# WHEN LESS IS ENOUGH: EFFICIENT INFERENCE VIA COLLABORATIVE REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In this work, we introduce DUET (**Du**al-model **E**fficient **T**wo-stage inference), a collaborative inference framework in which a capable model and a lightweight model work together to solve a task. Relying on a single large model to perform end-to-end reasoning and prediction often incurs substantial inference cost. In contrast, DUET decomposes inference into two stages: the capable model produces a reasoning signal, and the lightweight model interprets this signal to generate the final answer, allowing reasoning-intensive computation to be handled by the capable model while non–reasoning-intensive components are delegated to the lightweight model without sacrificing task performance. To achieve this objective, we propose a length-penalized joint training objective that encourages the capable model to transmit only the information that is sufficient for the lightweight model to solve the task. As a result, DUET maintains strong reasoning performance with substantially lower inference cost than end-to-end inference using a large model alone, saving up to 60% of the large model's output tokens on challenging reasoning benchmarks, including AIME and GPQA.

## 1 INTRODUCTION

To solve complex tasks such as mathematical or scientific reasoning, models increasingly rely on long reasoning chains in which the generation of long and extensive intermediate reasoning traces is correlated with accurate prediction (Wei et al., 2022; Muennighoff et al., 2025; Guha et al., 2025; Wang et al., 2022; Nye et al., 2021; Kaplan et al., 2020; Hoffmann et al., 2022; Wang et al., 2022; Wei et al., 2022; Nye et al., 2021; Sardana et al., 2023; Chowdhery et al., 2023; Snell et al., 2024; Wu et al., 2024). While effective, this approach to inference can prove computationally prohibitive. For example, on AIME 2024, reasoning models produce responses 5-10x longer than equivalently-sized instruct models, even on questions that both can solve correctly (Fan et al., 2025). Similarly, DeepSeek-R1 generates 31x more intermediate tokens than Llama 3.1 8B while running 54x slower, requiring 16 H100 GPUs just to serve (Garvey, 2025).

Several directions have been explored to improve this compute-performance tradeoff in LLMs. Methods such as model distillation (Hinton et al., 2015) and supervised fine-tuning (Muennighoff et al., 2025; Ye et al., 2025; Guha et al., 2025) reduce model size and thus the inference cost, but often fail to match the performance of larger models (Liu et al., 2025). Other approaches including pruning (Nayab et al., 2024) and constrained chain-of-thought generation (Munkhbat et al., 2025; Fatemi et al., 2025; Gao et al., 2025; Song & Zheng, 2025) aim to directly reduce the generated token count but risk discarding information critical for correctness (Łajewska et al., 2025).

Taken together, these limitations suggest that the inefficiency is not merely a byproduct of scale, but reflects a more fundamental architectural constraint. In particular, these models assign equal computational weight to all components of a task—whether a complex logical deduction or trivial syntactic formatting—and entangle reasoning with response generation in a single token stream. This is in contrast to how humans and other organisms allocate cognitive resources, and violates the basic principle of *cognitive economy* (Lieder & Griffiths, 2020; Attwell & Laughlin, 2001; Barlow et al., 1961) that establishes that biological organisms allocate metabolic resources in a manner proportional to task difficulty. In fact, this separation not only reduces total effort but also produces compressed, transferable representations. Unfortunately, the current reasoning models forego these benefits of distributed labor.



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.

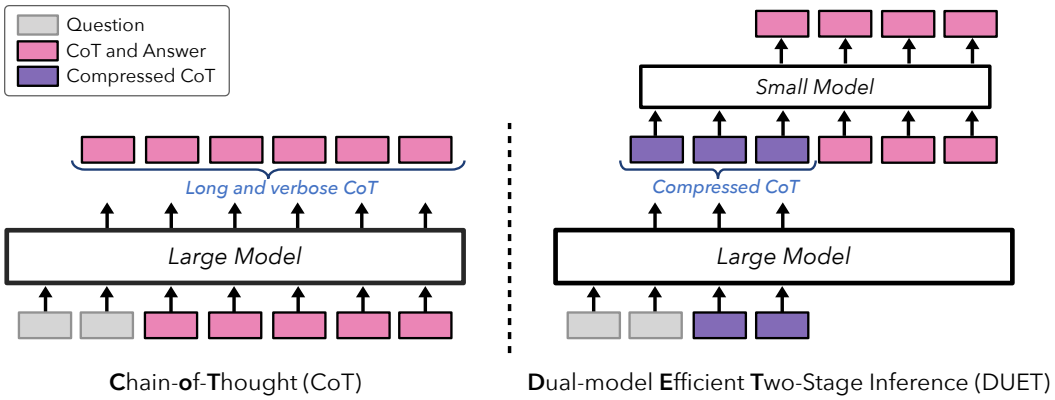


Figure 1: **High-level overview of the DUET framework.** Standard CoT relies on a large model to generate long reasoning traces, substantially increasing inference cost. DUET instead has the capable model produce a high-level reasoning signal that a lightweight model uses to generate the final answer, reducing generation length (and cost) without sacrificing accuracy.

In this work, we move beyond this monolithic design and propose a hierarchical division-of-labor principle, in which different parts of the inference process are handled by different models according to their computational demands. Consider, for example, a complex mathematical problem. Experienced mathematicians often begin by identifying a high-level proof strategy; only after this plan is in place do they turn to the more laborious work of filling in details, checking intermediate claims, and carrying out the necessary calculations. Taking inspiration from this example, we introduce DUET (**Du**al-model **E**fficient **T**wo-stage inference, Figure 1), a collaborative post-training framework that decomposes inference into two stages: (1) reasoning compression, where a large model $M_\theta$ outputs a compressed reasoning signal $z$, and (2) response generation, where a lightweight model $m_\phi$ conditions on $z$ to produce the final answer.

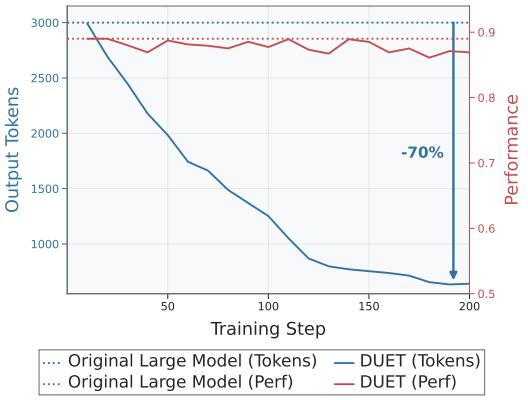


Figure 2: The number of output tokens generated by large model M vs performance over DUET iterations. DUET achieves a reduction in output length by up to 70% with only 200 steps of training. At the same time, performance is comparable to the original large model.

In practice, this is achieved by simultaneously training both models using a length-penalized objective that explicitly minimizes the number of tokens generated while maintaining accuracy. The objective combines (1) a marginal-utility reward that is positive only when the small model is likely to fail, thereby incentivizing use of the large model precisely when it improves outcomes, and (2) an adaptive length-penalty schedule that penalizes total token generation. Empirically, on AIME and GPQA (Rein et al., 2024), DUET matches large-model test performance while reducing its token usage by up to roughly $60\%$.

Our key contributions are:

- We introduce DUET, a dual-model, two-stage inference paradigm that decouples capability-intensive reasoning from inexpensive response generation.
- We propose a length-penalized joint training objective with alternating optimization. We augment this with marginal-utility rewards and an adaptive length-penalty schedule for stability and budget control.

- We demonstrate that DUET substantially reduces expensive model tokens, up to 60% on challenging benchmarks including AIME and GPQA, while preserving the baseline performance.

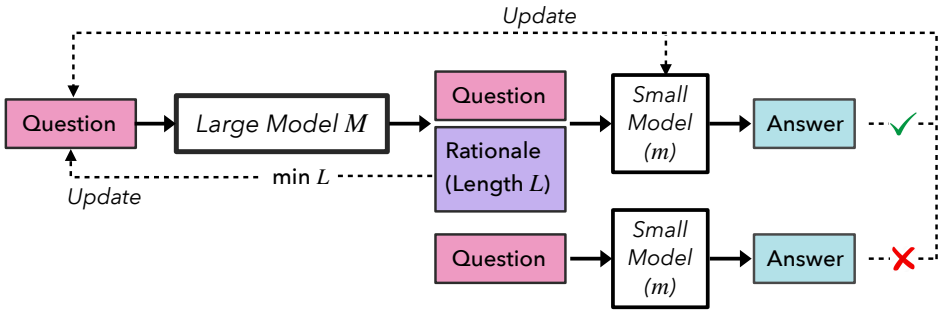


Figure 3: *For each input, the capable model generates a reasoning signal and the lightweight model produces an answer conditioned on (input + signal). In parallel, the lightweight model also produces a baseline answer from the input alone. The baseline score provides a reference point, and the gain achieved by adding the reasoning signal is used to update both models. The capable model is trained with an additional length penalty to keep its reasoning traces short.*

## 2 DUAL-MODEL EFFICIENT TWO-STAGE INFERENCE (DUET)

In this work, we propose DUET, a framework for efficient reasoning through collaboration between a large and a small model. DUET consists of two pretrained models: a capable (large) model $M_\theta$ and a lightweight (small) model $m_\phi$. When the context is clear, we omit parameter subscripts for simplicity.

