AdaSPEC: Selective Knowledge Distillation for Efficient Speculative Decoding

Yuezhou Hu[*1], Jiaxin Guo[*2], Xinyu Feng[3], Tuo Zhao[3]
[1]UC Berkeley, [2]Tsinghua University, [3]Georgia Tech

Presenter: Jiaxin Guo
Tsinghua University
https://xinyuerufei.github.io/

November 7, 2025

Speculative Decoding (SD)

Speculative Decoding accelerates LLM inference by:

- Using a **small draft model** to propose tokens
- Verifying them with a **large target model**

State-of-the-art SD methods use Knowledge Distillation (KD) to align draft model with target model, whose optimization target is minimizing the **global KL divergence** between the draft and target model output distributions over *all* tokens.

Motivation: Current Method's Bottleneck

However, minimizing the overall KL divergence does not necessarily lead to high acceptance rate!

- Due to its limited capacity, draft models often struggle to fully assimilate the knowledge of the target model.
- In extreme cases, the large size gap between the draft and target models can even cause training to fail to converge.

Key Insight: Not All Tokens Are Equal

Tokens vary in **learnability** for small draft models:

- "Learnable" tokens: tokens where model can improve the most
- "Non-learnable" tokens: too difficult for model yo learn

Insight: Focus distillation only on tokens that matter for α!

- Filter out hard-to-learn tokens
- Allocate model capacity to learn "easy" tokens well

Goal: Maximize token acceptance rate, not minimize global KL.

AdaSPEC: Two-Stage Selective Distillation

Stage 1: Build a Reference Model

- Train M_{ref} from target M_p using standard KD (e.g., DistillSpec)
- M_{ref} shares architecture with draft model M_q

Stage 2: Select Learnable Tokens

- Compute per-token KL losses:

$$L_{\text{ref}}(w) = \mathrm{KL}(P(w\|c)\|R(w\|c)), \quad L_{\text{draft}}(w) = \mathrm{KL}(P(w\|c)\|Q(w\|c))$$

- Define token "learnability" by margin:

$$\Delta L(w) = L_{\text{draft}}(w) - L_{\text{ref}}(w)$$

- Select top-k tokens with largest $\Delta L(w)$ (most improvable)

Examples: What Tokens AdaSPEC Selects

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.

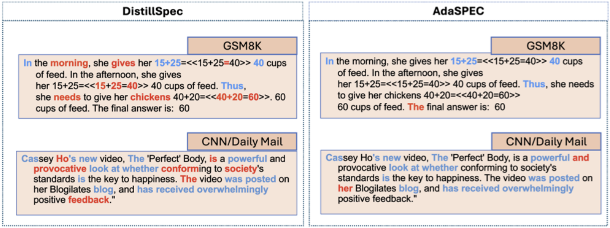

Figure: AdaSPEC vs. DistillSpec

Training Objective

Only selected tokens contribute to distillation loss:

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

where $S = \{w \mid \Delta L(w) \text{ in top } k\%\}$.

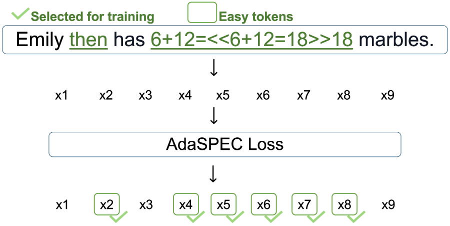

Figure: AdaSPEC distillation pipeline: token filtering via reference model.

Experimental Setup

Model Pairs:

- Pythia-31M $\rightarrow$ Pythia-1.4B (same family)
- CodeGen-350M $\rightarrow$ Phi-2 (cross-family, aligned tokenizer)

Tasks:

- GSM8K (math), Alpaca (instruction), MBPP (code), CNN/DailyMail & XSUM (summarization)

Baselines:

- DistillSpec (SOTA for SD)

Training Settings:

- 3-Epoch (resource-constrained)
- Optimal-Epoch (performance-maximized)

Main Results: Acceptance Rate α

Task	3-Epoch (α)				Optimal-Epoch (α)			
	Pythia-31M $\rightarrow$ 1.4B		CodeGen-350M $\rightarrow$ Phi-2		Pythia-31M $\rightarrow$ 1.4B		CodeGen-350M $\rightarrow$ 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%**

AdaSPEC consistently outperforms DistillSpec across all tasks and settings, with up to **+15%** gain in α.

Analysis: Why AdaSPEC Works

Logit Margin

- AdaSPEC has more **positive margins** → more correct predictions
- Fewer **negative margins** → fewer rejections

KL Divergence

- Lower KL across tokens → better alignment

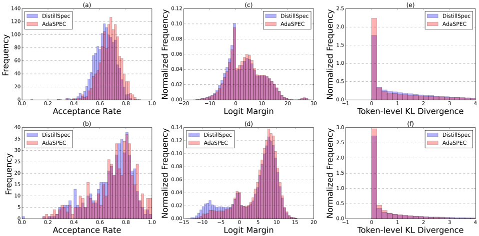

Figure: Logit margin and KL distributions (GSM8K & CNN/DM).

End-to-End Speedup & Scalability

		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**

	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%)

AdaSPEC is general, scalable, and orthogonal to SD frameworks.

Conclusion

- Proposed **AdaSPEC**: selective KD for speculative decoding
- Uses reference model to **filter hard tokens**, focus on learnable ones
- Achieves **higher acceptance rate** (+up to 15%) and **faster generation**
- Works across tasks, model families, and different SD variants (such as Eagle)

Code: `https://github.com/yuezhouhu/adaspec`