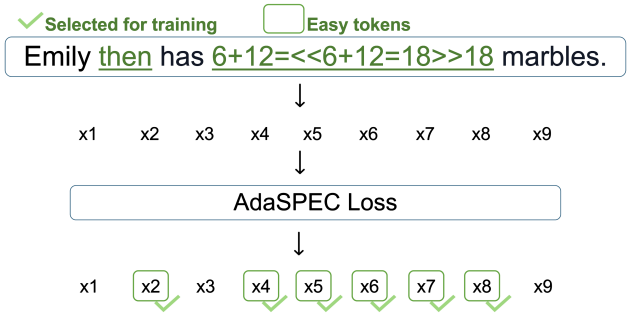

Figure 1: **Overview of AdaSPEC distillation process**: AdaSPEC selects the most training-effective tokens and distills on these tokens.

4 Experiments

We evaluate AdaSPEC through comprehensive experiments across diverse domains and conduct detailed ablation studies to analyze its impact on the acceptance rate α.

4.1 Experimental Setup

Our experimental framework employs GPT-like decoder-only Transformer models in two distinct configurations, designed to evaluate performance across different parameter scales while maintaining tokenizer consistency for SD:

• **Small-to-Large Model Configuration:** A draft model *Pythia-31M* paired with target model *Pythia-1.4B* [5]. These models share architecture and tokenizer, providing an ideal test case for same-family knowledge transfer.

• **Medium-to-Large Model Configuration:** A draft model *CodeGen-350M* paired with target model *Phi-2* [24, 1]. While from different families, these models use an aligned tokenizer to ensure token-level consistency, allowing us to evaluate cross-family KD.

We test these two configurations on a diverse set of five tasks, each representative of a specific domain to provide a robust evaluation framework for AdaSPEC: GSM8K [9] (A benchmark for multi-step arithmetic reasoning), Alpaca [27] (A comprehensive instruction following dataset), MBPP [3] (A Python programming challenge set for code generation), CNN/Daily Mail [22] (A long-form summarization task), and XSUM [23] (An extreme summarization challenge).

Reference Model Training. To ensure a consistent starting point and fair comparison, both the draft model and the reference model are initialized from the same pre-trained model. For each task, we first fine-tune the target model on the task-specific dataset to establish a strong baseline. The reference model is then trained using the method from DistillSpec [32].

4.2 Baseline Setup

We compare AdaSPEC against DistillSpec [32], the current state-of-the-art method for SD. Although AdaSPEC builds upon DistillSpec's training framework for its reference model, it introduces novel token selection mechanism. To evaluate its effectiveness, we evaluate both methods under two settings: a resource-efficient scenario with fixed training duration and a scenario optimized for maximum performance:

• **3-Epoch Setting:** Both reference and draft models are trained for exactly 3 epochs, a standard practice in LLM fine-tuning that balances task-specific performance with general capability retention [4]. This controlled training duration effectively prevents overfitting while ensuring adequate task adaptation. This setting evaluates model effectiveness under typical resource constraints and provides insights into rapid adaptation scenarios.

• **Optimal-Epoch Setting:** Models are trained for a variable number of epochs, treated as a tunable hyperparameter, to maximize task-specific performance. While this approach may lead to overfitting to the specific task at the expense of performance on other tasks, it allows us to thoroughly evaluate the upper bound of performance. The optimal number of epochs is determined empirically. Specifically, for GSM8K, the number of target epochs is chosen according to validation accuracy, while for the rest of the experiments it is chosen according to validation perplexity. Afterwards, we distill the reference model and pick the one with highest α on validation set. Eventually, this model serves as reference to train our draft model. For robustness, we only select the optimal epoch from 1, 3, 6, 10, 15, 20 and 30 (for XSUM and CNN/Daily Mail we select from 1, 3, 6, 10 for training efficiency). This configuration enables evaluation of both methods under less constrained scenarios, where achieving optimal task performance takes precedence over maintaining general capabilities.

While Zhou et al. [32] employs a more extensive training schedule in their DistillSpec experiments, our study adopts a more resource-efficient approach due to computational constraints. In the **Optimal-Epoch Setting**, we limit training to a maximum of 30 epochs, striking a balance between performance optimization and computational feasibility. Complete hyperparameter configurations and training specifications for both DistillSpec and AdaSPEC are detailed in Appendix A.2.

