

000 DIVERSE TEXT DECODING VIA ITERATIVE
001
002 REWEIGHTING
003
004

005 **Anonymous authors**

006 Paper under double-blind review

007
008 ABSTRACT
009
010

011 Recent advances in large language models (LLMs) have led to impressive results
012 in text generation. However, current decoding methods still lack diversity when
013 combined with popular sampling techniques. We propose a **Reweighting-based**
014 **Iterative DEcoding (OverRIDE)** approach that dynamically adjusts the decoding
015 process with history responses. Our method fine-tunes auxiliary output heads iter-
016 atively on previously generated sequences to capture and suppress semantic pat-
017 terns that appear in the history responses. This inference-time training process
018 only incurs minimal loss of efficiency. We conduct extensive experiments on var-
019 ious tasks, including code generation, mathematical reasoning and story genera-
020 tion, demonstrating that OverRIDE increases output diversity while maintaining
021 quality. We implement OverRIDE on LLM serving systems like vLLM, achieving
022 a 6.4% throughput loss for 72B models under parallel decoding.

023 1 INTRODUCTION
024

025 Recent advances in Large Language Models (LLMs) have brought significant improvements to ap-
026 plications in natural language processing, such as creative writing (Fan et al., 2018; Brown et al.,
027 2020; Stiennon et al., 2020; Rafailov et al., 2023), mathematical reasoning(Cobbe et al., 2021; Yu
028 et al., 2023; Luo et al., 2023; Romera-Paredes et al., 2024), and code generation (Li et al., 2022b;
029 Roziere et al., 2023; Li et al., 2023; El-Kishky et al., 2025). To enhance the quality and diversity of
030 LLM generations, various sampling techniques have been proposed. For instance, top- k sampling
031 (Fan et al., 2018) and top- p nucleus sampling (Holtzman et al., 2019) work by filtering out low-
032 confidence tokens in the distribution tail, which helps ensure the overall quality of the generated
033 text; Temperature sampling (Ackley et al., 1985) controls the randomness of the sampling process,
034 allowing for more creative outputs.

035 Despite these advances in sampling techniques, current generation strategies still struggle to pro-
036 duce adequately diverse outputs. This limitation comes from the autoregressive nature of mod-
037 ern language models. The standard decoding process for autoregressive language models involves
038 sequentially predicting each token conditioned on all previously generated tokens, creating a path-
039 dependent generation process. In practice, this often leads to responses that share similar beginnings,
040 and only diverge toward the end, as illustrated in Figure 1. Often, the generated samples are struc-
041 turally similar, with only minor variations in local details. For example, in code generation tasks,
042 models may produce different variable names or formatting styles, but the underlying logic remains
043 the same, which leads to the same mistake across multiple generations.

044 The issue lies in the decoding strategy: given the same context, the same next-token probability
045 distribution is applied at every round of response, regardless of what has been generated in previous
046 rounds. Although various sampling methods adjust this distribution by, for example, truncating the
047 tail or rescaling the distribution, this process is done only at the token level independently. While
048 these sampling-based approaches enable stochastic decoding, they fail to learn from the history of
049 generated samples, and lead to redundant exploitation of high-probability regions in the generation
050 space. This often causes degeneration problems like repetitions or unfavorable outputs. (Holtzman
051 et al., 2019; Welleck et al., 2019; Su et al., 2022)

052 To address the aforementioned limitation, we introduce OverRIDE (**Reweighting-based Iterative**
053 **DEcoding**), a decoding method that dynamically adjusts token probabilities across different de-
054 coding rounds based on previously generated responses. OverRIDE iteratively fine-tunes a guide