Given an input prompt $x$, the large model first produces an intermediate representation $z \sim M(\cdot|x)$, which serves as a communication medium between the two models. Conditioned on both the input $x$ and the intermediate representation $z$, the small model then generates the final response $y_{Mm} \sim m(\cdot|x, z)$.

The most direct application of the DUET framework is efficient inference. In this setting, $z$ corresponds to the reasoning process generated by the large model, while the small model conditions on this reasoning to produce the final answer. This decouples expensive reasoning from response generation, allowing inference cost to be significantly reduced.

Individually, the large model $M$ is capable of solving complex tasks but typically incurs high inference cost due to model complexity (large number of parameters) and the length of the reasoning traces. In contrast, the small model $m$ is computationally efficient but lacks sufficient capacity to reliably solve the given tasks on its own. DUET is a way to combine their strengths: the large model learns to handle capability-intensive reasoning, while the small model performs efficient response generation, thereby achieving a favorable trade-off between performance and inference efficiency.

### 2.1 ALGORITHM

DUET proposes the following constrained optimization objective:

$$\max_{M_\theta, m_\phi} \mathbb{E}_{x,z\sim M_\theta(\cdot|x)} \left[\mathbb{E}_{y_{Mm}\sim m_\phi(\cdot|z)}[R(x, y_{Mm})]\right] \text{ s.t. } \mathbb{E}_{x,z\sim M_\theta(\cdot|x)}[L(z)] \leq B, \tag{1}$$

for some bound $B$ on the expected length of the generated sequence of intermediate tokens. Intuitively, the objective in (1) corresponds to finding parameters for $M_\theta$ and $m_\phi$, such that they can maximize the expected reward, while keeping the expected length of intermediate representation (tokens) to be smaller than $B$. The expected length of $z$ directly correlates with inference-time computational cost, making it a natural objective for efficient reasoning. While the reward maximization objective ensures the total reward of the combined pipeline (i.e. $x$ to $z$ to $y_{Mm}$) remains high.

In practice, however, we optimize an approximate (Lagrangian) version of Equation (1):

$$\max_{M_\theta} \max_{m_\phi} \mathbb{E}_{x\sim\mathcal{D}} \left[ \mathbb{E}_{z\sim M_\theta(\cdot|x)} [\mathbb{E}_{y\sim m_\phi(\cdot|z)}[R(x,y)] - \lambda L(z)] \right]. \tag{2}$$

where $\lambda > 0$ corresponds to a regularization parameter. We optimize this objective via alternating optimization over the large model $M_\theta$ and the small model $m_\phi$. The pseudo-code is provided in Section B. During training step $t$, we collect a batch of samples generated by the two model pipeline, i.e. $\{x, z_M, y_{Mm}\}$, where $z_M \sim M_\theta(\cdot \mid x)$ and $y_{Mm} \sim m_\phi(\cdot \mid x, z_M)$. We then optimize $M$ and $m$ alternatingly via the following objectives:

$$\begin{aligned} M: &\quad \max_{M_\theta} \mathbb{E}_{x,z_M,y_{Mm}} \big[ R(x, y_{Mm}) - \lambda \frac{L(z_M)}{B} \big], \\ m: &\quad \max_{m_\phi} \mathbb{E}_{x,z_M,y_{Mm}} \big[ R(x, y_{Mm}) \big]. \end{aligned} \tag{3}$$

where $B$ can be thought of as the hyper-parameter controlling the target expected length bound from Equation (1). To further improve training stability and to avoid degenerate solutions, we introduce two practical techniques on top of the objective in Equation (3). In Section 3.2.5, we report ablation results showing how these two design choices affect the performance of DUET.

**Marginal Utility.** In our experiments, we observed that directly optimizing $M$ with the reward $R(x, y_{Mm}) - \lambda \frac{L(z_M)}{B}$ can lead to poor convergence. This issue was particularly pronounced on tasks for which the small model $m$ already achieves relatively strong baseline performance. In such settings, optimizing Equation (2) can drive the large model $M$ to emit extremely short reasoning traces, effectively offloading most of the generation to the small model. Intuitively, because the objective is non-convex, the relative scaling of the task reward $R$ and the compression cost $L$ may favor a degenerate solution in which the large model minimizes communication by setting $L(z_M)$ close to zero, even when doing so provides little useful guidance.

To mitigate this issue, we replace the reward term $R(x, y_{Mm})$ with a marginal utility reward $R(x, y_{Mm}) - R(x, y_m)$ where $y_m \sim m(\cdot \mid x)$ denotes the output of the small model when run without access to $z_M$. This yields the modified training objective $(R(x, y_{Mm}) - R(x, y_m)) - \lambda \frac{L(z_M)}{B}$. This marginal reward directly measures the contribution of the large model's reasoning by quantifying how much access to $z_M$ improves the small model's performance over its standalone baseline.

Note that when GRPO (Shao et al., 2024) is used as the policy optimization algorithm, the reward $r_m$, which depends only on the query, cancels out in the advantage computation. In our implementation, we sample $r_m$ independently for each $z_M$, which prevents the cancellation, and our experiments show that doing so resolves the collapse issue. We interpret $r_m$ as a natural form of random noise that helps the optimizer escape local minima, a mechanism widely studied in the optimization literature (Jin et al., 2017): although zero-mean noise leaves the expected gradient unchanged, it alters the stochastic optimization trajectory.

**Length Penalty Schedule.** Although Equation (2) is a reasonable relaxation of the constrained problem in Equation (1), a more faithful approach is to adapt the Lagrange multiplier online according to the degree of constraint violation. Concretely, we use the dual-ascent update $\lambda_t = \Big[\lambda_{t-1} + \eta \left( \frac{\mathbb{E}[L(z_M)]}{B} - 1 \right) \Big]_+$, where $\eta > 0$ is a stepsize parameter and $[\cdot]_+$ denotes projection onto the nonnegative reals. This is the standard primal-dual update for inequality-constrained optimization, specialized here to the budget constraint $\mathbb{E}[L(z_M)] \leq B$ (see for example (Boyd & Vandenberghe, 2004)).

The adaptive length penalty introduces a curriculum, prevents early collapse of the large model that can occur with a fixed $\lambda$, and eliminates the need for extensive hyperparameter tuning over penalty strengths. In practice, this schedule leads to more stable training and smoother convergence.

# 3 EXPERIMENTS

## 3.1 EXPERIMENTAL SETTING

**Models.** We adopt Qwen3-4B (thinking mode enabled) as the capable (large) model and Qwen3-0.6B as the lightweight (small) model (Yang et al., 2025). Given our limited computing resources of only 4 H100 GPUs, Qwen3-4B is a suitable choice for the large model in DUET, as it offers a strong balance between reasoning capability and training efficiency: it possesses competitive original performance on challenging reasoning benchmarks while remaining feasible to train efficiently on four H100 GPUs. In contrast, Qwen3-0.6B is significantly more lightweight and cost-effective, making it a natural fit for the small model in DUET framework.

**Training and Evaluation.** Both models are trained using the standard GRPO algorithm (Shao et al., 2024). Training is conducted on one of three datasets: MATH-LightEval (train split), DeepScaleR, or DAPO-Math-17k (Hendrycks et al., 2021; Luo et al., 2025; Yu et al., 2025). We evaluate the models on five challenging reasoning benchmarks: MATH500, AMC23, AIME2024, AIME2025, and GPQA-Diamond (Hendrycks et al., 2021; Rein et al., 2024).

**Metrics.** We report results using three evaluation metrics: *Accuracy, Token length*[1]*, and Intelligence-per-Token* (IPT) (Liu et al., 2025). Accuracy measures the task performance on each benchmark. Token length corresponds to the total number of output tokens generated by the large model. Finally, **IPT is defined as the ratio of accuracy to token length**, capturing the trade-off between the task performance and reasoning efforts. Note that a higher IPT score does not necessarily imply better overall performance; rather, IPT serves as an auxiliary efficiency measure and should be interpreted jointly with accuracy and token length.

**Baselines.** We compare DUET against the large model's original performance, in which the large model conducts end-to-end reasoning and answering alone. In addition, we include a naive chain-of-thought truncation baseline. In this setting, a fixed reasoning budget is pre-specified, and the large model is prompted to generate reasoning as usual; once the budget is exhausted, the reasoning process is manually truncated. The truncated trace, together with the original input, is then passed to the large model itself, which generates the final answer.

Beyond the naive baselines, we include four additional baselines that explicitly target concise reasoning, spanning both prompt-based and training-based approaches: Prompt-based, Single-model, ThinkPrune (Hou et al., 2025), and L-GRPO (Song & Zheng, 2025). Detailed descriptions and implementation specifics are provided in Appendix D.

Other detailed training configurations, including batch sizes, learning rates, and other hyperparameters, are provided in Section C.

## 3.2 MAIN RESULTS AND ABLATIONS

We summarize our main results in Figure 4. Across all evaluated benchmarks, DUET consistently achieves performance comparable to, and in some cases exceeding, that of the original large model, while substantially reducing inference cost quantified by the output token length. On average, DUET reduces the total number of output tokens generated by the large model by approximately $70\%$, demonstrating a favorable performance–efficiency trade-off enabled by DUET and joint training.