4.3 Main Results

We summarize the main experimental results in Table 1.

Table 1: Main experimental results for AdaSPEC compared to DistillSpec under two settings: 3-Epoch and Optimal-Epoch. Metrics include acceptance rate (α).

Task	**3-Epoch** (α)				**Optimal-Epoch** (α)			
	Pythia-31M → 1.4B		**CodeGen-350M → Phi-2**		**Pythia-31M → 1.4B**		**CodeGen-350M → Phi-2**	
	DistillSpec	AdaSPEC	DistillSpec	AdaSPEC	DistillSpec	AdaSPEC	DistillSpec	AdaSPEC
GSM8K	57.58%	**62.63%**	79.49%	**82.79%**	66.19%	**68.28%**	81.49%	**83.48%**
Alpaca	44.34%	**47.25%**	56.48%	**58.80%**	65.41%	**65.79%**	58.05%	**60.36%**
MBPP	46.88%	**47.73%**	87.36%	**88.76%**	49.88%	**65.12%**	86.60%	**87.70%**
CNN/Daily Mail	73.05%	**74.22%**	79.33%	**80.63%**	80.15%	**80.89%**	85.01%	**86.29%**
XSUM	47.24%	**49.11%**	58.88%	**59.93%**	56.11%	**57.80%**	66.78%	**68.19%**

Acceptance Rate Analysis. We evaluate performance using the acceptance rate α, defined as the proportion of draft-model-generated tokens validated by the target model. As shown in Table 1, AdaSPEC consistently achieves higher acceptance rates than DistillSpec across all tasks and model configurations, demonstrating superior draft-target model alignment.

4.4 Analysis

To provide detailed insights into AdaSPEC's effectiveness, we conduct in-depth analyses on two representative configurations:

• **Pythia-31M/1.4B on GSM8K (3-Epoch):** This configuration examines performance on arithmetic reasoning under constrained training conditions, representing scenarios with limited computational resources and the need for generalization. Since reasoning is typically considered as an additional capability beyond general language modeling, this setup ensures that the model retains its core abilities while effectively handling arithmetic tasks.

• **Pythia-31M/1.4B on CNN/Daily Mail (Optimal-Epoch):** This setup investigates extractive summarization with extended training, demonstrating the model's ability to optimize for task-specific objectives. In real-world applications, models are sometimes specifically deployed for summarizing long-form contents such as news reports, emails, or web pages, requiring dedicated fine-tuning. Thus, the Optimal-Epoch setting is chosen to maximize the model's summarization capabilities.

Task-Level Acceptance Rate Distribution. We first analyze the distribution of acceptance rates across tasks for both methods. As illustrated in Figure 2, AdaSPEC demonstrates consistently superior performance compared to DistillSpec. The acceptance rate histograms for both tasks exhibit a significant rightward shift under AdaSPEC, indicating more frequent successful draft predictions. This systematic improvement in acceptance rate suggests that AdaSPEC's selective distillation approach effectively enhances draft-target model alignment across diverse task contexts.

Logit Margin Distributions Across Tokens. Next, we analyze the distribution of top-2 logit margins across tokens for both methods. The logit margin, defined as the difference between the logits of the top-1 and top-2 predicted tokens, serves as a measure of prediction confidence. A positive margin indicates a correct draft model prediction, while a negative margin signifies an incorrect prediction that would be rejected in SD.

As shown in Figure 2, AdaSPEC demonstrates superior logit margin distributions compared to DistillSpec across both GSM8K and CNN/Daily Mail datasets. AdaSPEC exhibits:

• Higher frequency and magnitude of positive margins, indicating more frequent and confident correct predictions.

• Lower frequency and magnitude of negative margins, suggesting less frequent and less severe prediction errors.

These patterns demonstrate that AdaSPEC achieves better draft-target model alignment through its selective distillation approach, enabling more effective knowledge transfer from the target model to the draft model.

KL-Divergence Distribution Across Tokens. We further analyze the Kullback-Leibler (KL) divergence between draft and target models' token prediction distributions on both GSM8K and CNN/Daily Mail datasets. As illustrated in Figure 2, AdaSPEC exhibits consistently lower KL divergence values compared to DistillSpec across both tasks, demonstrated by significant leftward shifts in the distributions. This systematic reduction in KL divergence across different tasks and tokens indicates that AdaSPEC's selective distillation approach achieves tighter alignment between draft and target model predictions, corroborating our previous findings on acceptance rates and logit margins.