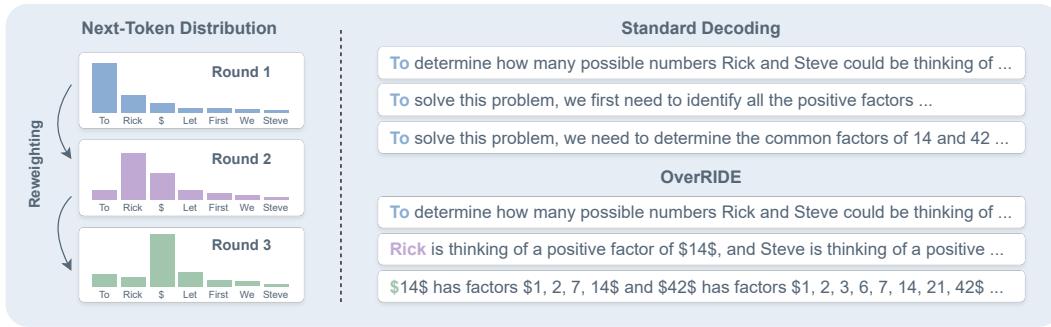


Figure 1: Standard autoregressive decoding often leads to similar outputs. OverRIDE dynamically reweights token probabilities by learning from the patterns of previous generations.

model on previously generated samples to identify common semantic patterns. This guide model is then used to reweight the next-token distribution of the original model by suppressing the probability of tokens that lead to previously seen patterns. In such a way, we encourage exploration of less-traveled paths in the generation space. By learning from previous generations, OverRIDE can effectively suppress repetitive patterns, leading to more diverse outputs, as illustrated in Figure 1. Although the training of the auxiliary model is performed at inference time, it is relatively efficient compared to the cost of autoregressively generating an entire response. Additionally, OverRIDE can be combined with existing sampling methods to further improve the generation process.

Furthermore, we design a parallel version of OverRIDE, which allows seamless integration with LLM serving systems like vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024), and benefits from the efficiency of parallel decoding. We restrict the trainable parameters to a tiny amount in the output head, allowing for minimal efficiency loss. We also synchronize the sampling and fine-tuning processes, which makes it compatible with the decoding mechanism of LLM serving systems.

We evaluate OverRIDE on a variety of tasks, including code generation (HumanEval (Chen et al., 2021b)), mathematical reasoning (MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021)) and story generation (CCNews (Common Crawl, 2007)). Our experiments demonstrate that OverRIDE improves the response diversity while maintaining or improving quality. Additionally, OverRIDE prevents models from generating over-confident responses. We further analyze the decoding dynamics of OverRIDE, showing how it explores high- and low-probability regions in the generation space to achieve better diversity. We implement OverRIDE on vLLM (Kwon et al., 2023), and achieve a 6.4% throughput loss for 72B models under parallel decoding. We test on models ranging from 3B to 72B, and verify that OverRIDE is effective on models of different sizes.

2 METHODOLOGY

2.1 OVERVIEW

Let M_p be a pretrained LLM, and $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^N$ be a set of contexts. For each context \mathbf{c}_i , our goal is to generate T diverse responses. As illustrated in Figure 2, at each round $t \in \{1, 2, \dots, T\}$, we: 1) Generate a new response for each context; 2) Fine-tune a guide model M_{q_t} on previously generated responses; 3) Use M_{q_t} to reweight the next-token distribution of M_p , producing the reweighted distribution p_t ; 4) Use p_t to generate new responses in the next round. The core idea is that the reweighted distribution should be close enough to the next-token distribution of the original model M_p , while avoiding the distribution that would likely result in previously generated responses. In this way, we encourage the model to explore less-traveled paths at every decoding round.

In the following sections, we explain each component of OverRIDE: Section 2.2 describes how to capture common patterns from history responses by fine-tuning the guide model M_{q_t} ; Section 2.3 describes the reweighting mechanism that adjusts token probabilities. Section 2.4 describes a parallel implementation of OverRIDE, which allows for efficient parallel decoding, and can be directly integrated with existing LLM serving systems.

