AdaSTaR: Adaptive Data Sampling for Training Self-Taught Reasoners

Woosung Koh, Wonbeen Oh, Jaein Jang, MinHyung Lee, Hyeongjin Kim, Ah Yeon Kim, Joonkee Kim, Junghyun Lee, Taehyeon Kim, Se-Young Yun

Paper Code

LG AI Research

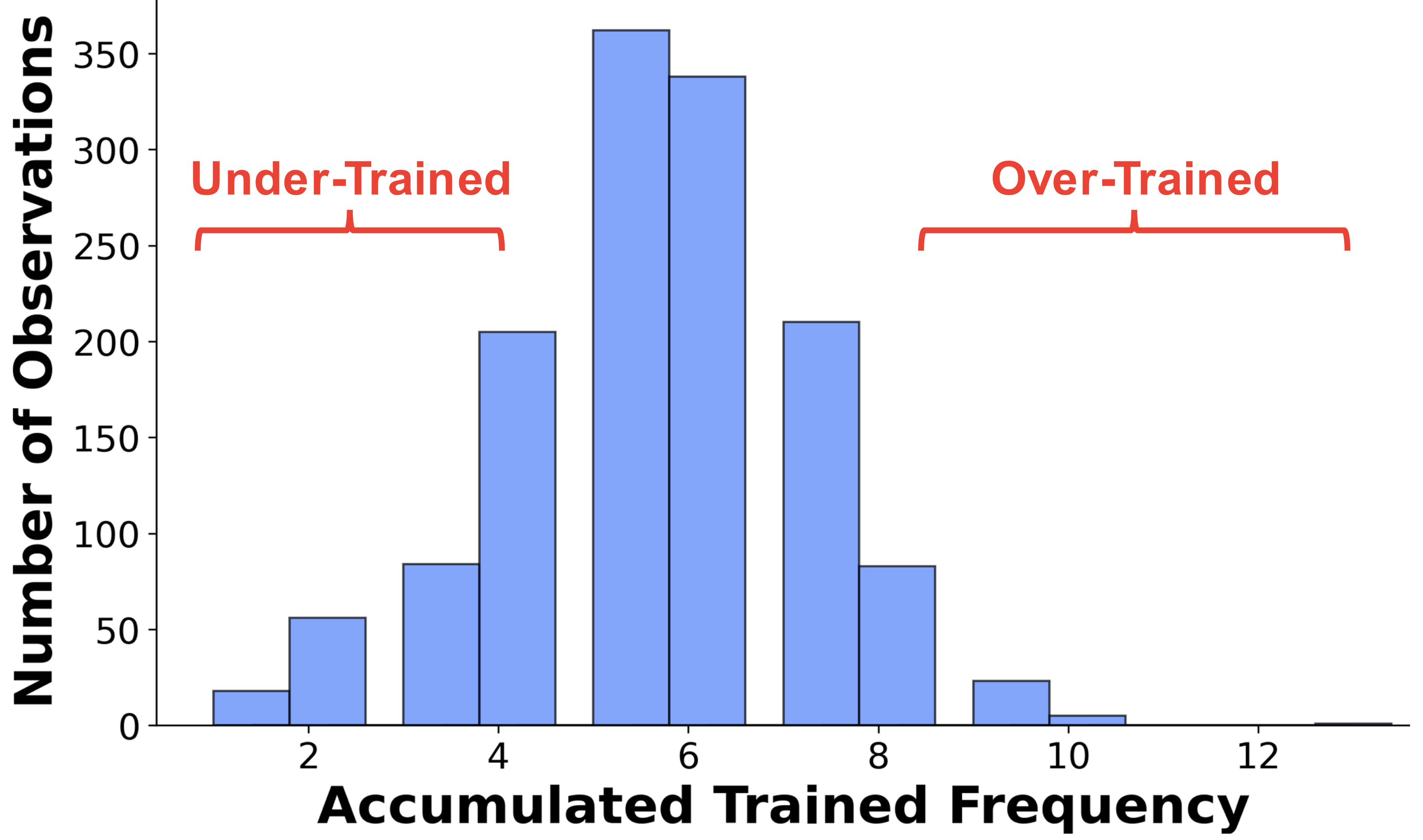
NEURAL INFORMATION PROCESSING SYSTEMS

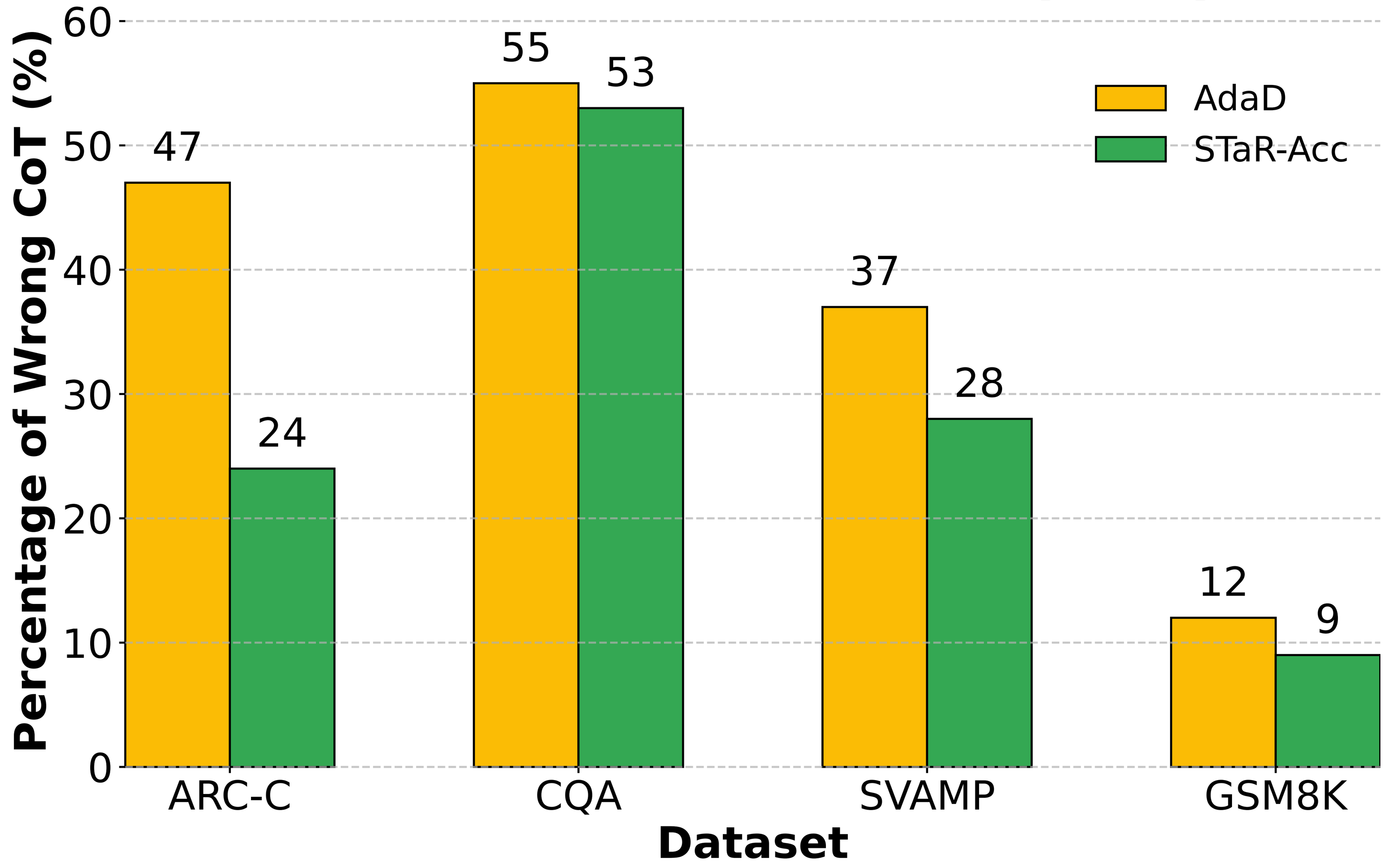
KAIST AI Kim Jaechul Graduate School

Still using sample inefficient Self-Taught Reasoners (Rejection Fine-tuning)?