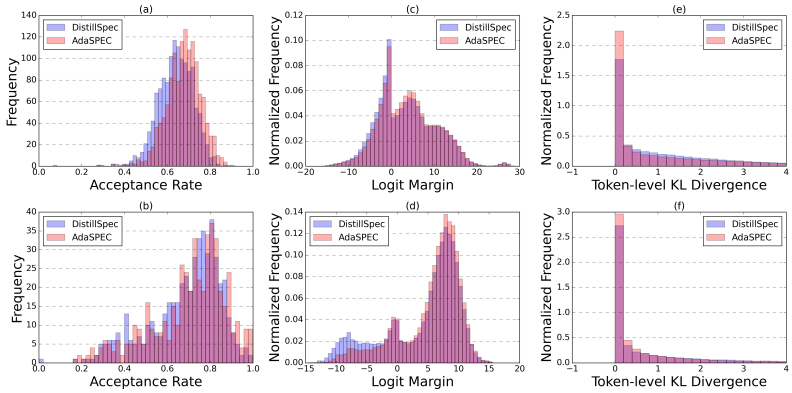

Figure 2: **Comparative analysis of AdaSPEC and DistillSpec performance across multiple metrics on GSM8K (a, c, e) and CNN/Daily Mail (b, d, f) datasets:** (a-b) Task-level acceptance rate distributions showing AdaSPEC's superior performance across tasks. (c-d) Logit margin distributions demonstrating AdaSPEC's improved prediction confidence with higher positive margins and lower negative margins. (e-f) Token-level KL divergence distributions indicating better draft-target model alignment for AdaSPEC with consistently lower divergence values. The results demonstrate AdaSPEC's more effective knowledge transfer and improved draft-target model alignment compared to DistillSpec across different evaluation metrics.

Case Studies. We conduct detailed case studies on GSM8K and CNN/Daily Mail datasets. A consistent pattern emerges: AdaSPEC's prediction errors form nearly a subset of DistillSpec's errors, as illustrated in Figure 3. This pattern demonstrates the general effectiveness of AdaSPEC's targeted training approach in improving alignment and reducing inference discrepancies.

GSM8K, with its natural division between mathematical and non-mathematical tokens, offers particularly insightful analysis. During training, AdaSPEC predominantly selects mathematics-related tokens for focused learning (see Appendix A.3). During inference, this selective approach translates into significantly improved prediction accuracy for mathematical tokens compared to DistillSpec, as shown in Figure 3. These results demonstrate AdaSPEC's ability to identify and prioritize task-critical tokens during training, leading to more precise draft-target model alignment.

4.5 Ablation Study

To systematically evaluate the effectiveness of different components in AdaSPEC, we conduct comprehensive ablation studies across the following four key dimensions. All experiments are conducted on GSM8K and MBPP with Pythia 1.4B (target) and Pythia 31M parameters (draft). All models are trained for 3 epochs.

Token Selection Mechanism. To evaluate our token selection strategy, we compare models trained on the top 40% of tokens (selected based on KL-divergence margin) against those trained on the bottom 40%. As shown in Table 2, models trained on the top 40% tokens consistently outperform those trained on the bottom 40%, with the latter performing even worse than the reference model. The improvement is particularly pronounced on the MBPP dataset, where token selection yields up to a 6% performance gain. These results demonstrate that AdaSPEC effectively enhances model alignment by focusing on more learnable tokens during Knowledge Distillation.

Training Method. To demonstrate that AdaSPEC's benefits extend beyond Knowledge Distillation, we replace the distillation process for both reference and draft models with direct fine-tuning. Table 3

