FEWER TOKENS, ADAPTIVE THINKING: DYNAMIC ROUTING FRAMEWORK WITH SOFT BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) improves large language model reasoning but requires costly token-by-token decoding. Latent reasoning reduces explicit decoding but can be unstable and opaque. Hybrid reasoning combines sampled tokens with continuous context. However, using Hybrid inputs at every step may introduce noise and leave the token-efficiency potential of latent reasoning underused. To address these limitations, we propose **DASH**, a **D**ynamic routing framework with **A**daptive **S**oft budgets for **H**ybrid reasoning. At each step, DASH selects between Token-only and Hybrid candidates using the current reasoning state and predictive uncertainty. This makes continuous-context use state-dependent while retaining token anchoring. An adaptive soft budget guides the reasoning scale without truncating generation. DASH learns routing and budget allocation through outcome-based reinforcement learning (RL). A three-stage curriculum establishes reliable routing, rewards shorter correct trajectories, and adapts the soft budget. No CoT annotations or intermediate labels are required. This avoids costly supervision and lets DASH discover reasoning paths that better balance correctness and length efficiency. Across five mathematical benchmarks and three model scales, DASH achieves higher accuracy with shorter trajectories than supervised fine-tuning with CoT (SFT-CoT) and outperforms representative latent-reasoning baselines. Compared with Hybrid Reasoning Policy Optimization (HRPO), it produces substantially shorter trajectories at an accuracy cost, achieving higher accuracy gain per reasoning position on four arithmetic benchmarks.

1 INTRODUCTION

Chain-of-Thought (CoT) prompting improves reasoning in large language models (LLMs) through explicit intermediate steps (Wei et al., 2022). Reinforcement learning (RL) with verifiable rewards provides further gains (Zhihong Shao, 2024; Guo et al., 2025). However, decoding each step increases latency, memory use, and serving cost. Recent methods shorten traces through reinforcement learning, token skipping, or trajectory compression (Aggarwal & Welleck, 2025; Xia et al., 2025; Wang et al., 2025c). They still rely on sequential token generation.

Latent reasoning moves intermediate computation into continuous states (Deng et al., 2023; 2024; Goyal et al., 2024; Pfau et al., 2024; Geiping et al., 2025). Chain of Continuous Thought (Coconut) and Continuous Chain-of-Thought via Self-Distillation (CODI) learn such states from explicit CoT supervision (Hao et al., 2025; Shen et al., 2025). Compressed Latent Reasoning (CoLaR) dynamically compresses reasoning (Tan et al., 2025). Render-of-Thought (RoT) uses rendered CoT as visual supervision (Wang et al., 2026). These methods reduce explicit decoding, but continuous recurrence can be unstable and difficult to inspect. Alignment with predefined CoT may also restrict reasoning to reference linguistic paths and obscure precision-critical steps.

Hybrid reasoning bridges explicit and latent reasoning. Hybrid Reasoning Policy Optimization (HRPO) retains sampled tokens and adds projected hidden features through learnable gates (Yue et al., 2025). This preserves readable trajectories while supplying continuous context. However, HRPO applies Hybrid inputs at every reasoning step. Such fusion may add noise when a sampled token already provides a reliable transition. HRPO optimizes answer correctness without an explicit length objective. These properties motivate selective context use and explicit length optimization.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

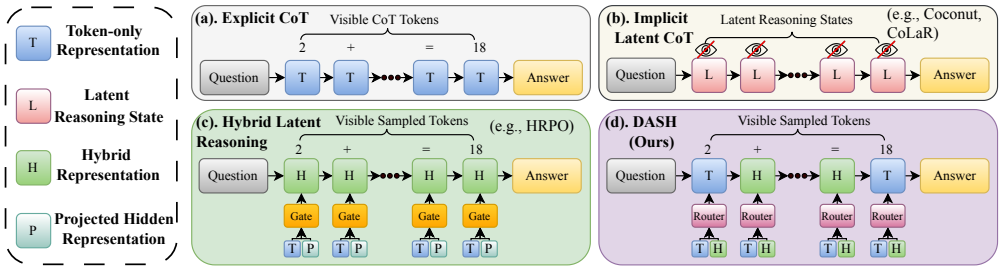


Figure 1: Comparison of reasoning paradigms. (a) Explicit CoT uses visible tokens. (b) Latent CoT uses hidden continuous states. (c) HRPO fuses token embeddings and projected hidden features during reasoning. (d) DASH dynamically selects Token-only or Hybrid inputs during reasoning.

To address these gaps, we propose **DASH**, a **D**ynamic routing framework with **A**daptive **S**oft budgets for **H**ybrid reasoning. Figure 1 illustrates the main difference. At each reasoning step, a context-aware router selects between Token-only and Hybrid candidates using its learned, state-dependent policy. The Token-only candidate preserves standard token transitions. The Hybrid candidate adds continuous context. Both candidates remain anchored to sampled tokens.

DASH uses predicted soft budgets to guide reasoning without forcing termination. Outcome-based RL trains the reasoning policy through a three-stage reward curriculum. Stage 1 learns reliable routing from correctness rewards. Stage 2 adds length bonuses for shorter correct trajectories. Stage 3 uses budget-aware rewards to optimize the policy. Rollout outcomes also supervise budget prediction. All stages use final-answer feedback, without CoT annotations or intermediate labels.

Our main contributions are as follows:

- We propose DASH, a length-aware RL framework for token-anchored hybrid reasoning. Staged reward optimization and soft budgets guide compact reasoning.
- We introduce a three-stage curriculum that establishes correctness, rewards shorter successful trajectories, and jointly optimizes the policy and budget.
- Contextual routing selects Token-only or Hybrid inputs. Input-conditioned soft budgets guide length. Both learn from final-answer feedback without CoT annotations.
- Evaluations across five mathematical benchmarks and three model scales demonstrate compact trajectories and higher accuracy than representative latent-reasoning baselines.

2 Related Work

Explicit Chain-of-Thought Reasoning. Path aggregation, self-training, and explicit search strengthen explicit CoT reasoning (Wang et al., 2023; Zelikman et al., 2022; Yao et al., 2023). Step-level supervision further improves intermediate reasoning (Lightman et al., 2024; Zelikman et al., 2024). Recent methods reduce token use through budgets, trajectory refinement, or adaptive compression (Wang et al., 2024; Han et al., 2025; Xiao et al., 2025). These methods remain in token space, where compression may discard information needed later. DASH combines length-aware RL with token-anchored hybrid representations, retaining a continuous-context channel alongside visible reasoning.

Latent and Hybrid Reasoning. Latent reasoning moves intermediate computation into continuous representations. Existing methods use weighted token embeddings, dynamic shortcuts, parallel latent states, or continuous token mixtures (Zhang et al., 2025; Wang et al., 2025b; Wu et al., 2025; Gozeten et al., 2026). HRPO fuses sampled-token embeddings with projected hidden features (Yue et al., 2025). This preserves readable trajectories and adds continuous context. However, pure latent recurrence can be unstable and opaque. CoT alignment requires trajectory supervision and may inherit teacher-defined patterns. Fixed compression may also remove precision-critical information. DASH instead selects Token-only or Hybrid inputs according to the current state. Token anchoring preserves readability, while final-answer feedback avoids CoT alignment.

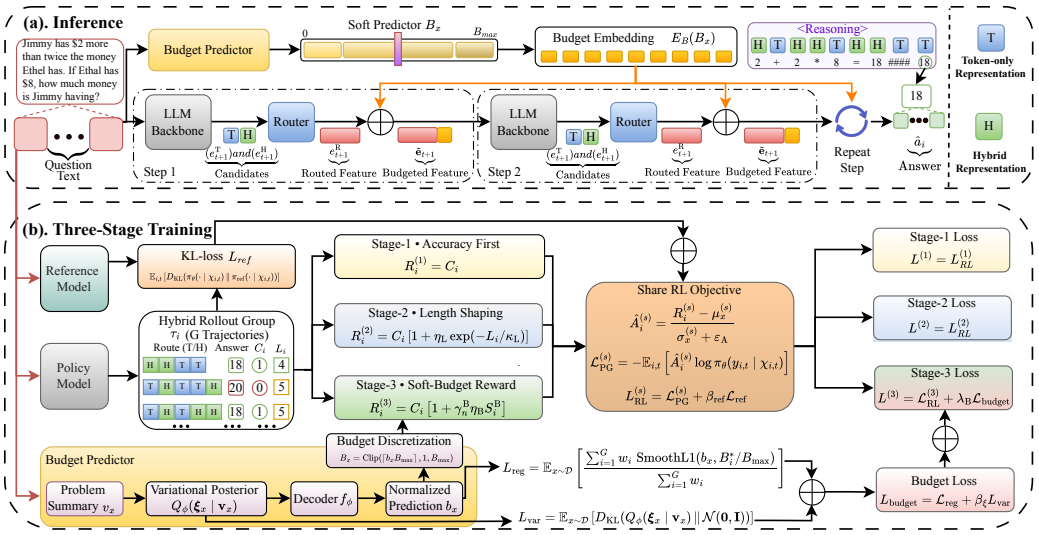


Figure 2: DASH overview. **(a) Inference.** Budget B_x conditions reasoning steps, while the router selects a Token-only or Hybrid input. **(b) Training.** Correctness C_i and length L_i define three-stage rewards for shared group-relative RL. Stage 3 jointly optimizes the policy and Budget Predictor.

Reinforcement Learning. Group Relative Policy Optimization (GRPO) compares multiple responses through group-relative rewards (Zhihong Shao, 2024). Recent variants add length regularization, difficulty weighting, adaptive reasoning decisions, or selective rollouts (Zhang & Zuo, 2025; Wang et al., 2025a; Li et al., 2025; Zheng et al., 2025a). Reinforcement learning has also been extended to continuous reasoning (Ren, 2025; Zheng et al., 2025b). DASH first establishes reliable routing, then introduces length shaping and joint soft-budget optimization. This ordering delays compression pressure until after accuracy-first training. Final-answer feedback trains routing and budget allocation without CoT annotations.

3 Methodology

DASH couples representation selection with length optimization (Figure 2). Routing controls the next input representation. Length rewards and budget conditioning guide trajectory length. Three stages progressively introduce these coupled controls.

Notation. Let $\mathcal{D}$ be a dataset of questions x and reference answers a^*. For each $(x, a^*) \sim \mathcal{D}$, the policy samples a *hybrid rollout group* of G trajectories under one decoding setting. Trajectory $i \in \{1, \dots, G\}$ is $\tau_i = (\mathbf{y}_i, \mathbf{z}_i)$, with T_i tokens $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,T_i})$ and routes $\mathbf{z}_i = (z_{i,t}^m)_{t,m}$, where t indexes tokens and $m \in \{1, \dots, M\}$ indexes the M routing heads. Group members share x and its soft budget B_x when active. Let $L_i \le T_i$ be the reasoning-token count, $\hat{a}_i$ the extracted answer, and n the within-stage update index. Correctness is $C_i = \mathbb{I}[\text{Verify}(\hat{a}_i, a^*) = 1]$, where $\mathbb{I}$ is the indicator function and Verify checks answer correctness and format.

3.1 Dynamic Token–Hybrid Routing

Token-only and Hybrid representations. The vocabulary is $\mathcal{V}$, and the hidden dimension is d. At step t, the backbone state is $\mathbf{h}_t \in \mathbb{R}^d$. The vocabulary projection $\mathbf{W}_o$ produces the next-token distribution $\mathbf{p}_{t+1} = \text{Softmax}(\mathbf{W}_o \mathbf{h}_t)$. We sample $y_{t+1} \sim \text{Cat}(\mathbf{p}_{t+1})$ from this categorical distribution. The pretrained embedding table $\mathbf{E}$ provides the Token-only candidate $\mathbf{e}_{t+1}^{\text{T}} = \mathbf{E}(y_{t+1})$. Superscripts T and H denote the two branches.

The Hybrid candidate combines the hidden state with an expected top-k embedding. The set $\mathcal{K}_{t+1} \subset \mathcal{V}$ contains the k most probable tokens. Within this set, $\tilde{p}_{t+1,v}$ is the renormalized probability of

token v. We use HRPO-style fusion (Yue et al., 2025):

$$\begin{aligned}\mathbf{s}_{t+1} &= \sum_{v\in\mathcal{K}_{t+1}} \tilde{p}_{t+1,v}\mathbf{E}(v),\\ \mathbf{c}_{t+1} &= \mathrm{RMSMatch}\big((1-\alpha)\mathbf{s}_{t+1}+\alpha\mathbf{h}_t,\, \mathbf{e}^{\mathrm{T}}_{t+1}\big),\\ \mathbf{e}^{\mathrm{H}}_{t+1} &= \boldsymbol{\lambda}_{t+1}\odot\mathbf{e}^{\mathrm{T}}_{t+1} + \sqrt{1-\boldsymbol{\lambda}^2_{t+1}}\odot\mathbf{g}_{t+1}\odot\mathbf{c}_{t+1}.\end{aligned} \tag{1}$$

Weighted averaging gives the expected embedding $\mathbf{s}_{t+1}$. Mixing it with the hidden state gives the root-mean-square (RMS)-matched context $\mathbf{c}_{t+1}$, with α controlling the hidden-state contribution. Fusion produces the Hybrid candidate $\mathbf{e}^{\mathrm{H}}_{t+1}$. The gates $\boldsymbol{\lambda}_{t+1}, \mathbf{g}_{t+1} \in (0,1)^d$ control token retention and continuous-context input, respectively. Multiplication ($\odot$), squares, and roots are element-wise. $\mathrm{RMSMatch}$ rescales its first argument to the root-mean-square norm of its second. The sampled-token term preserves token anchoring. Appendix A.1 details the gates and head partitioning.

Alongside the sampled-token embedding, the Hybrid input carries $\mathbf{c}_{t+1}$ from the predictive distribution and preceding hidden state. Thus, the same token can yield different inputs across reasoning states, influencing subsequent predictions without decoding this context into text. DASH performs token-anchored hybrid latent reasoning: every reasoning position emits a visible token, while selected Hybrid inputs also carry continuous information into the next step.

Context-aware routing. A shared contextual feature $\mathbf{r}_{t+1}$ produces per-head Token-only/Hybrid probabilities and offsets to the Hybrid fusion gates. Training uses straight-through (ST) routing: the forward pass uses the selected branch, while the backward pass differentiates through its routing probabilities. Appendix A.2 defines the distribution and replay procedure. For head m, $z^m_{t+1} \in \{\mathrm{T},\mathrm{H}\}$ denotes the selected branch. The routed input combines the selected feature blocks:

$$\mathbf{e}^{\mathrm{R}}_{t+1} = \underset{m=1}{\overset{M}{\mathrm{Concat}}}\left(\mathbf{e}^{z^m_{t+1},m}_{t+1}\right). \tag{2}$$

For routing head m, $\mathbf{e}^{z^m_{t+1},m}_{t+1}$ denotes the selected candidate's m-th d/M-dimensional block. Concat joins these blocks into the next input. Routing heads partition features, not backbone attention. Only reasoning tokens are routed. Answer tokens use standard embeddings. Both routes pass through all Transformer layers. Routing changes the input representation at each reasoning step.

3.2 Adaptive Soft Budgets

DASH predicts an adaptive soft budget B_x, shared by the G rollouts of input x. It guides reasoning without enforcing a length limit.

For an input of N tokens, $\mathbf{h}^x_j$ denotes the j-th hidden state. The *Budget Predictor* has parameters ϕ. It produces a normalized prediction b_x and an integer budget B_x bounded by $B_{\max}$:

$$\begin{aligned}\mathbf{v}_x &= \frac{1}{2}\left(\mathbf{h}^x_N + \frac{1}{N}\sum_{j=1}^{N}\mathbf{h}^x_j\right),\\ Q_\phi(\cdot\mid\mathbf{v}_x) &= \mathcal{N}\big(\boldsymbol{\mu}_\phi(\mathbf{v}_x), \mathrm{diag}\big(\boldsymbol{\sigma}^2_\phi(\mathbf{v}_x)\big)\big), \boldsymbol{\epsilon}_x \sim \mathcal{N}(\mathbf{0},\mathbf{I}),\\ \boldsymbol{\xi}_x &= \boldsymbol{\mu}_\phi(\mathbf{v}_x) + \boldsymbol{\sigma}_\phi(\mathbf{v}_x)\odot\boldsymbol{\epsilon}_x, \qquad \boldsymbol{\xi}_x \sim Q_\phi(\cdot\mid\mathbf{v}_x),\\ b_x &= \mathrm{Sigmoid}(f_\phi(\boldsymbol{\xi}_x)), B_x = \mathrm{Clip}(\lceil b_x B_{\max}\rceil, 1, B_{\max}).\end{aligned} \tag{3}$$

Conditioning on the input summary $\mathbf{v}_x$ defines the variational Gaussian posterior Q_ϕ. Its mean and standard deviation are $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$. Training uses standard Gaussian noise $\boldsymbol{\epsilon}_x$ to sample the latent vector $\boldsymbol{\xi}_x$ by reparameterization. Inference uses the posterior mean. diag forms a diagonal matrix. $\mathbf{I}$ is the identity. A sigmoid maps decoder f_ϕ's output to $b_x \in (0,1)$. The ceiling and $\mathrm{Clip}(v,l,u) = \min(\max(v,l),u)$ give $B_x \in \{1,\ldots,B_{\max}\}$. This budget does not limit generation. It conditions the next input:

$$\tilde{\mathbf{e}}_{t+1} = \mathbf{e}^{\mathrm{R}}_{t+1} + \mathbf{E}_B(B_x), \tag{4}$$

We embed each budget in d dimensions using the learned mapping $\mathbf{E}_B$. The resulting $\tilde{\mathbf{e}}_{t+1}$ is the budget-conditioned backbone input. Each reasoning token receives the same budget embedding. Answer tokens receive none. The predictor is trained in Stage 3.

3.3 THREE-STAGE OUTCOME-BASED OPTIMIZATION

DASH stages correctness rewards, length shaping, and joint soft-budget optimization.

Outcome-to-loss conversion. At stage $s \in \{1, 2, 3\}$, trajectory τ_i receives reward $R_i^{(s)}$. The G rewards for input x are standardized within its rollout group:

$$\hat{A}_i^{(s)} = \frac{R_i^{(s)} - \mu_x^{(s)}}{\sigma_x^{(s)} + \varepsilon_{\mathrm{A}}}, \tag{5}$$

Normalization yields the group-relative advantage $\hat{A}_i^{(s)}$. The group reward mean and standard deviation are $\mu_x^{(s)}$ and $\sigma_x^{(s)}$. The constant $\varepsilon_{\mathrm{A}} > 0$ stabilizes the denominator.

The state before token t is $\chi_{i,t} = (x, \mathbf{y}_{i,<t}, \mathbf{z}_{i,<t})$. It includes all preceding tokens and routes. Shared budget conditioning is implicit in Stage 3. We optimize the policy with

$$\begin{aligned} \mathcal{L}_{\mathrm{PG}}^{(s)} &= -\mathbb{E}_{i,t}\left[\hat{A}_i^{(s)} \log \pi_\theta(y_{i,t} \mid \chi_{i,t})\right], \\ \mathcal{L}_{\mathrm{ref}} &= \mathbb{E}_{i,t}\left[D_{\mathrm{KL}}(\pi_\theta(\cdot \mid \chi_{i,t}) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid \chi_{i,t}))\right], \\ \mathcal{L}_{\mathrm{RL}}^{(s)} &= \mathcal{L}_{\mathrm{PG}}^{(s)} + \beta_{\mathrm{ref}} \mathcal{L}_{\mathrm{ref}}. \end{aligned} \tag{6}$$

$\mathbb{E}_{i,t}$ averages trajectories and positions. The policy π_θ has trainable parameters θ. The reference policy π_{ref} remains frozen. D_{KL} denotes Kullback–Leibler (KL) divergence. The RL loss $\mathcal{L}_{\mathrm{RL}}^{(s)}$ combines the advantage-weighted policy-gradient loss $\mathcal{L}_{\mathrm{PG}}^{(s)}$ with the reference penalty $\mathcal{L}_{\mathrm{ref}}$. The coefficient β_{ref} weights this penalty. Each trajectory advantage weights all its generated tokens. During replay, gradients from subsequent token predictions reach the router through reconstructed inputs and ST routing weights. The token objective trains routing without a separate route-level likelihood term. Appendix A.2 specifies this surrogate gradient.