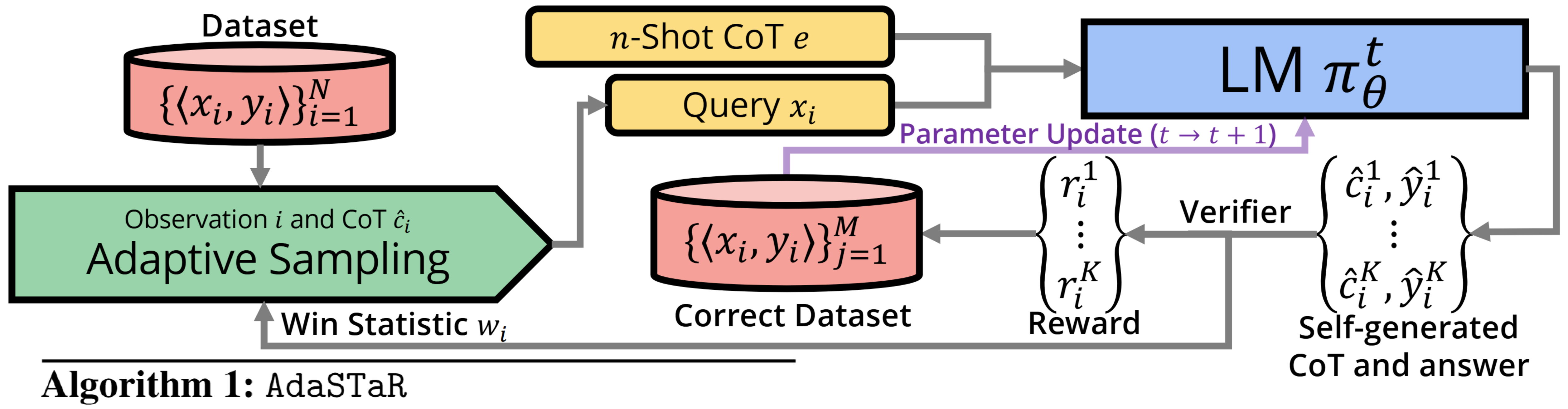
Existing Problem

- **STaR (or RFT)** allows LMs to improve iteratively by training on their own correct CoT solutions.
- Standard STaR relies on **random data sampling.**
- **Inefficiency**: Random sampling causes the model to over-train on easy examples (solved repeatedly) while under-training on challenging ones.
- **False Positives**: Simply prioritizing "hard" problems increases false positives (correct final answer, but flawed reasoning/CoT).





Method: Adaptive Sampling for Diversity and Curriculum



Algorithm 1: `AdaSTaR`

Input: $\mathcal{D}$, $\pi_\theta^{t=1}$, e

`/* AdaD (§3.1; lines 1-14) */`

1 $\tilde{t} \leftarrow \text{dict}\{i : \tilde{t}_i = 0\}_{i=1}^N$;

2 $w \leftarrow \text{dict}\{i : w_i = 0\}_{i=1}^N$;

3 init `HieMinHeap`$(\mathcal{D}, \tilde{t}, w)$;

4 **for** iteration $t = 1, \cdots$ **do**

5 $\quad \mathcal{D}_+^t \leftarrow \emptyset$, $m \leftarrow 0$;

6 $\quad w^{tmp} \leftarrow \text{dict}\{i : w_i^{tmp} = 0\}_{i=1}^N$;

7 $\quad$ **while** $|\mathcal{D}_+^t| < \beta^t$ **do**

8 $\quad\quad i \leftarrow$ `HieMinHeap`$.peek\_next$;

9 $\quad\quad m \leftarrow m + 1$;

10 $\quad\quad$ **for** sample $k = 1, \cdots, K$ **do**

11 $\quad\quad\quad \langle \hat{c}_i, \hat{y}_i \rangle \leftarrow \pi_\theta^t(e, x_i)$;

12 $\quad\quad\quad w_i^{tmp} \leftarrow \frac{k-1}{k} w_i^{tmp} + \frac{1}{k}\mathbb{I}[\hat{y}_i = y_i]$;

