Gold-Switch: Training-Free Superposition of Slow- and Fast- Thinking LLMs

Anonymous ACL submission

Abstract

Large Reasoning Models (LRMs) excel at structured tasks by emulating deliberate human reasoning, but they often suffer from overthinking, degrading performance and wasting resources. One possible baseline is to deploy both LLM and LRM, then route input by predicting whether it requires reasoning and may cause overthinking. However, deploying multiple models can be costly or impractical. We propose a **superposed deployment** strategy that enables switching a single model between slow and fast thinking using a lightweight, **training-free** regulation module. Our central finding is surprisingly simple: a low-rank approximation of the parameter difference between an LRM and its base LLM is already sufficient to suppress overthinking. We further introduce a layer-wise rank selection method based on the cumulative energy of singular values to identify low-rank projections that regulate reasoning "just right." Our module is architecturally identical to LoRA-style adapters, making it easy to integrate with existing deployment pipelines. Experiments show that our approach brings LRMs closer to the Goldilocks zone, delivering faster inference with negligible accuracy loss while requiring substantially less memory overhead than router-based solutions.

1 Introduction

Large reasoning models (LRMs), such as DeepSeek-R1 (DeepSeek-AI et al., 2025) and QwQ (Qwen, 2025), have recently emerged as a powerful class of reasoning-oriented models. Built on top of base large language models (LLMs) and further trained with extensive chain-of-thought (CoT) supervision, LRMs achieve strong performance on complex reasoning tasks. LRMs represent a new generation of language models optimized for "slow thinking"—producing elaborate intermediate steps that mimic human reasoning.

However, overthinking (Sui et al., 2025; Chen et al., 2025; Cuadron et al., 2025) has recently been identified as a central failure mode of LRM: they often generate unnecessarily long or repetitive reasoning paths even for relatively simple problems (Sui et al., 2025). Such behavior not only increases inference cost, but can also hurt task performance itself (Cuadron et al., 2025). This highlights the need to switch between slow- and fast- thinking in LRMs.

One possible direction is to train a single model to support both modes through prompting, as in Qwen3 (Yang et al., 2025). However, due to their limited performance, recent model releases have increasingly reverted to providing slow- and fast-thinking variants separately. For example, Qwen3-2507, Qwen3-VL, Qwen3-Next (Yang et al., 2025), Qwen3-Omni (Xu et al., 2025a), and Qwen3-Max (Team, 2025) are all released as separate reasoning and non-reasoning models.

An alternative direction is to deploy both an LRM and its corresponding LLM, and route each query to the more appropriate model (Chuang et al., 2025b,a; Aytes et al., 2025; Ong et al., 2025). For instance, RouteLLM (Ong et al., 2025) trains a lightweight classifier to estimate input difficulty and dispatch easy queries to the fast model while reserving the reasoning model for hard ones. However, deployment of both LRM and LLM can be costly in practice.

In this paper, we propose a **training-free superposed deployment** strategy that achieves a "Goldilocks" balance between slow-thinking LRMs and fast-thinking LLMs, while keeping deployment lightweight. Our starting point is a simple but surprising observation: we find that *simply extracting a low-rank component from the parameter difference* between an LRM and its base LLM can suppress excessive reasoning. This result is notable because it suggests that overthinking is encoded in a compact and removable subspace, rather than being inseparably entangled with reasoning ability. Specifically, for linear layers in LRM and LLM,

denoted as W_R and W_B, instead of full difference $\Delta W = W_B - W_R$, we hypothesize the existence of low-rank component L such that $L + W_R$ unlearns overthinking only, by low intrinsic dimensionality (Li et al., 2018).

This formulation is appealing for both scientific and practical reasons. Scientifically, it reveals that a substantial portion of overthinking can be removed through a compact low-rank correction. Practically, it is entirely training-free, and the resulting module is smoothly compatible with existing LoRA implementations and deployment pipelines.

We derive L by low-rank approximating ΔW with rank r. To choose r effectively, we introduce an energy-based, layer-wise rank selection method inspired by (Xu et al., 2020). This layer-wise strategy yields both faster inference and better task performance than naïvely applying a uniform fixed rank ratio to all layers.

The key contributions of Gold-Switch are:

- We propose Gold-Switch, a practical **training-free superposed deployment** framework that balances slow-thinking LRMs and fast-thinking LLMs without requiring two full models at serving time.
- Surprisingly, a simple training-free *low-rank approximation of the parameter difference* between an LRM and its base LLM is already sufficient to reduce overthinking. This makes Gold-Switch to be simply compatible with existing LoRA implementations and deployment pipelines.
- Experiments with QwQ-32B and DeepSeek-R1-Distill-32B on ASDIV, GSM8K, AIME, and GPQA show that our method moves LRMs closer to the Goldilocks zone: it achieves up to 2.7× speedup with negligible performance drop, while reducing additional GPU memory by 9× compared with router-based solutions.
- We release our code and models to the public,[1] enabling further research in this area.

2 Related Works

2.1 Overthinking in LRMs

LRMs like DeepSeek-R1 and QwQ reportedly excel at complex multi-step tasks through chain-of-thought (CoT) training. Meanwhile, they may generate long and repeated reasoning paths even for simple problems (Sui et al., 2025; Chen et al., 2025; Cuadron et al., 2025). Chen et al. (2025) define overthinking as the inefficient use of computation, where models spend too much effort on simple tasks without much accuracy gain. For example, they show that even for a simple question like "What is the answer of 2 plus 3?", LRM such as QwQ-32B-Preview requires 901 tokens to reach the answer '5', while an LLM from the same family, Qwen2.5-72B, requires only 7 tokens.