The large model alone is capable of solving these reasoning tasks, but its performance typically relies on generating long reasoning traces. Naive truncation reduces inference cost by imposing a fixed reasoning budget, but it discards reasoning tokens indiscriminately, regardless of their importance. As a result, truncation often removes critical intermediate information, leading to noticeable performance degradation, especially on more challenging benchmarks.

In contrast, DUET explicitly trains the capable model to compress its reasoning into a concise and informative signal that is tailored to the lightweight model. Through joint training with a length-penalized objective, the capable model learns to preserve the most task-relevant information while

[1]Output token length refers to tokens generated by the large model. The small model adds negligible cost due to its smaller size and shorter outputs, so large-model decoding dominates inference cost. For the total cost, please refer to Appendix E.1.

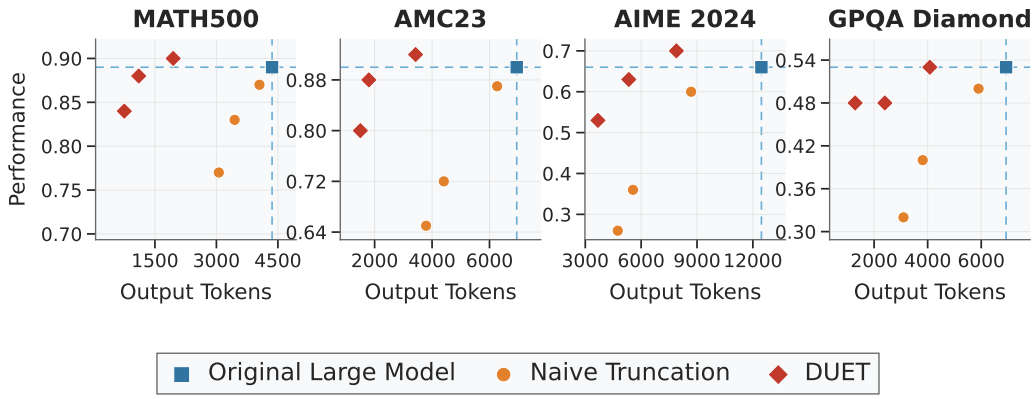


Figure 4: *We compare DUET with the original large model and naive truncation on MATH500, AMC23, AIME2024, and GPQA Diamond benchmarks. The x-axis shows the total number of output tokens generated by the large model, and the y-axis shows task performance. Dashed vertical and horizontal lines indicate the output tokens and performance of the original large model, respectively. Each DUET point corresponds to a checkpoint of different training steps (50, 100, and 150). Points for naive truncation correspond to fixed reasoning budgets of 3072, 4096, and 8192 tokens.*

eliminating unnecessary verbosity. This enables DUET to maintain strong reasoning performance while consuming much less tokens, highlighting the effectiveness and advantage of our DUET framework.

Unlike approaches that aim to improve a model's capability, DUET is designed to preserve the existing capability of a strong model while reducing the amount of reasoning efforts required to realize that capability via collaboration with the small model. To study how sensitive DUET is to the choice of training data, we jointly train the dual-model system on three different datasets: MATH-LightEval, DeepScaleR, and DAPO-Math-17k. The results are summarized in Table 2.

Table 1: *Performance of DUET compared with different baselines across benchmarks. Each cell reports Intelligence-per-Token (IPT), followed by (Accuracy, Large-model output tokens). IPT is measured as intelligence per 1000 tokens.*

| Method | Benchmark | | | |
|---|---|---|---|---|
| | AMC 23 | AIME 2024 | GPQA Diamond | MATH 500 |
| **Original model** | **0.130** (0.90, 6948) | **0.053** (0.66, 12458) | **0.077** (0.53, 6928) | **0.204** (0.89, 4358) |
| **DUET** | **0.489** (0.88, 1800) | **0.118** (0.63, 5337) | **0.200** (0.48, 2402) | **0.796** (0.88, 1106) |
| **Prompt-based** | **0.145** (0.90, 6209) | **0.061** (0.67, 11025) | **0.121** (0.52, 4303) | **0.258** (0.89, 3444) |
| **Single model** | **0.390** (0.82, 2103) | **0.086** (0.50, 5786) | **0.229** (0.45, 1969) | **0.764** (0.87, 1139) |
| **ThinkPrune** | **0.162** (0.87, 5353) | **0.053** (0.53, 10007) | **0.079** (0.44, 5534) | **0.273** (0.87, 3187) |
| **L-GRPO** | **0.162** (0.92, 5668) | **0.067** (0.66, 9835) | **0.101** (0.54, 5332) | **0.260** (0.89, 3429) |

### 3.2.1 Comparison Relative to Baselines

We compare DUET with several prompt-based and training-based baselines designed to encourage concise reasoning. As shown in Table 1, DUET consistently outperforms all baselines across benchmarks, maintaining the high accuracy of the original model while using significantly fewer large-model tokens. For example, on AMC23, DUET improves IPT from 0.130 to 0.489 and reduces token usage from 6948 to 1800 while maintaining the original performance, with similar gains observed on AIME24, GPQA Diamond, and MATH500. Overall, DUET achieves the best efficiency–performance trade-off, reducing large-model tokens by up to 60–70% while maintaining or even improving accuracy.

In contrast, the baseline approaches provide limited gains: prompt-based methods yield only modest improvements on the efficiency while single-model and ThinkPrune may degrade performance,

indicating that naive compression either discards essential reasoning or fails to achieve truly concise reasoning. Beyond performance, DUET offers additional advantages. Its dual-model design decouples reasoning from final task completion, enabling the large model to produce compact, task-relevant signals while allowing the small model to be flexibly prompted or fine-tuned for different objectives—something that is difficult to achieve with a single concise model. Moreover, DUET is well-suited for practical deployment: a large model can reside on a remote server and provide a single reasoning signal, after which a lightweight local model performs downstream tasks on-device, enabling more efficient, privacy-preserving, and secure inference.

Table 2: *Performance of DUET trained on different datasets across benchmarks. Each cell reports Intelligence-per-Token (IPT), followed by (Accuracy, Large-model output tokens). IPT is measured as intelligence per 1000 tokens.* ***Top row:*** *original large model (end-to-end) performance.* ***Bottom rows:*** *DUET trained on different datasets.*

| Method / Training Data | Benchmark | | | | |
|---|---|---|---|---|---|
| | MATH 500 | AMC 23 | AIME 2024 | AIME 2025 | GPQA Diamond |
| **Original large model** | **0.20** (0.89, 4358) | **0.13** (0.90, 6948) | **0.05** (0.66, 12458) | **0.04** (0.53, 12625) | **0.08** (0.53, 6928) |
| **DUET trained on:** | | | | | |
| MATH-LightEval | **0.81** (0.90, 1109) | **0.45** (0.85, 1878) | **0.13** (0.63, 5013) | **0.07** (0.47, 6418) | **0.19** (0.52, 2689) |
| DeepScaleR | **0.80** (0.88, 1106) | **0.49** (0.88, 1800) | **0.12** (0.63, 5337) | **0.08** (0.47, 5887) | **0.20** (0.47, 2402) |
| DAPO-MATH-17k | **0.69** (0.88, 1271) | **0.33** (0.85, 2544) | **0.10** (0.57, 5898) | **0.05** (0.43, 7819) | **0.16** (0.52, 3190) |

### 3.2.2 Generalization Across Difficulty Levels

A notable observation from Table 2 is that DUET exhibits strong generalization across substantial difficulty gaps between training and evaluation problems. When trained on MATH-LightEval, DUET successfully performs efficient reasoning while maintaining performance comparable to the original large model on significantly harder benchmarks such as AIME2024 and AIME2025, reducing large-model token usage by $60\%$ and $49\%$, respectively.

This behavior is fundamentally different from training for capability. Due to the large difficulty gap between MATH-LightEval and AIME-level problems, it is typically infeasible to train a model to solve AIME problems by learning directly from MATH-LightEval alone. In contrast, DUET does not attempt to acquire new problem-solving skills; instead, it learns to compress and distill the reasoning required to express the model's existing intelligence. These results provide strong empirical evidence that DUET is not training for improved capability, but rather for compressed intelligence—enabling the model to achieve similar accuracy with substantially fewer reasoning tokens.

### 3.2.3 Sensitivity to Training Data

As shown in Table 2, DUET trained on all three datasets consistently achieves substantially higher inference efficiency while maintaining task performance comparable to the original large model, as evidenced by markedly higher IPT (Information Per Token) and accuracy.

Notably, the large model $M$ has original accuracies of $89\%$, $61\%$ and $77\%$ on Math-LightEval, DAPO-Math-17k and DeepScaleR, respectively. These results indicate that DUET is effective across diverse training datasets as long as the large model exhibits sufficient baseline capability on the training data. Since DUET is designed to compress existing intelligence rather than to improve a model's underlying capability, a natural question arises:

*Can DUET be used to improve performance on problems where the large model itself struggles?*

To investigate this question, we trained DUET on the LIMO dataset (Ye et al., 2025), on which the original large model achieves only around $10$–$20\%$ accuracy. The results are shown in Figure 5.

When trained on LIMO, DUET fails to achieve meaningful compression. After 150 steps of training, DUET only improves IPT by a small margin compared to training on the other three datasets. The large model continues to produce long reasoning traces, and the overall token length remains high throughout training.