13 $\quad\quad\quad$ **if** $\hat{y}_i = y_i$ **then**

14 $\quad\quad\quad\quad \mathcal{D}_+^t \leftarrow \mathcal{D}_+^t \cup \{\langle x_i, \hat{c}_i, \hat{y}_i \rangle\}$;

$\quad$ `/* AdaC (§3.2; lines 15-19) */`

15 $\quad \alpha, \pi_\theta^{t+1} \leftarrow$ `Train`$(\pi_\theta^t, \mathcal{D}_+^t)$;

16 $\quad$ **for** $1, \cdots, \lfloor m\alpha^2 \rfloor$ **do**

17 $\quad\quad i \leftarrow$ `HieMinHeap`$.pop$;

18 $\quad\quad \tilde{t}_i \leftarrow t$, $w_i \leftarrow w_i^{tmp}$;

19 $\quad\quad$ `HieMinHeap`$.push(i, \tilde{t}_i, w_i)$;

- **Pseudocode: Color-coded**
  - **Black:** Existing STaR
  - **Green**: AdaSTaR
- **Diversity (AdaD):** Encourage more diverse training samples (↓ standard deviation)
  - $\tilde{t}_i$: Last sampled iteration
  - w_i: Win statistic (accuracy of i at $\tilde{t}_i$)
- **Curriculum (AdaC):** Sample easier observations earlier and harder later
  - α: Train accuracy

☑ TL;DR: Main Contribution

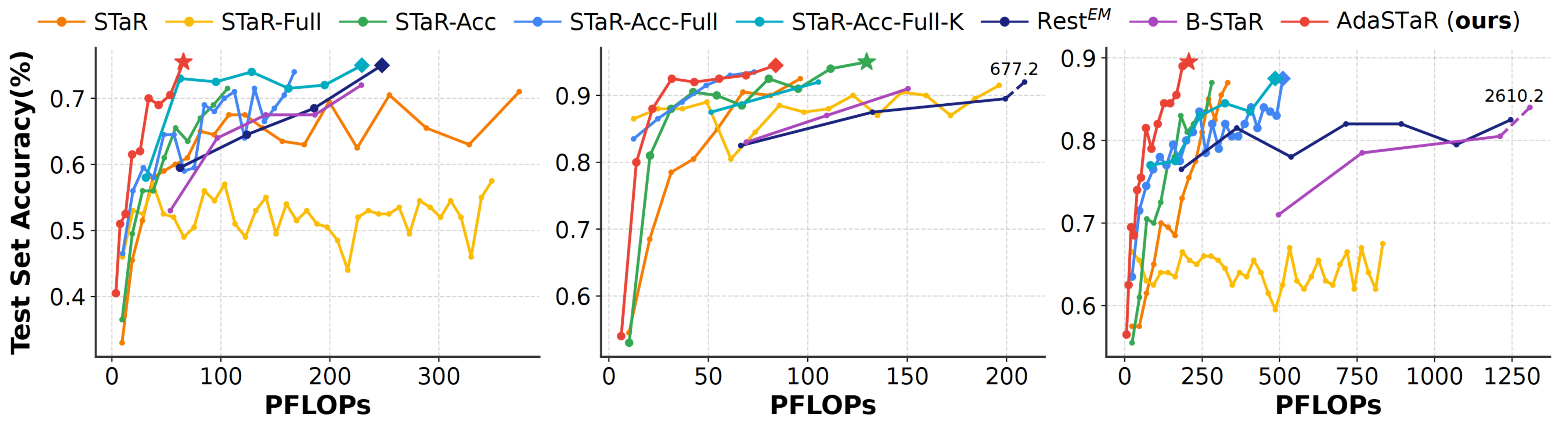
1. AdaSTaR solves the **under-training** and **over-training** of examples (data points) during rejection fine-tuning (RFT) self-improvement post-training.
2. This is achieved using a **compute-overhead free** sampling algorithm.
3. This algorithm results in improved post-training **performance**, and **significantly reduces training FLOPs.**

Empirical Study

- Achieves best test accuracy in **6/6 benchmarks** and reduced training FLOPs by an average of **58.6%.**
- **Pre-trained Base Models:** Llama 3.2 3B, Qwen 2.5 3B, Gemma 7B.