2.2 Reducing Overthinking

2.2.1 Training-Based

Reinforcement learning or supervised fine-tuning approaches, by rewarding concise, accurate answers and penalize long or incorrect ones, can encourage the model to generate shorter reasoning paths (Liu et al., 2024; Yu et al., 2024; Cui et al., 2025; Munkhbat et al., 2025; Ma et al., 2025b; Kang et al., 2025; Shen et al., 2025; Aggarwal and Welleck, 2025; Qu et al., 2025; Arora and Zanette, 2025; Team et al., 2025; Luo et al., 2025; Yeo et al., 2025; Xia et al., 2025). For example, models like Kimi k1.5 (Team et al., 2025) apply length-aware rewards during their RL process to encourage shorter responses. It also utilizes rejection sampling, generating and evaluating candidate responses to guide efficient generation. Other methods create length-optimized datasets by removing redundant steps or tokens (Xia et al., 2025; Kang et al., 2025), or use self-training to sample shorter paths as pseudo-labels (Munkhbat et al., 2025).

Models like Qwen3 (Yang et al., 2025) are trained to allow both slow- and fast- thinking by predefined prompts. However, due to their limited performance, the recent trend is back to releasing slow- and fast- thinking models separately (Yang et al., 2025; Xu et al., 2025a; Team, 2025).

Distinction While these methods usually require extensive data- and compute- resources, Gold-Switch is training-free, removing such a burden.

2.2.2 Training-Free

Routing-based Routing-based method classifies the input prompt to route to an appropriate model (Chuang et al., 2025b,a; Aytes et al., 2025; Ong et al., 2025). For example, RouteLLM (Ong et al., 2025) trains a lightweight classifier to classify whether the input prompt is relatively hard or easy,

[1] https://anonymous.4open.science/r/GoldSwitch-377D/

then uses this classifier to route input prompts.

Prompt-based Prompt-based approaches adjust the input format to limit reasoning steps (Renze and Guven, 2024; Xu et al., 2025b; Ma et al., 2025a). For example, minimal prompting strategies like Nothinking (Ma et al., 2025a) encourage direct answers without intermediate steps, by early closure of <think> tag with </think> (Figure 6).

Interpolation-based Interpolation-based methods merges the weights of LRM and LLM (Wu et al., 2025). For example, MI (Wu et al., 2025) linearly interpolates the weight of LRM and LLM, similarly to Task Arithmetic (Ilharco et al., 2023). By adjusting the interpolation ratio, MI can balance between the reasoning capabilities of LRM and the efficiency of LLM.

Distinction While the routing-based method requires deploying both LRM and LLM, we propose a superposed deployment strategy. Moreover, our method is orthogonal to prompt-based methods or interpolation-based methods, which can be combined to further enhance the efficiency.

2.3 Low-Rank Representation

Parameter-Efficient Fine-Tuning (PEFT) methods, especially LoRA (Hu et al., 2022), have become a standard for adapting large language models (LLMs) to downstream tasks. These methods rely on the hypothesis that the parameter changes required to learn a specific task or capability have a much lower intrinsic dimension than the full parameter space (Hu et al., 2022; Aghajanyan et al., 2021).

MoDeGPT (Lin et al., 2025) modularizes model parameters by task and recombines them, demonstrating that LLMs can be compressed and reused efficiently without additional training.

Distinction Gold-Switch demonstrates that unlearning reasoning capacities can also be captured in a compact, low-dimensional subspace without additional training.

3 Proposed Method

3.1 Preliminaries

3.1.1 LRMs

To enhance the reasoning capabilities of large language models (LLMs), researchers have developed various reasoning-oriented models, such as

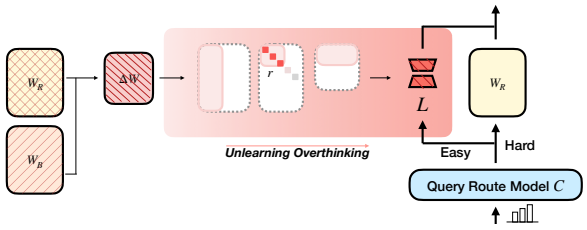

Figure 1: Overview of Gold-Switch. Gold-Switch extracts the overthinking parameters from the reasoning-enhanced model using low-rank approximation. It then dynamically adjusts the reasoning capabilities based on the difficulty of the input, allowing for efficient reasoning without retraining.

QwQ (Qwen, 2025) and DeepSeek-R1 (DeepSeek-AI et al., 2025). These models are typically trained on a base model (e.g., Qwen2.5-32B) with extensive chain-of-thought (CoT) supervision, enabling them to perform complex reasoning tasks.

Formally, we denote the base model as W_B. By updating this model we obtain a reasoning-enhanced model W_R:

$$W_R = W_B + \Delta W \tag{1}$$

where ΔW represents the parameter update needed for complex reasoning.

However, this reasoning-enhanced model often suffers from overthinking, where it generates unnecessarily long or convoluted reasoning paths (Sui et al., 2025). This behavior can lead to inefficiencies and even degradation of task performance (Cuadron et al., 2025).