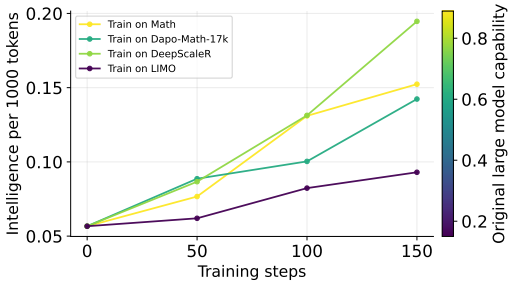


Figure 5: *The performance evolution from DUET training. Each curve corresponds to a different training dataset, on which the large model exhibits a different level of accuracy. Lighter-colored curves indicate training datasets for which the large model has higher original accuracy. The y-axis reports the IPT metric averaged across all benchmarks except MATH500.*

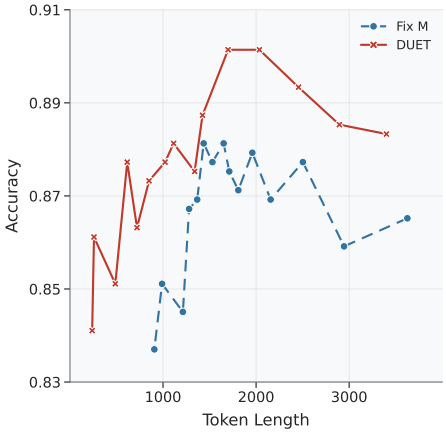


Figure 6: *Performance trajectories of DUET and the Fix m variant during training. Performance is logged every 10 training steps. The x-axis shows the total number of output tokens generated by the large model, and the y-axis reports task performance. Models are trained on DeepScaleR and evaluated on MATH500. Training proceeds from right to left.*

This behavior follows from the DUET objective in Equation (3). When the large model lacks baseline capability on the training distribution, the marginal-utility signal $R(x, y_{Mm}) - R(x, y_m)$ becomes weak and noisy. Thus, without a reliable reasoning signal to compress, DUET cannot learn meaningful communication, leading the large model to either retain long traces or collapse toward uninformative outputs under the length penalty, as observed on the LIMO dataset.

Figure 5 illustrates the evolution of the intelligence-per-token metric during training for DUET models trained on datasets where the large model exhibits varying levels of initial performance. Training on DeepScaleR yields the strongest gains, suggesting that the most effective training datasets are those on which the large model can solve problems confidently, yet which are not so easy as to limit opportunities for meaningful reasoning compression.

### 3.2.4 Is it Necessary to Train the Small Model?

In the default setting of DUET, the small model is trained jointly with the large model, allowing the two to co-evolve during optimization. To understand whether this co-training is essential, we also consider a variant in which the small model is kept fixed. In this setting, we keep the reward formulation unchanged but disable updates to the small model.

Figure 6 compares the evolution of the accuracy and token length on the MATH500 benchmark during training with a fixed small model (*Fix* $m$) and with joint training (*DUET*). Interestingly, we find that DUET remains effective even when the small model is not trained, still achieving meaningful compression of the large model's reasoning. This suggests that part of DUET's compression capability arises from optimizing the large model alone under the DUET objective. Further, throughout training, the DUET curve consistently lies above that of *Fix* $m$, indicating a better trade-off between accuracy and inference efficiency. Jointly training the small model enables the large model to produce more concise reasoning without sacrificing performance. This behavior is intuitive: as the small model co-evolves with the large model, the large model can transmit increasingly compressed information that remains interpretable by the small model. In contrast, a fixed small model lacks the capacity to adapt to progressively more compact representations.

### 3.2.5 Ablations on Marginal Utility and Length Penalty Schedule

In Section 2.1, we introduce two practical design choices to stabilize the training: marginal utility and length penalty schedule. We conducted ablation experiments to study their effects, in which we removed each component individually or removed both. Specifically, without marginal utility means that the reward for the large model is $R(x, y_{Mm})$ alone without subtracting $R(x, y_m)$ (see Section Section 2.1 for details on this). Without the length penalty schedule means that a constant length-penalty coefficient is used throughout training.

Figure 7 shows the ablation results. For each algorithmic variant, we evaluate the checkpoint after 150 training steps on the AMC23 benchmark. All models are trained on the Math-Lighteval dataset.

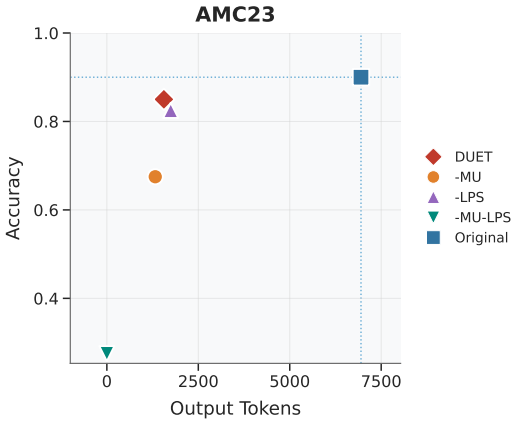


Figure 7: *The ablation results of marginal utility and length penalty schedule. –MU removes marginal-utility; –LPS removes the length-penalty schedule and sets a constant length-penalty coefficient as 0.5. Dashed blue vertical lines indicate the original large-model performance. The performance is evaluated on the AMC23 benchmark.*

When marginal utility is removed, the large model still learns to reduce its output length, however, it also incurs a noticeable drop in accuracy which is not preferred. This observation is consistent with expectations. Math-Lighteval is a relatively easy training dataset, on which the small model has a decent accuracy (around 40%) without any help from the large model. Hence, without marginal utility, the large model tends to output few tokens to avoid length penalty, while still achieving a moderate accuracy relying solely on the small model's ability. With marginal utility, this behavior of relying completely on the small model itself is not rewarded.

When both marginal utility and the length penalty schedule are omitted, the training collapses: the large model converges to producing empty or extremely short outputs. For the full ablation results, we refer readers to Section E.4.

## 4 Conclusions and Future Works

In this work, we introduced DUET for efficient inference by splitting it into two stages: a high-capacity model that outputs a compact reasoning signal, and a lightweight model that decodes it into the final response. We show that DUET can eliminate up to 60% of large-model output tokens on AIME and GPQA without sacrificing accuracy. More broadly, our results shift the efficiency lens from compressing *parameters* to compressing the *information* a model must transmit.

**Limitation.** Due to limited computing resources, we evaluate DUET using Qwen3-4B as the large model and do not explore larger model scales. It would be interesting to see the behavior of DUET at larger model scales and under different model-size combinations. In addition, we evaluate DUET only on tasks with outcome-based rewards. It would be interesting to extend the framework to settings involving supervised fine-tuning losses or process rewards.

**Future Work.** Looking ahead, DUET offers several natural extensions: replacing text-token communication with continuous codes (e.g., latent codes or soft prompts) (Hao et al., 2024b; Zou et al., 2025a) may increase compression and enable smoother end-to-end training, albeit with reduced interpretability. Moving from a one-shot hand-off to an interactive protocol where the small model queries the bigger model only under low confidence could further reduce computational overhead. More broadly, training a single large model to output a model-agnostic reasoning code that multiple heterogeneous small models can reliably decode, amortizes expensive inference across many lightweight, task-specific decoders, and forms an exciting avenue for future work. Finally, DUET is complementary to existing efficiency techniques and can be combined with methods such as Flash Attention, speculative decoding, etc.

## AI USE STATEMENT

In this work, we used generative AI tools for polishing the language, improving the presentation and implementing methods. We have not used generative AI tools for theoretical models development, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypothesis, designing methodology or experiments, assisting with translation, interpreting results. Synthetic data generations, cleaning and reformating dataset, supporting qualitative and thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

David Attwell and Simon B Laughlin. An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism*, 21(10):1133–1145, 2001.

Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01):217–233, 1961.

Adarsh Prasad Behera, Jaya Prakash Champati, Roberto Morabito, Sasu Tarkoma, and James Gross. Towards efficient multi-llm inference: Characterization and analysis of llm routing and hierarchical techniques. *arXiv preprint arXiv:2506.06579*, 2025.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Federico Cassano and Alexander M. Rush. Training composer for longer horizons. Cursor Research Blog, 2026.

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

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin C Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242, 2024.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

Siqi Fan, Bowen Qin, Peng Han, Shuo Shang, Yequan Wang, and Aixin Sun. The price of a second thought: On the evaluation of reasoning efficiency in large language models. *arXiv preprint arXiv:2505.22017*, 2025.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*, 2025.

Chengqian Gao, Haonan Li, Taylor W Killian, Jianshu She, Renxi Wang, Liqun Ma, Zhoujun Cheng, Shibo Hao, and Zhiqiang Xu. Concise reasoning in the lens of lagrangian optimization. *arXiv preprint arXiv:2510.10168*, 2025.

Cormac Garvey. Performance analysis of DeepSeek R1 AI inference using vLLM on ND-H100-v5. Microsoft Tech Community, Azure High Performance Computing Blog, August 2025.

Gabriel Grand, Joshua B Tenenbaum, Vikash K Mansinghka, Alexander K Lew, and Jacob Andreas. Self-steering language models. *arXiv preprint arXiv:2504.07081*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Sharut Gupta, Phillip Isola, Stefanie Jegelka, David Lopez-Paz, Kartik Ahuja, Mark Ibrahim, and Mohammad Pezeshki. Reasoncache: Teaching llms to reason without weight updates. *arXiv preprint arXiv:2602.02366*, 2026.