Stage 1: Accuracy-First Routing. The first stage uses only final-answer correctness. Writing $\mathcal{L}^{(s)}$ for the total Stage-s loss,

$$R_i^{(1)} = C_i, \qquad \mathcal{L}^{(1)} = \mathcal{L}_{\mathrm{RL}}^{(1)} = \mathcal{L}_{\mathrm{PG}}^{(1)} + \beta_{\mathrm{ref}} \mathcal{L}_{\mathrm{ref}}. \tag{7}$$

No length, budget, or route labels are used. The standardized correctness reward weights Equation 6 and guides the learning of the state-dependent routing policy.

Stage 2: Correctness-Conditioned Length Reduction. Starting from the Stage-1 checkpoint, Stage 2 introduces a small length bonus:

$$\begin{aligned} R_i^{(2)} &= C_i \left[1 + \eta_{\mathrm{L}} \exp(-L_i/\kappa_{\mathrm{L}})\right], \\ \mathcal{L}^{(2)} &= \mathcal{L}_{\mathrm{RL}}^{(2)}. \end{aligned} \tag{8}$$

The maximum bonus is $\eta_{\mathrm{L}} \geq 0$. Its decay scale $\kappa_{\mathrm{L}} > 0$ controls the preference for shorter correct trajectories. Incorrect trajectories receive no bonus. This stage adds no length loss or truncation.

Stage 3: Adaptive Soft-Budget Optimization. Stage 3 jointly optimizes the budget-conditioned policy and Budget Predictor. Budgets guide generation and RL rewards. Rollout outcomes supervise budget learning. The predictor learns through its auxiliary loss, without differentiating through the integer budget. For the shared budget B_x, the policy reward is:

$$\begin{aligned} S_i^{\mathrm{B}} &= \begin{cases} 1, & L_i \leq B_x, \\ \exp\big(-(L_i - B_x)/\kappa_{\mathrm{B}}\big), & L_i > B_x, \end{cases} \\ R_i^{(3)} &= C_i \left[1 + \gamma_n^{\mathrm{B}} \eta_{\mathrm{B}} S_i^{\mathrm{B}}\right]. \end{aligned} \tag{9}$$

S_i^{B} measures budget compliance. The scale κ_{B} controls over-budget decay, and η_{B} sets the maximum bonus. During Stage 3, $\gamma_n^{\mathrm{B}} = \min(n/K_{\mathrm{B}}, 1)$ ramps up the bonus over K_{B} updates. Correct trajectories retain their base reward beyond budget. Incorrect ones receive no bonus.

Outcome-based budget targets and loss. For each trajectory, we construct a target budget B_i^*:

$$B_i^* = \begin{cases} \mathrm{Clip}(L_i + \Delta, 1, B_{\max}), & C_i = 1, \\ B_{\max}, & C_i = 0, \end{cases} \tag{10}$$

Correct trajectories target their observed length plus the safety margin $\Delta \geq 0$. Failures target $B_{\max}$, so they cannot encourage compression.

The regression weight is $w_i = 1$ for correct trajectories and $w_i = w_{\mathrm{F}} > 1$ for failures. Using the robust regression loss $\mathrm{SmoothL1}$, we optimize the Budget Predictor by

$$\begin{aligned} \mathcal{L}_{\mathrm{reg}} &= \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{\sum_{i=1}^{G} w_i \ \mathrm{SmoothL1}(b_x, B_i^*/B_{\max})}{\sum_{i=1}^{G} w_i}\right], \\ \mathcal{L}_{\mathrm{var}} &= \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}(Q_\phi(\cdot \mid \mathbf{v}_x) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I}))\right], \\ \mathcal{L}_{\mathrm{budget}} &= \mathcal{L}_{\mathrm{reg}} + \beta_\xi \mathcal{L}_{\mathrm{var}}. \end{aligned} \tag{11}$$

$\mathcal{L}_{\mathrm{reg}}$ fits continuous b_x to normalized targets. β_ξ weights the variational regularizer $\mathcal{L}_{\mathrm{var}}$ in the total budget loss $\mathcal{L}_{\mathrm{budget}}$. The final objective is

$$\mathcal{L}^{(3)} = \mathcal{L}_{\mathrm{RL}}^{(3)} + \lambda_{\mathrm{B}} \mathcal{L}_{\mathrm{budget}}, \tag{12}$$

The coefficient $\lambda_{\mathrm{B}} \geq 0$ weights Budget Predictor supervision.

4 Experiments

4.1 Experiment Settings

Datasets and Evaluation Metrics. We train DASH on GSM8K and evaluate on its test split (Cobbe et al., 2021). Without further training, we evaluate out-of-distribution (OOD) robustness on GSM-Hard, SVAMP, and MultiArith (Gao et al., 2023; Patel et al., 2021; Roy & Roth, 2015), and transfer to competition-level mathematical reasoning on MATH500 (Lightman et al., 2024). We report Pass@1, average reasoning length (#L), accuracy gain per position over supervised fine-tuning without CoT (SFT-w/o CoT, $\Delta\mathrm{Acc}/L$), and end-to-end latency. #L counts positions with representation-dependent costs. Main results report means and 95% confidence intervals (CIs) over five inference seeds per checkpoint (Appendix C).

Base Models and Baselines. We use Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Team, 2025). DASH freezes the backbone and trains low-rank adaptation (LoRA) adapters (Hu et al., 2022) and its auxiliary modules. Baseline adaptation settings are specified in Appendix D. RoT uses the same language backbones and a frozen visual encoder from Qwen3-VL-4B-Instruct. We compare against SFT-w/o CoT, supervised fine-tuning with CoT (SFT-CoT), HRPO (Yue et al., 2025), implicit Chain-of-Thought (iCoT) (Deng et al., 2024), Coconut (Hao et al., 2025), CoLaR-2/5 (Tan et al., 2025), and RoT (Wang et al., 2026). Baseline settings follow their original papers, with total training epochs matched to DASH. Appendix D details reproduction.

Implementation Details. We train DASH on three NVIDIA A800 graphics processing units (GPUs, 80GB each). Stages 1–3 run for 1, 1, and 10 epochs. The per-device batch size is 16, with $G = 4$ rollouts per input. We set the maximum completion length to 1,024 and the maximum soft budget to $B_{\max} = 64$. Module, optimizer, and prompt settings appear in Appendix A.

4.2 Main Results

Reasoning Performance and Generalization. Table 1 shows that DASH improves average Pass@1 over SFT-CoT by 4.96–12.75 percentage points across the three backbones, while reducing reasoning length by 54.1–67.4%. These joint gains show that DASH combines compact reasoning with improved answer quality in these settings. DASH also exceeds iCoT, Coconut, CoLaR, and RoT in average accuracy. It retains visible intermediate reasoning while using continuous context to support subsequent predictions, combining readable traces with higher accuracy.

DASH achieves a favorable accuracy–token cost trade-off relative to HRPO. Across the three backbones, it reduces reasoning tokens by 67.5–79.5%. On Qwen3-4B, DASH cuts average reasoning length from 178.82 to 36.72 tokens, a 79.5% reduction, while maintaining 78.19% Pass@1 compared with HRPO's 83.09%. Thus, DASH saves nearly four-fifths of the reasoning tokens at a cost of 4.90 accuracy points. This makes DASH an efficient choice when reasoning-token cost matters.

Table 1: Pass@1 and reasoning length (#L) across three backbones and four benchmarks. Means and 95% CIs over five inference seeds. Average is unweighted across benchmarks.

Method	GSM8K		GSM-Hard		SVAMP		MultiArith		Average	
	Pass@1↑	#L↓	Pass@1↑	#L↓	Pass@1↑	#L↓	Pass@1↑	#L↓	Pass@1↑	#L↓
Qwen3-1.7B										
SFT-w/o CoT	12.89 ± 0.94	0.00 ± 0.00	3.38 ± 0.53	0.00 ± 0.00	41.40 ± 1.07	0.00 ± 0.00	35.70 ± 1.74	0.00 ± 0.00	23.34	0.00
SFT-CoT	49.39 ± 1.46	118.11 ± 2.93	22.53 ± 0.40	174.67 ± 1.88	60.52 ± 1.08	57.56 ± 1.61	89.90 ± 0.43	59.10 ± 1.49	55.59	102.36
HRPO	74.45 ± 0.86	152.14 ± 0.37	45.73 ± 0.38	264.43 ± 6.36	90.56 ± 0.71	90.62 ± 1.61	97.83 ± 0.64	71.08 ± 0.41	77.14	144.57
iCoT	18.01 ± 0.43	0.00 ± 0.00	4.61 ± 0.72	0.00 ± 0.00	44.60 ± 0.30	0.00 ± 0.00	41.77 ± 0.43	0.00 ± 0.00	27.25	0.00
Coconut	14.50 ± 0.41	6.00 ± 0.00	4.08 ± 0.23	6.00 ± 0.00	36.04 ± 0.66	6.00 ± 0.00	27.67 ± 0.77	6.00 ± 0.00	20.57	6.00
CoLaR-2	17.56 ± 0.54	39.72 ± 0.09	3.85 ± 0.39	43.38 ± 0.13	27.52 ± 0.91	21.21 ± 0.07	52.97 ± 0.71	23.63 ± 0.11	25.48	31.99
CoLaR-5	10.86 ± 0.36	18.99 ± 0.06	2.88 ± 0.20	23.01 ± 0.09	21.74 ± 0.57	10.04 ± 0.03	30.93 ± 0.35	10.85 ± 0.03	16.60	15.72
RoT	7.55 ± 0.64	32.00 ± 0.00	2.08 ± 0.51	32.00 ± 0.00	29.62 ± 0.71	32.00 ± 0.00	18.07 ± 1.16	32.00 ± 0.00	14.33	32.00
DASH (Ours)	64.25 ± 0.62	54.26 ± 0.51	34.28 ± 0.92	76.16 ± 0.66	80.88 ± 0.63	28.23 ± 0.37	93.93 ± 0.19	29.31 ± 0.10	68.34	46.99
Qwen3-4B										
SFT-w/o CoT	27.07 ± 1.18	0.00 ± 0.00	8.01 ± 0.33	0.00 ± 0.00	67.56 ± 0.45	0.00 ± 0.00	85.67 ± 1.11	0.00 ± 0.00	47.08	0.00
SFT-CoT	74.01 ± 0.75	119.14 ± 0.86	42.21 ± 0.80	169.86 ± 3.59	78.34 ± 1.42	51.35 ± 0.95	97.27 ± 0.60	55.66 ± 0.67	72.96	99.00
HRPO	86.10 ± 0.55	142.14 ± 0.45	56.24 ± 0.71	433.31 ± 0.89	92.20 ± 0.25	80.28 ± 0.69	97.80 ± 0.17	59.54 ± 0.31	83.09	178.82
iCoT	29.86 ± 0.64	0.00 ± 0.00	9.25 ± 0.56	0.00 ± 0.00	68.94 ± 0.41	0.00 ± 0.00	84.80 ± 0.27	0.00 ± 0.00	48.21	0.00
Coconut	25.50 ± 0.93	6.00 ± 0.00	7.69 ± 0.49	6.00 ± 0.00	65.40 ± 0.28	6.00 ± 0.00	81.10 ± 1.02	6.00 ± 0.00	44.92	6.00
CoLaR-2	30.98 ± 1.04	39.25 ± 0.08	8.05 ± 0.35	48.52 ± 0.16	54.30 ± 0.53	20.10 ± 0.10	79.03 ± 1.30	21.28 ± 0.07	43.09	32.29
CoLaR-5	26.78 ± 0.45	23.43 ± 0.12	6.26 ± 0.20	32.45 ± 0.15	51.88 ± 0.16	10.99 ± 0.15	74.93 ± 0.48	11.63 ± 0.02	39.96	19.63
RoT	24.99 ± 1.06	32.00 ± 0.00	7.40 ± 0.25	32.00 ± 0.00	67.90 ± 0.39	32.00 ± 0.00	83.37 ± 1.66	32.00 ± 0.00	45.92	32.00
DASH (Ours)	82.06 ± 0.90	38.25 ± 0.37	44.76 ± 0.57	71.21 ± 1.01	87.84 ± 0.45	18.54 ± 0.36	98.10 ± 0.50	18.89 ± 0.07	78.19	36.72
Qwen3-8B										
SFT-w/o CoT	27.35 ± 0.34	0.00 ± 0.00	9.16 ± 0.53	0.00 ± 0.00	70.16 ± 0.44	0.00 ± 0.00	85.43 ± 0.75	0.00 ± 0.00	48.03	0.00
SFT-CoT	75.22 ± 0.94	119.08 ± 0.97	43.79 ± 0.39	170.44 ± 0.71	79.10 ± 0.43	52.34 ± 1.13	96.73 ± 0.65	57.15 ± 0.52	73.71	99.75
HRPO	86.41 ± 0.45	150.34 ± 0.80	57.60 ± 0.43	219.96 ± 2.03	93.78 ± 0.73	81.63 ± 0.81	97.50 ± 0.15	76.25 ± 0.80	83.82	132.05
iCoT	32.16 ± 0.31	0.00 ± 0.00	9.98 ± 0.20	0.00 ± 0.00	67.90 ± 0.30	0.00 ± 0.00	88.27 ± 0.19	0.00 ± 0.00	49.58	0.00
Coconut	26.57 ± 0.59	6.00 ± 0.00	8.17 ± 0.53	6.00 ± 0.00	65.66 ± 0.73	6.00 ± 0.00	83.67 ± 0.87	6.00 ± 0.00	46.02	6.00
CoLaR-2	40.61 ± 0.49	46.13 ± 0.15	10.36 ± 0.17	51.26 ± 0.17	53.30 ± 0.84	24.46 ± 0.09	88.80 ± 1.04	26.95 ± 0.08	48.27	37.20
CoLaR-5	32.87 ± 0.64	21.68 ± 0.10	7.90 ± 0.37	25.65 ± 0.10	46.68 ± 0.46	10.61 ± 0.05	71.70 ± 0.87	11.17 ± 0.01	39.79	17.28
RoT	25.75 ± 1.02	32.00 ± 0.00	8.17 ± 0.40	32.00 ± 0.00	67.78 ± 0.93	32.00 ± 0.00	82.40 ± 1.03	32.00 ± 0.00	46.03	32.00
DASH (Ours)	82.53 ± 0.75	33.50 ± 0.70	45.53 ± 1.11	65.92 ± 1.32	90.14 ± 0.40	16.40 ± 0.38	96.47 ± 0.49	14.42 ± 0.15	78.67	32.56

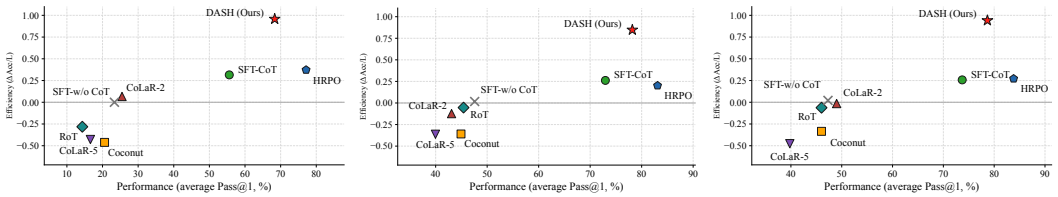
Table 2: MATH500 Pass@1 and reasoning length (#L): means with 95% confidence intervals over five inference seeds. Average aggregates the three backbones with equal weights.

Method	Qwen3-1.7B		Qwen3-4B		Qwen3-8B		Average	
	Pass@1↑	#L↓	Pass@1↑	#L↓	Pass@1↑	#L↓	Pass@1↑	#L↓
SFT-w/o CoT	12.16 ± 0.81	0.00 ± 0.00	20.12 ± 0.96	0.00 ± 0.00	22.12 ± 2.20	0.00 ± 0.00	18.13	0.00
SFT-CoT	22.16 ± 0.97	243.15 ± 4.40	27.68 ± 1.59	269.41 ± 14.96	44.20 ± 2.87	336.95 ± 7.03	31.35	283.17
HRPO	54.48 ± 1.94	478.55 ± 8.93	63.80 ± 1.01	532.18 ± 3.21	69.24 ± 0.98	437.63 ± 4.89	62.51	482.79
iCoT	14.56 ± 1.31	0.00 ± 0.00	23.36 ± 0.44	0.00 ± 0.00	27.80 ± 0.50	0.00 ± 0.00	21.91	0.00
Coconut	11.00 ± 1.43	6.00 ± 0.00	17.36 ± 1.55	6.00 ± 0.00	13.60 ± 0.84	6.00 ± 0.00	13.99	6.00
CoLaR-2	6.12 ± 1.11	36.92 ± 0.16	14.84 ± 1.31	51.61 ± 0.56	15.08 ± 0.89	44.73 ± 0.17	12.01	44.42
CoLaR-5	4.08 ± 1.09	22.08 ± 0.26	13.60 ± 1.50	45.90 ± 0.70	13.24 ± 1.20	33.56 ± 0.64	10.31	33.85
RoT	6.92 ± 0.91	64.00 ± 0.00	16.68 ± 2.44	64.00 ± 0.00	18.52 ± 1.20	64.00 ± 0.00	14.04	64.00
DASH (Ours)	37.12 ± 1.08	98.12 ± 2.16	53.92 ± 1.78	104.05 ± 2.93	54.48 ± 2.48	98.48 ± 2.64	48.51	100.22

On MATH500 (Table 2), DASH transfers without additional training. Across backbones, average Pass@1 rises from SFT-CoT's 31.35 to 48.51 with 64.6% shorter reasoning. These gains extend to more challenging mathematics beyond the training distribution. Compared with HRPO, DASH reduces average reasoning length from 482.79 to 100.22 tokens (79.2%). This substantial saving comes with a 14.00-point accuracy gap (48.51 versus 62.51).

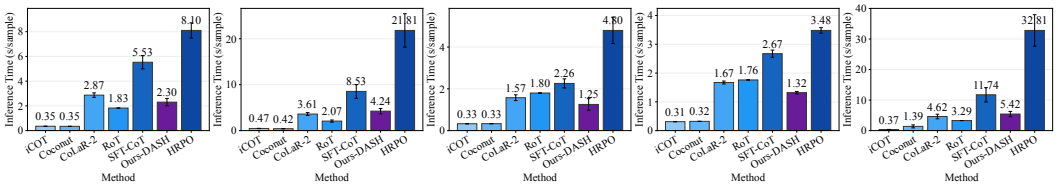
Accuracy Gain per Position and Inference Latency. Figure 4 reports accuracy gain per reasoning position across the four arithmetic benchmarks. DASH obtains the highest scores among the plotted methods: 0.957, 0.847, and 0.941 across the three backbones. These scores are substantially higher than those of HRPO and SFT-CoT. This ratio measures accuracy gain over direct-answer prediction per reasoning position. Wall-clock latency captures runtime, including the per-position costs of each representation.

Figure 5 reports Qwen3-4B end-to-end performance with method-specific decoding (Appendix C.3). Timings include all DASH modules. Its compact trajectories reduce sequential decoding steps and yield faster responses in this evaluation. On the timed examples, DASH achieves 76.4% accuracy at a macro-average latency of 2.91 seconds per sample. Relative to SFT-CoT, it reduces latency by 52.7% and improves accuracy by 9.6 percentage points. Compared with HRPO, DASH cuts latency



(a) Qwen3-1.7B (b) Qwen3-4B (c) Qwen3-8B

Figure 4: Accuracy–efficiency trade-offs across Qwen3-family backbones. Performance is macro Pass@1, and efficiency is $\Delta\mathrm{Acc}/L$ relative to SFT-w/o CoT. Upper right is better.