3.1.2 Baseline: Routing Input Queries to LRM and LLMs

To avoid overthinking, we can adopt RouteLLM (Ong et al., 2025) to route the input query to the appropriate model.

Formally, given a lightweight classifier C that predicts the difficulty of the input, RouteLLM routes the input x to W_R if $C(x) \leq t$, and to W_B otherwise. In theory, the classifier C can be either black-boxed, or a BERT-based classifier trained on a preference dataset.

However, this solution blindly deploys both W_R and W_B, which can be costly in practice.

3.2 Gold-Switch: Training-Free Superposition Between LRM and LLM

Figure 1 illustrates the overall flow of Gold-Switch. We start by defining L, for unlearning only overthinking from LRM (§3.2.1). This lightweight mod-

ule could be deployed alongside W_R, with minimal overhead. Then we introduce how to superpose L with the given LRM, for each given query (§3.2.2).

3.2.1 Training-Free L to Unlearn Only Overthinking from LRM

L as a low-rank approximation of ΔW Related works (Section 2.2.1) succeeded to unlearn overthinking, while keeping reasoning capability. This would mean that overthinking has lower intrinsic dimensionality than the reasoning capability.

This leads to the key intuition– constructing L that **unlearns overthinking** while preserving reasoning is possible. In other words, unlike $\Delta W = W_B - W_R$, which captures all finetuned capabilities including reasoning, L focuses solely on overthinking, implying lower intrinsic dimensionality than ΔW (Li et al., 2018). With low intrinsic dimensionality, L could be low-rank decomposed, enabling lightweight deployment alongside W_R.

Our second key intuition is that L and ΔW have similar direction, both shifting W_R to W_B. Therefore we hypothesize that L can be a low-rank approximation of ΔW. Our experiments (Section 4) will reveal that when we low-rank approximate the full unlearning matrix ΔW, slow-thinking unlearning is captured first in these low-rank approximations.

Formally, we decompose ΔW as :

$$\Delta W \approx USV^T \tag{2}$$

where $U \in \mathbb{R}^{N \times \min(N,M)}$, $S \in \mathbb{R}^{\min(N,M) \times \min(N,M)}$ and $V \in \mathbb{R}^{M \times \min(N,M)}$. Following the Eckart-Young-Mirsky theorem (Eckart and Young, 1936), we can approximate ΔW with a low-rank matrix by selecting the top r singular values from S. This allows us to select the essential dimensions of the delta matrix ΔW. Formally, we get the low-rank approximation of ΔW as follows:

$$\hat{U} = U[:, :r], \quad \hat{S} = S[:r, :r], \quad \hat{V} = V[:, :r] \tag{3}$$

Finally, we get the Gold-Switch L as follows:

$$L = L_U \cdot L_V, \quad L_U = \hat{U} \cdot \sqrt{\hat{S}}, \quad L_V = \sqrt{\hat{S}} \cdot \hat{V}^T \tag{4}$$

The resulting L can selectively remove unnecessary overthinking components while preserving core reasoning. This approach is crucial for retaining the reasoning of large LRMs while enhancing efficiency during inference.

Energy-based r selection Now, the problem narrows down to how to effectively select r for each layer. A blind selection of a high rank would make L too similar to ΔW, causing the unlearning process to strip away not only overthinking components but also critical reasoning abilities.

To address this, inspired by Xu et al. (2020), we employ an energy-based selection mechanism driven by the energy percentage e. According to matrix perturbation theory (Stewart and Sun, 1990), the difference between two matrices is bounded by the differences in their singular values, as expressed by the inequality:

$$\|B - A\|_F^2 \geq \sqrt{\sum_i |s_{b,i} - s_{a,i}|^2} \tag{5}$$

where $s_{b,i}, s_{a,i}$ denotes the singular values of B and A respectively. Applying this to our setting, we derive the lower bound of the error between L and ΔW:

$$\|L - \Delta W\|_F^2 \geq \sqrt{\sum_{i=r}^{\min(N,M)} s_i^2} \tag{6}$$

where s_i denotes the singular values of ΔW. This lower bound serves as an indicator to prevent L from becoming too close to ΔW, ensuring that essential reasoning is not inadvertently discarded. For each layer, we determine the optimal r by regulating the right-hand term of Equation 6, preventing it from diminishing excessively.

To this end, we introduce the reversed cumulative energy $e(r)$, defined as follows:

$$e(r) = \frac{\sum_{i=r}^{\min(N,M)} s_i^2}{\sum_{i=1}^{\min(N,M)} s_i^2} \tag{7}$$

We then select r as the minimum rank that maintains $e(r)$ above an empirically tuned threshold τ. This ensures sufficient information retention while eliminating unnecessary overthinking components.

3.2.2 Hard and Soft Superposition

Now the remaining part is how to superpose slow- and fast-thinking LRM with L and W_R. We introduce hard or soft superposition.