Evaluation	ARC-C			CQA			CLadder 1.5		
Metric	Acc. (↑)	t	PFLOPs (↓)	Acc. (↑)	t	PFLOPs (↓)	Acc. (↑)	t	PFLOPs (↓)
SFT	61.4	1.0 e	7.0	71.8	1.0 e	24.0	31.0	7.0 e	382.3
SFT + 8-CoT	59.0	1.5 e	10.5	71.6	2.5 e	60.1	43.6	3.0 e	163.9
SFT + 5-SC	63.8	4.5 e	31.6	76.4	2.5 e	60.1	45.2	8.0 e	437.0
`STaR`	71.6	13 it	351.4	72.2	25 it	2877.8	53.4	25 it	8427.3
`STaR-Full`	69.8	27 it	739.4	72.2	12 it	1502.7	53.8	19 it	6523.7
`STaR-Acc`	<u>73.2</u>	18 it	639.8	74.6	19 it	1745.3	<u>94.2</u>	28 it	9663.0
`STaR-Acc-Full`	71.8	5 it	**135.8**	<u>76.0</u>	10 it	1158.3	<u>94.2</u>	15 it	4465.4
`STaR-Acc-Full-K`	71.4	3 it	302.2	73.0	4 it	1760.9	80.0	6 it	6382.3
`ReST`^{EM}	70.8	4 it	637.1	72.8	2 it	1548.4	53.4	5 it	10498.3
`B-STaR`	67.8	2 it	222.8	68.4	2 it	<u>800.9</u>	52.8	4 it	<u>3937.3</u>
`AdaSTaR` (**ours**)	**73.8**	10 it	<u>174.4</u> (↓ 72.7%)	**78.0**	20 it	**779.3** (↓ 32.7%)	**95.6**	23 it	**3610.0** (↓ 19.2%)

Evaluation	ANLI			GSM8K			SVAMP		
Metric	Acc. (↑)	t	PFLOPs (↓)	Acc. (↑)	t	PFLOPs (↓)	Acc. (↑)	t	PFLOPs (↓)
SFT	64.2	4 e	262.9	61.0	2.5 e	177.3	57.0	5.5 e	21.7
SFT + 8-CoT	65.2	5 e	328.7	68.0	1 e	70.9	61.5	7.5 e	29.6
SFT + 5-SC	49.2	2 e	131.5	67.2	2.5 e	177.3	61.5	5.5 e	21.7
`STaR`	61.0	23 it	4195.3	<u>76.0</u>	4 it	409.2	71.0	20 it	373.8
`STaR-Full`	57.6	13 it	2604.6	72.6	4 it	684.8	57.5	37 it	348.5
`STaR-Acc`	<u>64.8</u>	22 it	3528.4	**77.0**	3 it	<u>305.2</u>	71.5	10 it	<u>106.2</u>
`STaR-Acc-Full`	64.6	5 it	**986.0**	74.6	2 it	333.0	74.0	18 it	167.3
`STaR-Acc-Full-K`	58.8	4 it	2528.4	**77.0**	2 it	1456.5	<u>75.0</u>	7 it	229.3
`ReST`^{EM}	63.0	9 it	10938.5	**77.0**	2 it	2229.1	<u>75.0</u>	4 it	247.8
`B-STaR`	59.4	10 it	6373.4	73.6	3 it	2120.2	72.0	5 it	228.9
`AdaSTaR` (**ours**)	**66.8**	21 it	<u>1340.9</u> (↓ 62.0%)	**77.0**	2 it	**19.3** (↓ 93.7%)	**75.5**	9 it	**65.7** (↓ 71.3%)



- Star ★ and diamond ◆ represents the **best** and **second best** accuracy.
- Training curve visualized up to the peak performance; afterwards it overfits.
- Dashed line represents methods that use large FLOPs that lead outside of the figure; numeric annotation denotes FLOPs at peak performance.