(a) GSM8K (b) GSM-Hard (c) SVAMP (d) MultiArith (e) MATH500

Figure 5: Inference-time comparison on Qwen3-4B across five mathematical reasoning benchmarks. Bars report mean wall-clock time per sample with batch size 1 over 100 examples per benchmark. Error bars denote 95% Student-t confidence intervals over examples. Runs use method-specific decoding configurations. Appendix C.3 specifies the timing protocol.

by 79.5%, with accuracy of 76.4% versus 82.6%. Faster iCoT and Coconut reach only 43.0% and 43.4% accuracy. DASH therefore combines much faster responses than HRPO with substantially better answer quality than the faster latent baselines.

Soft-Budget Trade-offs. We compare input-adaptive and fixed soft budgets across budget settings under the same Stage 3 reward. Figure 3 plots average accuracy against reasoning length. Larger settings improve accuracy but lengthen traces. Smaller adaptive ceilings lose more accuracy, while larger ceilings offer modest gains with longer trajectories. We choose an adaptive ceiling of 64 to retain much of the accuracy of larger budgets while favoring compact reasoning. Its input-conditioned guidance does not force termination. Appendix E.4 reports the detailed results.

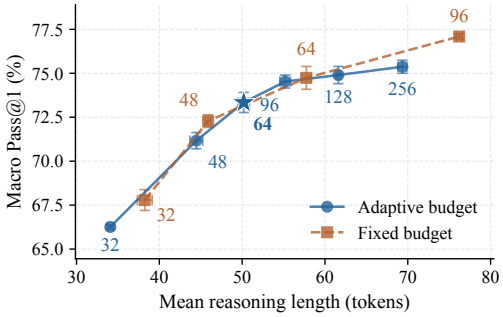
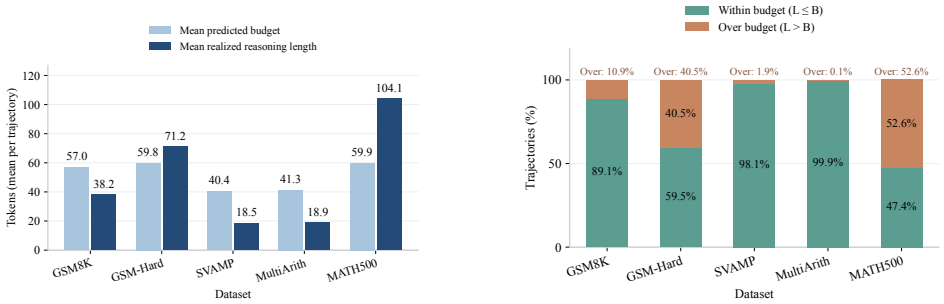


Figure 3: Qwen3-4B five-benchmark means with 95% inference-seed CIs.

Soft-Budget Behavior. Figure 6 examines budget allocation on Qwen3-4B. Predicted budgets are lower on SVAMP and MultiArith, where almost all trajectories remain within budget. On GSM-Hard and MATH500, budgets are higher, yet 40.5% and 52.6% of trajectories exceed them. Short solutions can finish early, while longer ones can continue beyond their allowance. Correct over-budget trajectories retain their base reward, and shortening a within-budget solution brings no further bonus. This separates a preferred reasoning scale from a stopping rule. Predicted budgets encode length preferences learned from rollout outcomes.

4.3 Ablation Study

Table 3 evaluates routing and training stages. Appendix D.7 specifies the controls.



(a) Predicted budgets and reasoning lengths

(b) Budget compliance

Figure 6: Soft-budget behavior of DASH across five benchmarks. (a) Mean predicted soft budgets and realized reasoning lengths. (b) Proportions of trajectories within ($L \leq B_x$) and over ($L > B_x$) their predicted budgets. Aggregation and length counting follow Appendix C.2.

Table 3: Ablations on Qwen3-4B (12 epochs, Appendix D.7). w/o Stage 3 replaces joint soft-budget optimization with extended Stage 2. Means ± 95% CIs.

Variant	Metric	GSM8K	GSM-Hard	SVAMP	MultiArith	MATH500	Average
DASH (Full)	Pass@1↑	82.06 ± 0.90	44.76 ± 0.57	87.84 ± 0.45	98.10 ± 0.50	53.92 ± 1.78	73.34 ± 0.57
	#L↓	38.25 ± 0.37	71.21 ± 1.01	18.54 ± 0.36	18.89 ± 0.07	104.05 ± 2.93	50.19 ± 0.68
Fixed Token-only	Pass@1↑	80.79 ± 0.56	43.43 ± 0.28	87.80 ± 0.59	98.10 ± 0.24	53.44 ± 1.64	72.71 ± 0.32
Routing	#L↓	36.57 ± 0.43	67.41 ± 1.43	17.94 ± 0.48	17.99 ± 0.08	101.18 ± 2.81	48.22 ± 0.60
Fixed Hybrid	Pass@1↑	81.27 ± 0.76	43.97 ± 0.42	88.08 ± 0.60	97.33 ± 0.21	53.84 ± 1.14	72.90 ± 0.33
Routing	#L↓	37.71 ± 0.39	68.48 ± 1.31	18.24 ± 0.55	18.53 ± 0.35	103.70 ± 3.07	49.33 ± 0.44
w/o Stage 2	Pass@1↑	81.02 ± 0.92	43.70 ± 0.25	87.26 ± 0.57	98.13 ± 0.23	52.84 ± 1.52	72.59 ± 0.45
(Length Shaping)	#L↓	36.91 ± 0.62	68.55 ± 1.05	18.60 ± 0.59	18.60 ± 0.09	101.77 ± 2.97	48.89 ± 0.85
w/o Stage 3	Pass@1↑	57.26 ± 0.69	24.85 ± 0.79	81.08 ± 0.30	91.77 ± 0.80	35.40 ± 1.89	58.07 ± 0.70
(Soft-Budget Opt.)	#L↓	19.88 ± 0.34	28.45 ± 0.88	5.62 ± 0.39	7.83 ± 0.23	45.35 ± 0.97	21.42 ± 0.37

Dynamic Token–Hybrid Routing. Fixed routes retain the full curriculum, including length shaping and soft-budget optimization. DASH gains 0.63 and 0.44 accuracy points over Fixed Token-only and Fixed Hybrid, using 1.97 and 0.86 more tokens. These small mean gains recur across three training runs (Appendix D.10). These results support the effectiveness of dynamic routing.

Correctness-Conditioned Length Shaping. Reallocating Stage 2's epoch to Stage 3 reduces Pass@1 from 73.34 to 72.59 and length from 50.19 to 48.89 tokens. Stage 2 distinguishes correct trajectories by length, preparing the policy for budget adaptation without rewarding failures. Its length bonus provides an efficiency signal even when all rollouts in a group are correct.

Joint Soft-Budget Optimization. Replacing extended Stage 2 with Stage 3 raises Pass@1 from 58.07 to 73.34 and length from 21.42 to 50.19 tokens. All benchmarks improve. Equal bonuses for correct within-budget traces remove further compression incentives. Correct over-budget solutions retain their base reward. Budgets condition generation. Successful lengths supply safety-margin targets, and failures receive maximum-budget targets. Together, these changes ease compression pressure while rewarding longer successful solutions.

5 Conclusion

In this paper, we presented DASH for efficient hybrid reasoning. DASH dynamically selects between Token-only and Hybrid representations. An adaptive soft budget guides reasoning length without truncating generation. A three-stage curriculum learns routing, length preferences, and budget allocation from final-answer feedback. No CoT annotations or intermediate supervision are required. Experiments across five mathematical benchmarks and three model scales show that DASH outperforms representative latent-reasoning baselines. It also achieves higher accuracy with shorter trajectories than SFT-CoT. Ablation results support the contributions of learned routing and staged optimization. Future work will further improve token efficiency. We will refine routing and budget learning to shorten reasoning while preserving accuracy.

Artificial Intelligence Use Disclosure

In this work, we used generative artificial intelligence (AI) tools to improve the grammar, clarity, concision, and organization of the manuscript. These tools also assisted with LaTeX formatting, figure and table presentation, and limited development of evaluation and analysis scripts. All AI-assisted text and code were reviewed, revised, and verified by the authors. The authors made all final decisions regarding the research design, method, experimental setup, claims, and interpretation of results. Generative AI was not used to generate datasets or fabricate experimental results. We take full responsibility for the final content of this work.

Ethics Statement

The authors have read and adhere to the ICLR Code of Ethics. This work studies token-efficient reasoning in large language models. It does not involve human subjects, personal or sensitive data, or the collection of new datasets. All experiments use publicly available benchmarks and pretrained models. We follow their licenses and intended research use. DASH may inherit limitations and biases from its base models, but we do not identify risks specific to its routing or budget mechanisms. By reducing unnecessary reasoning tokens, this work may also lower inference cost. Code and model checkpoints will be released to support transparent research.

Reproducibility Statement

The main paper and appendix describe DASH's components and training objectives. Section 3 presents routing, soft budgets, and three-stage optimization. Appendix A.2 specifies the routing probabilities, straight-through estimator, and replay gradient boundaries. The experimental section reports the datasets, baselines, evaluation metrics, model backbones, hardware, optimization settings, and decoding parameters. Additional implementation details and hyperparameters are provided in Appendix. We report results over five inference seeds with 95% confidence intervals. Training scripts, evaluation code, configurations, and final model checkpoints will be released.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

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.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799, 2023.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024.

Alperen Gozeten, Muhammed Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 82881–82914, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gang Li, Ming Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. Disco: Reinforcing large reasoning models with discriminative constrained optimization. *Advances in Neural Information Processing Systems*, 38:57304–57331, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024.

Hangliang Ren. Lsrl: Process-supervised grpo on latent recurrent states improves mathematical reasoning. In *EMNLP (Findings)*, pp. 12534–12545, 2025.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. 38, Main Conference:4646–4668, 2025.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Jiaqi Wang, Kevin Qinghong Lin, James Cheng, and Mike Zheng Shou. Think or not? selective reasoning via reinforcement learning for vision-language models. *Advances in Neural Information Processing Systems*, 38:111196–111234, 2025a.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: Budget-aware evaluation of llm reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19916–19939, 2024.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. System-1.5 reasoning: Traversal in language and latent spaces with dynamic shortcuts. *Advances in Neural Information Processing Systems*, 38:65328–65351, 2025b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Yibo Wang, Haotian Luo, Huanjin Yao, Tiansheng Huang, Haiying He, Rui Liu, Naiqiang Tan, Jiaxing Huang, Xiaochun Cao, Dacheng Tao, et al. R1-compress: Long chain-of-thought compression via chunk compression and search. *arXiv preprint arXiv:2505.16838*, 2025c.

Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Zheng Wei, and Yang Tang. Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45236–45253, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Yang Xiao, Jiashuo Wang, Ruifeng Yuan, Chunpu Xu, Kaishuai Xu, Wenjie Li, and Pengfei Liu. Limopro: Reasoning refinement for efficient and effective test-time scaling. *Advances in Neural Information Processing Systems*, 38:157001–157026, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. 38, Main Conference:5501–5530, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*, 2024.

Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5642–5665, 2025.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Advances in Neural Information Processing Systems*, 38:168990–169012, 2025.

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Advances in Neural Information Processing Systems*, 38:124321–124346, 2025a.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025b.

Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

APPENDIX CONTENTS

A Complete Implementation Details **14**
- A.1 Model Components and Hybrid Fusion . . . 14
- A.2 Token–Hybrid routing. . . . 15
- A.3 Adaptive soft budget. . . . 16
- A.4 Training Objectives and Optimization . . . 17
- A.5 Generation Settings and Prompt . . . 18

B Training and Inference Algorithms **18**
- B.1 Three-Stage Training . . . 18
- B.2 Soft-Budget Inference . . . 23

C Data and Evaluation Protocol **23**
- C.1 Datasets and Generation Protocol . . . 23
- C.2 Evaluation Metrics and Statistical Reporting . . . 24
- C.3 Inference-Latency Protocol . . . 25

D Baseline Reproduction Details **26**
- D.1 Shared Setup and Training Controls . . . 26
- D.2 Training configuration. . . . 27
- D.3 Hyperparameter configuration. . . . 27
- D.4 Comparison with DASH. . . . 27
- D.5 Method-Specific Implementations . . . 29
- D.6 Evaluation alignment. . . . 30
- D.7 Ablation Training Protocol . . . 30
- D.8 Paired Analysis of Routing Ablations . . . 31
- D.9 Routing Ablations with Greedy Token Decoding . . . 32
- D.10 Routing Ablations across Training Seeds . . . 34

E Joint Soft-Budget Optimization and Adaptive Budget Trade-offs **35**
- E.1 Evaluation Setup . . . 35
- E.2 Effectiveness of Joint Soft-Budget Optimization . . . 35
- E.3 Sensitivity to the Maximum Soft Budget . . . 36
- E.4 Adaptive versus Fixed Soft Budgets . . . 36

F Routing Behavior Analysis **38**
- F.1 Routing Preferences across Reasoning Positions . . . 38
- F.2 Probability Changes within Individual Trajectories . . . 39
- F.3 Interpretation and Relation to Fixed-Routing Ablations . . . 39

G Qualitative Analysis and Failure Cases **40**
- G.1 Successful Cases . . . 40
- G.2 Failure Cases . . . 41
- G.3 Case-selection safeguards. . . . 43

A Complete Implementation Details

A.1 Model Components and Hybrid Fusion

Trainable components. The backbone remains frozen throughout training. We optimize the LoRA adapters and DASH-specific modules. These modules include the fusion gates, keep-ratio parameters, and contextual router. Stage 3 additionally trains the Budget Predictor and budget embedding. Table 4 summarizes the module dimensions, fusion settings, routing configuration, and budget initialization. The listed values initialize the gates and budget predictions. These quantities are subsequently learned during training.

Table 4: Module configurations used by DASH.

Module	Parameter	Value
LoRA	Target projections	`q`, `k`, `v`, `o`, `gate`, `up`, `down`
LoRA	Rank / alpha / dropout	32 / 64 / 0
Hybrid candidate	Top-k vocabulary support	$k = 16$
Hybrid candidate	Hidden-state mixing coefficient	0.35
Hybrid fusion	Initial keep-ratio range	$[0.99, 0.999]$
Contextual router	Hidden dimension / routing heads	128 / 1
Budget Predictor	Variational latent dimension	32
Budget Predictor	Maximum budget	$B_{\max} = 64$
Budget Predictor	Initial normalized prediction	0.99
Budget embedding	Dimension / initialization	d / zero

Hybrid fusion gates. Let d denote the backbone hidden dimension. The Hybrid candidate uses a reset gate to determine token retention and an input gate to control the continuous contribution. Both gates depend on the sampled-token embedding and contextual offsets from the router:

$$\begin{aligned} \mathbf{r}_{t+1}^{\text{reset}} &= \text{Sigmoid}\big(\mathbf{W}_r \mathbf{e}_{t+1}^{\mathrm{T}} + \mathbf{b}_r + \bar{\boldsymbol{\delta}}_{t+1}^{r}\big), \\ \mathbf{g}_{t+1} &= \text{Sigmoid}\big(\mathbf{W}_g \mathbf{e}_{t+1}^{\mathrm{T}} + \mathbf{b}_g + \bar{\boldsymbol{\delta}}_{t+1}^{g}\big), \\ \boldsymbol{\lambda}_{t+1} &= \exp\big(-c\ \text{Softplus}(-\boldsymbol{\kappa}) \odot \mathbf{r}_{t+1}^{\text{reset}}\big), \end{aligned} \tag{13}$$

where $\mathbf{W}_r, \mathbf{W}_g \in \mathbb{R}^{d\times d}$, $\mathbf{b}_r, \mathbf{b}_g, \boldsymbol{\kappa} \in \mathbb{R}^d$, and $c = 8$ is a fixed scale. The gate weights, biases, and retention vector $\boldsymbol{\kappa}$ are trainable. The resulting $\boldsymbol{\lambda}_{t+1}$ and $\mathbf{g}_{t+1}$ are d-dimensional vectors with entries in $(0, 1)$. They define the Hybrid candidate in Equation 1. For numerical stability, the implementation adds 10^{-8} inside the square root of its continuous coefficient.

Gate initialization and head partitioning. The fusion module is shared across generation steps. It is applied when constructing the next backbone input, rather than separately within each Transformer layer. For each routing head m, the shared router feature $\mathbf{r}_{t+1}$ produces two scalar offsets:

$$\begin{aligned} \delta_{t+1}^{r,m} &= (\mathbf{w}_{\delta r}^{m})^{\top} \mathbf{r}_{t+1} + b_{\delta r}^{m}, \\ \delta_{t+1}^{g,m} &= (\mathbf{w}_{\delta g}^{m})^{\top} \mathbf{r}_{t+1} + b_{\delta g}^{m}. \end{aligned} \tag{14}$$

Each offset is broadcast to the corresponding contiguous d/M feature coordinates to form $\bar{\boldsymbol{\delta}}_{t+1}^{r}$ and $\bar{\boldsymbol{\delta}}_{t+1}^{g}$. Unlike the shared affine biases $\mathbf{b}_r$ and $\mathbf{b}_g$, these offsets vary with the current reasoning state. They enter the gate logits before the sigmoid, not the token embedding directly. The reset offset affects token retention through $\boldsymbol{\lambda}_{t+1}$, while the input offset controls the continuous contribution through $\mathbf{g}_{t+1}$. Gate modulation and representation selection are distinct outputs of the same contextual module. Gate projections use the full embedding. Only contextual offsets and route choices are partitioned. We use $M = 1$ routing head, separate from backbone attention.

The affine gate layers use the default PyTorch initialization: their weights and biases are sampled uniformly from $[-1/\sqrt{d}, 1/\sqrt{d}]$. The router output projection for the gate offsets is initialized to

zero. For each coordinate j, we sample $u_j \sim \mathcal{U}(0.99, 0.999)$ and initialize

$$\kappa_j = -\log\Big(u_j^{-1/c} - 1\Big). \tag{15}$$

This yields $\lambda_{t+1,j} = u_j^{r_{t+1,j}^{\text{reset}}}$ at initialization, favoring the sampled-token contribution. The interval $[0.99, 0.999]$ specifies the initial base retention. Training subsequently updates the retention parameters and fusion gates.

A.2 Token–Hybrid routing.

The Hybrid candidate is constructed from the top-k vocabulary predictions. Their probabilities are renormalized to form an expected embedding, which is combined with the current hidden state. The context-aware router selects a Token-only or Hybrid representation according to its learned, state-dependent policy. The selected route is recorded with each generated token and retained during trajectory replay. Only reasoning tokens are routed. Answer tokens use standard embeddings.

Contextual features. The router combines the sampled-token embedding, RMS-matched context, current hidden state, and predictive uncertainty:

$$\begin{aligned}\mathbf{r}_{t+1} = \mathrm{SiLU}\big(&\mathbf{W}_{\mathrm{T}}\,\mathrm{LN}(\mathbf{e}^{\mathrm{T}}_{t+1}) + \mathbf{W}_{\mathrm{C}}\,\mathrm{LN}(\mathbf{c}_{t+1}) \\ &+ \mathbf{W}_{\mathrm{H}}\,\mathrm{LN}(\mathbf{h}_t) + \mathbf{W}_{\mathrm{U}}\mathbf{u}_{t+1}\big).\end{aligned} \tag{16}$$

LN denotes non-affine layer normalization (LN). SiLU is the sigmoid linear unit (SiLU) activation. The first three projections map d features to 128. $\mathbf{W}_{\mathrm{U}}$ maps three uncertainty features to 128. All four projections are bias-free. The uncertainty vector contains top-k entropy normalized by $\log k$, the top-two probability margin, and the top-one probability. Entropy uses probabilities renormalized within the top-k set. The margin and top-one value use the original probabilities. The same feature supplies the gate offsets in Equation 14. Uncertainty enters alongside the token and contextual features. The learned projection jointly maps these inputs to route probabilities. Its response to uncertainty therefore depends on the current token and reasoning context.