	QwQ / Qwen2.5-Instruct				DS-R1-32B / Qwen2.5				Q3T-4B / Q3I-4B / Q3-4B			
	GSM8K	time	ASDIV	time	GSM8K	time	ASDIV	time	GSM8K	time	ASDIV	time
LRM	96.2%	1.00×	87.6%	1.00×	88.7%	1.00×	83.7%	1.00×	95.2%	1.00×	89.5%	1.00×
LLM	94.2%	0.17×	87.6%	0.22×	85.5%	0.91×	80.7%	1.03×	90.4%	0.30×	77.6%	0.40×
RouteLLM	93.8%	0.61×	88.3%	0.37×	88.6%	0.84×	83.0%	0.81×	91.4%	0.47×	78.2%	0.53×
NoThinking	95.9%	0.98×	87.2%	1.06×	89.2%	0.54×	87.1%	0.44×	94.7%	1.06×	89.6%	1.07×
Training-Based	-	-	-	-	-	-	-	-	92.0%	0.48×	88.4%	0.81×
Gold-Switch	95.8%	0.37×	87.5%	0.39×	92.3%	0.85×	86.0%	0.82×	93.9%	0.47×	88.5%	0.56×

Table 1: GSM8K and ASDIV performance of Gold-Switch and baselines on QwQ-32B, DeepSeek-R1-Distill-Qwen-32B (DS-R1-32B), and Qwen3-Thinking-4B (Q3T-4B). Q3I-4B and Q3-4B denote Qwen3-Instruct-4B and Qwen3-4B, respectively.

Hard Superposition Given a classifier C that predicts the difficulty of the input (§3.1.2), we can switch between slow-thinking and fast-thinking based on the predicted difficulty:

$$W'_R = \begin{cases} W_R + L & \text{if } C(x) \leq t \\ W_R & \text{if } C(x) > t \end{cases} \quad (8)$$

where W'_R is the weight we use for the inference of input x, and t is a threshold that determines whether the input is considered easy or hard. This enables us to deal with the complex tasks that require sophisticated reasoning capabilities via slow-thinking, while still benefiting from the reduced overthinking for easier tasks via fast-thinking.

Soft Superposition Moreover, we investigate soft superposition to continuously adjust the weight of Gold-Switch based on the difficulty of the input, by generalizing the concept of routing in Ong et al. (2025). Formally, the softly superposed W'_r is defined as:

$$W'_r = W_R + \alpha L \quad (9)$$

where α is a scaling factor that controls the strength of Gold-Switch. However, we find that scaling Gold-Switch may increase the overthinking in some cases, making it harder to control the reasoning capabilities (Figure 3). Therefore we use the hard superposition for the final experiments.

4 Experiments

Models We employ three open-sourced LRMs: QwQ-32B (Qwen, 2025), DeekSeek-R1-Distill-Qwen-32B (DeepSeek-AI et al., 2025), and Qwen3-4B-Thinking-2507 (Yang et al., 2025). As the base LLMs, we utilize Qwen2.5-32B-Instruct, Qwen2.5-32B (Qwen et al., 2025), and Qwen3-4B-Instruct-2507, respectively.

Tasks and Datasets We mainly evaluate with ASDIV (Miao et al., 2020) and GSM8K (Cobbe et al., 2021), where overthinking is reported to happen (Chen et al., 2025). As challenging reasoning problems where slow thinking is favored, we employ AIME2024. We also include GPQA (Rein et al., 2024), the non-math task Chen et al. (2025) employed. All evaluations are done in a 0-shot manner.

Implementation Details For AIME2024 and GPQA, we employ OPEN-R1.[2] To evaluate ASDIV and GSM8K, we extend LM-EVALUATION-HARNESS[3] (Gao et al., 2021) to use the suggested format by DeepSeek and Qwen. Following the suggestion, we use temperature of 0.6, topP of 0.95, topK of 30, and minP of 0. For Qwen3 models, we use topK of 20, and for Qwen3-4B-Instruct-2507 and the non-thinking mode of Qwen3-4B, we use temperature of 0.7, topP of 0.8. We use max new tokens of 32K, while for ASDIV and GSM8K evaluation of DeepSeek-R1-Distill-Qwen-32B and Qwen2.5-32B, we use max new tokens of 2K, to align with the default configuration of huggingface Qwen2.5-32B. We use $t = 0.78$, and $\tau = 0.6$.[4] For classificer C, we use a pretrained BERT-based model introduced in Ong et al. (2025), which is claimed to generalize over diverse settings. All experiments are conducted on 2×A6000.

Comparisons We compare the following methods:

- **LRM** breaks down structured problems into multiple steps for deeper understanding but

[2] github.com/huggingface/open-r1

[3] github.com/neuralmagic/lm-evaluation-harness/tree/llama_3.1_instruct

[4] We conducted small grid search once on QwQ-32B with 100 samples of GSM8K to choose this value, and used over all models and tasks, as elaborated in Appendix B.

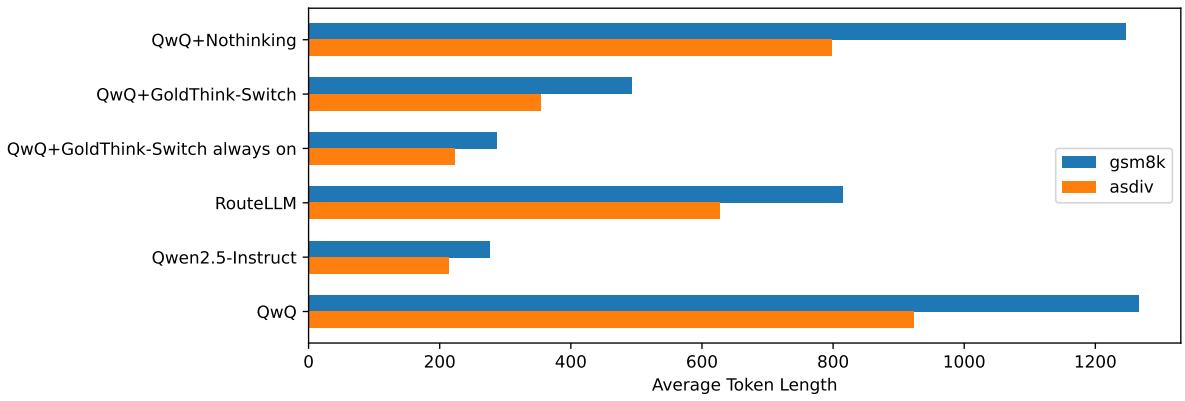

Figure 2: Average token length of GSM8K and ASDIV on QwQ

often overthink even on simple tasks.