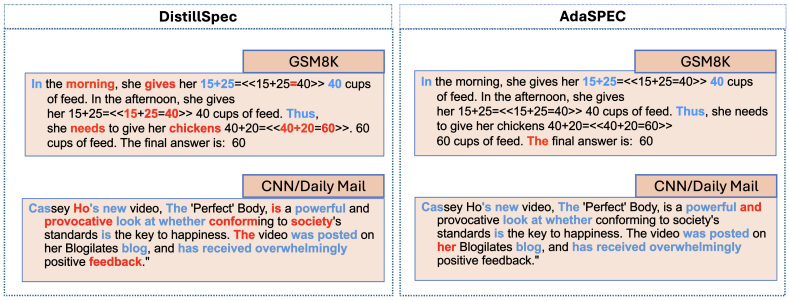

Figure 3: **Comparison of prediction errors between AdaSPEC and DistillSpec on GSM8K and CNN/Daily Mail Datasets:** Tokens highlighted in blue represent errors made by both methods, while tokens highlighted in red indicate errors unique to the corresponding method. As can be seen, AdaSPEC's errors form nearly a subset of DistillSpec's errors, demonstrating the effectiveness of AdaSPEC's selective training approach in reducing inference discrepancies.

Table 2: Ablation study results for token selection strategies.

Sub-Strategy	**GSM8K**		**MBPP**	
	Reference α	Draft α	Reference α	Draft α
Top 40%	59.77%	**63.22%**	42.22%	**48.22%**
Bottom 40%	**59.77%**	49.03%	**42.22%**	39.75%

Table 3: Ablation study results for training methods.

Sub-Strategy	**GSM8K**		**MBPP**	
	Reference α	Draft α	Reference α	Draft α
Distillation	59.77%	**63.22%**	42.22%	**48.22%**
Fine-tuning	59.64%	**63.13%**	41.42%	**45.61%**

Table 4: Ablation study results for distillation methods.

Sub-Strategy	**GSM8K**		**MBPP**	
	Reference α	Draft α	Reference α	Draft α
KL	59.77%	**63.22%**	42.22%	**48.22%**
TVD	9.32%	9.09%	3.86%	6.76%
RKL	30.22%	30.05%	13.17%	15.61%

reveals two key findings: (1) fine-tuned draft models outperform the distillation baseline (reference model) across both datasets, confirming that our token selection process aids model convergence; and (2) fine-tuned draft models achieve up to 4% improvement over their reference counterparts, indicating that our token selection mechanism's benefits generalize beyond distillation to broader training scenarios.

Distillation Method. We expand AdaSPEC to more distillation approaches: Reverse KL (RKL) and Total Variation Distance (TVD) [28]. With $k = 0.4$ for all methods, we observe that token selection significantly improves the acceptance rate by 6% on MBPP when using forward KL. However, when using RKL and TVD, the acceptance rate performance degrades. This is primarily attributed to the inherent limitations of RKL and TVD as distillation objectives, which struggle to effectively align the draft and target models in the context of SD. It is worth noting that DistillSpec [32] uses TVD as the distillation function with a batch size of 32 and a training step of 300,000. This prolonged training process not only requires substantial computational resources but also results in the problem of overfitting. Considering these factors, we ultimately select forward KL divergence as our distillation objective.

Token Selection Ratio. To investigate the impact of token selection ratio, we vary k and compare the final acceptance rate of the draft model. Results in Fig 4 show that typically, lower k values result in better final acceptance rate. To strike a balance between training efficiency and performance, we finally choose $k = 0.4$ in most cases.

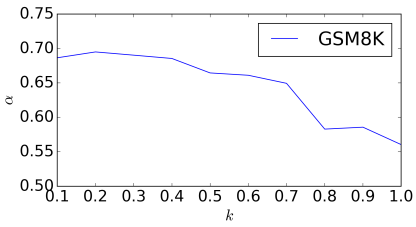

Figure 4: **Impact of token selection ratio k on acceptance rate for GSM8K:** Results show a general trend where lower k values (0.2-0.4) yield higher acceptance rates.

4.6 Additional Experimental Results

Wall Clock Speed-up. To investigate AdaSPEC's potential to accelerate end to end decoding in a real world setting, we use frontier inference engine vLLM [15] on one single A100 GPU and report speed-up in Table 5. Results show that an expected 10∼20% speed-up could be easily achieved compared with DistillSpec, demonstrating the effectiveness of our approach.

Table 5: Generation speed for AdaSPEC with VLLM on one single A100 GPU. We use greedy decoding and report time to generate a sentence and one token. We use Pythia-31M→1.4B and the models are trained for 3 epochs. On these tasks, AdaSPEC exhibits 10∼20% speed-up.