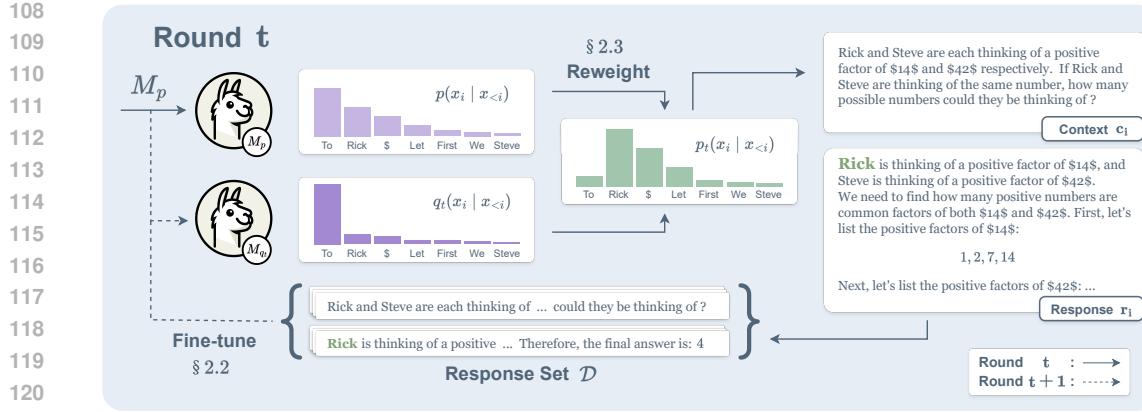


Figure 2: The overall framework of OverRIDE, implemented sequentially.

2.2 CAPTURE HISTORY PATTERNS

A standard autoregressive language model M_p predicts the next token with probability $p(x_i | x_{<i})$, where x_i is the i -th token in the sequence, and $x_{<i}$ is the sequence of tokens before x_i . In the initial round, the model generates a response r_i for each context c_i by sampling each token sequentially from the original distribution $p(x_i | x_{<i})$. The generated responses and corresponding contexts are stored in a response set $\mathcal{D} = \{(c_i, r_i)\}_{i=1}^N$. Since no prior generations exist at the initial round, we decode responses with the original model to explore its default behavior and high-probability paths.

Starting from the second round, we fine-tune a guide model M_{q_t} from the original model M_p , using the previously generated responses in \mathcal{D} :

$$\mathcal{L}_{q_t} = -\frac{1}{|\mathcal{D}|} \sum_{(\mathbf{c}, \mathbf{r}) \in \mathcal{D}} \log q_t(\mathbf{r} | \mathbf{c}). \quad (1)$$

This guide model M_{q_t} learns to predict the common semantic patterns that appeared in previous samples. The key insight here is that we cannot directly use previously generated samples to guide new generations, since the sampling process is non-deterministic, and responses vary at each round. By fine-tuning a guide model on previous responses, we enable it to learn and generalize the common patterns that have emerged across samples. This allows us to capture and suppress these patterns for more diverse responses in subsequent rounds.

2.3 REWEIGHTING THE NEXT-TOKEN DISTRIBUTION

After fine-tuning the guide model M_{q_t} , we use it to reweight the original model’s next-token distribution. We construct a reweighted probability distribution $p_t(x_i | x_{<i})$ that leverages both the original model M_p and the guide model M_{p_t} :

$$p_t(x_i | x_{<i}) = \frac{1}{Z} \left(\frac{p(x_i | x_{<i})}{q_t(x_i | x_{<i})} \right)^\lambda p(x_i | x_{<i}), \quad (2)$$

where $p(x_i | x_{<i})$ is the next-token distribution from the original model, $q_t(x_i | x_{<i})$ is the next-token distribution from the guide model, λ is a hyperparameter that controls the strength of reweighting, and Z is the normalization term.

The intuition behind this reweighting mechanism is that since the guide model M_{p_t} predicts patterns that appear in previous samples, if it assigns higher probability to a token, the weighting ratio $\frac{p(x_i | x_{<i})}{q_t(x_i | x_{<i})}$ becomes smaller, reducing the probability of selecting this token in a new round. Conversely, tokens that were less frequently chosen in previous rounds receive relatively higher probability, encouraging exploration of alternative responses. The hyperparameter λ controls the diversity-quality trade-off. This reweighting process is performed at each decoding round, allowing the model to adaptively adjust its generation strategy based on the history of generated responses.