- **LLM** can produce concise responses with low computation, but often lacks deep logical reasoning for complex problems.
- **RouteLLM** (Ong et al., 2025) selects different language models (such as LLM or LRM) depending on the difficulty of the input.
- **Nothinking** (Ma et al., 2025a) skips the reasoning process by pre-filling an assistant message with '</think>'. It also uses a Token Budget (set to 3500 in our tests) to force short responses, reducing overthinking.
- **MI** (Wu et al., 2025) interpolates the weights of LRM and LLM, with a controllable interpolation ratio λ to balance reasoning capability and efficiency. We denote this baseline as MI-λ.
- **Gold-Switch** is our proposal.

4.1 Experimental Results

RQ1: Gold-Switch Effectively Reduces Overthinking in LRMs Table 1 shows the results of Gold-Switch on QwQ-32B, DeepSeek-R1-Distill-Qwen-32B, and Qwen3-4B. Taking QwQ-32B as an example, on GSM8K, while QwQ-32B achieves higher accuracy compared with Qwen2.5-32B-Instruct, it takes almost 6× longer time to finish the same task. With Gold-Switch, we significantly reduce the performance gap between LRM and LLM, with a moderate time cost. Gold-Switch sometimes achieves higher accuracy than LRM by reducing overthinking (GSM8K on DS-R1-32B), which is consistent with Ma et al. (2025a). On ASDIV, we can observe a similar trend– the performance gap is almost similar, while the inference time is drastically reduced.

Figure 2 shows the average token length of GSM8K and ASDIV on QwQ-32B. We can observe that Gold-Switch can significantly reduce the average token length. More detailed statistics are provided in the Appendix.

When compared to an existing router-based solution RouteLLM, Gold-Switch achieves similar or better performance, while drastically decreasing the GPU memory usage (Table 5). For example, on GSM8K, Gold-Switch improves the accuracy by 1.7-3.7% (Table 1), while requiring similar or only 0.6× of time to finish. When compared to a prompt-based baseline NoThinking, while NoThinking often fails to decrease the generation time, Gold-Switch always succeeds to decrease the time drastically, up to 2.7×.

Finally, when compared to a training-based baseline (Qwen3-4B), which is trained to have slow- or fast- thinking switch, training-free superposition of Qwen3-4B-Thinking and Qwen3-4B-Instruct by Gold-Switch is much better– GSM8K accuracy is 1.9% higher, while ASDIV speed is 30% faster. The trend is similar in AIME and GPQA as well (Table 2,3).

RQ2: Gold-Switch Is Orthogonal to Existing Training-Free Methods Table 3 shows that Gold-Switch can be combined with interpolation-based method MI to further improve the efficiency. For example, while MI-0.8 or MI-0.5 fails to reduce the inference time of GPQA, MI-0.8+Gold-Switch can reduce the inference time to 0.7×, with marginal performance drop. We stress that such inference time reduction is not achievable with con-

	QwQ-32B / Qwen2.5-Instruct				DS-R1-32B / Qwen2.5			
	AIME	time	GPQA	time	AIME	time	GPQA	time
LRM	66.70%	1.00x	63.10%	1.00x	60.00%	1.00x	65.10%	1.00x
LLM	13.30%	0.07x	45.50%	0.01x	6.70%	0.77x	39.40%	0.26x
RouteLLM	66.70%	1.00x	62.10%	0.69x	60.00%	1.00x	55.60%	0.70x
Gold-Switch	66.70%	1.00x	61.10%	0.69x	60.00%	1.00x	56.60%	0.79x

Table 2: AIME, GPQA performance of Gold-Switch and baselines on QwQ-32B, DeepSeek-R1-Distill-Qwen-32B (DS-R1-32B).

	AIME	time	GPQA	time
LRM (Q3T-4B)	72.22%	1.00x	66.86%	1.00x
MI-0.8	84.44%	0.61x	66.29%	0.94x
LLM (Q3I-4B)	64.44%	0.56x	44.49%	0.02x
Training (Q3-4B)	76.67%	0.54x	47.80%	0.58x
RouteLLM	72.22%	1.00x	55.43%	0.60x
GoldSwitch	72.22%	1.00x	62.89%	0.70x
MI-0.5	76.67%	0.43x	62.36%	0.90x
MI-0.8+GoldSwitch	84.44%	0.61x	64.66%	0.73x

Table 3: AIME, GPQA performance of Gold-Switch and baselines on Qwen3-Thinking-4B (Q3T-4B). Q3I-4B and Q3-4B denote Qwen3-Instruct-4B and Qwen3-4B, respectively.