		Speed (s/sentence)	Speed (tokens/s)
MBPP	DistillSpec	0.69	149.15
	AdaSPEC	**0.57**	**181.67**
GSM8K	DistillSpec	0.51	227.86
	AdaSPEC	**0.48**	**241.34**
CNN/DailyMail	DistillSpec	0.76	248.49
	AdaSPEC	**0.67**	**283.50**

Integration with Advanced SD Methods. To demonstrate the orthogonality and generalizability of AdaSPEC beyond vanilla speculative decoding (SD), we integrate our method with EAGLE [17], an advanced SD algorithm featuring tree attention and adaptive expansion strategies. Following the standard 3-Epoch training setup on the ShareGPT dataset, we evaluate both training accuracy and generation speed on MT-Bench. As shown in Table 6, AdaSPEC consistently improves both accuracy and decoding efficiency within the EAGLE framework.

Table 6: Vicuna-7B-v1.3 [8, 31] with 3-Epoch finetuning following original EAGLE recipe. Here, training accuracy refers to first-generated-token accuracy in the training set.

	Eagle	Eagle + AdaSPEC
Training Accuracy ↑	75.3%	**76.3%**
Speed (s/sentence) ↓	8.85	**8.06** (-8.9%)
Speed (tokens/s) ↑	63.48	**68.21** (+7.45%)

Table 7: Acceptance rate of larger model configuration with 3-Epoch GSM8K.

	GSM8K
DistillSpec	84.43%
AdaSPEC	**86.21%**

Results on Larger Models. We conduct an additional GSM8K evaluation using a combination of the Qwen2.5-0.5B and Qwen2.5-32B models. When trained with 3 epochs, AdaSPEC reaches an acceptance rate of 86.21% while DistillSpec achieves 84.43%, as shown in Table 7. This shows that our approach can easily scale up to larger models.

Results on Mixed Dataset. To further validate AdaSPEC's ability to work on blended tasks, we mix GSM8K with MBPP in training and validate α separately. Specifically, we first train the target on MBPP and then on GSM8K, each for 3 epochs. The reference and draft models also follow the same process. The

Table 8: Performance on mixed datasets.

	MBPP	GSM8K
DistillSpec	69.63%	72.75%
AdaSPEC	**70.69%**	**78.41%**

results in Table 8 reveals that AdaSPEC strives to retain the original model's capabilities as much as possible, with less forgotten knowledge.

5 Dicussion

Size Gap Between Target and Draft Models. In traditional SD settings, the size gap between the draft model and the target model is often within 10x. In this work, we demonstrate that AdaSPEC effectively bridges the performance gap, even when the size difference is substantial — up to 64 times in our experiments. By leveraging selective token filtering and Knowledge Distillation, AdaSPEC can enhance token acceptance rates and help maintain generation quality, providing more opportunities to use significantly smaller draft models.

Model Size Gap and Performance Gains. From Table 1, we observe that the performance gain of AdaSPEC over DistillSpec becomes more pronounced as the size gap between the reference and target models increases (e.g., from CodeGen-350M $\rightarrow$ Phi-2 to Pythia-31M $\rightarrow$ 1.4B). This trend is consistent across both 3-Epoch and Optimal-Epoch settings. The result aligns well with our motivation: when the capacity discrepancy between models widens, direct Knowledge Distillation tends to suffer from representation mismatch, making it harder for the smaller model to absorb all teacher signals uniformly. AdaSPEC's adaptive mechanism mitigates this issue by selectively aligning easier tokens first, effectively narrowing the transfer gap. Consequently, the larger the size difference, the greater the relative improvement AdaSPEC achieves.

Connection with Lin et al. [19]. A similar token selection method is proposed in Lin et al. [19], which focuses on identifying and prioritizing harder-to-learn tokens (opposite to the motivation of AdaSPEC) during pre-training. Different from their design, our approach focuses on addressing the limited capacity of the draft model in SD. Specifically, we focus on identifying and filtering out challenging tokens, allowing the draft model to concentrate on learning easier-to-predict tokens. Our selective distillation process ensures that the draft model aligns more effectively with the target model on tokens that are more tractable, given its constrained capacity. By doing so, we maximize the draft model's limited resources while maintaining high-quality predictions in SD tasks. Thus, the essential difference lies in the distinct objectives of pre-training and Speculative Decoding.