Routing distribution. For head m, an affine projection of the shared feature produces two active logits for Token-only and Hybrid:

$$\begin{aligned}\boldsymbol{\ell}^m_{t+1} &= \mathbf{W}^m_{\mathrm{R}}\mathbf{r}_{t+1} + \mathbf{b}^m_{\mathrm{R}} + \gamma^{\text{route}}_n \log \boldsymbol{\rho}_0, \\ \boldsymbol{\pi}^m_{\mathrm{R},t+1} &= \mathrm{Softmax}\left(\boldsymbol{\ell}^m_{t+1}/\tau_{\mathrm{R}}\right).\end{aligned} \tag{17}$$

Here, $\mathbf{W}^m_{\mathrm{R}} \in \mathbb{R}^{2\times 128}$ and $\mathbf{b}^m_{\mathrm{R}} \in \mathbb{R}^2$ parameterize the active route probabilities, conditioned on the sampled token and the contextual inputs defining $\mathbf{r}_{t+1}$. The implementation retains an inactive third route logit for checkpoint compatibility. It is masked before normalization. During training and evaluation, each head draws a route from its learned state-conditioned distribution:

$$z^m_{t+1} \sim \mathrm{Cat}(\boldsymbol{\pi}^m_{\mathrm{R},t+1}).$$

Each draw selects one representation for the forward computation. Training records these choices for ST replay. Evaluation applies the learned routing policy without updating parameters.

The route-output weights and biases are initialized to zero. Stage 1 uses the Token-favoring prior $\boldsymbol{\rho}_0 = (0.8, 0.2)$ with $\gamma^{\text{route}}_n = \max(1 - n/K_{\mathrm{w}}, 0)$ for $K_{\mathrm{w}} > 0$, where n counts Stage-1 optimizer updates and K_{w} is the prior warm-up duration. Setting $K_{\mathrm{w}} = 0$ disables the prior. The prior is disabled in Stages 2 and 3 and evaluation. Temperature τ_{R} scales logits before normalization.

Straight-through routing. Let $\mathbf{q}^m_{t+1} = \mathrm{OneHot}(z^m_{t+1})$. Replay reconstructs the routing weights using

$$\bar{\mathbf{z}}^m_{t+1} = \mathbf{q}^m_{t+1} + \boldsymbol{\pi}^m_{\mathrm{R},t+1} - \mathrm{sg}\left(\boldsymbol{\pi}^m_{\mathrm{R},t+1}\right), \tag{18}$$

where sg stops gradients. The routed input is

$$\mathbf{e}^{\mathrm{R}}_{t+1} = \operatorname*{Concat}_{m=1}^{M}\left(\bar{z}^m_{t+1,\mathrm{T}}\mathbf{e}^{\mathrm{T},m}_{t+1} + \bar{z}^m_{t+1,\mathrm{H}}\mathbf{e}^{\mathrm{H},m}_{t+1}\right). \tag{19}$$

The forward value of $\bar{\mathbf{z}}_{t+1}^{m}$ equals the stored one-hot choice, while its backward derivative follows the routing probabilities. Subsequent token losses therefore update the route-selection parameters through the routed input. The advantage-weighted token objective provides outcome-based supervision, while ST specifies how its gradient reaches the route probabilities. This is a biased surrogate gradient for discrete routing, rather than an unbiased policy-gradient estimator for route actions.

Trajectory replay and gradient flow. Rollout generation records tokens, route choices, continuous-context vectors, decision hidden states, and uncertainty features. These rollout-derived features are treated as fixed inputs during replay. Using the current trainable parameters, replay recomputes the contextual router, fusion gates, and routing probabilities, while retaining the recorded route choices in Equation 18.

The reconstructed embeddings are passed through the backbone to compute the token objective in Equation 6. Its gradients update LoRA, the fusion parameters, and the contextual router. Router parameters receive gradients through both routing weights and contextual gate offsets. Replay differentiates through this reconstructed computation, not through the cached feature-generation process of the rollout. The advantage-weighted token objective supplies the learning signal. We use no additional route-level REINFORCE loss or routing-entropy regularizer.

Connection to latent reasoning. DASH belongs to the token-anchored hybrid latent-reasoning family represented by HRPO (Yue et al., 2025). Its latent component is the continuous context carried into the next backbone input, alongside the sampled-token embedding. In Equation 1, $\mathbf{c}_{t+1}$ combines a distribution-weighted embedding with the preceding hidden state. Consequently, the Hybrid input depends on more than the identity of the sampled token: the continuous context can influence subsequent predictions without being decoded into additional text at that position. Token-only inputs retain the sampled-token embedding and the same budget conditioning when active.

This distinction concerns the inter-step input representation, not the mere presence of hidden states within a Transformer. Pure latent recurrence can advance through continuous states without emitting an intermediate token at every step. DASH retains a visible token at every reasoning position and selectively augments its embedding. The variational variable $\boldsymbol{\xi}_x$ in the Budget Predictor predicts the reasoning budget, whereas $\mathbf{c}_{t+1}$ supplies continuous context at each reasoning step.

Computation and readable outputs. Each reasoning position emits a visible token and constructs candidate representations for the next backbone input. Both routes use a full backbone forward pass. Shorter trajectories reduce the number of autoregressive positions, while top-k embedding construction, fusion gates, routing, and budget conditioning add computation. Visible tokens provide the readable part of the reasoning trace. Continuous context and budget embeddings additionally influence the next prediction through the backbone input.

A.3 Adaptive soft budget.

The Budget Predictor outputs a normalized scalar $b_x \in (0, 1)$. We convert it into an integer budget:

$$B_x = \mathrm{Clip}\left(\lceil b_x B_{\max} \rceil, 1, B_{\max}\right), \qquad B_{\max} = 64.$$

The predictor is initialized with $b_x = 0.99$. Its initial integer budget is therefore 64. The budget embedding is initialized to zero. Its dimension matches the backbone hidden size.

The predicted budget conditions every reasoning step. It does not terminate or truncate generation. The model may exceed the budget and stop naturally. The implementation reuses the planner container from the original codebase. Only the Budget Predictor and budget embedding remain active. All visual-memory branches are disabled. DASH uses no rendered CoT, optical character recognition (OCR), or visual memory.

Budget targets are derived from the current policy's successful lengths and failures. They encode an outcome-conditioned preference for reasoning length. The safety margin and higher failure weight favor larger budgets after unsuccessful rollouts. This preference guides generation through the budget embedding and reward, while the global completion limit provides the stopping safeguard. Dataset-level budget differences describe how the trained system allocates its preferred reasoning lengths across benchmarks.

A.4 Training Objectives and Optimization

Stage-wise objectives. Stage 1 optimizes final-answer correctness:

$$R_i^{(1)} = C_i.$$

Correctness requires an answer matching the reference and exactly one valid answer delimiter.

Stage 2 adds a length bonus to correct trajectories:

$$R_i^{(2)} = C_i \left[1 + 0.1 \exp\left(-\frac{L_i}{64}\right)\right].$$

Incorrect trajectories receive no length bonus.

Stage 3 introduces the soft-budget reward. Its budget-compliance score is

$$S_i^{\mathrm{B}} = \begin{cases} 1, & L_i \le B_x, \\ \exp\left(-\dfrac{L_i - B_x}{16}\right), & L_i > B_x. \end{cases}$$

The resulting reward is

$$R_i^{(3)} = C_i \left[1 + 0.1\, \gamma_n^{\mathrm{B}} S_i^{\mathrm{B}}\right], \qquad \gamma_n^{\mathrm{B}} = \min\left(\frac{n}{250}, 1\right).$$

Here n counts updates within Stage 3, starting from zero at its entry. Every correct trajectory retains its correctness reward. Exceeding the budget only reduces the efficiency bonus. It does not invalidate a correct response.

The Budget Predictor uses outcome-derived targets:

$$B_i^* = \begin{cases} \mathrm{Clip}(L_i + 8, 1, 64), & C_i = 1, \\ 64, & C_i = 0. \end{cases}$$

SmoothL1 regression uses targets normalized by $B_{\max}$, with weight 4 for failures. For scalar prediction u and target v, it is defined as

$$\mathrm{SmoothL1}(u, v) = \begin{cases} \frac{1}{2}(u - v)^2, & |u - v| < 1, \\ |u - v| - \frac{1}{2}, & |u - v| \ge 1. \end{cases}$$

This uses the default transition threshold of 1. Regression acts on continuous b_x, before rounding and clipping. The integer B_x controls the budget embedding and reward. The variational KL and overall budget-loss weights are 0.01 and 1, respectively.

Stage-wise optimization. Table 5 reports the epoch allocation and learning rate for each parameter group. "Gates" denotes the two Hybrid fusion gates. "Keep ratio" denotes the learnable fusion keep-ratio parameters. "Budget" includes the Budget Predictor and budget embedding. Dashes indicate that the budget modules are inactive in Stages 1 and 2. Stage 3 activates these modules while using lower learning rates for LoRA, routing, and fusion than in the preceding stages.

Table 5: Epochs and learning rates used in each training stage.

Stage	Epochs	LoRA	Router	Gates	Keep ratio	Budget
Stage 1	1	5×10^{-6}	3×10^{-4}	1×10^{-4}	1×10^{-3}	–
Stage 2	1	3×10^{-6}	1×10^{-4}	1×10^{-4}	1×10^{-3}	–
Stage 3	10	2×10^{-6}	3×10^{-5}	3×10^{-5}	1×10^{-4}	1×10^{-4}

Stage 1 starts from the pretrained backbone with newly initialized adapters. Stage 2 loads the Stage 1 adapter. Stage 3 loads the Stage 2 adapter and activates the budget modules. LoRA, fusion, and routing parameters remain trainable in all three stages.

Optimization settings. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay 0.1. The learning rate follows a cosine schedule. The warmup ratio is 0.1. The maximum gradient norm is 0.1. The reference-policy KL coefficient is 0.005.

We sample $G = 4$ trajectories for each input. The training temperature is 0.5. The per-device batch size is 16. Gradient accumulation uses eight steps. The maximum prompt and completion lengths are both 1,024 tokens. Training uses Brain Floating Point 16 (BF16), gradient checkpointing, and DeepSpeed's stage-2 Zero Redundancy Optimizer (ZeRO-2). We do not use vLLM during training. All models are trained on three NVIDIA A800 GPUs with 80GB memory each. The default training seed is 42.

Training runs and checkpoint selection. For the main results and ablations in Table 3, we evaluate one final checkpoint from a training schedule with seed 42. Stages 1–3 use 1, 1, and 10 epochs, respectively. Each active stage continues from the preceding active stage's checkpoint. Stage-1 and Stage-2 checkpoints are intermediate states of the same training run. The ablation schedules retain 12 total epochs and are specified in Appendix D.7. We select the final checkpoint after the prescribed schedule, without checkpoint selection based on test performance. Each checkpoint is evaluated with inference seeds $\{0, 1, 2, 3, 4\}$. The resulting confidence intervals describe inference variability conditional on that checkpoint, with the training seed held fixed. Appendix D.10 additionally evaluates the routing variants across three training seeds.

A.5 Generation Settings and Prompt

Generation and evaluation. We use the native Qwen3 chat template during training and evaluation. Native Qwen thinking mode is disabled. Each input contains a system message and one user question. No reference CoT, route label, or budget value is included in the textual prompt. The predicted budget conditions generation through the budget embedding.

The model generates concise visible reasoning followed by exactly one final-answer line beginning with `####`. Reasoning length counts tokens before this delimiter, excluding the delimiter and answer tokens. Evaluation uses temperature 1.0 and top-p 0.9.

Prompt template. Training and evaluation share a system prompt specifying the reasoning and answer format, without task-specific examples, reference rationales, or target answers.

System Prompt Used by DASH

```
Solve the user's problem and show a concise, visible reasoning
process.  End the response with exactly one final-answer line that
starts with '#### ' and is followed immediately by the actual answer.
For example, if the computed answer is forty-two, the last line
must be '#### 42'.  Never print template placeholders, braces, or
angle-bracket fields.  Do not use boxed notation for the final answer.
Do not write #### anywhere except the final-answer line.
```

B Training and Inference Algorithms

Figure 7 summarizes reward construction and optimization across the three training stages. From left to right, DASH uses correctness rewards, correctness-conditioned length bonuses, and soft-budget rewards with an auxiliary budget loss. The T and H blocks denote Token-only and Hybrid route choices in a rollout group. Algorithms 1 and 2 detail training and inference.

B.1 Three-Stage Training

B.1.1 Shared rollout procedure.

All stages use group-relative reinforcement learning. For each input x, the policy samples G trajectories under the same decoding configuration. Each trajectory is

$$\tau_i = (\mathbf{y}_i, \mathbf{z}_i),$$

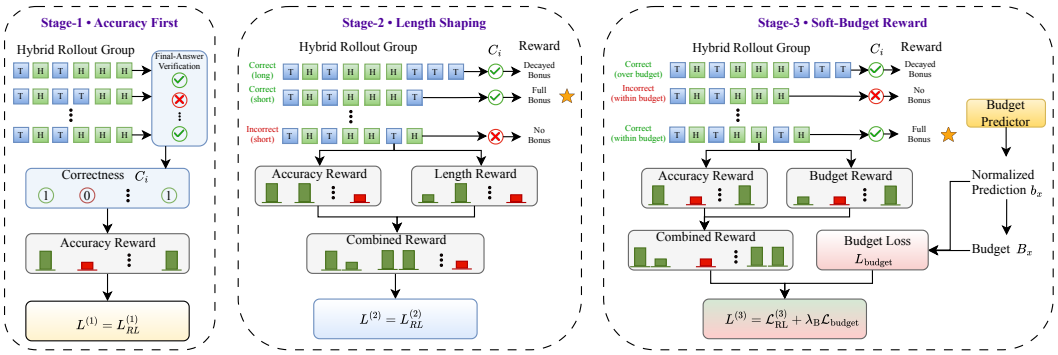


Figure 7: Three-stage outcome-based training of DASH. T and H denote Token-only and Hybrid route choices. **Left: Accuracy first.** Final-answer verification produces correctness rewards for group-relative RL. **Middle: Length shaping.** The accuracy reward and correctness-conditioned length bonus form the combined reward $R_i^{(2)}$. **Right: Soft-budget optimization.** The accuracy reward and correctness-conditioned budget bonus form $R_i^{(3)}$. Correct trajectories receive the full scheduled bonus within budget and a decayed bonus beyond it. Incorrect trajectories receive zero reward in every stage. Combined rewards drive group-relative RL. Stage 3 additionally trains the Budget Predictor through $\mathcal{L}_{\text{budget}}$ using outcome-derived targets. The soft budget guides generation without imposing a stopping point.

where $\mathbf{y}_i$ and $\mathbf{z}_i$ are the sampled token and route sequences. Rollouts share the input and, in Stage 3, budget B_x. Tokens, routes, reasoning lengths, and answers may differ.

At each reasoning step, the policy first samples a visible token $y_{i,t}$. It then constructs a Token-only candidate and a Hybrid candidate. The context-aware router selects between them according to its learned, state-dependent policy. The selected representation becomes the input to the next decoding step. Both the sampled token and selected route are stored.

Generation continues until the model produces its final answer or reaches the global completion limit. The predicted budget is never used as a stopping condition. After generation, the verifier extracts the answer $\hat{a}_i$. It checks answer correctness and output format:

$$C_i = \mathbb{I}[\text{Verify}(\hat{a}_i, a^*) = 1].$$

The reasoning length L_i counts tokens generated before the final-answer delimiter.

Each stage converts C_i and L_i into rewards, normalized within each group of G rollouts:

$$\hat{A}_i^{(s)} = \frac{R_i^{(s)} - \mu_x^{(s)}}{\sigma_x^{(s)} + \varepsilon_{\text{A}}}.$$

This compares trajectories generated for the same input. Replay uses the stored tokens, route choices, and cached contextual features to reconstruct the inputs with current parameters. The ST estimator in Appendix A.2 retains the recorded forward choices and propagates gradients through the routing probabilities. The resulting loss updates the active LoRA, fusion, and routing parameters. A frozen reference policy provides token-level KL regularization.

B.1.2 Stage 1: Accuracy-first routing.

The left panel of Figure 7 follows each rollout from Token–Hybrid generation to final-answer verification. The verifier assigns correctness C_i, which directly defines the Stage 1 reward. This stage initializes the LoRA adapters, fusion modules, and contextual router. The backbone remains frozen, and the Budget Predictor is inactive.

For each input, the policy generates G trajectories whose tokens and route decisions may differ. Some reasoning positions use Token-only representations. Others use Hybrid representations. The model receives no position-level route labels. It learns only from the final answer.

The Stage 1 reward is

$$R_i^{(1)} = C_i.$$

Correct trajectories receive a reward of 1. Incorrect or malformed trajectories receive 0. No length reward is used. No budget reward is used. The policy must first learn to produce reliable answers.

The contextual router is optimized together with the policy through the ST replay procedure in Appendix A.2. Group-relative advantages weight the token objective. Gradients from subsequent predictions update the route-selection parameters through reconstructed inputs. The same loss also trains the contextual gate offsets.

Stage 1 produces a correctness-oriented routing policy that initializes Stage 2.

B.1.3 STAGE 2: CORRECTNESS-CONDITIONED LENGTH REDUCTION.

The middle panel of Figure 7 adds a correctness-conditioned length bonus to the accuracy reward. Stage 2 loads the Stage 1 checkpoint. The Budget Predictor remains inactive. LoRA, fusion, and routing parameters remain trainable.

Stage 2 uses the same group rollout process. Each trajectory is evaluated by correctness C_i and reasoning length L_i. The reward becomes

$$R_i^{(2)} = C_i \left[1 + \eta_{\mathrm{L}} \exp\left(-\frac{L_i}{\kappa_{\mathrm{L}}} \right) \right].$$

We use $\eta_{\mathrm{L}} = 0.1$ and $\kappa_{\mathrm{L}} = 64$.

Correct trajectories retain their base reward, with larger efficiency bonuses for shorter traces. Incorrect trajectories receive no bonus at any length.

The three illustrated trajectories distinguish length from correctness. The long correct trajectory receives a smaller bonus than the short correct one. The short incorrect trajectory receives zero reward. The *Combined Reward* block merges the accuracy reward with the correctness-conditioned length bonus to obtain $R_i^{(2)}$. This reward is normalized within the group before computing $\mathcal{L}_{\mathrm{RL}}^{(2)}$. The length term shapes the policy reward. It is not a separate supervised loss.

Stage 2 bridges accuracy-first reasoning and adaptive budget learning by introducing a preference for shorter correct trajectories. It imposes neither a target length nor budget-based truncation. This transition prepares the policy for Stage 3. The final schedule comparison in Section 4.3 evaluates its contribution to answer quality under length pressure.

B.1.4 STAGE 3: ADAPTIVE SOFT-BUDGET OPTIMIZATION.

The right panel of Figure 7 combines budget-aware policy optimization with Budget Predictor training. Stage 3 loads the Stage 2 checkpoint and activates the budget modules. The rollout examples on the left illustrate the policy reward. The predictor on the right produces b_x and the corresponding budget B_x. The two learning branches meet in the joint objective at the bottom.

Policy-learning branch. For each input x, the Budget Predictor first forms an input summary $\mathbf{v}_x$. During training, it samples

$$\boldsymbol{\xi}_x \sim Q_\phi(\cdot \mid \mathbf{v}_x)$$

through reparameterization. The decoder produces a normalized prediction $b_x \in (0, 1)$. It is converted into an integer budget:

$$B_x = \mathrm{Clip}\left(\lceil b_x B_{\max} \rceil, 1, B_{\max} \right).$$

All G trajectories for input x share B_x, keeping the group's budget condition fixed.

After routing to a Token-only or Hybrid candidate, DASH adds the budget embedding:

$$\tilde{\mathbf{e}}_{t+1} = \mathbf{e}_{t+1}^{\mathrm{R}} + \mathbf{E}_B(B_x).$$

The same budget embedding conditions every reasoning step. It is not added to final-answer tokens.