	GSM8K	time	ASDIV	time
Gold-Switch	92.3%	0.85×	86.0%	0.82×
NoThinking (NT)	89.2%	0.54×	87.1%	0.44×
Gold-Switch + NT	89.9%	0.57×	86.6%	0.48×

Table 4: GSM8K, ASDIV performance of Gold-Switch, NoThinking, and their combination on DeepSeek-R1-Distill-Qwen-32B.

Model size	Q3T-4B	QwQ-32B
Build GPU mem (GB)	1	4
Build GPU hours (h)	0.06	0.67
Serve size of Gold-Switch	1.17×	1.11×
Serve size of RouteLLM	2.00×	2.00×

Table 5: Required GPU memory (GB) and hours (h) to build Gold-Switch with one A6000, additional GPU memory (percentage) required for serving Gold-Switch and RouteLLM compared to the base models.

	Total TTFT (s)	Routing Time (s)
GSM8K	0.2348	0.0098
ASDIV	0.2125	0.0070

Table 6: Time comparison between the total time-to-first-token (TTFT) and the routing time on QwQ-32B with Gold-Switch.

trolling the parameters with MI only, since MI-0.5 shows lower performance yet slower than MI-0.8+Gold-Switch. Table 4 shows that Gold-Switch can be combined with prompt-based method No-Thinking as well, to explore better performance with reduced latency.

RQ3: Gold-Switch Is Cost-Efficient to Build and Serve Table 5 shows that the building cost of Gold-Switch is negligible– this training-free preprocessing step requires less than 1 hour with only 4GB of GPU memory. The serve size rows show that Gold-Switch significantly saves the deployment cost. While the routing-based solution, RouteLLM additionally requires 1× compared with serving one LRM, Gold-Switch only requires around 0.11×, which is 9× smaller.

Finally, Table 6 shows that the routing time is negligible as well. Even if we consider the time-to-first-token latency, routing time takes only 0.04×.

4.2 Ablation Studies

Switching On/Off Gold-Switch Is Important Table 7 shows the performance of Gold-Switch by enabling/disabling the switching behavior. While always enabling Gold-Switch accelerates the inference further, it significantly reduces the performance. In contrast, enabling the switching mechanism can successfully maintain the reasoning capability in the complex tasks as well.

Energy-Based Rank Selection Is Better than Fixed Ratio Table 8 shows the performance of Gold-Switch with different rank selection methods on QwQ-32B. We can observe that energy-based rank selection can obtain not only a smaller size compared to the fixed ratio method, but also achieves better performance. This emphasizes the importance of our layer-wise energy-based method to keep distance from ΔW (Equation 6).

Soft Switching Fails to Reduce Overthinking Figure 3 illustrates the relative average token length

	QwQ / Qwen2.5-Instruct				DS-R1-32B / Qwen2.5			
	AIME	time	GPQA	time	AIME	time	GPQA	time
Gold-Switch	66.70%	1.00x	61.10%	0.69x	60.00%	1.00x	56.60%	0.79x
- Switching (Always On)	16.70%	0.80x	42.90%	0.02x	23.30%	0.83x	43.40%	0.57x

Table 7: AIME, GPQA performance of Gold-Switch and baselines by enabling/disabling the switching behavior.

Rank r selection	relative size of L	GSM8K	time(h)	ASDIV	time (h)
fixed ratio	14%	90.8%	6.2	85.7%	7.5
energy-based	11%	92.3%	3.9	86.0%	6.3

Table 8: The size of L and the performance of GSM8K and ASDIV on DeepSeek-R1-Distill-Qwen-32B

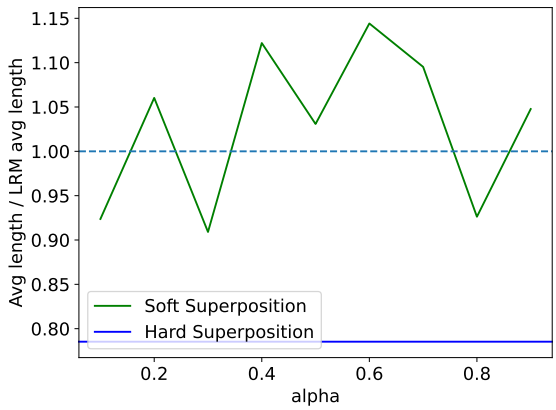

Figure 3: Average token length ratio of ASDIV on DeepSeek-R1-Distill-Qwen-32B.

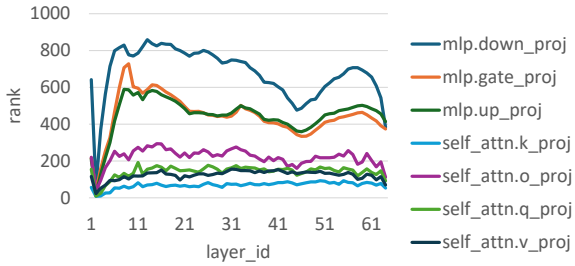

Figure 4: Rank distribution of Gold-Switch over layer ids (x-axis) on QwQ.

ratio of hard and soft superposition compared to LRM (DeepSeek-R1-Distill-Qwen-32B), using 90 samples from ASDIV. As α varies between 0 and 1, the average token length of output from $W_R + \alpha L$ may increase, suggesting that the soft switching mechanism is ineffective in mitigating overthinking. In contrast, for easy problems, hard superposition effectively reduces token length.