Shiqin Han, Manning Gao, Menghua Jiang, Yuncheng Jiang, Haifeng Hu, and Sijie Mai. Uncertainty-aware collaborative system of large and small models for multimodal sentiment analysis. *arXiv preprint arXiv:2509.04459*, 2025a.

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, 2025b.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024a.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 14852–14882, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pp. 1724–1732. PMLR, 2017.

Gurusha Juneja, Subhabrata Dutta, Soumen Chakrabarti, Sunny Manchanda, and Tanmoy Chakraborty. Small language models fine-tuned to coordinate larger language models improve complex reasoning. *arXiv preprint arXiv:2310.18338*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Steven Kolawole, Don Dennis, Ameet Talwalkar, and Virginia Smith. Agreement-based cascading for efficient inference. *arXiv preprint arXiv:2407.02348*, 2024.

Weronika Łajewska, Momchil Hardalov, Laura Aina, Neha Anna John, Hang Su, and Lluís Màrquez. Understanding and improving information preservation in prompt compression for llms. *arXiv preprint arXiv:2503.19114*, 2025.

Shiye Lei, Zhihao Cheng, Kai Jia, and Dacheng Tao. Revisiting llm reasoning via information bottleneck. *arXiv preprint arXiv:2507.18391*, 2025.

Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. Steering llm thinking with budget guidance. *arXiv preprint arXiv:2506.13752*, 2025.

Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, et al. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*, 2025.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? *Advances in Neural Information Processing Systems*, 37: 45359–45385, 2024.

Ximing Lu, Seungju Han, David Acuna, Hyunwoo Kim, Jaehun Jung, Shrimai Prabhumoye, Niklas Muennighoff, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, et al. Retro-search: Exploring untaken paths for deeper and efficient reasoning. *arXiv preprint arXiv:2504.04383*, 2025.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 2: short papers)*, pp. 1773–1781, 2023.

Kai Mei, Wujiang Xu, Minghao Guo, Shuhang Lin, and Yongfeng Zhang. Omnirouter: Budget and performance controllable multi-llm routing. *ACM SIGKDD Explorations Newsletter*, 27(2): 107–116, 2025.

Meta. Introducing muse spark: Msl's first model, purpose-built to prioritize people. Meta Newsroom, 2026.

Giovanni Monea, Yair Feldman, Shankar Padmanabhan, Kianté Brantley, and Yoav Artzi. Breadcrumbs reasoning: Memory-efficient reasoning with compression beacons. *arXiv preprint arXiv:2510.13797*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*, 2025.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*, 2024.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. 2021.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mingyang Song and Mao Zheng. Walk before you run! concise llm reasoning via reinforcement learning. *arXiv preprint arXiv:2505.21178*, 2025.

Dijia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57144–57163. PMLR, 2025.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. Ieee, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. Understanding chain-of-thought in llms through information theory. *arXiv preprint arXiv:2411.11984*, 2024.

Jingyao Wang, Wenwen Qiang, Zeen Song, Changwen Zheng, and Hui Xiong. Learning to think: Information-theoretic reinforcement fine-tuning for llms. *arXiv preprint arXiv:2505.10425*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zhenyi Wang, Li Zou, Shengyun Wei, Feifan Liao, Jia Zhuo, Haibo Mi, and Rongxuan Lai. Large language model enabled semantic communication systems. *arXiv preprint arXiv:2407.14112*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *International Conference on Learning Representations*, 2025b.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*, 2025.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025a.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025b.

# A Related Work

**Model collaboration.** DUET belongs to a broader class of frameworks that leverage collaboration among multiple models. One prominent line of research is speculative decoding, which accelerates generation by allowing a small model to first produce draft tokens that are subsequently verified or refined by a larger model (Chen et al., 2023; 2024). Another widely studied paradigm is model routing, where a learned router assigns queries to the small or large model based on the predicted query difficulty and the desired quality level (Ding et al., 2024; Mei et al., 2025; Behera et al., 2025). Similarly, model cascading (Kolawole et al., 2024) builds a cascade of models of different sizes to dynamically assign inputs for efficient inference.

A popular line of model collaboration research studies role-based collaboration, in which different models are assigned complementary responsibilities within a task. Juneja et al. (2023) proposed a two-model system consisting of a decomposition generator that breaks a task into subproblems and a solver that addresses each subproblem. Similarly, Grand et al. (2025) introduced a planner–follower paradigm, where a planner model generates a task-specific inference program that is executed by a population of follower models. Han et al. (2025a) proposed an uncertainty-aware collaborative system that offloads high-confidence inputs to a smaller model while reserving uncertain cases for a larger model. Relatedly, Zhang et al. (2025a) proposed Recursive Language Models, in which a model programmatically inspects long inputs and recursively invokes itself on selected subcontexts. While these methods emphasize decomposition and recursive control rather than explicit reasoning compression, they share with DUET the broader view that inference efficiency can arise from structuring computation across models or calls rather than relying on a single monolithic generation pass.

**Knowledge distillation.** Knowledge distillation (Hinton et al., 2015) is another form of model collaboration in which a large teacher model transfers knowledge to a smaller student model to improve efficiency. Several works (Ho et al., 2023; Hsieh et al., 2023; Magister et al., 2023) have explored generating reasoning traces from powerful teacher models to fine-tune smaller students. On-policy distillation (Gu et al., 2023; Agarwal et al., 2024) also emerged as a popular method where student models are trained on data generated by themselves under the teacher's supervision.

**Memory compaction and context compression.** Another related direction studies how to compress the model's working context or internal state rather than only its final answer tokens. Zhang et al. (2025b) proposed Activation Beacon, which compresses long contexts at the activation/KV-cache level to improve efficiency. More recently, Monea et al. (2025) introduced learned compression beacons that periodically summarize past KV states during reasoning, and Wu et al. (2025) studied periodic context summarization for long-horizon agentic search. These approaches are complementary to DUET: they primarily compress memory or history within a single reasoning process, whereas DUET learns a concise reasoning signal that is explicitly optimized to be consumed by a second model (in an independent auto-regressive run).

At the systems level, Cursor reports that Composer is trained to periodically self-summarize its working context to continue long-horizon problem solving under context constraints, while Meta reports that Muse Spark can launch multiple subagents in parallel for harder queries, illustrating a related trend toward system-level decomposition in deployed assistants (Cassano & Rush, 2026; Meta, 2026).

**Concise reasoning.** A growing body of work investigates efficient reasoning. One line of research explicitly encourages conciseness through fine-tuning or optimization objectives. Munkhbat et al. (2025) showed that models can produce concise reasoning traces by fine-tuning on self-generated examples. Fatemi et al. (2025) observed that standard reinforcement learning objectives tend to favor overly long responses and introduce a secondary RL fine-tuning phase explicitly rewarding brevity. Song & Zheng (2025) proposed a two-stage RL algorithm which first trains the model for more steps, and subsequently fine-tunes it for conciseness in the second stage. Gao et al. (2025) considered a length-constrained optimization problem closely related to ours, however, their approach considers a single model and enforces a hard constraint on the token budget during generation, whereas DUET adopts a dual-model framework that allows for more flexible reasoning compression. Arora & Zanette (2025) also train reasoning models to reduce inference-time compute using a reinforcement-learning objective that incentivizes shorter chains of thought while maintaining accuracy, yielding a family of models with different efficiency–accuracy trade-offs. Similar to Gao et al. (2025), this approach

operates in a single-model setting, whereas DUET uses a dual-model decomposition in which a large model communicates a compressed reasoning signal to a lightweight model.

Towards concise reasoning, one popular line of work focuses on explicit budget control. Budget forcing methods terminate the reasoning process once a predefined token limit is reached (Muennighoff et al., 2025). Liu et al. (2025) studied the accuracy degradation caused by truncation, and showed that it arises from inadequate RL optimization rather than the lack of sophisticated penalties. Other approaches guide reasoning to adhere to a budget by predicting the remaining thinking length (Li et al., 2025) or dynamically adjusting the token budget based on problem complexity (Han et al., 2025b). Gupta et al. (2026) show that training a prefix cache while keeping the base model frozen is sufficient to improve reasoning performance, and strikingly induces more concise reasoning even without explicitly optimizing for brevity.

There are also several heuristic approaches. These include prompt-based methods for eliciting concise reasoning (Nayab et al., 2024), controlled frameworks that encourage step-skipping behavior to generate shorter yet accurate reasoning paths (Liu et al., 2024), and reinforcement learning paradigms that employ control tokens (e.g., <short> and <think>) to switch between concise and detailed reasoning modes (Fang et al., 2025). Lu et al. (2025) proposed Retro-Search, which retrospectively revises reasoning paths to discover higher-quality yet shorter traces, enabling student models to achieve faster inference with strong reasoning performance. Relatedly, Hou et al. (2025) and Xia et al. (2025) explicitly prune or skip low-utility reasoning tokens to improve the performance–length trade-off.