The model may generate fewer or more than B_x reasoning tokens. It still terminates naturally. After generation, DASH compares the observed length L_i with B_x:

$$S_i^{\mathrm{B}} = \begin{cases} 1, & L_i \le B_x, \\ \exp\left(-\dfrac{L_i - B_x}{\kappa_{\mathrm{B}}}\right), & L_i > B_x. \end{cases}$$

The Stage 3 reward is

$$R_i^{(3)} = C_i \left[1 + \gamma_n^{\mathrm{B}} \eta_{\mathrm{B}} S_i^{\mathrm{B}}\right].$$

With maximum bonus $\eta_{\mathrm{B}} = 0.1$ and decay scale $\kappa_{\mathrm{B}} = 16$, we gradually activate the bonus:

$$\gamma_n^{\mathrm{B}} = \min\left(\frac{n}{K_{\mathrm{B}}}, 1\right), \qquad K_{\mathrm{B}} = 250.$$

The right panel illustrates three reward cases. A correct over-budget trajectory retains its base reward and receives a decayed bonus. An incorrect within-budget trajectory receives zero reward. A correct within-budget trajectory receives the full scheduled bonus, $\gamma_n^{\mathrm{B}} \eta_{\mathrm{B}}$. Unlike Stage 2, further shortening within budget does not increase this bonus. The *Combined Reward* block represents $R_i^{(3)}$, which combines correctness with the correctness-conditioned budget bonus. Group-relative advantages from this reward define $\mathcal{L}_{\mathrm{RL}}^{(3)}$.

The integer budget is treated as a fixed index in the policy branch. Policy gradients update the budget embedding. They do not pass through the ceiling operation. The Budget Predictor is trained separately by the budget-learning branch.

Budget-learning branch. The Budget Predictor and budget-loss block in the rightmost panel of Figure 7 represent the budget-learning branch. Its supervision comes from the same rollout group's correctness and observed lengths. The regression compares the continuous prediction b_x with normalized outcome-derived targets. The integer B_x is used for generation conditioning and reward computation, not as a differentiable regression output. No annotated CoT lengths are required.

Each trajectory produces a target budget:

$$B_i^* = \begin{cases} \mathrm{Clip}(L_i + \Delta, 1, B_{\max}), & C_i = 1, \\ B_{\max}, & C_i = 0. \end{cases}$$

We use $\Delta = 8$ and $B_{\max} = 64$.

A correct trajectory uses its observed length plus a safety margin. The margin preserves room for additional computation. An incorrect trajectory uses the maximum budget. Therefore, a short failed trajectory cannot encourage the Budget Predictor to reduce future budgets.

The input-level prediction b_x is compared with all targets in the rollout group. Correct trajectories receive weight 1. Incorrect trajectories receive weight $w_{\mathrm{F}} = 4$. The regression loss is

$$\mathcal{L}_{\mathrm{reg}} = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{\sum_{i=1}^{G} w_i \,\mathrm{SmoothL1}\left(b_x, B_i^*/B_{\max}\right)}{\sum_{i=1}^{G} w_i}\right].$$

The variational posterior is regularized by

$$\mathcal{L}_{\mathrm{var}} = \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(Q_\phi(\cdot \mid \mathbf{v}_x) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\right)\right].$$

The complete budget objective is

$$\mathcal{L}_{\mathrm{budget}} = \mathcal{L}_{\mathrm{reg}} + \beta_\xi \mathcal{L}_{\mathrm{var}},$$

where $\beta_\xi = 0.01$.

Joint Stage-3 update. The bottom of the right panel combines the RL loss derived from $R_i^{(3)}$ with the weighted budget loss. The combined reward is therefore used to construct the policy loss, rather than being added directly to $\mathcal{L}_{\mathrm{budget}}$:

$$\mathcal{L}^{(3)} = \mathcal{L}_{\mathrm{RL}}^{(3)} + \lambda_{\mathrm{B}} \mathcal{L}_{\mathrm{budget}},$$

where $\lambda_{\mathrm{B}} = 1$.

The policy loss updates LoRA, fusion, routing, and budget-embedding parameters. The budget loss updates the Budget Predictor. Both objectives use the same sampled trajectories. The budget policy can therefore adapt to the current generation behavior.

Across the three panels, correctness remains the prerequisite for a positive trajectory reward. Stage 1 uses correctness alone. Stage 2 distinguishes correct trajectories by length. Stage 3 evaluates their budget compliance and additionally trains the Budget Predictor. Incorrect trajectories receive no policy reward, but they still supply maximum-budget targets in Stage 3.

Algorithm 1 implements the three rewards in Figure 7. Each stage uses group-relative RL. Stage 3 also adds $\mathcal{L}_{\mathrm{budget}}$. The final checkpoint runs inference via Algorithm 2.

Algorithm 1 Three-stage outcome-based training of DASH

Require: Dataset $\mathcal{D}$, group size G, maximum budget $B_{\max}$
Require: Frozen backbone and reference policy π_{ref}
Ensure: Trained policy π_θ and Budget Predictor Q_ϕ

Initialize LoRA, fusion, and routing parameters
Freeze the backbone and reference policy
for $s = 1, 2, 3$ **do**
  if $s = 1$ **then**
    Keep Q_ϕ and $\mathbf{E}_B$ inactive
  else if $s = 2$ **then**
    Load the Stage 1 checkpoint
  else
    Load Stage 2 and activate Q_ϕ and $\mathbf{E}_B$
  end if
  for each Stage-s minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
    $\mathcal{L}_{\mathcal{B}} \leftarrow 0$
    for each $(x, a^*) \in \mathcal{B}$ **do**
      if $s = 3$ **then**
        Encode x and construct $\mathbf{v}_x$
        Sample $\boldsymbol{\xi}_x \sim Q_\phi(\cdot \mid \mathbf{v}_x)$ and predict B_x
      else
        $B_x \leftarrow \varnothing$
      end if
      for $i = 1, \ldots, G$ **do**
        Generate $\tau_i = (\mathbf{y}_i, \mathbf{z}_i)$ with routes drawn from Equation 17 with $\mathbf{E}_B(B_x)$ added to each reasoning input if $s = 3$
        Stop at completion or the global limit, independently of B_x
        Record L_i, extract $\hat{a}_i$, and compute $C_i = \mathbb{I}[\mathrm{Verify}(\hat{a}_i, a^*) = 1]$
      end for
      if $s = 1$ **then**
        $R_i^{(1)} \leftarrow C_i$, $\mathcal{L}_{\mathrm{aux}} \leftarrow 0$
      else if $s = 2$ **then**
        $R_i^{(2)} \leftarrow C_i[1 + \eta_{\mathrm{L}} \exp(-L_i/\kappa_{\mathrm{L}})]$
        $\mathcal{L}_{\mathrm{aux}} \leftarrow 0$
      else
        Compute S_i^{B} and $R_i^{(3)} \leftarrow C_i[1 + \gamma_n^{\mathrm{B}} \eta_{\mathrm{B}} S_i^{\mathrm{B}}]$
        Construct $\{B_i^*, w_i\}_{i=1}^G$ and compute $\mathcal{L}_{\mathrm{budget}}$
        $\mathcal{L}_{\mathrm{aux}} \leftarrow \lambda_{\mathrm{B}} \mathcal{L}_{\mathrm{budget}}$
      end if
      Normalize $\{R_i^{(s)}\}_{i=1}^G$ into $\{\hat{A}_i^{(s)}\}_{i=1}^G$
      Reconstruct inputs from stored routes and cached features using ST weights (Equation 18)
      Compute $\mathcal{L}_{\mathrm{RL}}^{(s)} = \mathcal{L}_{\mathrm{PG}}^{(s)} + \beta_{\mathrm{ref}} \mathcal{L}_{\mathrm{ref}}$
      $\mathcal{L}_{\mathcal{B}} \leftarrow \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathrm{RL}}^{(s)} + \mathcal{L}_{\mathrm{aux}}$
    end for
    Update all active trainable parameters using $\mathcal{L}_{\mathcal{B}}$
  end for
  Save the Stage-s checkpoint
end for
return π_θ, Q_ϕ

B.2 SOFT-BUDGET INFERENCE

Algorithms 1 and 2 form two connected parts of DASH. Algorithm 1 learns routing and budget allocation from sampled outcomes. Algorithm 2 applies the learned components to unseen inputs. No additional optimization is performed during inference.

The Budget Predictor uses its posterior mean:

$$\boldsymbol{\xi}_x = \boldsymbol{\mu}_\phi(\mathbf{v}_x).$$

It then produces b_x and B_x. Sampling from the variational posterior is disabled. The resulting budget is therefore deterministic for a fixed input and checkpoint.

At each reasoning step, the model decodes a visible token and constructs its Token-only and Hybrid candidates. Each routing head draws a representation choice from the learned, state-conditioned distribution in Equation 17, with the route prior disabled. The budget embedding is added to the selected representation. The resulting input is used at the next decoding step. Final-answer tokens use standard token embeddings.

The stopping rule remains independent of B_x. The model can terminate below the budget or continue beyond it. Generation stops only when the answer is completed, an end token is produced, or the global completion limit is reached. Algorithm 2 summarizes this process.

Algorithm 2 DASH inference with an adaptive soft budget

Require: Input x, trained policy π_θ, and Budget Predictor Q_ϕ
Require: Maximum budget $B_{\max}$
Ensure: Trajectory $\mathbf{y}$, answer $\hat{a}$, and budget B_x
Encode x and construct $\mathbf{v}_x$
$\boldsymbol{\xi}_x \leftarrow \boldsymbol{\mu}_\phi(\mathbf{v}_x)$
$b_x \leftarrow \mathrm{Sigmoid}(f_\phi(\boldsymbol{\xi}_x))$
$B_x \leftarrow \mathrm{Clip}(\lceil b_x B_{\max} \rceil, 1, B_{\max})$
Initialize the prefix with x and set $\mathbf{y} \leftarrow \varnothing$
while no end token is produced and the safety limit is not reached **do**
    Compute hidden state $\mathbf{h}_t$
    Compute next-token distribution $\mathbf{p}_{t+1}$
    Decode y_{t+1} and append it to $\mathbf{y}$
    $\mathbf{e}^{\mathrm{T}}_{t+1} \leftarrow \mathbf{E}(y_{t+1})$
    if reasoning is active **then**
        Construct $\mathbf{e}^{\mathrm{H}}_{t+1}$
        Compute $\boldsymbol{\pi}^m_{\mathrm{R},t+1}$ for each head using Equation 17
        $z^m_{t+1} \sim \mathrm{Cat}(\boldsymbol{\pi}^m_{\mathrm{R},t+1})$
        Assemble $\mathbf{e}^{\mathrm{R}}_{t+1}$ from the selected blocks as in Equation 2
        $\tilde{\mathbf{e}}_{t+1} \leftarrow \mathbf{e}^{\mathrm{R}}_{t+1} + \mathbf{E}_B(B_x)$
    else
        $\tilde{\mathbf{e}}_{t+1} \leftarrow \mathbf{e}^{\mathrm{T}}_{t+1}$
    end if
    Feed $\tilde{\mathbf{e}}_{t+1}$ into the next step
▷ B_x is not a termination condition
end while
Extract the final answer $\hat{a}$ from $\mathbf{y}$
return $\mathbf{y}, \hat{a}, B_x$

Together, Algorithms 1 and 2, with the ST replay definition in Appendix A.2, specify DASH's routing and budget workflow. Training learns representation routing and budget allocation from final-answer outcomes. Inference applies both mechanisms at every reasoning step. Neither procedure requires CoT annotations, route labels, or token-importance labels.

C DATA AND EVALUATION PROTOCOL

C.1 DATASETS AND GENERATION PROTOCOL

Datasets and splits. We use the GSM8K training split as the only training source (Cobbe et al., 2021). It contains 7,473 arithmetic word problems. During reinforcement learning, we retain only

the questions and final answers. The provided CoT rationales are not used as supervision. We evaluate in-domain performance on the 1,319-example GSM8K test split.

We evaluate out-of-distribution generalization on GSM-Hard, SVAMP, and MultiArith (Gao et al., 2023; Patel et al., 2021; Roy & Roth, 2015). GSM-Hard contains harder numerical variants of GSM8K problems. SVAMP and MultiArith use different problem structures and linguistic templates. We also evaluate transfer to competition-level mathematical reasoning on MATH500 (Lightman et al., 2024). It contains 500 problems from the MATH test set. All five benchmarks evaluate mathematical reasoning. None of these four transfer benchmarks is used for training or adaptation.

Table 6 summarizes each dataset's evaluation role, training use, and test-set size. The training-use column refers to the dataset's training split, not to the evaluation examples. Test sizes refer to full accuracy evaluation. Latency uses 100-example subsets (Appendix C.3).

Table 6: Datasets used for training and evaluation. "OOD" denotes out-of-distribution evaluation.

Dataset	Role	Used for Training	Test Size
GSM8K	In-domain	Yes	1,319
GSM-Hard	OOD	No	1,319
SVAMP	OOD	No	1,000
MultiArith	OOD	No	600
MATH500	OOD mathematics	No	500

Input and output format. All methods receive the same question text. No few-shot demonstrations are used. We apply the model-specific chat template and disable the native Qwen thinking mode. The instruction asks the model to place its final answer after exactly one #### delimiter. Text before this delimiter is treated as the reasoning trace. Text after it is treated as the final answer.

Decoding protocol. Unless otherwise stated, we evaluate one fixed trained checkpoint. Each example produces one response per inference seed. We use temperature 1.0 and top-p 0.9. The five inference seeds are $\{0, 1, 2, 3, 4\}$. Each seed controls the stochastic generation operations. The prompt and completion limits remain fixed across methods. We evaluate every example in each test set without subsampling. Appendix D.9 reports a separate five-checkpoint analysis with greedy token decoding and sampled routing. Appendix D.10 extends the sampling-based routing ablations to three training seeds.

DASH uses soft-budget inference: the predicted budget conditions generation without terminating it. Answer and end-token transitions govern stopping, subject to the global completion limit.

Answer verification. A prediction is correct only when its answer and output format are both valid. Numeric answers are normalized by removing commas, currency symbols, and other formatting differences. Equivalent integers, decimals, and fractions are accepted. MATH500 predictions use the standard MATH answer normalization. This procedure supports symbolic expressions, tuples, and equivalent numeric forms. A response with a missing or repeated final-answer delimiter is counted as incorrect. Evaluation failures are also counted as incorrect.

C.2 Evaluation Metrics and Statistical Reporting

Evaluation metrics. We report Pass@1 as the primary accuracy metric. For dataset d, seed s, and test-set size N_d, it is computed as

$$\text{Pass@1}_{d,s} = \frac{1}{N_d} \sum_{j=1}^{N_d} C_{d,j}^{(s)}, \tag{20}$$

where $C_{d,j}^{(s)} = 1$ only if both the answer and format are correct.

We use #L to measure reasoning sequence length:

$$\#L_{d,s} = \frac{1}{N_d} \sum_{j=1}^{N_d} L_{d,j}^{(s)}. \tag{21}$$

For DASH, HRPO, and explicit-reasoning methods, $L_{d,j}^{(s)}$ counts reasoning tokens before the final-answer delimiter. Prompt tokens, the delimiter, and final-answer tokens are excluded. For DASH, a missing delimiter causes the full completion to be counted as reasoning and the response to be marked incorrect. For Coconut and CoLaR, length counts recurrent latent positions. For RoT, it counts visual latent positions. SFT-w/o CoT and the evaluated fully internalized iCoT configuration have $\#L = 0$. Thus, $\#L$ measures reasoning sequence length across these representations. Wall-clock latency captures their runtime costs under the protocol in Appendix C.3.

Accuracy gain per reasoning position. Figure 4 aggregates GSM8K, GSM-Hard, SVAMP, and MultiArith with equal dataset weights. MATH500 is not included. For method m, let $\bar{A}_m$ and $\bar{L}_m$ denote the arithmetic means of the four dataset-level accuracies and reasoning lengths reported in Table 1. Accuracy is expressed in percent. Let $\bar{A}_{\mathrm{ref}}$ denote the corresponding SFT-w/o CoT accuracy. The efficiency score is

$$\mathrm{Eff}_m = \frac{\bar{A}_m - \bar{A}_{\mathrm{ref}}}{\bar{L}_m}, \qquad \bar{L}_m > 0. \tag{22}$$

Thus, the score measures percentage-point gain per reasoning position. We divide the difference in macro-average accuracy by macro-average length, rather than averaging dataset-specific ratios. For zero-length methods, the ratio is undefined. SFT-w/o CoT is assigned a vertical coordinate of zero solely as an accuracy reference, not as a measured efficiency score. Other zero-length methods, including iCoT, are omitted from this plot. The score summarizes accuracy gain over the direct-answer reference per reasoning position at the reported operating point. Pass@1, $\Delta\mathrm{Acc}/L$, and measured latency describe answer quality, sequence-length efficiency, and end-to-end runtime, respectively. Reporting all three captures both reasoning effectiveness and the execution cost of each representation.

Statistical reporting. For each checkpoint in sampling-based evaluation, we compute each metric per inference seed. We report the five-seed mean with a 95% confidence interval:

$$\bar{m} \pm t_{0.975,4} \frac{s_m}{\sqrt{5}}, \tag{23}$$

where $\bar{m}$ and s_m are the mean and standard deviation of the five seed-level results. The statistical unit is an inference seed. These intervals quantify stochastic inference variation conditional on each trained checkpoint. Appendix D.10 separately reports the mean and sample standard deviation across three training-run means.

The average column in the main results table is the unweighted mean over GSM8K, GSM-Hard, SVAMP, and MultiArith. MATH500 is reported separately to evaluate transfer to competition-level mathematics. The same accuracy-evaluation protocol is applied to all locally reproduced baselines.

Soft-budget statistics. Figure 6 pools inference seeds within each benchmark. Both means use the same trajectories with valid budget and reasoning-length records. Incorrect answers and length-limited outputs are retained. Error records and records without valid values are excluded. Within- and over-budget rates use this same population. For this descriptive analysis, reasoning lengths are estimated by re-tokenizing the saved reasoning text. The figure reports pooled means and proportions. The result tables separately report seed-level estimates with confidence intervals.

C.3 Inference-Latency Protocol

C.3.1 Hardware and software.

The latency benchmark uses Qwen3-4B as the language backbone for every method. RoT additionally uses the frozen visual encoder from Qwen3-VL-4B-Instruct. Training and latency evaluation use NVIDIA A800 GPUs with 80GB memory each. Training uses three GPUs. During latency evaluation, each method runs in a separate worker on one A800 GPU. The complete model remains resident throughout evaluation. The scheduler assigns at most one benchmark worker to each GPU and requires identical GPU models across methods. Inference uses BF16 with PyTorch and the Compute Unified Device Architecture (CUDA). The project dependency configuration specifies Transformers 4.57.3. Each method retains its native generation implementation and model-loader

attention settings. The benchmark records the GPU model, device memory, PyTorch version, and CUDA version in its output metadata. All reported latency measurements use the 4B configuration.

C.3.2 WORKLOAD AND DECODING.

We measure latency on the five evaluation benchmarks with a batch size of 1. For each benchmark, we select 100 examples without replacement. The selection seed is $20260902 + d$, where $d \in \{0, \ldots, 4\}$ follows the dataset order in the configuration. All methods process the same examples in the same order. We use greedy decoding with sampling disabled and top-p set to 1.0. The random seed is reset to the zero-based sample index before each generation call. The prompt limit is 2,048 tokens. Each method uses its own decoding configuration, native prompt format, and stopping rules. Coconut uses six latent steps, and CoLaR-2 uses a compression factor of two. RoT uses 32 latent tokens on the four arithmetic benchmarks and 64 on MATH500. The main accuracy tables use the five-seed sampling protocol, while this benchmark pairs latency with accuracy on the same timed examples. The paired accuracies are macro-averages over those examples. The runtime comparison evaluates each complete inference configuration, including its model, decoding settings, and generation implementation.

C.3.3 WARM-UP AND TIMING SCOPE.

Each worker loads its model once and performs five warm-up generations before measurement. Warm-up uses the first five selected examples from the first benchmark and is performed once per worker. For each measured example, we synchronize the GPU before starting the timer and after generation returns. Elapsed time is measured with `time.perf_counter()`. Timing includes prompt preparation, tokenization, input transfer, prefill, reasoning, and answer generation. It also includes method-specific computation inside the generation call. For DASH, this covers budget prediction, routing, and feature fusion. Model and dataset loading, warm-up, external answer scoring, result serialization, and plotting are excluded.