4.3 Analysis

Case Study Figure 5, 6 and 7 in the Appendix show a case study for the same question "Sophia and Rose went together to the market to buy onions and potatoes. Rose bought 4 times the number of onions and potatoes Sophia bought. If Rose bought 12 onions and 4 potatoes, how many onions and potatoes in total did Sophia buy at the market?".

First, the base LRM, QwQ-32B, overthinks over and over again– repeatedly obtaining the same answer "4" for 10 times– resulting in 1353 tokens to reach the final answer. NoThinking (Ma et al., 2025a), by closing the </think> beforehand, reduces the thinking process, but fails to avoid repeated answer of "4" for 6 times, resulting in 984 tokens to reach the final answer.

In contrast, Gold-Switch directly dives into a single CoT process, without any repetitive answering, resulting in 328 tokens, almost 4× reduction of token length compared to the base LRM. Moreover, it is interesting to observe that the LRM seems to forget the <think> token, not reacting at all, even does not close with </think> token. The output token length is almost similar to the fast-thinking LLM, Qwen2.5-32B-Instruct (Figure 8).

Rank Distribution over Layers Figure 4 shows the rank distribution of Gold-Switch over layer ids on QwQ-32B. We can observe that specific layers, such as the earliest layers, requires lowest rank.

5 Conclusion

Our work introduces lightweight, training-free LoRA modules that suppress overthinking, enabling LLMs to dynamically modulate their inference budget based on input complexity.

Limitations

Gold-Switch leverages classifiers to route hard or easy tasks from existing baselines (Ong et al., 2025). Therefore we follow the limitations from

those baselines, such as interpretability, or scarcity of gold-labeled dataset to train such classifiers.

Gold-Switch assumes access to both the base and fine-tuned weights, which may not be feasible for closed-source or commercial LLMs. However, our target is making local deployment more efficient with superposed deployment, thus supporting the closed-source models is out of our scope.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *Preprint*, arXiv:2503.04697.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Daman Arora and Andrea Zanette. 2025. Training Language Models to Reason Efficiently. *Preprint*, arXiv:2502.04463.

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-Thought: Efficient LLM Reasoning with Adaptive Cognitive-Inspired Sketching. *Preprint*, arXiv:2503.05179.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do NOT Think That Much for 2+3=? On the Overthinking of Long Reasoning Models. In *Forty-Second International Conference on Machine Learning*.

Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. 2025a. Confident or Seek Stronger: Exploring Uncertainty-Based On-device LLM Routing From Benchmarking to Generalization. *Preprint*, arXiv:2502.04428.

Yu-Neng Chuang, Helen Zhou, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. 2025b. Learning to Route LLMs with Confidence Tokens. *Preprint*, arXiv:2410.13284.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *Preprint*, arXiv:2110.14168.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The Danger of Overthinking: Examining the Reasoning-Action Dilemma in Agentic Tasks. *Preprint*, arXiv:2502.08235.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. 2025. Stepwise Perplexity-Guided Refinement for Efficient Chain-of-Thought Reasoning in Large Language Models. In *arXiv.Org*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Preprint*, arXiv:2501.12948.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation. Zenodo.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3oT: Generating Shorter Chain-of-Thought Without Compromising Effectiveness. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24312–24320.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2025. MoDeGPT: Modular Decomposition for Large Language Model Compression. *Preprint*, arXiv:2408.09632.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-Pruner: Length-Harmonizing Fine-Tuning for O1-Like Reasoning Pruning. *Preprint*, arXiv:2501.12570.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. Reasoning Models Can Be Effective Without Thinking. *Preprint*, arXiv:2504.09858.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. CoT-Valve: Length-Compressible Chain-of-Thought Tuning.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-Training Elicits Concise Reasoning in Large Language Models. *Preprint*, arXiv:2502.20122.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. Optimizing test-time compute via meta reinforcement finetuning. In *Workshop on Reasoning and Planning for Large Language Models*.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 Technical Report. *Preprint*, arXiv:2412.15115.

Team Qwen. 2025. QwQ-32B: Embracing the power of reinforcement learning.

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

Matthew Renze and Erhan Guven. 2024. The Benefits of a Concise Chain of Thought on Problem-Solving in Large Language Models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models. *Preprint*, arXiv:2503.04472.

Gilbert W Stewart and Ji-guang Sun. 1990. Matrix perturbation theory. *(No Title)*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *Preprint*, arXiv:2501.12599.

Qwen Team. 2025. Qwen3-max: Just scale it.

Taiqiang Wu, Runming Yang, Tao Liu, Jiahao Wang, and Ngai Wong. 2025. Revisiting Model Interpolation for Efficient Reasoning.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. TokenSkip: Controllable Chain-of-Thought Compression in LLMs. *Preprint*, arXiv:2502.12067.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, and 19 others. 2025a. Qwen3-Omni Technical Report. *Preprint*, arXiv:2509.17765.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025b. Chain of Draft: Thinking Faster by Writing Less. *Preprint*, arXiv:2502.18600.

Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. 2020. TRP: Trained Rank Pruning for Efficient Deep Neural Networks. *Preprint*, arXiv:2004.14566.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 Technical Report. *Preprint*, arXiv:2505.09388.

Edward Yeo, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long CoT reasoning in LLMs. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Ping Yu, Jing Xu, Jason E Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

A Token-Length Statistics

Table 9, 10 show the detailed statistics of token-length distribution.

B Ablation Study

Effect of Thresholds We conducted ablation study to investigate how the performance of Gold-Switch varies with different values of the difficulty threshold t and energy threshold τ.