Limitations. As a preliminary study on selective training for SD, we limit our study on simple loss-related token filter. In future work, one can design more adaptive filtering strategies as well as integrate AdaSPEC with tree-based or multi-step verification frameworks to further improve both speed and quality of LLM inference.

6 Conclusion

We present AdaSPEC, a novel approach for training more efficient draft models for SD. AdaSPEC introduces selective token filtering based on reference model perplexity gaps, enabling draft models to focus limited capacity on tokens where alignment with the target model is most achievable. Experiments show it outperforms baselines in arithmetic reasoning, instruction following, code generation, and summarization with higher acceptance rates.

References

[1] Marah Abdin, Jyoti Aneja, Sebastien B, Caio Mendes, Weizhu Chen, Allie Giorno, Ronen Eldan, Sivakanth Gopi, Suriya Gunasekar, Mojan Javaheripi, Piero Kauffmann, Yin Tat Lee, Yuanzhi Li, Anh Nguyen, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Michael Santacroce, and Yi Zhang. Phi-2: The surprising power of small language models, 12 2023.

[2] Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. Raft: A real-world few-shot text classification benchmark, 2022. URL https://arxiv.org/abs/2109.14076.

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[5] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[6] 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.

[7] 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*, 2023.

[8] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[12] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018. URL https://arxiv.org/abs/1706.02677.

[13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[14] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

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

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

[18] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024.

[19] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need, 2024. URL https://arxiv.org/abs/2404.07965.

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

[21] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949, 2024.

[22] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[23] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018. URL https://arxiv.org/abs/1808.08745.

[24] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[25] OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[26] Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. *arXiv preprint arXiv:2406.02532*, 2024.

[27] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[28] Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-divergence minimization for sequence-level knowledge distillation, 2023. URL https://arxiv.org/abs/2307.15190.

[29] 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.

[30] 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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.607.

[31] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

[32] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Both the abstract and beginning part of introduction indicate the scope of the paper's contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in "conclusions" chapter.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: This article is based on experiments and does not require strict theoretical proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Results can be reproduced according to "method" and "experiments" chapters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [Yes]

Justification: Please refer to the "abstract" part.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Please refer to "experiments" chapter.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The experiments related to our method is relatively costly so it is difficult to repeat each setting. However, we testify our method on a wide range of settings and tasks, which exhibits the feasibility of our method.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Please refer to Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Please refer to Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper doesn't release any new models, generators or datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Please refer to the "experiments" chapter.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Technical Appendices and Supplementary Material

A.1 Full Algorithms for AdaSPEC

Algorithm 1 Greedy Speculative Decoding

Input: target model M_p, draft model M_q, input sequence $\boldsymbol{x}$
accept $\leftarrow 0$, reject $\leftarrow 0$, $t \leftarrow \text{len}(\boldsymbol{x})$
▷ Sample γ tokens $z_1, ..., z_\gamma$ from M_q autoregressively.
for $i = 1$ to γ **do**
  $z_i = S_q(\boldsymbol{z}_{<i})$; $\boldsymbol{z} \leftarrow \boldsymbol{z} + z_i$
end for
▷ Verify in parallel
$\boldsymbol{z}' \leftarrow [S_p(\boldsymbol{z}_{<1}), S_p(\boldsymbol{z}_{<2}), ..., S_p(\boldsymbol{z}_{<\gamma})] \triangleq [z_1^{'}, z_2^{'}, ..., z_\gamma^{'}]$
for $i = 1$ to γ **do**
  if $z_i^{'} \neq z_i$ **then**
    reject $\leftarrow$ reject $+ 1$; **break**
  end if
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + z_i$, accept $\leftarrow$ accept $+ 1$
  if $z_i = <\text{eos}>$ **then**
    return $\boldsymbol{x}$, accept, reject
  end if
end for
$\boldsymbol{x} \leftarrow \boldsymbol{x} + S_p(\boldsymbol{z}_{<1})$; **goto** 4