162
163

2.4 PARALLEL IMPLEMENTATION FOR LLM SERVING SYSTEMS

164
165
166
167
168
169
170

The described OverRIDE method operates sequentially, requiring all responses to be collected and the guide model to be fine-tuned before proceeding to the next round. However, modern LLM serving systems like vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024) have implemented parallel decoding that significantly reduces sampling time and computational overhead. These efficiency gains primarily come from the reuse of the KV cache. For example, when generating multiple responses for the same context, the KV cache for the context only needs to be computed once, and subsequent sampled tokens or generation paths can also be reused if they are identical.

171
172
173
174
175
176

We now describe our architectural and procedural improvements for parallelizing OverRIDE. Our design is directly compatible with existing LLM serving systems, as demonstrated by our implementation, which maintains high-performance parallel decoding without compromising the method’s core objective of generating diverse responses.

177
178
179
180
181
182
183
184

Output head adapters To minimize the efficiency loss from fine-tuning, we restrict the trainable parameters to the output heads. As shown in Figure 3, we set a low-rank adapter for each decoding round, with a structure and training approach similar to LoRA (Hu et al., 2022). This allows us to compute both p_t and q_t simultaneously in a single forward pass given hidden states h :

$$\text{logit}_p = Wh, \text{ and } \text{logit}_{q_t} = Wh + W_B^t W_A^t h. \quad (3)$$

185
186
187
188
189
190
191

Intuitively, the adapter captures the difference between the model’s original distribution and the categorical distribution of the actual sampled token. In the next decoding round, this difference is suppressed by a factor of λ to avoid previous patterns. Since the guide model M_{q_t} only drifts slightly from the original distribution, this tiny amount of trainable parameters is sufficient to capture the difference.

192
193
194
195
196

Additionally, the use of adapters come with a satisfactory benefit: as the model size grows larger, the efficiency loss gets even smaller, since the adapter takes up a smaller proportion of the total parameters. We conduct experiments on different model sizes, and show that this method is still effective.

197
198
199
200
201
202
203
204
205

Synchronized fine-tuning The original OverRIDE method requires collecting all the sampled responses before fine-tuning, which is not acceptable in the case of parallel decoding. To solve this problem, we propose to perform fine-tuning right after the sampling process. As shown in Figure 4, at the t -th round, we use head t to compute the reweighted distribution p_t and sample the next token. Then, we update the adapter weights in head $t+1$ using cross entropy loss with respect to the sampled token, which is the same as Equation 1 but at the token level.

206
207
208
209
210
211
212
213
214
215

This design aligns with the implementation of LLM serving systems. To perform parallel decoding, the same context is duplicated into T identical requests, and sent to the model for execution. These requests are consecutively processed in order to reuse KV cache in the memory. While this original implementation doesn’t require an execution order, we sort these requests by the order of their round number. In this way, after the t -th request is processed by head t , head $t+1$ is updated subsequently, and is immediately used to perform the $t+1$ -th round of decoding. The parallel version of OverRIDE is described in Algorithm 1.

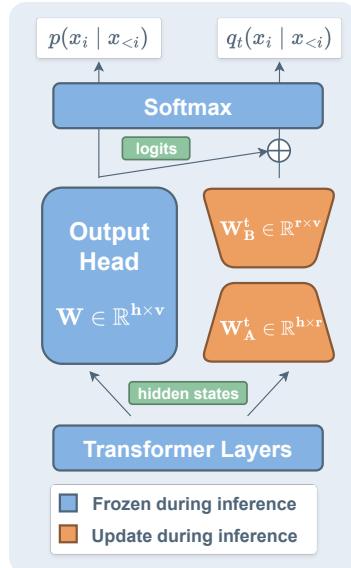


Figure 3: Architecture of a single output head. h is the dimension of the hidden states, r is the rank of the adapter, and v is the dimension of the vocabulary.

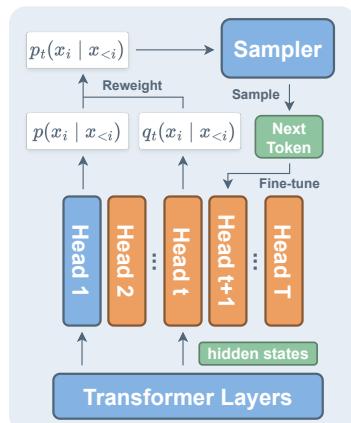


Figure 4: Decoding process at round t . Head 1 is the original head. Head t consists of the original head and the adapter. Here, we simplify them as a whole.