**Latent reasoning and semantic communication.** A closely related line of work asks whether reasoning or inter-model communication should occur in natural language at all. Hao et al. (2024a) introduced Chain of Continuous Thought, where intermediate reasoning unfolds in continuous latent space rather than through discrete text tokens. Shen et al. (2025) compress explicit chain-of-thought into continuous representations via self-distillation, while Su et al. (2025) mix latent and text tokens to shorten reasoning traces while preserving learnability. Zhu et al. (2025) provided theoretical support for such latent reasoning by showing that continuous thoughts can encode superpositions of multiple search frontiers. In multi-model settings, Fu et al. (2025), Zheng et al. (2025), and Zou et al. (2025b) explore direct semantic communication through KV caches, latent thoughts, or shared latent working memory instead of textual messages. At a broader level, this perspective is closely connected to semantic communication, where the goal is to transmit task-relevant meaning rather than faithfully reproduce surface form (Wang et al., 2024). Compared with these approaches, DUET deliberately retains a textual, inspectable communication channel between the large and small models, trading off some compression efficiency for interpretability and ease of training.

**Information bottleneck.** The way DUET decomposes answer generation into an initial reasoning trace produced by a large model and a subsequent, potentially longer, generation by a smaller model is closely related to the information bottleneck principle. The information bottleneck (IB) principle was originally introduced by Tishby et al. (2000) and later extended to deep learning models (Tishby & Zaslavsky, 2015). More recently, several works have used information-theoretic perspectives to study and improve reasoning in large language models. Lei et al. (2025), for example, view reasoning as an information bottleneck and use this perspective as a regularizer to encourage reasoning trajectories to remain informative about the correct answer. Ton et al. (2024) analyze chain-of-thought reasoning through an information-theoretic lens by quantifying the information gain associated with each reasoning step. Similarly, Wang et al. (2025) propose an information-theoretic reinforcement fine-tuning framework that encourages accurate reasoning with fewer generated tokens.

## B ALGORITHM

**Algorithm 1** DUET: Training Algorithm

**Input:** Dataset $\mathcal{D}$, capable model $M_\theta$, lightweight model $m_\phi$, reward function $R(\cdot)$, initial length penalty weight $\lambda_0$, number of rollouts $G$, batch size $S$, training steps $T$, length penalty schedule factors $\eta$ and $B$.
**for** $t = 1$ **to** $T$ **do**
  Sample a minibatch $\{(x_i, y_i^*)\}_{i=1}^{S} \sim \mathcal{D}$ of $S$ samples, consisting of prompts $x$ and a response $y$ (containing the answer).
  **for** $i = 1$ **to** $S$ **do**
    Sample $G$ reasoning signals from the capable model:

$$z_{i,1:G} \sim M_\theta(\cdot \mid x_i).$$

    Sample $G$ responses from the lightweight model (without the reasoning signal):

$$y^m_{i,1:G} \sim m_\phi(\cdot \mid x_i).$$

    Compute baseline reward $r^m_{i,j} \leftarrow R(x_i, y^m_{i,j}, y_i^*) = \mathbf{1}\{y^m_{i,j} = y_i^*\}, \forall j = 1, \ldots, G.$
    **for** $j = 1$ **to** $G$ **do**
      Generate a response from the lightweight model conditioned on $(x_i, z_{i,j})$:

$$\hat{y}_{i,j} \sim m_\phi(\cdot \mid x_i, z_{i,j}).$$

      Compute reward $r_{i,j} \leftarrow R(x_i, \hat{y}_{i,j}, y_i^*)$ and reasoning length $\ell_{i,j} \leftarrow L(z_{i,j})$.
    **end for**
  **end for**
  Update the lightweight model $m_\phi$ using GRPO with the rewards: $r_{i,j}$.
  Update the capable model $M_\theta$ using GRPO with the rewards: $r_{i,j} - r^m_{i,j} - \lambda_{t-1}\, \ell_{i,j}/B$.
  Update the length penalty coefficient $\lambda$:

$$\lambda_t \leftarrow \lambda_{t-1} + \eta \left( \frac{E[L]}{B} - 1 \right), \quad \text{and} \quad E[L] = \frac{1}{nS} \sum_{i=1}^{S} \sum_{j=1}^{n} \ell_{i,j}.$$

**end for**
**Return:** Trained models $(M_\theta, m_\phi)$.

## C EXPERIMENTAL SETUP

All experiments are conducted on four NVIDIA H100 GPUs. We set hyperparameters $\eta = 0.01$ and $B = 1,000$. The training batch size is set to 64. The learning rate for both the large and small model is $1 \times 10^{-6}$. For both the large and small models, we sample four rollouts per input to compute advantages during GRPO training. We applied a KL-divergence regularization term to the reward with a coefficient of 0.001. The maximum response lengths for the large model $M$ and the small model $m$ are set to 16,384 and 2,048 tokens, respectively. The maximum prompt length is set to 1,024 tokens for the large model and 18,432 tokens for the small model. The rollout temperature for training and validation is set to 1.0 and 0.0, respectively.

All training is implemented using VERL (Sheng et al., 2024) as the underlying framework.

## D BASELINE ALGORITHMS

We compare DUET against the following baseline algorithms:

- **Prompt-based:** In this baseline, we encourage concise reasoning through prompt engineering. Specifically, we augment the input with the instruction: "Note that you should keep the reasoning concise: include only essential reasoning steps and avoid repetition or unnecessary explanations."
- **Single-model:** This baseline removes the small model and retains only the large model. The large model is trained using the same reward function as in DUET, but without the two-stage collaboration.
- **ThinkPrune (Hou et al., 2025):** This method enforces a maximum response length during training. Only the portion of the response within the length limit is decoded to extract the final answer and compared against the ground truth. Consequently, a response receives a reward only if it produces the correct answer before exceeding the preset length. The maximum length is progressively reduced in a stepwise schedule: we initialize it at 8000 tokens and decrease it by 1000 every 20 training steps, with a minimum threshold of 1000 tokens.
- **L-GRPO (Song & Zheng, 2025):** This baseline introduces a group-based length penalty. Specifically, a length penalty term is added to the reward, defined as the ratio between the response length and the maximum length among correct responses within the group. The penalty is applied only to correct responses.

# E Additional Experimental Details

## E.1 Total Cost

For a model with $B$ parameters, given a prompt of length $P$, the FLOPs required to generate a response of length $L$ can be approximated as $2BL + 4nd\left(LP + \frac{L(L-1)}{2}\right)$, where $n$ is the number of transformer layers and $d$ is the hidden dimension. In practice, the parameter-dependent term dominates, and the total cost can be well approximated by $2BL$. Therefore, throughout the paper, we use the output token length as a proxy for computational cost. We report only the large model's output tokens in the main page, as its parameter size dominates the overall computation compared to the small model. For completeness, we provide the token counts of both models in Table 3.

In addition, we measure the wall-clock inference time on the full evaluation dataset. The original model requires 63m27.599s, while DUET reduces this to 41m54.337s, demonstrating a substantial improvement in practical efficiency.

Table 3: *The performance and efficiency of DUET under different training steps. Each cell reports (accuracy, large-model tokens, small-model tokens).*

| Training steps | Benchmark | | | |
|---|---|---|---|---|
| | MATH 500 | AMC 23 | AIME 2024 | GPQA Diamond |
| 50 | (0.90, 1946, 402) | (0.92, 3426, 538) | (0.70, 7892, 705) | (0.53, 4082, 6) |
| 100 | (0.88, 1106, 452) | (0.88, 1800, 634) | (0.63, 5337, 837) | (0.48, 2402, 13) |
| 150 | (0.84, 753, 376) | (0.80, 1504, 571) | (0.53, 3681, 812) | (0.48, 1292, 6) |

## E.2 Values Used to Plot Main Figures

In this subsection, we report the numerical values used to generate the figures in the main paper, in order to facilitate reproducibility and detailed inspection of the results.

Table 4: *Numerical values used to plot Figure 4. Each cell reports (accuracy, large-model output tokens).*

| Method | Benchmark | | | |
|---|---|---|---|---|
| | MATH 500 | AMC 23 | AIME 2024 | GPQA Diamond |
| **Original large model** | (0.89, 4358) | (0.90, 6948) | (0.66, 12458) | (0.53, 6928) |
| **Naive truncation with token budget:** | | | | |
| 3072 | (0.77, 3059) | (0.65, 3792) | (0.26, 4744) | (0.32, 3097) |
| 4096 | (0.83, 3445) | (0.72, 4409) | (0.36, 5561) | (0.40, 3817) |
| 8192 | (0.87, 4050) | (0.87, 6266) | (0.60, 8684) | (0.50, 5900) |
| **DUET at training steps:** | | | | |
| 50 | (0.90, 1946) | (0.92, 3426) | (0.70, 7892) | (0.53, 4082) |
| 100 | (0.88, 1106) | (0.88, 1800) | (0.63, 5337) | (0.48, 2402) |
| 150 | (0.84, 753) | (0.80, 1504) | (0.53, 3681) | (0.48, 1292) |

Table 5: *Numerical values used to plot Figure 5. Each cell reports the intelligence-per-1000-tokens metric evaluated at 50, 100, and 150 training steps, respectively.*

| Training dataset | Benchmark | | |
|---|---|---|---|
| | AIME 2024 | AIME 2025 | GPQA Diamond |
| **MATH-LightEval** | (0.075, 0.126, 0.121) | (0.045, 0.073, 0.075) | (0.113, 0.162, 0.230) |
| **DeepScaleR** | (0.089, 0.119, 0.109) | (0.042, 0.079, 0.106) | (0.129, 0.196, 0.369) |
| **DAPO-MATH-17k** | (0.085, 0.096, 0.124) | (0.068, 0.043, 0.073) | (0.113, 0.162, 0.230) |
| **LIMO** | (0.05, 0.074, 0.094) | (0.046, 0.065, 0.055) | (0.09, 0.108, 0.130) |

Table 6: *Numerical values used to plot Figure 7. –MU removes marginal-utility; –LPS removes the length-penalty schedule. The performance is evaluated on the AMC23 benchmark.*

| Algorithm | Accuracy | Output tokens |
|---|---|---|
| DUET | 0.85 | 1559 |
| -MU | 0.67 | 1323 |
| -LPS | 0.82 | 1746 |
| -MU -LPS | 0.27 | 0 |

### E.3 Necessity of Training the Small Model

In the main paper, we use Qwen3-0.6B as our small model. Despite having only 0.6 billion parameters, it is a relatively capable model that can generate accurate answers when conditioned on the large model's reasoning from the very beginning. We also experimented with Qwen2.5-0.5B-Instruct as the small model, which initially has limited ability to decode the large model's reasoning into correct answers. Through joint training, it quickly learns to solve problems by leveraging reasoning guidance from the large model, demonstrating that training the small model $m$ is necessary in this case.