C.3.4 AGGREGATION AND UNCERTAINTY.

Each example is timed once per method. We report the arithmetic mean over 100 examples per benchmark. Error bars represent 95% Student-t confidence intervals: $\bar{t} \pm 1.984\, s_t/\sqrt{100}$, where $\bar{t}$ and s_t are the mean and sample standard deviation of per-example latency. The lower endpoint is clipped at zero. The statistical unit is a timed example. The macro-average gives equal weight to the five benchmark means. Latency ratios compare the complete run configurations described above.

D BASELINE REPRODUCTION DETAILS

D.1 SHARED SETUP AND TRAINING CONTROLS

Common setup. We reproduce the baselines at the three Qwen3 backbone scales used by DASH. Adaptation is method-specific. For LoRA-based configurations, the pretrained backbone weights remain frozen and LoRA adapters (Hu et al., 2022) are trained together with any active auxiliary modules. The adaptation settings are listed in Table 9. The Coconut implementation uses language-model parameter training without LoRA, as described below.

All methods use the same 7,473 GSM8K training problems. The questions and final answers are identical across methods. Baselines that require CoT supervision also receive the original rationales. HRPO and DASH learn from questions and final-answer feedback. Neither method uses the provided rationales as supervision. We verify all converted data files with Secure Hash Algorithm 256-bit (SHA-256) checksums.

GSM8K has no official validation split. We sample 512 training examples as a deterministic monitoring subset. These remain in training and are neither held out nor used for final results. We evaluate the final checkpoint after the prescribed schedule.

Training control. We retain the original training order and objectives of each baseline. Method-specific settings follow the corresponding papers and official implementations. We only modify the

model loader, data interface, and parameter-efficient adaptation when needed for Qwen3 compatibility. All reported baselines use the same total epoch budget as DASH. The batch size is set to 16. BF16 and gradient checkpointing are enabled when supported.

Table 7 compares the supervision, reasoning representations, routing policies, and length controls of the reproduced methods. "Fixed" in the routing column denotes use of the same representation throughout reasoning. The separate length-control column specifies token removal during training, latent-step counts, compression factors, and DASH's soft budgets.

Table 7: Comparison of DASH with the reproduced baselines. "Routing" indicates whether the reasoning representation is selected dynamically during inference.

Method	CoT Sup.	Representation	Routing	Length Control
SFT-w/o CoT	No	Direct answer	–	$L = 0$
SFT-CoT	Yes	Token-only	Fixed	None
HRPO	No	Token-anchored hybrid latent	Fixed	No explicit length objective
iCoT	Yes	Internalized CoT, direct answer	Fixed	Progressive token removal
Coconut	Yes	Continuous latent	Fixed	Six latent steps
CoLaR-2	Yes	Compressed latent	Fixed	Compression factor 2
CoLaR-5	Yes	Compressed latent	Fixed	Compression factor 5
RoT	Yes	Visual latent	Fixed	Fixed latent length
DASH (Ours)	**No**	**Token-only + hybrid latent**	**Dynamic**	**Adaptive soft budget**

D.2 Training configuration.

All methods use the same GSM8K training problems. The total training budget is 12 epochs, and the per-device batch size is 16. Baselines retain their method-specific objectives and training interfaces. Table 8 separates the shared experimental controls from DASH-specific implementation settings. Rows marked "Shared" apply across methods. Rows marked "DASH" specify our optimizer, rollout, and implementation settings. The final rows summarize the stage allocations. Table 10 gives the variant-level breakdown. The shared controls match training data and total epochs. Floating-point operations (FLOPs), GPU time, and trainable parameter counts vary with each method's configuration. LoRA ranks and auxiliary modules follow the method-specific settings in Table 9.

D.3 Hyperparameter configuration.

Table 9 lists method-specific training settings. For multi-stage methods, learning rates are listed in training order. DASH uses separate learning rates for its adapters and auxiliary modules, as detailed in Table 5. The ablation variants inherit the shared DASH hyperparameters. Only the specified components and stage allocations are changed. We evaluate the final checkpoint after the prescribed schedule, without checkpoint selection based on test performance.

Hyperparameter selection and monitoring. Fixed-routing variants reuse the DASH hyperparameters without variant-specific tuning. The 512-example monitoring subset is used for in-training monitoring and remains part of the training data. The checkpoint rule above specifies which trained state is evaluated. The final hyperparameter values are given in Tables 4, 5, and 9.

D.4 Comparison with DASH.

Table 7 compares the supervision, representations, and length controls of the baselines. Coconut and CoLaR use recurrent continuous representations, while RoT learns visual latent representations from rendered rationales. These methods move intermediate reasoning into latent positions, with reasoning scale controlled by latent-step counts or compression factors. HRPO and DASH instead

Table 8: Shared training protocol and DASH implementation settings.

Scope	Setting	Value
Shared	Training dataset	GSM8K
Shared	Total training epochs	12
Shared	Per-device batch size	16
DASH	Optimizer	AdamW
DASH	Adam coefficients (β_1, β_2)	$(0.9, 0.99)$
DASH	Learning-rate schedule	Cosine
DASH	Weight decay	0.1
DASH	Learning-rate warm-up ratio	0.1
DASH	Maximum gradient norm	0.1
DASH	Gradient accumulation steps	8
DASH	Rollouts per input G	4
DASH	Training temperature	0.5
DASH	Reference KL coefficient	0.005
DASH	Maximum prompt length	1,024
DASH	Maximum completion length	1,024
DASH	Maximum soft budget $B_{\max}$	64
DASH	Numerical precision	BF16
DASH	Gradient checkpointing	Enabled
DASH	Distributed training	DeepSpeed ZeRO-2
DASH	Training seed	42
Full / fixed-route	Stage-wise epochs	$(1, 1, 10)$
Fixed soft budget	Stage-wise epochs	$(1, 1, 10)$
Without Stage 2	Stage-wise epochs	$(1, 0, 11)$
Without Stage 3	Stage-wise epochs	$(1, 11, 0)$

Table 9: Method-specific optimizer and adaptation settings. LR denotes learning rate, WD denotes weight decay, and GA denotes gradient accumulation steps. LoRA settings are rank / alpha / dropout. DASH and HRPO LRs apply to LoRA. Coconut uses no adapter ("–").

Method	Optimizer	LR	WD	GA	LoRA
SFT-w/o CoT	AdamW	2e−5	0	8	16/32/0.05
SFT-CoT	AdamW	2e−5	0	8	16/32/0.05
HRPO	AdamW	5e−6	0.1	8	32/64/0
iCoT	AdamW	1e−5	0.01	8	64/128/0
Coconut	AdamW	1e−5 / 1e−5	0.01	8	–
CoLaR-2/5	AdamW	1e−4	0.01	8	128/32/0
RoT	AdamW	2e−5 / 2e−5	0.01	8	16/32/0.05
DASH	AdamW	5e−6 / 3e−6 / 2e−6	0.1	8	32/64/0

couple visible tokens with continuous features at the backbone input, placing both within hybrid latent reasoning. iCoT provides a complementary internalization approach: it progressively removes rationale tokens during training and uses direct answer generation in our evaluated configuration.

HRPO (Yue et al., 2025) is the closest hybrid latent baseline. It shares token anchoring, gated continuous-feature fusion, and outcome-based learning without reference CoT with DASH. DASH builds on this foundation with a length-aware training framework. Its curriculum introduces correctness-conditioned length shaping and joint soft-budget optimization. Within this framework, routing selects Token-only or Hybrid inputs, and predicted budgets condition generation and define the preferred length range in the reward. Generation can continue beyond this range. The representation controls and training objectives together define DASH's approach to compact token-anchored hybrid reasoning.

The comparisons therefore serve two purposes. Coconut, CoLaR, and RoT compare token-anchored hybrid reasoning with latent-sequence representations under the reported adaptation settings. HRPO tests the accuracy–length trade-off against an outcome-trained hybrid latent method without DASH's routing and budget controls. HRPO attains higher average accuracy in the reported results, whereas DASH uses substantially shorter reasoning trajectories. This comparison characterizes effort allocation within hybrid latent reasoning through the two methods' accuracy–length trade-offs.

D.5 Method-Specific Implementations

Standard SFT baselines. SFT-w/o CoT receives only the question and final answer. All intermediate steps are removed. The model is trained to answer directly. We assign $\#L = 0$ to this baseline. SFT-CoT uses the complete human-provided rationale. It represents the standard explicit-reasoning baseline. Both methods use the same optimizer and data order where applicable.

HRPO. We include Hybrid Reasoning Policy Optimization (HRPO) (Yue et al., 2025) as an outcome-based hybrid latent-reasoning baseline. HRPO uses projected hidden features obtained by mapping the next-token distribution to a continuous embedding. Learnable gates combine these features with the sampled-token embedding at each reasoning step. The gates initially favor the sampled-token contribution. Final-answer tokens use standard embeddings. Training uses final-answer rewards, group-relative advantages, and reference-policy KL regularization. No reference CoT is required. Unlike DASH, HRPO does not select between Token-only and Hybrid branches. It also has no explicit length bonus or learned soft budget.

iCoT. We follow iCoT's progressive internalization procedure (Deng et al., 2024), starting with full rationales and removing CoT tokens from the left under the original curriculum. The evaluated fully internalized model generates answers autoregressively without explicit reasoning tokens or additional latent steps, yielding $\#L = 0$ (Tables 1 and 2).

Coconut. We reproduce Coconut using its original two-part training procedure (Hao et al., 2025). We first obtain a CoT-initialized checkpoint. We then replace selected reasoning segments with recurrent continuous states. Each latent state is produced from the preceding hidden state. It is then passed back to the language model. The final answer is decoded after the latent sequence. We use the six-step latent configuration adopted by the GSM8K implementation. The supplied training implementation optimizes the language-model parameters directly and does not attach a LoRA adapter. This adaptation scope differs from the frozen-backbone LoRA setup used by DASH.

CoLaR. We reproduce CoLaR with its compressed-latent training objective (Tan et al., 2025). The model learns to predict compressed representations of consecutive CoT tokens. Training samples compression factors from the range specified by the original implementation. At inference, the compression factor controls how many source tokens are represented by each latent transition. We report two settings. CoLaR-2 uses a factor of 2. CoLaR-5 uses a factor of 5. Other model weights and decoding settings remain unchanged.

RoT. We follow the two-stage RoT procedure (Wang et al., 2026). Stage 1 learns the projection module from rendered CoT targets. Stage 2 freezes this module and trains the language-side LoRA

adapters. We use the frozen vision encoder from Qwen3-VL-4B-Instruct for all backbone scales. The encoder receives no gradient. The same encoder checkpoint is used in every RoT experiment.

D.6 Evaluation alignment.

Each baseline retains its native reasoning interface. Coconut uses recurrent continuous states. CoLaR uses compressed latent transitions. RoT uses visual latent representations. HRPO generates visible tokens with gated Hybrid inputs during reasoning. iCoT and the SFT baselines use token generation. The question content, test examples, answer verifier, sampling parameters, and inference seeds remain fixed across methods.

Following prior latent-reasoning work, #L counts reasoning positions before answer generation. For explicit methods, HRPO, and DASH, these positions emit visible tokens. HRPO and DASH additionally carry continuous features through Hybrid inputs. For Coconut and CoLaR, length counts recurrent latent positions. For RoT, it counts visual latent positions. We use #L to compare sequence lengths and measured latency to compare runtime, including representation-specific per-step costs.

For accuracy evaluation, all locally reproduced methods use temperature 1.0 and top-p 0.9. We evaluate one fixed checkpoint with five inference seeds. The answer parser and correctness verifier are shared across methods. GSM-Hard, SVAMP, MultiArith, and MATH500 are excluded from baseline training and adaptation.

D.7 Ablation Training Protocol

D.7.1 Initialization and epoch allocation.

All ablations use the same pretrained backbone, training data, and initial LoRA configuration as DASH. Fixed-routing variants follow their own Stage-1 and Stage-2 trajectories. Each active stage loads the preceding active stage's checkpoint. The fixed-soft-budget variant starts Stage 3 from the same Stage-2 checkpoint as Full, with a new Stage-3 optimizer. These intermediate checkpoints initialize successive stages within each training run. The full model uses 1, 1, and 10 epochs for Stages 1–3, respectively. We keep the total training budget at 12 epochs for every ablation. When Stage 2 is removed, its epoch is reassigned to Stage 3. When Stage 3 is removed, its ten epochs are reassigned to Stage 2. Table 10 summarizes the resulting schedules. All other training settings, including the optimizer, learning-rate schedule, batch size, rollout group size, and random seed, remain unchanged.

Table 10: Training schedules for DASH ablations, each totaling 12 epochs.

Variant	**Stage 1**	**Stage 2**	**Stage 3**	**Total**	**Controlled modification**
DASH (Full)	1	1	10	12	None
Fixed Token-only Routing	1	1	10	12	Token-only routing
Fixed Hybrid Routing	1	1	10	12	Hybrid routing
Fixed Soft Budget ($B = 64$)	1	1	10	12	Constant soft budget, no predictor loss
w/o Stage 2	1	0	11	12	No length shaping
w/o Stage 3	1	11	0	12	No soft-budget optimization

D.7.2 Fixed-routing variants.

The fixed-routing ablations compare contextual routing with a fixed representation policy. The selected route is used throughout all three training stages and during inference. The Fixed Token-only variant uses the sampled-token embedding as its reasoning representation, with budget conditioning added in Stage 3. The Fixed Hybrid variant always uses the Hybrid representation. Both variants use DASH's full training schedule and hyperparameters without separate tuning.

Each fixed-routing variant trains its own Budget Predictor and budget embedding in Stage 3. The predictor receives the prompt-state summary defined in Equation 3. Its targets are constructed from the correctness and lengths of rollouts generated by that fixed policy. LoRA and the budget embedding continue to be optimized under the corresponding Stage-3 objective. Each control therefore learns under its fixed representation policy throughout the full curriculum.

Relation to HRPO. Fixed Hybrid evaluates HRPO-style hybrid reasoning under the DASH training curriculum. It uses DASH's Hybrid candidate at every reasoning position and retains Stage-2 length shaping together with Stage-3 budget prediction, conditioning, rewards, and supervision. It therefore provides a length-controlled hybrid comparison for assessing the additional value of dynamic representation selection. Fixed Token-only provides the corresponding visible-token control. Both comparisons evaluate representation choice within the same length-aware curriculum. The separately reproduced HRPO baseline (Yue et al., 2025) retains its original outcome-based objective without DASH's length and budget optimization. Tables 1 and 2 report that baseline. Table 3 reports Fixed Hybrid.

D.7.3 FIXED SOFT-BUDGET CONTROL.

This variant replaces the input-conditioned budget with $B_x = 64$ for every question during Stage 3 training and evaluation. The Budget Predictor is inactive, and its regression and variational losses are disabled. The budget embedding remains trainable and conditions every reasoning step with $\mathbf{E}_B(64)$. LoRA, dynamic routing, and Hybrid fusion remain trainable. The variant retains the Stage 3 epoch allocation, optimizer settings, policy-gradient objective, and reference-policy regularization.

The constant budget enters both generation conditioning and the budget-compliance reward. With the same bonus schedule and decay scale as Full, the policy reward is

$$R_i^{\text{fixed}} = C_i \left[1 + 0.1\,\gamma_n^{\text{B}} \exp\!\left(-\frac{\max(L_i - 64, 0)}{16}\right)\right], \qquad \gamma_n^{\text{B}} = \min(n/250, 1). \tag{24}$$

The total objective is the corresponding group-relative RL loss with no auxiliary budget loss. Thus, this control preserves budget-aware policy optimization while replacing learned input-conditioned budgets with a shared allowance.

The value 64 is not a required trajectory length or a stopping rule. Generation may finish earlier or continue beyond 64 reasoning tokens, subject to the same global completion limit. This control differs from removing Stage 3, which removes budget conditioning and budget-aware rewards and extends Stage 2 training. We use the same five inference seeds, sampling parameters, answer verification, and reasoning-length definition as the other ablations. Table 17 reports this control alongside the additional fixed-budget settings.

D.7.4 STAGE-REMOVAL CONTROLS.

The Stage 2 ablation removes the correctness-conditioned length bonus. It proceeds directly from Stage 1 to eleven epochs of Stage 3 optimization. The Stage 3 ablation removes the Budget Predictor, budget embedding, budget-aware reward, and budget loss. Its ten unused epochs are assigned to Stage 2, resulting in eleven epochs of correctness-conditioned length shaping. The Stage 3 intervention removes the joint soft-budget optimization mechanism, including both policy learning under predicted budgets and outcome-based budget learning. Equal total epochs preserve the data-pass budget, while reallocating epochs changes the reward regime and active parameter groups. The comparison evaluates this joint policy and budget-learning transition.

Without Stage 2, budget-aware optimization begins directly from the accuracy-first checkpoint. Without Stage 3, length shaping continues instead of transitioning to equal bonuses for correct within-budget trajectories and outcome-based predictor supervision. Routing and Hybrid fusion remain active in both variants.

D.8 PAIRED ANALYSIS OF ROUTING ABLATIONS

We evaluate the routing variants on Qwen3-4B using five matched inference seeds $\{0, 1, 2, 3, 4\}$, with temperature 1.0 and top-p 0.9. Both fixed-routing controls retain DASH's three-stage training and soft-budget optimization. Table 11 reports mean Pass@1. We use two-sided paired t-tests on the five seed-level accuracies for each comparison. Pairing matches inference seeds between methods. Each configuration uses one trained checkpoint, and the tests characterize inference-sampling variability with training held fixed. These p-values concern individual dataset comparisons. Question-level analysis below and Appendix D.10 assess question and training variation.

Table 11: Mean Pass@1 (%) for the routing ablations on Qwen3-4B. Bold entries mark improvements with unadjusted $p < 0.05$ in two-sided paired t-tests. Superscripts identify the comparison: † against Fixed Token-only and ‡ against Fixed Hybrid. Average is the unweighted mean across the five benchmarks.

Method	GSM8K	GSM-Hard	SVAMP	MultiArith	MATH500	Average
Fixed Token-only	80.79	43.43	87.80	98.10	53.44	72.71
Fixed Hybrid	81.27	43.97	88.08	97.33	53.84	72.90
DASH	$\mathbf{82.06}^{\dagger}$	$\mathbf{44.76}^{\dagger\ddagger}$	87.84	$\mathbf{98.10}^{\ddagger}$	53.92	73.34

DASH achieves higher accuracy means than Fixed Hybrid on four of five benchmarks. Against Fixed Token-only, it improves four benchmarks and matches MultiArith. Relative to Fixed Hybrid, the improvements on GSM-Hard and MultiArith are 0.79 and 0.77 percentage points, with paired-test p-values of 0.01343 and 0.01488. Relative to Fixed Token-only, the improvements on GSM8K and GSM-Hard are 1.27 and 1.33 points, with p-values of 0.01799 and 0.006828. The remaining comparisons have unadjusted $p \geq 0.05$.

These gains are obtained over controls that already include DASH's length and budget optimization. The comparison therefore measures observed differences between learned and fixed representation policies under the same curriculum. The advantage varies across tasks, with Fixed Hybrid retaining a slightly higher mean on SVAMP. Overall, DASH achieves the highest average accuracy, using 0.86 more reasoning tokens than Fixed Hybrid and 1.97 more than Fixed Token-only. The observed trade-off favors improved answer quality at a small additional token cost.

D.8.1 Question-level paired analysis.

We further examine the sampling-based Qwen3-4B ablations at the question level. For each question, we average correctness and reasoning length over its five observed inference seeds. We then compute DASH's differences from each fixed route on that question. Questions are resampled within each dataset to obtain 95% confidence intervals. Each resampled question retains both methods' outputs and all five seeds together. This preserves the paired comparison without treating repeated outputs as independent questions. The macro-average assigns equal weight to the five datasets. These intervals are conditional on the trained checkpoints and observed inference seeds. They summarize question-level uncertainty rather than variation across independent training runs.