$\tau \setminus t$	0.76	0.77	0.78	0.79
0.6	93%	92%	**95%**	91%
0.7	93%	93%	93%	92%

Table 11: Comparison on GSM8K with Qwen3-4B

As shown in Table 11, we empirically selected the best hyperparameters based on performance across a small grid search.

	mean	min	max	var	Q(0.25)	Q(0.5)	Q(0.75)
QwQ+Nothinking	1342.59	58	14261	1.43E+06	750	1025	1487.75
QwQ+GoldThink-Switch always on	285.61	108	749	7.17E+03	224	275	335
QwQ+GoldThink-Switch	494.00	110	9966	3.99E+05	233	298	419.75
Qwen2.5-Instruct	275.82	110	796	5.67E+03	222	266	318
QwQ-32B	1268.87	285	9484	1.11E+06	697.5	968	1397.75

Table 9: Token-length statistics on GSM8K with QwQ-32B. We report mean, min, max, variance (var), quantiles (Q).

	mean	min	max	var	Q(0.25)	Q(0.5)	Q(0.75)
QwQ+Nothinking	918.94	41	32768	3.34E+06	229	660	1025
QwQ+GoldThink-Switch always on	221.80	79	32768	4.70E+05	141	182	245
QwQ+GoldThink-Switch	351.98	77	32768	7.95E+05	144	189	291
Qwen2.5-Instruct	212.77	79	1113	8.36E+03	157	187	241
QwQ-32B	921.37	225	12999	7.81E+05	501	680	1005

Table 10: Token-length statistics on ASDIV with QwQ-32B. We report mean, min, max, variance (var), quantiles (Q).

<think> Okay, let's see. So the problem is about Sophia and Rose buying onions and potatoes at the market. Rose bought 4 times the number of onions and potatoes that Sophia bought. And we know Rose bought 12 onions and 4 potatoes. The question is asking how many onions and potatoes Sophia bought in total.
Hmm, first, **let me parse the problem again.** (. . . omitted. . .) Therefore, Sophia bought 3 onions and 1 potato. *The total would be 3 + 1 = 4.*
Wait, but let me make sure I didn't misinterpret the problem (. . . omitted. . .)
Alternatively, (. . . omitted. . .) **Alternatively,** (. . . omitted. . .) Therefore, *total for Sophia is 3 + 1 = 4.*
Alternatively, (. . . omitted. . .) *Sophia's total is 4.*
Hmm, so either way, whether we take it per item or combined, the answer comes out the same? Wait, that's interesting. **Let me check both interpretations again.** (. . . omitted. . .)
So either way, *the answer is 4.* **Wait,** (...omitted...) Therefore, *the answer is 4.*
Therefore, I think *the answer is 4.* **Let me just confirm once again.** (. . . omitted. . .) So Sophia's *total is 4.*
Alternatively, (. . . omitted. . .) Therefore, Sophia bought 3 onions and 1 potato, *totaling 4.*
I think that's solid. So *the answer should be 4.* </think>
Sophia bought onions and potatoes such that Rose, who purchased 4 times as many of each, ended up with 12 onions and 4 potatoes. Let O be the number of onions and P the number of potatoes Sophia bought. Since Rose bought 4 times as many:

$$4O = 12 \implies O = \frac{12}{4} = 3$$

$$4P = 4 \implies P = \frac{4}{4} = 1$$

Thus, Sophia bought $3 + 1 = 4$ *items in total.* $\boxed{4}$ (1353 tokens)

Figure 5: Case study of QwQ

<think> Okay I have finished thinking.</think>
Alright, let's tackle this problem step by step. So Sophia and Rose went to the market to buy onions and potatoes. The problem says that Rose bought 4 times the number of onions and potatoes that Sophia bought. Then it tells us that Rose bought 12 onions and 4 potatoes. We need to find out how many onions and potatoes Sophia bought in total. Hmm, let me break this down. (...omitted...)
Adding those together, the *total would be 3 + 1 = 4*.
Wait, let me check that again to make sure I didn't make a mistake. (. . . omitted. . .) Adding those together, *3 + 1 is indeed 4*. That seems straightforward.
Is there another way to approach this? Maybe by setting up equations. (. . . omitted. . .) *Total for Sophia is O_s + P_s = 3 +1 =4*. Yep, same answer.
Hmm, maybe I should check if there's any possibility that (. . . omitted. . .) the *final total for Sophia ends up being 4*.
But the problem says "4 times the number of onions and potatoes", which is a bit ambiguous. (. . . omitted. . .)
Alternatively, (. . . omitted. . .) So either way, the answer comes out the same.
Therefore, regardless of the interpretation, the total number of onions and potatoes *Sophia bought is 4*.
So the *final answer should be 4*. I think that's solid. Both interpretations lead to the same result, so there's no ambiguity here.
Final Answer The total number of onions and potatoes *Sophia bought is* $\boxed{4}$. (984 tokens)

Figure 6: Case study of QwQ+Nothinking

<think> Let's break down the problem step by step. (...omitted...) Therefore, the total number of onions and potatoes that Sophia bought is $\boxed{4}$. (328 tokens)

Figure 7: Case study of QwQ+Gold-Switch

To find out how many onions and potatoes Sophia bought, (...omitted...) Therefore, Sophia bought a total of 4 items (onions and potatoes) at the market. (273 tokens)

Figure 8: Case study of Qwen2.5-32B-Instruct