Figure 8 shows the training progress of DUET compared to the variant with a fixed small model $m$. When the small model has limited initial capability, fixing $m$ causes it to struggle with generating correct responses based on the large model's reasoning, whereas DUET successfully trains the small model to generate correct answers through joint optimization.

Another interesting observation is that even without training the small model, it still shows slight performance improvement, suggesting that the large model adapts its reasoning to be more interpretable for the small model.

### E.4 Additional Ablation Results on Marginal Utility and Length Penalty Schedule

#### E.4.1 Trained on MATH

In Figure 7 in the main paper, we present ablation results evaluated on the AMC23 benchmark. Here in Figure 9, we report the corresponding results on the remaining benchmarks. The observations are consistent with those in the main paper: marginal utility plays a more important role than the length penalty schedule. Using a constant length-penalty coefficient still yields reasonable performance, but the full DUET configuration achieves the best overall results. When both marginal utility and the length penalty schedule are removed, training becomes unstable and often collapses, causing the large model to converge to producing empty or near-empty outputs.

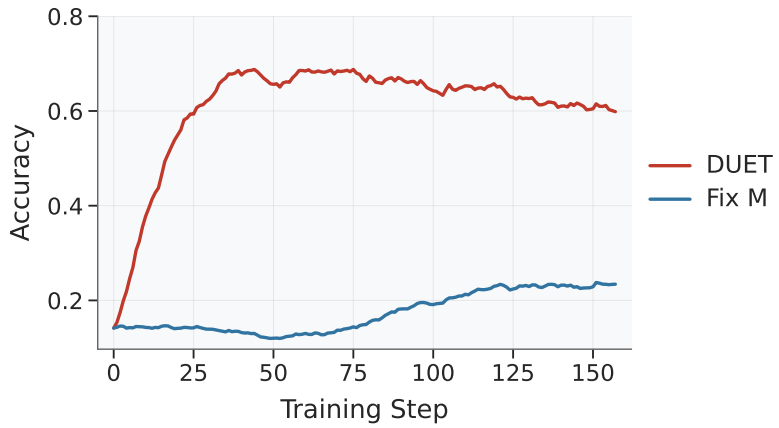


Figure 8: *Training dataset performance of DUET versus the fixed-$m$ variant during training. The small model is Qwen2.5-0.5B-Instruct and models are trained on the DeepScaleR dataset.*

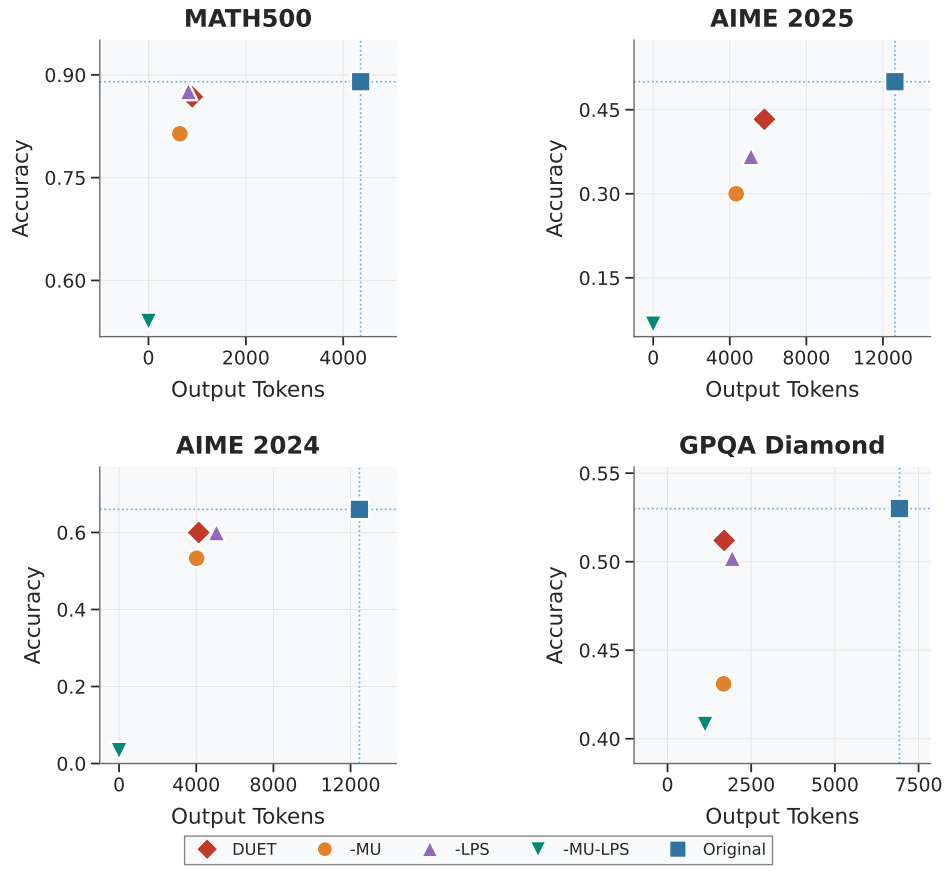


Figure 9: *The ablation results of marginal utility and length penalty schedule. –MU removes marginal-utility; –LPS removes the length-penalty schedule. Dashed blue vertical lines indicate the original large-model performance. The models are trained on MATH-LightEval dataset for 150 steps.*

### E.4.2 Trained on DeepScaleR

Figures 7 and 9 present results for models trained on the MATH-LightEval dataset. In Figure 10, we report the same ablation results when training on the DeepScaleR dataset. The overall obser-

t[!htb]

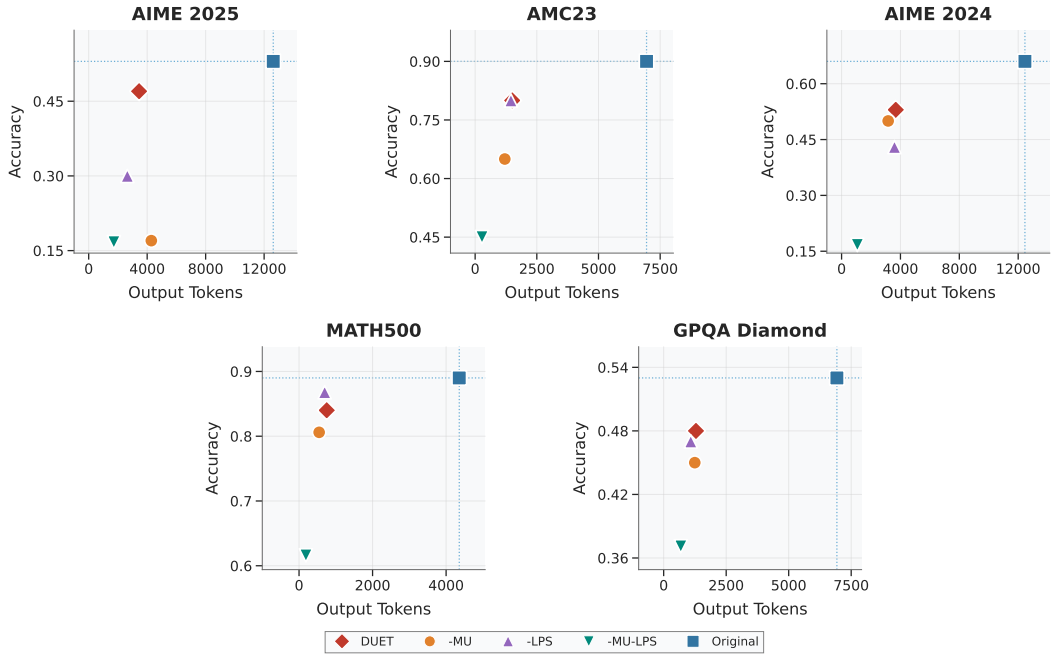


Figure 10: *The ablation results of marginal utility and length penalty schedule. –MU removes marginal-utility; –LPS removes the length-penalty schedule. Dashed blue vertical lines indicate the original large-model performance. The models are trained on DeepScaleR dataset for 150 steps.*

vations are consistent across datasets. Marginal utility plays a more important role than the length penalty schedule—using a constant length-penalty coefficient still yields good performance on most benchmarks except AIME2025. And when both marginal utility and the length penalty schedule are removed, training becomes unstable and often collapses, causing the large model to converge to producing empty or near-empty outputs.