Some legacy outputs omit router-decoding metadata. For these files, the analysis assumes the evaluator's default categorical route sampling. The saved outputs do not independently record this mode.

Accuracy gains and additional length. Both macro-average accuracy intervals lie above zero (Table 12). DASH gains 0.44 percentage points over Fixed Hybrid and 0.63 points over Fixed Token-only. The corresponding length increases are 0.86 and 1.97 tokens, whose intervals also lie above zero. This paired evidence supports improved average answer quality at a small additional token cost under the evaluated configurations. It does not compare routing policies at matched reasoning lengths.

The positive accuracy intervals occur on GSM-Hard and MultiArith against Fixed Hybrid, and on GSM8K and GSM-Hard against Fixed Token-only. The other dataset-level intervals include zero at the reported precision. The question counts describe how broadly the differences occur, while the accuracy differences also reflect their magnitude. For example, on SVAMP, DASH improves over Fixed Hybrid on 48 questions and declines on 42, yet its mean accuracy is 0.24 points lower. The losses across those questions outweigh the gains. Together, the paired differences and question counts describe both the aggregate advantage and its variation across tasks.

D.9 Routing Ablations with Greedy Token Decoding

Evaluation setup. We complement the sampling-based ablation with greedy token decoding on Qwen3-4B. For each routing variant, we evaluate five saved checkpoints from a single training run. At each step, token decoding selects the highest-probability token. Dynamic routing retains categorical sampling from its learned route probabilities. The fixed-routing variants retain their

Table 12: Question-level paired analysis of routing ablations on Qwen3-4B. Differences are DASH minus the named fixed route, with 95% confidence intervals in brackets. Accuracy differences are in percentage points (pp). Length differences are reasoning tokens per answer. N is the number of questions. More, Less, and Tied count questions on which DASH is correct more often, less often, or equally often across the five seeds.

Dataset	N	ΔPass@1 (pp)	Δ#L	More	Less	Tied
DASH minus Fixed Hybrid						
GSM8K	1,319	0.79 [−0.00, 1.58]	0.54 [0.09, 1.00]	137	101	1,081
GSM-Hard	1,319	0.79 [0.03, 1.55]	2.73 [1.52, 3.96]	138	110	1,071
SVAMP	1,000	−0.24 [−0.90, 0.40]	0.30 [−0.32, 0.93]	48	42	910
MultiArith	600	0.77 [0.17, 1.43]	0.37 [0.19, 0.53]	14	7	579
MATH500	500	0.08 [−1.44, 1.60]	0.35 [−1.91, 2.58]	67	67	366
Average	–	0.44 [0.02, 0.85]	0.86 [0.32, 1.39]	–	–	–
DASH minus Fixed Token-only						
GSM8K	1,319	1.27 [0.49, 2.05]	1.68 [1.18, 2.18]	151	95	1,073
GSM-Hard	1,319	1.33 [0.49, 2.18]	3.81 [2.56, 5.06]	159	109	1,051
SVAMP	1,000	0.04 [−0.62, 0.70]	0.60 [0.05, 1.17]	44	43	913
MultiArith	600	0.00 [−0.33, 0.33]	0.90 [0.73, 1.08]	6	6	588
MATH500	500	0.48 [−1.04, 2.00]	2.87 [0.78, 5.02]	74	65	361
Average	–	0.63 [0.21, 1.02]	1.97 [1.45, 2.50]	–	–	–

respective Token-only or Hybrid route. Thus, token selection is greedy, while DASH still has route-level randomness.

Table 13 reports mean Pass@1 and reasoning length over the five checkpoints. Entries report the mean $\pm$ standard deviation (SD). The checkpoints share a training history. These statistics summarize the selected checkpoint evaluations, including route-sampling variation for DASH.

Table 13: Routing ablations on Qwen3-4B with greedy token decoding. Entries show mean $\pm$ standard deviation across five checkpoints from one training run per method. DASH retains stochastic routing. Average gives equal weight to the five benchmarks.

	DASH		**Fixed Hybrid**		**Fixed Token-only**	
Dataset	Pass@1 (%)	#L	Pass@1 (%)	#L	Pass@1 (%)	#L
GSM8K	81.90 ± 0.61	38.52 ± 0.48	81.11 ± 0.52	38.08 ± 0.36	81.36 ± 0.25	36.63 ± 0.71
GSM-Hard	44.28 ± 0.35	69.31 ± 0.47	43.67 ± 0.16	67.81 ± 0.51	42.77 ± 0.28	67.08 ± 0.85
SVAMP	88.10 ± 0.19	18.46 ± 0.16	88.58 ± 0.08	17.82 ± 0.17	87.58 ± 0.19	17.73 ± 0.21
MultiArith	98.23 ± 0.15	19.01 ± 0.18	97.33 ± 0.35	18.66 ± 0.25	98.17 ± 0.00	18.34 ± 0.26
MATH500	54.76 ± 0.43	104.85 ± 1.04	54.48 ± 0.58	103.99 ± 1.93	56.28 ± 0.87	103.67 ± 2.40
Average	73.45 ± 0.20	50.03 ± 0.16	73.03 ± 0.21	49.27 ± 0.58	73.23 ± 0.11	48.69 ± 0.84

Accuracy–length comparison. DASH attains the highest macro-average Pass@1 at 73.45%. It exceeds Fixed Hybrid by 0.42 percentage points and Fixed Token-only by 0.22 points. Average reasoning length is 50.03 tokens, compared with 49.27 and 48.69 tokens for the two fixed routes. The additional lengths are 0.76 and 1.34 tokens, respectively. DASH also achieves the highest macro-average accuracy in the sampling-based ablation. Both token-decoding settings therefore show higher mean answer quality with dynamic routing at a small additional token cost.

The dataset-level results show complementary strengths. DASH has higher mean accuracy than each fixed route on four of the five benchmarks. Its largest gain over Fixed Token-only is 1.50 percentage points on GSM-Hard. Fixed Hybrid performs best on SVAMP, while Fixed Token-only performs best on MATH500. The macro-average advantage reflects DASH's aggregate performance across these different tasks. Greedy token selection still leaves representation choice active. The learned

router conditions this choice on the evolving reasoning state, extending the comparison beyond sampled token generation.

D.10 Routing Ablations across Training Seeds

Evaluation and aggregation. We evaluate each Qwen3-4B routing variant with three different random training seeds. For each trained checkpoint, we average results over five inference seeds under the sampling-based evaluation protocol. Cross-training statistics use these run-level means as observations. Both fixed-routing controls retain the three-stage curriculum and soft-budget optimization. Table 14 lists the macro-average results of each run, with equal weights for the five benchmarks.

Table 14: Macro-average results for three training runs on Qwen3-4B. Each entry averages five inference seeds and five benchmarks. Each row corresponds to a different training seed. DASH has the highest mean Pass@1 in all three runs.

	DASH		**Fixed Hybrid**		**Fixed Token-only**	
Training run	Pass@1 (%)	#L	Pass@1 (%)	#L	Pass@1 (%)	#L
Run 1	73.34	50.19	72.90	49.33	72.71	48.22
Run 2	73.24	50.14	72.75	49.73	72.76	48.40
Run 3	73.09	49.76	72.72	48.63	72.72	48.04

Table 15 summarizes the three training-run means. We report their mean and sample standard deviation (SD), computed from the reported run-level means. The statistical unit is a training run after averaging inference seeds. The within-run inference confidence intervals are not averaged or reused as across-training uncertainty. For the Average row, we first obtain each run's macro-average and then compute its mean and SD across the three runs.

Table 15: Routing ablations across three training seeds on Qwen3-4B. Entries are mean $\pm$ sample SD across training-run means. Each run-level result averages five inference seeds. Average gives equal weight to the five benchmarks.

	DASH		**Fixed Hybrid**		**Fixed Token-only**	
Dataset	Pass@1 (%)	#L	Pass@1 (%)	#L	Pass@1 (%)	#L
GSM8K	81.55 ± 0.44	38.14 ± 0.15	81.35 ± 0.09	37.66 ± 0.42	81.06 ± 0.35	36.45 ± 0.47
GSM-Hard	44.50 ± 0.22	70.82 ± 0.34	44.17 ± 0.18	68.29 ± 0.21	43.39 ± 0.14	67.42 ± 0.28
SVAMP	87.86 ± 0.15	18.73 ± 0.18	88.02 ± 0.10	18.05 ± 0.37	87.77 ± 0.08	17.94 ± 0.04
MultiArith	98.10 ± 0.07	19.00 ± 0.12	97.38 ± 0.14	18.47 ± 0.15	98.01 ± 0.09	18.02 ± 0.07
MATH500	54.09 ± 0.30	103.47 ± 0.94	53.03 ± 0.72	103.69 ± 1.73	53.43 ± 0.06	101.27 ± 0.62
Average	73.22 ± 0.13	50.03 ± 0.24	72.79 ± 0.10	49.23 ± 0.56	72.73 ± 0.03	48.22 ± 0.18

Consistency of the routing gains. DASH achieves higher macro-average accuracy than both fixed routes in all three training runs. Its gains over Fixed Hybrid range from 0.37 to 0.49 percentage points. The gains over Fixed Token-only range from 0.37 to 0.63 points. Across runs, DASH reaches 73.22% accuracy with 50.03 reasoning tokens. This improves accuracy by 0.43 points over Fixed Hybrid and 0.49 points over Fixed Token-only, with 0.80 and 1.81 additional tokens. The positive mean gains therefore recur across training seeds rather than appearing in a single trained checkpoint. They support dynamic representation selection as a repeatable source of incremental answer-quality improvement within DASH's length- and budget-aware curriculum.

Dataset-level averages further show where this gain arises. On MATH500, DASH reaches 54.09% accuracy compared with 53.03% for Fixed Hybrid, with slightly shorter mean reasoning (103.47 versus 103.69 tokens). Fixed Hybrid retains the higher average accuracy on SVAMP. These differences show task-dependent benefits rather than uniform improvements on every benchmark. Overall, the three-run results support higher macro-average accuracy at a small additional token cost. The comparison remains between the evaluated operating points rather than matched-length policies.

E JOINT SOFT-BUDGET OPTIMIZATION AND ADAPTIVE BUDGET TRADE-OFFS

E.1 EVALUATION SETUP

We examine soft-budget optimization on Qwen3-4B through stage removal, an adaptive-budget sweep, and fixed-budget controls. The adaptive sweep uses maximum soft-budget values of 32, 48, 64, 96, 128, and 256. This parameter bounds the predicted budget and the outcome-derived budget targets in Stage 3. Budgets vary by question, and generation can continue beyond them. For each setting, we evaluate one trained checkpoint with inference seeds $\{0, 1, 2, 3, 4\}$, temperature 1.0, and top-p 0.9. Table 16 reports the adaptive sweep. Table 17 reports fixed soft budgets of 32, 48, 64, and 96. Both tables report Pass@1 and reasoning length across the five benchmarks. Average gives each benchmark equal weight. The 95% confidence intervals summarize stochastic inference variation conditional on each trained checkpoint, following Appendix C.

Table 16: Results for six maximum soft-budget settings. Entries are means with 95% confidence intervals over five inference seeds. The shaded rows mark the default $B_{\max} = 64$, not the highest accuracy. Reasoning may continue beyond the predicted budget.

$B_{\max}$	GSM8K	GSM-Hard	SVAMP	MultiArith	MATH500	Average
(a) Pass@1 (%, higher is better)						
32	72.72 ± 0.58	35.92 ± 0.72	83.46 ± 0.29	96.27 ± 0.43	42.92 ± 1.12	66.26 ± 0.14
48	78.57 ± 0.94	41.46 ± 0.54	86.52 ± 0.59	98.33 ± 0.21	50.96 ± 1.43	71.17 ± 0.46
64	82.06 ± 0.90	44.76 ± 0.57	87.84 ± 0.45	98.10 ± 0.50	53.92 ± 1.78	73.34 ± 0.57
96	83.99 ± 0.99	46.26 ± 0.26	88.12 ± 0.77	98.20 ± 0.45	56.08 ± 0.99	74.53 ± 0.37
128	85.67 ± 0.31	47.07 ± 0.51	87.68 ± 0.60	98.40 ± 0.28	55.68 ± 1.90	74.90 ± 0.49
256	87.67 ± 0.51	49.55 ± 0.61	88.58 ± 0.30	98.67 ± 0.15	52.40 ± 1.02	75.37 ± 0.37
(b) Reasoning length (#L, lower is better)						
32	25.87 ± 0.51	45.58 ± 0.75	12.22 ± 0.24	12.45 ± 0.04	74.29 ± 1.98	34.08 ± 0.36
48	34.82 ± 0.69	60.33 ± 1.47	15.97 ± 0.44	17.89 ± 0.09	93.31 ± 2.57	44.47 ± 0.77
64	38.25 ± 0.37	71.21 ± 1.01	18.54 ± 0.36	18.89 ± 0.07	104.05 ± 2.93	50.19 ± 0.68
96	42.27 ± 0.45	77.77 ± 1.64	19.80 ± 0.74	19.41 ± 0.08	116.73 ± 2.22	55.20 ± 0.64
128	46.64 ± 0.47	86.87 ± 0.96	20.25 ± 0.88	19.79 ± 0.11	134.56 ± 1.44	61.62 ± 0.28
256	54.06 ± 0.37	97.93 ± 0.66	22.13 ± 0.58	20.63 ± 0.07	151.98 ± 2.39	69.35 ± 0.53

E.2 EFFECTIVENESS OF JOINT SOFT-BUDGET OPTIMIZATION

The Stage 3 ablation in Table 3 replaces joint soft-budget optimization with additional Stage 2 training. Both schedules use 12 epochs. The full schedule reaches 73.34 Pass@1 with 50.19 reasoning tokens, compared with 58.07 and 21.42 without Stage 3. Accuracy improves on all five benchmarks, including gains of 24.80 points on GSM8K, 19.91 on GSM-Hard, and 18.52 on MATH500. These results support the transition from continued length shaping to joint soft-budget optimization.

The transition changes how correct trajectories are rewarded. Stage 2 assigns larger bonuses to shorter correct traces. Stage 3 assigns the same bonus to all correct traces within budget. Thus, two correct solutions within budget are no longer ranked by length through the efficiency bonus. The policy can preserve useful intermediate steps without losing this bonus. Beyond budget, the bonus decays but the base correctness reward remains. Incorrect trajectories receive zero reward, even when short. This structure makes compression conditional on success and stops rewarding every further reduction within the preferred range.

Budget embeddings communicate that range to the policy. Successful rollout lengths provide prediction targets with a safety margin. Failed rollouts instead supply maximum-budget targets. This prevents short failures from supplying smaller target budgets. The longer, more accurate traces under the full schedule are consistent with this shift away from continuous compression pressure. The comparison evaluates the combined reward, conditioning, and budget-learning mechanism. Both

adaptive and fixed soft budgets below use this budget-aware reward structure. The stage ablation tests the joint training transition. The fixed-budget comparisons examine how input-conditioned versus shared allowances change the accuracy–length trade-off.

E.3 Sensitivity to the Maximum Soft Budget

The choice of $B_{\max}$ sets the range of preferred reasoning lengths that the Budget Predictor can express. It also bounds the targets derived from successful lengths and failed rollouts. Under Equation 9, correct trajectories receive the full efficiency bonus within their predicted budget. Beyond that budget, the bonus decays while the correctness reward remains intact. A smaller ceiling therefore limits the length range eligible for the full bonus. A larger ceiling allows the predictor to assign more generous budgets. Choosing $B_{\max}$ balances this preference for compact reasoning against the room available for longer successful trajectories.

The smallest settings favor compact outputs but sacrifice answer quality. At $B_{\max} = 32$, average reasoning length is 34.08 tokens and Pass@1 is 66.26. Increasing the ceiling to 64 recovers 7.08 accuracy points. The recovery is especially pronounced on GSM8K, GSM-Hard, and MATH500, where the gains are 9.34, 8.84, and 11.00 points. Even relative to 48, the setting of 64 improves these three benchmarks by 3.49, 3.30, and 2.96 points, respectively. These results favor retaining more reasoning capacity than the smallest budget ranges provide. The additional capacity is most useful on tasks where strong compression is accompanied by larger accuracy losses.

Above 64, further increases in average accuracy come with longer traces. The settings of 96, 128, and 256 gain 1.19, 1.56, and 2.03 points over 64, while increasing average reasoning length by 10.0%, 22.8%, and 38.2%. For comparison, moving from 48 to 64 gains 2.17 points with 5.72 additional tokens. Moving from 64 to 256 adds 19.16 tokens for a slightly smaller accuracy gain. The extra length therefore yields a less favorable aggregate return than the earlier increase to 64. On SVAMP and MultiArith, the mean accuracies at 64 are already within 0.74 and 0.57 points of the highest values in the sweep. The value of the additional reasoning varies across benchmarks.

A ceiling of 64 still allows extended reasoning on difficult inputs. Average lengths reach 71.21 on GSM-Hard and 104.05 on MATH500, while remaining below 19 tokens on SVAMP and MultiArith. This variation is compatible with a shared ceiling because the budget guides the preferred reasoning scale rather than enforcing a stopping point. Consequently, the ceiling need not match the length of a long solution. On MATH500, increasing it to 96 improves Pass@1 from 53.92 to 56.08. At 256, however, Pass@1 falls to 52.40 despite an average length of 151.98 tokens. MATH500 therefore favors an intermediate ceiling in this sweep, with its highest mean accuracy at 96.

At the default $B_{\max} = 64$, the results favor compact reasoning while recovering accuracy lost under stronger compression. It provides a wider range for budget learning than 32 or 48, yet keeps average reasoning length near 50 tokens across the five benchmarks. Relative to 256, it uses 27.6% fewer reasoning tokens at a cost of 2.03 average accuracy points. Soft continuation allows longer trajectories while keeping the preferred budget range compact. These results characterize the default setting's accuracy–length trade-off. Larger ceilings remain useful when additional average accuracy is worth the longer traces.

E.4 Adaptive versus Fixed Soft Budgets

Both adaptive and fixed soft budgets are viable configurations within our budget-aware reinforcement learning framework. Both retain budget conditioning and the Stage 3 reward, and neither imposes a hard stopping point. They differ in how the full-bonus range is assigned. A fixed budget uses the same range for every input. The adaptive predictor assigns an input-conditioned range and learns from outcome-derived targets.

We use adaptive soft budgets as the default to prioritize compact reasoning with input-conditioned length preferences. Compared with fixed $B = 64$, the default adaptive setting reduces average reasoning length from 57.75 to 50.19 tokens, a 13.1% reduction, with a 1.40-percentage-point decrease in Pass@1. Smaller predicted budgets cause the bonus to decay earlier while preserving the base reward for correct longer solutions. Fixed $B = 64$ provides a wider full-bonus range for inputs assigned smaller adaptive budgets. It achieves higher accuracy with longer traces. Our default

therefore favors shorter reasoning, while fixed $B = 64$ offers an alternative when accuracy takes priority.

Comparison setup. Adaptive budgets satisfy $B_x \leq B_{\max}$ and vary by input. A fixed soft budget assigns $B_x = B$ to every question. The fixed controls retain budget conditioning, dynamic routing, Hybrid fusion, and budget-aware policy optimization. The predictor and its auxiliary losses are inactive. Appendix D.7.3 gives the fixed-budget mechanism.

Table 17 gives all five-benchmark results for the four fixed budgets. These complement the adaptive sweep in Table 16. A shared numerical value denotes a prediction ceiling for adaptive budgets and a uniform allowance for fixed budgets. We therefore compare observed accuracy and reasoning length, rather than treating equal budget values as equal computational cost.

Table 17: Fixed soft-budget results on Qwen3-4B. Entries are means with 95% confidence intervals over five inference seeds for each trained checkpoint. Average gives equal weight to the five benchmarks. The budget conditions generation and rewards but does not truncate reasoning.