Algorithm 2 AdaSPEC: Selective Distillation for Speculative Decoding

Input: dataset $\mathcal{D}$, target model M_p, draft model M_q, fraction k
Step 1. Fine-tune M_p:
  Train M_p on $\mathcal{D}$ (via standard LM fine-tuning) to obtain M_p^*.
Step 2. Reference Model Training:
  Initialize $M_{\text{ref}} \leftarrow M_q$.
  Distill M_{ref} from M_p^* on $\mathcal{D}$ (e.g. forward KL).
Step 3. Selective Distillation:
  (a) Compute losses: For each token w and context,

$$\mathcal{L}_{\text{ref}}(w) = \mathcal{K}\left(P(w \mid \text{context}) || R(w \mid \text{context})\right),$$
$$\mathcal{L}_{\text{draft}}(w) = \mathcal{K}\left(P(w \mid \text{context}) || Q(w \mid \text{context})\right),$$
$$\Delta_{\mathcal{L}}(w) = \mathcal{L}_{\text{draft}}(w) - \mathcal{L}_{\text{ref}}(w).$$

  (b) Filter tokens:

$$\mathbb{S} = \{\, w \mid \Delta_{\mathcal{L}}(w) \text{ is among the top } k \times 100\% \text{ of all tokens} \,\}, \quad k \in [0, 1].$$

  (c) Distill on filtered set:

$$\min_{M_q} \mathcal{L}_{\text{distill}} = \frac{1}{k \cdot |\boldsymbol{y}|} \sum_{i=1}^{|\boldsymbol{y}|} \mathbb{I}\left[y_i \in \mathbb{S}\right] \cdot \mathcal{L}_{\text{draft}}(y_i)$$

return M_q (draft model in SD)

A.2 Implementation Details

We use the hyperparameters in Table 9. For 3-Epoch setting, both reference and draft model are distilled for 3 epochs. For Optimal-Epoch setting, the target model is first fine-tuned to maximize performance on validation set. Specifically, for GSM8K, the number of target epochs is chosen

according to validation accuracy, while for the rest of the experiments it is chosen according to validation perplexity. Afterwards, we distill the reference model and pick the one with highest α on validation set. Eventually, this model serves as the reference model to train our draft model. For robustness, we only select the optimal epoch from 1, 3, 6, 10, 15, 20 and 30 (for XSUM and CNN/Daily Mail we select from 1, 3, 6, 10 for training efficiency). Note that when performing token selection, we apply the linear scaling rule for learning rate adjustment [12].

Table 9: Experimental hyperparameters.

Task	Hyperparameter	3-Epoch		Optimal-Epoch	
		Pythia 31M→1.4B	Codegen-350M→Phi-2	Pythia 31M→1.4B	Codegen-350M→Phi-2
GSM8K	Batch size	16	16	16	16
	Learning rate	3e-4	3e-4	3e-4	3e-4
	Epochs for target model	3	3	6	3
	Epochs for reference model	3	3	15	30
	Epochs for draft model	3	3	30	30
	Filter fraction k	0.4	0.4	0.4	0.4
Alpaca	Batch size	16	16	16	16
	Learning rate	3e-4	3e-4	3e-4	3e-4
	Epochs for target model	3	3	1	1
	Epochs for reference model	3	3	15	20
	Epochs for draft model	3	3	30	15
	Filter fraction k	0.4	0.4	0.4	0.4
MBPP	Batch size	8	8	8	8
	Learning rate	1e-5	1e-4	1e-5	1e-4
	Epochs for target model	3	3	1	1
	Epochs for reference model	3	3	30	10
	Epochs for draft model	3	3	6	6
	Filter fraction k	0.4	0.4	0.4	0.4
CNN/Daily Mail	Batch size	16	16	16	16
	Learning rate	1e-4	1e-4	1e-4	1e-4
	Epochs for target model	3	3	1	1
	Epochs for reference model	3	3	10	10
	Epochs for draft model	3	3	10	10
	Filter fraction k	0.4	0.4	0.4	0.4
XSUM	Batch size	16	16	16	16
	Learning rate	3e-4	1e-4	3e-4	1e-4
	Epochs for target model	3	3	1	1
	Epochs for reference model	3	3	10	10
	Epochs for draft model	3	3	10	10
	Filter fraction k	0.4	0.4	0.4	0.4