### E.5 Ablation on Training Dataset

To verify that the observed behavior arises from the DUET framework and joint training rather than the dataset itself, we train the large model on the same training data using a standard reward objective with GRPO. The results are shown in Table 7. Training only the large model with a standard RL algorithm fails to produce concise reasoning, demonstrating that the ability to generate concise reasoning stems from the DUET framework and length-penalized joint training, rather than from the training dataset itself.

Table 7: *Large model performance across different training datasets using outcome reward only with GRPO. All models are evaluated at 100 training steps.*

| Training dataset | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| **Original large model** | 89.4 | 4358 | 90.0 | 6948 | 66.7 | 12458 | 53.3 | 12625 | 53.4 | 6928 |
| **MATH-LightEval** | 90.4 | 4702 | 90.0 | 7410 | 66.7 | 11233 | 60.0 | 13125 | 59.9 | 7345 |
| **DeepScaleR** | 89.0 | 4739 | 97.5 | 7108 | 73.3 | 12750 | 60.0 | 13635 | 52.8 | 7163 |
| **DAPO-MATH-17k** | 89.6 | 4789 | 90.0 | 7569 | 60.0 | 11716 | 60.0 | 12857 | 51.8 | 7186 |

### E.6 Ablation on Hyperparameters

We conduct ablation studies on the hyperparameters in the length penalty, including $B$ and $\lambda$ (with constant schedule).

#### E.6.1 The Value of $B$

The hyperparameter $B$ is the length normalization factor which controls the strength of length penalty. We evaluate three values: $B \in \{500, 1000, 2000\}$, where $B = 1000$ is the default used in the main paper. Results are shown in Table 8 (trained on MATH-LightEval) and Table 9 (trained on DeepScaleR). Smaller values of $B$ apply stronger length penalties. When $B = 2000$, task performance is slightly higher than $B = 500$ or $B = 1000$, but produces more tokens. Conversely, when $B = 500$, reasoning is most compressed but sacrifices more performance. Overall, DUET demonstrates stable performance across all tested values of $B$ in the range $[500, 2000]$.

Table 8: *Ablation study of hyperparameter $B$. Models are trained on the **DeepScaleR** dataset and evaluated at 100 training steps across multiple benchmarks. Acc. denotes accuracy (%) and Tokens denotes average output tokens per sample.*

| Value of B | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| $B = 500$ | 87.0 | 954 | 77.5 | 2054 | 50.0 | 4980 | 36.7 | 4876 | 45.2 | 1914 |
| $B = 1000$ | 88.0 | 1106 | 87.5 | 1801 | 63.3 | 5338 | 46.7 | 5888 | 47.2 | 2403 |
| $B = 2000$ | 88.8 | 1375 | 90.0 | 2256 | 53.3 | 6095 | 46.7 | 6624 | 51.3 | 2644 |

Table 9: *Ablation study of hyperparameter $B$. Models are trained on the **MATH** dataset and evaluated at 100 training steps across multiple benchmarks. Acc. denotes accuracy (%) and Tokens denotes average output tokens per sample.*

| Value of B | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| $B = 500$ | 86.4 | 776 | 85.0 | 1906 | 56.7 | 3472 | 33.3 | 5918 | 47.7 | 2221 |
| $B = 1000$ | 89.8 | 1110 | 85.0 | 1879 | 63.3 | 5014 | 46.7 | 6419 | 52.3 | 2690 |
| $B = 2000$ | 91.0 | 2019 | 90.0 | 3923 | 70.0 | 7424 | 46.7 | 9438 | 50.3 | 4894 |

#### E.6.2 Length Penalty Coefficient

In the main paper, we adopt an adaptive length penalty schedule. Here, we also compare it with constant penalty schedules using different coefficients. The results are summarized in Table 10. Larger values of $\lambda$ apply stronger length penalties, achieving more compressed reasoning. As shown in the ablations on marginal utility and length penalty schedule (Section E.4), the constant schedule achieves similar performance to the adaptive schedule, demonstrating DUET's robustness to the choice of length penalty coefficient. Furthermore, performance remains stable across different values of $\lambda$ in the range $[0.1, 2.0]$, which further confirms the stability of DUET.

### E.7 The Influence of KL Penalty in the Reward

In training, we include a KL penalty term in the reward for the large model, which is standard practice in the RL post-training paradigm. The KL penalty is computed as $\mathrm{KL}(M_t \| M_0)$, where $t$ denotes the training step and $M_0$ is the initial model. We conduct ablations on the effect of this KL term, with results shown in Tables 11 and 12. We find that including the KL penalty achieves better performance and training stability in DUET, while without the KL penalty, it still achieves similar performance. However, when removing marginal utility and switching from the adaptive length penalty schedule to

Table 10: *Ablation study of hyperparameter $\lambda$ when using **constant length penalty schedule**. Models are trained on the **MATH** dataset and evaluated at 100 training steps across multiple benchmarks. Acc. denotes accuracy (%) and Tokens denotes average output tokens per sample.*

| Value of $\lambda$ | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| Adaptive schedule | 89.8 | 1110 | 85.0 | 1879 | 63.3 | 5014 | 46.7 | 6419 | 52.3 | 2690 |
| Constant schedule: | | | | | | | | | | |
| $\lambda = 0.1$ | 90.4 | 2278 | 92.5 | 4063 | 66.7 | 9302 | 66.7 | 9500 | 48.7 | 5002 |
| $\lambda = 0.5$ | 90.6 | 1267 | 87.5 | 2623 | 60.0 | 5862 | 43.3 | 7304 | 48.7 | 2839 |
| $\lambda = 1.0$ | 90.2 | 1005 | 82.5 | 1768 | 53.3 | 4923 | 40.0 | 6584 | 47.7 | 2506 |
| $\lambda = 2.0$ | 87.0 | 871 | 87.5 | 1388 | 56.7 | 4713 | 53.3 | 5552 | 46.2 | 2000 |

a constant schedule, including the KL penalty can prevent training collapse to some extent, though performance remains significantly worse than DUET with default settings.

Table 11: *Effect of KL penalty on DUET performance. Models are trained on the MATH-LightEval dataset and evaluated at 100 training steps. Acc. denotes accuracy (%) and Tokens denotes average output tokens per sample.*

| | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| w/ KL penalty | 89.8 | 1110 | 85.0 | 1879 | 63.3 | 5014 | 46.7 | 6419 | 52.3 | 2690 |
| w/o KL penalty | 89.2 | 1174 | 90.0 | 2404 | 53.3 | 6084 | 43.3 | 7295 | 51.8 | 2624 |

Table 12: *Effect of KL penalty on the ablated variant without marginal utility and adaptive length penalty schedule. Models are trained on the MATH-LightEval dataset and evaluated at 150 training steps. Acc. denotes accuracy (%) and Tokens denotes average output tokens per sample.*

| | MATH500 | | AMC23 | | AIME24 | | AIME25 | | GPQA Diamond | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| w/ KL penalty | 84.2 | 684 | 77.5 | 1394 | 40.0 | 3945 | 30.0 | 4116 | 41.1 | 1688 |
| w/o KL penalty | 54.0 | 0 | 27.5 | 0 | 3.3 | 0 | 6.7 | 0 | 47.7 | 1121 |

### E.8 Effect of Clipping the Length Penalty Coefficient

In our algorithm, we apply a clipping mechanism on $\lambda$ to prevent explosive length penalties:

$$\lambda_t \leftarrow \max\{0, \min\{\lambda_t, 1\}\}.$$

Here, we explore the effect of this clipping mechanism. Figure 11 shows task performance and output tokens throughout training. In the early training stage, clipping has minimal effect since $\lambda$ has not yet grown to large values. In the later training stage, clipping helps prevent training collapse and maintains good performance compared to the unclipped variant.

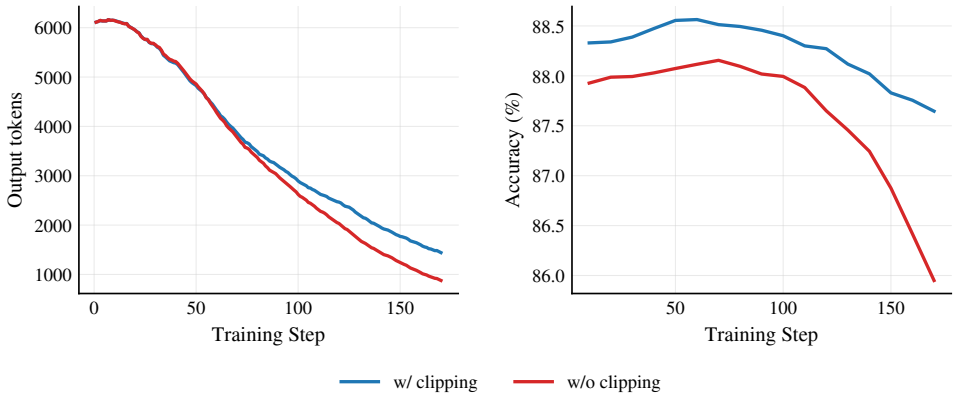


Figure 11: *Comparison of training with and without $\lambda$ clipping. Models are trained on the DeepScaleR dataset and evaluated on MATH500 benchmark.*