B	GSM8K	GSM-Hard	SVAMP	MultiArith	MATH500	Average
			(a) Pass@1 (%, higher is better)			
32	72.98 ± 0.72	37.98 ± 0.80	85.64 ± 0.57	97.10 ± 0.40	45.24 ± 1.40	67.79 ± 0.59
48	81.18 ± 0.65	43.18 ± 0.55	86.56 ± 0.55	98.40 ± 0.28	52.08 ± 1.92	72.28 ± 0.36
64	84.81 ± 0.79	47.22 ± 1.03	87.76 ± 0.71	98.57 ± 0.19	55.36 ± 2.04	74.74 ± 0.65
96	89.22 ± 0.72	52.04 ± 0.57	91.28 ± 0.67	98.97 ± 0.17	53.96 ± 0.62	77.09 ± 0.15
			(b) Reasoning length (#L, lower is better)			
32	29.05 ± 0.18	50.08 ± 1.69	13.79 ± 0.28	14.77 ± 0.10	83.51 ± 3.64	38.24 ± 0.89
48	34.33 ± 0.46	65.27 ± 1.43	17.22 ± 0.55	15.65 ± 0.10	96.98 ± 1.23	45.89 ± 0.35
64	44.56 ± 0.63	80.16 ± 1.04	19.90 ± 0.56	20.19 ± 0.09	123.94 ± 2.12	57.75 ± 0.59
96	62.56 ± 0.56	111.02 ± 0.92	30.72 ± 0.29	25.17 ± 0.11	151.56 ± 1.08	76.21 ± 0.33

How budget assignment changes length pressure. Fixed and adaptive budgets are both soft. Fixed $B = 64$ does not require 64 reasoning tokens. The question still conditions generation, so different inputs can produce different lengths under a fixed budget. Adaptation adds an input-dependent reward threshold and budget embedding to this existing variation.

Consider the same correct trace under fixed $B = 64$ and a predicted budget $B_x < 64$. Both receive the full bonus when $L \leq B_x$. For $B_x < L \leq 64$, the fixed setting still gives the full bonus, while the adaptive bonus decays. Above 64, both bonuses decay and both retain the base correctness reward. The fixed setting therefore leaves a wider range of correct traces equally rewarded. Adaptive guidance introduces a preference for shorter traces earlier on inputs assigned smaller budgets. This is a difference in training incentives, not a hard length constraint.

Budget targets connect this threshold to the policy's own outcomes. Short successful traces can supply smaller targets, while long successes and failures push targets toward the ceiling. The resulting prediction expresses a learned length preference. It is not an oracle for the minimum length needed to solve a question. The safety margin and continued base reward leave room for reasoning beyond that preference.

Aggregate accuracy–length trade-offs. Figure 3 summarizes the macro averages. At budget values of 32, 48, 64, and 96, the fixed controls achieve higher macro-average accuracy and generate longer traces than their adaptive counterparts. Adaptive guidance reduces reasoning length by 10.9%, 3.1%, 13.1%, and 27.6%, respectively. The corresponding accuracy costs are 1.53, 1.11, 1.40, and 2.56 percentage points. This pattern is consistent with the wider full-bonus range under fixed budgets. Longer correct solutions remain equally rewarded over that range, relaxing the preference for compression. Adaptive guidance moves the aggregate results toward shorter traces, with some loss of answer quality.

The cost of that accuracy gain depends on the setting. At 48, the fixed control gains 1.11 accuracy points with just 1.42 additional tokens on average. On GSM8K, it is also more accurate (81.18

versus 78.57) and slightly shorter (34.33 versus 34.82). Thus, a lower adaptive reward threshold does not guarantee shorter realized traces on every task. Both variants learn their generation policies under their respective budget conditions. At 96, the fixed control gains 2.56 macro-average accuracy points but uses 21.01 additional tokens. The stronger compression under adaptive guidance yields a larger length saving at this setting.

Cross-setting comparisons place these results on a common accuracy–length scale. Adaptive $B_{\max} = 96$ reaches 74.53 Pass@1 with 55.20 tokens. Fixed $B = 64$ reaches 74.74 with 57.75 tokens. The adaptive setting uses 4.4% fewer tokens with a 0.21-point lower accuracy mean. Adaptive $B_{\max} = 128$ reaches 74.90 with 61.62 tokens, placing fixed 64 between these two adaptive settings in both metrics. These results describe the measured operating points. They do not establish a uniformly better accuracy–length frontier for adaptive budgets.

Task-dependent effects on MATH500. The aggregate accuracy gain at larger fixed budgets is driven by the arithmetic benchmarks. Moving from fixed 64 to 96 improves GSM8K by 4.41 points and GSM-Hard by 4.82 points. MATH500 follows a different pattern. Fixed 64 achieves 55.36 Pass@1 with 123.94 tokens. Increasing the fixed budget to 96 lengthens reasoning to 151.56 tokens, while accuracy falls to 53.96. The wider reward range permits longer successful traces, but it does not check the correctness of intermediate steps. Final-answer rewards cannot make additional steps useful by themselves. The MATH500 result shows why a larger allowance and longer output need not translate into higher accuracy.

Adaptive $B_{\max} = 96$ achieves 56.08 Pass@1 with 116.73 tokens. Compared with fixed 64, this is a 0.72-point higher accuracy mean with 5.8% fewer tokens. Compared with fixed 96, accuracy is 2.12 points higher and length is 23.0% lower. Here, input-conditioned guidance accompanies better accuracy and shorter traces than either fixed setting. Together with the fixed-48 advantage on GSM8K, this shows that the benefit of budget adaptation is task-dependent.

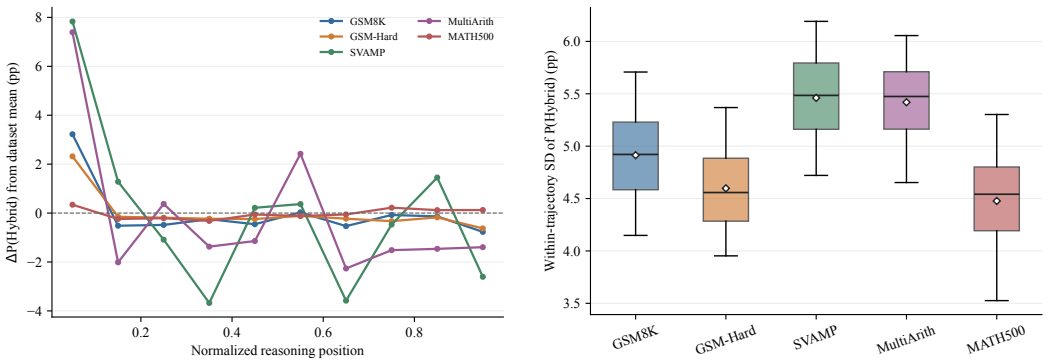
Efficiency of the shared optimization framework. Fixed 96 also illustrates the value of the budget-aware framework with a uniform allowance. On the four arithmetic benchmarks, it averages 82.88 Pass@1 with 57.37 reasoning tokens. The reproduced HRPO baseline averages 83.09 with 178.82 tokens (Table 1). The fixed-budget variant thus uses 67.9% fewer reasoning tokens with a 0.21-point lower accuracy mean. Even a uniform budget can therefore retain much of the arithmetic accuracy at a substantially lower token cost under the DASH curriculum. This result belongs to the complete fixed-budget variant, which retains dynamic routing, Stage 2 length shaping, and Stage 3 budget-aware RL. It supports the practical value of the shared optimization framework, while the adaptive predictor provides an additional control over input-specific length preferences.

F Routing Behavior Analysis

We analyze DASH's routing behavior on Qwen3-4B from two perspectives in Figure 8. Panel (a) asks whether different reasoning positions show a shared routing preference across questions. Panel (b) asks whether the routing probability changes within each individual trajectory. Both panels use Hybrid probabilities recorded before branch selection. We include reasoning transitions whose inputs are consumed by the next decoding step and exclude final-answer generation. Across the five benchmarks, mean Hybrid probabilities are approximately 0.59–0.62. The analysis below examines changes around this moderate overall preference for Hybrid inputs.

F.1 Routing Preferences across Reasoning Positions

Panel (a) compares the beginning, middle, and end of reasoning across trajectories of different lengths. We divide each trajectory into ten normalized-position bins. For each bin, we average probabilities over positions, then over observed heads and inference seeds per question. Each contributing question receives equal weight. The vertical axis shows the difference from the question-weighted dataset mean in percentage points. Positive values indicate a stronger Hybrid preference than the dataset average. Negative values indicate a Hybrid probability below that average. Zero corresponds to the dataset mean.



(a) Routing preferences across positions

(b) Probability variation within trajectories

Figure 8: Routing behavior of DASH on Qwen3-4B across five mathematical benchmarks. **(a)** Mean Hybrid probability at each normalized reasoning position, centered on the corresponding dataset mean. **(b)** Within-trajectory standard deviation of Hybrid probabilities, averaged over observed seeds and heads per question. Boxes span the 25th–75th percentiles. Black lines mark medians, white diamonds mark means, and whiskers mark the 5th–95th percentiles. Outliers are not displayed. Both panels use percentage-point units.

On SVAMP and MultiArith, Hybrid probabilities are highest near the beginning, approximately 7–8 percentage points above their dataset means. They subsequently decrease and show local increases later in reasoning. GSM8K and GSM-Hard exhibit smaller early increases, followed by values closer to their dataset means. MATH500 has a nearly flat average curve. Thus, a stronger early Hybrid preference is shared across the arithmetic benchmarks, but its strength and subsequent pattern vary by dataset. The curves capture dataset-specific changes in Hybrid preference as reasoning progresses.

F.2 Probability Changes within Individual Trajectories

Panel (b) distinguishes changes in routing preference from randomness in the selected branches. Even a constant Hybrid probability can produce different branch choices under sampling. We therefore measure how much the probability itself changes within each trajectory. For each question, seed, and routing head, we compute its population standard deviation over reasoning positions. We exclude trajectories with fewer than two recorded positions and average the resulting values over observed heads and seeds per question. The boxplots give each question equal weight.

Median within-trajectory standard deviations are approximately 4.5–5.5 percentage points across the five benchmarks. These values quantify local changes in Hybrid probability during reasoning. They describe the variation in the learned distribution before each branch is sampled.

MATH500 illustrates the distinction between the two views. Its average position curve is nearly flat, yet individual trajectories exhibit probability variation. Changes at different positions across questions can cancel in the aggregate. A flat average curve can therefore coexist with local probability adjustments. Together, the panels characterize how the router varies its preferences during reasoning.

F.3 Interpretation and Relation to Fixed-Routing Ablations

Routing and soft-budget optimization serve different roles in DASH. The router selects the input representation for the next reasoning step. Both branches pass through all Transformer layers. Budget conditioning and length-aware training guide the number of reasoning steps. DASH therefore adapts representation choice and trajectory length through complementary mechanisms. Hybrid proportions describe branch allocation, while reasoning lengths describe the number of decoding steps.

The observed probabilities indicate moderate local adjustments around a shared Hybrid preference. The two panels resolve this behavior at different levels. Position profiles show where Hybrid preference rises or falls across questions. Within-trajectory dispersion captures local adjustments within individual solutions. On MATH500, these local changes coexist with a nearly flat aggregate position curve. State-dependent routing therefore takes the form of probability modulation around a shared preference.

Figure 8 describes local changes in routing probabilities. Table 3 separately evaluates answer quality against fixed Token-only and fixed Hybrid routing under the DASH curriculum. Full improves mean Pass@1 by 0.63 and 0.44 percentage points, respectively, with slightly longer trajectories. The macro-average advantage recurs across three training seeds (Appendix D.10). The plots characterize probability modulation. The ablations measure the overall learned-routing configuration against always selecting either branch.

G Qualitative Analysis and Failure Cases

We examine ten representative outputs produced by DASH. The five successful cases cover all five evaluation datasets. They include both short arithmetic problems and longer compositional problems. The five failure cases capture distinct error types. These examples are used only for qualitative analysis. Controlled experiments evaluate the quantitative effects of routing and budget allocation.

G.1 Successful Cases

C1: Compact reasoning on SVAMP

Question. Rachel had to complete 2 pages of reading homework and 4 pages of math homework. How many more pages of math homework than reading homework did she have?

Reasoning.

$$4 - 2 = 2.$$

Prediction. 2
Gold answer. 2
Budget and length. $B_x = 15,\ \#L = 8.$
Observation. DASH assigns a small budget and solves the problem with one arithmetic step. This case illustrates compact reasoning on a simple problem.

C2: Multi-step arithmetic on MultiArith

Question. Will bought 7 boxes of chocolate candy and gave 3 boxes to his little brother. Each box contains 4 pieces. How many pieces did Will still have?

Reasoning.

$$7 \times 4 = 28, \qquad 3 \times 4 = 12, \qquad 28 - 12 = 16.$$

Prediction. 16
Gold answer. 16
Budget and length. $B_x = 36,\ \#L = 25.$
Observation. The model retains the quantities needed for the final subtraction without unrelated steps.

C3: Relational reasoning on GSM8K

Question. Dana can run four times faster than she can walk. She can skip at half her running speed. If she skips at 3 miles per hour, how far can she travel in six hours when she spends one-third of the time running and two-thirds walking?

Reasoning.

$$\text{Skip} = 3 \implies \text{Run} = 6 \implies \text{Walk} = 1.5.$$

She runs for 2 hours and walks for 4 hours. Therefore,

$$6 \times 2 + 1.5 \times 4 = 12 + 6 = 18.$$

Prediction. 18
Gold answer. 18
Budget and length. $B_x = 61,\ \#L = 61.$

Observation. This problem requires several dependent steps. The model correctly derives the three speeds before computing the final distance.

C4: OOD numerical reasoning on GSM-Hard

Question. The Doubtfire sisters adopted 5,813,040 kittens. Patchy then had three times as many kittens as the adopted group, while Trixie had 12 kittens. How many kittens does the family have in total?

Reasoning.

$$5{,}813{,}040 + 3 \times 5{,}813{,}040 + 12 = 4 \times 5{,}813{,}040 + 12 = 23{,}252{,}172.$$

Prediction. 23,252,172
Gold answer. 23,252,172
Budget and length. $B_x = 62,\ \#L = 62.$
Observation. The model transfers the correct reasoning structure to unusually large values. This case illustrates robustness to the numerical distribution shift in GSM-Hard.

C5: Soft-budget reasoning on MATH500

Question. In how many ways can 8 people sit around a round table if Pierre, Rosa, and Thomas must sit together? Two arrangements are identical if one is a rotation of the other.

Reasoning. Treat Pierre, Rosa, and Thomas as one block. This gives six entities around the table. Their circular arrangements are

$$(6-1)! = 120.$$

The three people inside the block can be arranged in

$$3! = 6$$

ways. Therefore,

$$120 \times 6 = 720.$$

Prediction. 720
Gold answer. 720
Budget and length. $B_x = 63,\ \#L = 64.$
Observation. Reasoning exceeds the predicted budget by one token. The soft budget permits continuation, allowing the model to complete the final step.

Successful-case analysis. The successful cases illustrate different reasoning scales. The simple SVAMP example receives a small budget and uses only eight reasoning tokens. The GSM8K and MATH500 examples receive larger budgets and use longer traces. In Case C5, generation continues beyond the predicted budget to complete the answer. Together, the cases illustrate early completion and flexible continuation under soft guidance. Figure 6 reports the aggregate relation between predicted budgets and observed lengths.

G.2 Failure Cases

F1: Incorrect percentage reference on GSM8K

Question. Josh buys a house for \$80,000 and spends \$50,000 on repairs. The repairs increase the value of the house by 150%. How much profit does he make?

Reasoning.

$$80{,}000 + 50{,}000 = 130{,}000$$

is treated as the total cost. The model then computes

$$150\% \text{ of } 80{,}000 = 120{,}000$$

and incorrectly adds this increase to the total cost:

$$130{,}000 + 120{,}000 = 250{,}000.$$

It finally predicts

$$250{,}000 - 130{,}000 = 120{,}000.$$

Prediction. 120,000
Gold answer. 70,000

Budget and length. $B_x = 63,\ \#L = 72.$
Failure type. The model uses the wrong reference value when computing the final house value. The failure is conceptual rather than computational.

F2: Incorrect inverse-process model on GSM8K

Question. Melanie sold one-third of her vacuum cleaners at the green house. She sold 2 more at the red house. She then sold half of what remained at the orange house. If she had 5 cleaners left, how many did she initially have?

Reasoning. The model defines the initial number as x, but constructs

$$\frac{x}{3} + 2 + \left(\frac{x}{2} - \frac{x}{3}\right) = x - 5.$$

It simplifies this equation and obtains

$$x = 14.$$

Prediction. 14
Gold answer. 18
Budget and length. $B_x = 63,\ \#L = 99.$
Failure type. The sales operations are represented by an incorrect equation. The later algebra is internally consistent, but it follows the wrong model.

F3: Reversed profit direction on GSM-Hard

Question. A hospital sees 500 patients per day. Each patient is seen for 24 minutes. Doctors charge the hospital \$6,914,510 per hour, while the hospital charges patients \$200 per hour. What profit does the hospital make?

Reasoning. The total service time is computed correctly:

$$500 \times 24 = 12{,}000 \text{ minutes} = 200 \text{ hours}.$$

The model then computes

$$6{,}914{,}510 \times 200 = 1{,}382{,}902{,}000$$

and

$$200 \times 200 = 40{,}000.$$

However, it subtracts revenue from cost:

$$1{,}382{,}902{,}000 - 40{,}000 = 1{,}382{,}862{,}000.$$

Prediction. 1,382,862,000
Gold answer. $-1{,}382{,}862{,}000$
Budget and length. $B_x = 63,\ \#L = 104.$
Failure type. The model computes both quantities correctly but reverses the subtraction required for profit.

F4: Distractor interference on SVAMP

Question. Danny found 46 wrappers and 50 bottle caps at the park. He now has 21 bottle caps and 52 wrappers in his collection. How many more bottle caps than wrappers did he find at the park?

Reasoning. The model first obtains the correct quantity:

$$50 - 46 = 4.$$

It then introduces the current collection counts:

$$21 + 52 = 73, \qquad 73 - (50 + 46) = -23.$$

It combines this unrelated result with the correct answer:

$$4 - (-23) = 27.$$

Prediction. 27
Gold answer. 4
Budget and length. $B_x = 56,\ \#L = 56.$
Failure type. The model identifies the correct operation but continues reasoning with irrelevant information. The extra steps overwrite the initially correct result.

F5: Coefficient omission on MATH500

Question. Find the quotient when $x^6 - 3$ is divided by $x + 1$.

Reasoning. The model applies synthetic division with the root -1. However, it writes the coefficient sequence as

$$1,\ 0,\ 0,\ 0,\ 0,\ -3.$$

The correct sequence is

$$1,\ 0,\ 0,\ 0,\ 0,\ 0,\ -3.$$

The missing zero shifts the final calculation. The model therefore obtains

$$x^5 - x^4 + x^3 - x^2 + x - 4.$$

Prediction.

$$x^5 - x^4 + x^3 - x^2 + x - 4$$

Gold answer.

$$x^5 - x^4 + x^3 - x^2 + x - 1$$

Budget and length. $B_x = 57,\ \#L = 123.$
Failure type. A missing coefficient corrupts the synthetic-division procedure. The model continues for substantially longer than its predicted budget but does not recover from the early bookkeeping error.

Failure analysis. The failures arise from identifiable errors in reasoning content. Cases F1 and F2 follow incorrect problem formulations. Case F3 reverses a subtraction. Case F4 introduces irrelevant information. Case F5 contains an early symbolic bookkeeping error. Several traces continue beyond their predicted budgets while retaining the initial error. In these cases, longer generation extends an already incorrect reasoning path.

Soft budgets guide the preferred reasoning scale, while the generation policy determines each reasoning step. These examples locate failures in problem formulation, operation selection, and symbolic bookkeeping. Table 3 separately quantifies answer quality under dynamic, fixed Token-only, and fixed Hybrid routing.

G.3 Case-selection safeguards.

We manually exclude examples with ambiguous reference answers. We also exclude outputs that are mathematically equivalent to the reference but rejected because of notation differences. This prevents evaluation artifacts from being presented as reasoning failures. The ten cases above illustrate distinct behaviors. Aggregate performance is evaluated on the full benchmark test sets.