A.3 AdaSPEC Example Tokens

Here, we showcase some example tokens (Listing 1) that AdaSPEC selects while training on GSM8K. These selected tokens are typically mathematical related tokens, such as digits and operators.

```
{ "scored", "8", "in", "thus", "9", "x", "1", "=", "<<", "9", "*", "91", "=", "19",
    ">>", "8", "19", "Em", "because", "28", "28", "\+", "8", "28", "+", "90", "18",
     "9", "18", "18", "18", "-", "8", "=", "99", "99", "99", "+", "100", "The", "
    final", "answer", "100", "equal", "12", "+", "7", "=", "19", ">>", "19", "packs
    ", "19", "5", "24", "total", "(", "24", "*(", "2", ")=", "16", ">>", "16", "J",
     "spends", "inside", "because", "-", "(", "inside", "16", "iley", "3", "18", "
    spends", "12", "In", "total", "they", "+", "12", "=", "<<", "8", "+", "=", "20",
     "/", "=", "10", "10", "The", "earned", "final", "answer", "difference", "-",
    "=", "13", "*", "2", "26", ">>", "26", "twice", "26", "18", "=", "26", "18",
    "8", "The", "final", "answer", ":", " ", "8", }
```

Listing 1: Selected tokens during GSM8k training.

A.4 Minimal Code Implementation of AdaSPEC

The core of AdaSPEC could be implemented with $\sim$ 100 lines of code. We show it in Listing 2.

```
def compute_loss(
       self,
       model,
       inputs,
       return_outputs=False,
```

```
        num_items_in_batch=None
):
    labels = inputs["labels"][:, 1:]

    outputs = model(**inputs)
    with torch.no_grad():
        target_outputs = self.target_model(**inputs)
        ref_outputs = self.ref_model(**inputs)

    logits = outputs["logits"]
    target_logits = target_outputs["logits"]
    ref_logits = ref_outputs["logits"]

    loss_fct = KLDivLoss(reduction="none")

    shift_logits = logits[..., :-1, :].contiguous()
    shift_target_logits = (target_logits[..., :-1, :]
                           .contiguous())
    shift_ref_logits = (ref_logits[..., :-1, :]
                        .contiguous())

    shift_logits = shift_logits.view(
        -1, model.config.vocab_size)
    shift_target_logits = (shift_target_logits
                           .view(-1, model.config.vocab_size))
    shift_ref_logits = (shift_ref_logits
                        .view(-1, model.config.vocab_size))
    mask = labels.ne(IGNORE_INDEX).flatten().unsqueeze(-1)

    shift_logits = (
        torch.masked_select(shift_logits, mask=mask)
        .view(-1, model.config.vocab_size))
    shift_target_logits = \
        (torch.masked_select(shift_target_logits, mask=mask)
         .view(-1, model.config.vocab_size))
    shift_ref_logits = \
        (torch.masked_select(shift_ref_logits, mask=mask)
         .view(-1, model.config.vocab_size))

    p = F.softmax(shift_target_logits, dim=-1)
    q_log = F.log_softmax(shift_logits, dim=-1)
    actual = loss_fct(q_log, p)

    q_log = F.log_softmax(shift_ref_logits, dim=-1)
    ref = loss_fct(q_log, p)

    actual = actual.sum(dim=-1)
    ref = ref.sum(dim=-1)

    k = self.k
    delta = actual - ref
    mask = delta >= torch.quantile(
        delta, 1 - k, dim=0, keepdim=True)

    if num_items_in_batch is not None:
        loss = torch.masked_select(actual, mask=mask).sum()
        loss = loss / num_items_in_batch
    else:
        loss = torch.masked_select(actual, mask=mask).mean()

    return (loss, outputs) if return_outputs else loss
```

Listing 2: AdaSPEC implementation with PyTorch.

A.5 Broader Impact

AdaSPEC can be used mainly to improve generation speed of LLMs, which is a positive influence to reduce potential electric energy consumption for serving LLMs. However, this technique may also be used for some models that is non-compliance with regulations and ethics, such as models that generate discriminatory contents.

A.6 Experiments compute resources

Here we list the estimated GPU hours; see Table 10.

Table 10: GPU hours of training models on A100 GPUs.

Task	**3-Epoch**		**Optimal-Epoch**	
	Pythia-31M → 1.4B	**CodeGen-350M → Phi-2**	**Pythia-31M → 1.4B**	**CodeGen-350M → Phi-2**
GSM8K	1	3	15	50
Alpaca	1	3	15	50
MBPP	1	3	15	50
CNN/Daily Mail	60	200	200	700
XSUM	60	200	200	700