216 **Algorithm 1** OverRIDE: Reweighting-based Iterative Decoding

217 **Input:** model M_p ; context \mathbf{c} ; number of rounds T ; reweighting parameter λ .

218 1: **for** $t \leftarrow 1$ to T **do**

219 2: **if** $t = 1$ **then**

220 3: $\mathbf{r}_1 \sim p(\cdot \mid \mathbf{c})$ ▷ Sample response with the original model

221 4: **else**

222 5: $p_t(x_i \mid x_{<i}) = \text{normalize}\left(\left(\frac{p(x_i \mid x_{<i})}{q_t(x_i \mid x_{<i})}\right)^\lambda p(x_i \mid x_{<i})\right)$ ▷ Reweighting

223 6: $\mathbf{r}_t \sim p_t(\cdot \mid \mathbf{c})$ ▷ Sample response with the reweighted distribution

224 7: **end if**

225 8: **if** $t < T$ **then**

226 9: $M_{q_{t+1}} \leftarrow \text{train}(M_{q_{t+1}}, \mathcal{L}_{q_t}, \{(\mathbf{c}, \mathbf{r}_t)\})$ ▷ Fine-tune head $t+1$ with the sampled response

227 10: **end if**

228 11: **end for**

229 12: **Output:** Diverse responses $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T\}$.

232 **3 EXPERIMENTS AND ANALYSIS**234 **3.1 DATASETS AND METRICS**

236 **Datasets** We evaluate OverRIDE’s effectiveness across different domains, including code generation (HumanEval(Chen et al., 2021b)), mathematical reasoning (MATH500 (Hendrycks et al., 2021), GSM8K(Cobbe et al., 2021)), and story generation (CCNews Common Crawl (2007)).

240 **Evaluation metrics** To evaluate the quality and diversity of multiple generated responses, we 241 employ the following metrics: (1) PASS@ k : the pass rate of a problem if allowed to sample k 242 times; (2) Cosine similarity: the average pairwise cosine similarity between the embeddings of 243 generated responses; (3) CodeBLEU (Ren et al., 2020): the average pairwise CodeBLEU score 244 between all generated code snippets, which indicates the degree of matching between different code; 245 (4) MAUVE (Pillutla et al., 2021): a metric to determine the similarity between model generations 246 and human answers. Details of the datasets and metrics can be found in Appendix A.

248 **3.2 MAIN RESULTS**

250 We evaluate OverRIDE using [Qwen-2.5-7B-Instruct](#) (Yang et al., 2024) and [Mistral-7B-Instruct-v0.3](#) (Jiang et al., 2023) as our base models. For each model, we implement OverRIDE with the 251 following sampling methods: (1) Greedy decoding; (2) Top- p sampling (Holtzman et al., 2019) 252 with different temperature settings ($\tau = 0.6, 1.0$); (3) Top- k sampling (Fan et al., 2018); (4) Min- p 253 sampling (Nguyen et al., 2024). The implementation details can be found in Appendix B.

256 **OverRIDE balances quality and diversity.** Table 1 shows the results on HumanEval(Chen et al., 257 2021b) and MATH500(Hendrycks et al., 2021). Results on GSM8K(Cobbe et al., 2021) are 258 presented in Appendix D. We report PASS@ k accuracy, pairwise CodeBLEU score and pairwise cosine 259 similarity. While PASS@ k accuracy mainly indicates generation quality, it also reflects generation 260 diversity. Lower CodeBLEU score and cosine similarity indicates higher diversity between generated 261 responses. For PASS@5 accuracy, PASS@10 accuracy, and the similarity score, we present 262 paired results where the left value represents the baseline sampling method, and the right value 263 represents the same method integrated with OverRIDE.

264 OverRIDE increases PASS@5 and PASS@10 accuracy while reducing response similarity across 265 sampling methods with both models. Additionally, OverRIDE’s effectiveness persists across different 266 temperature settings. This derives from OverRIDE’s ability to dynamically explore both high 267 and low probability responses, avoiding the typical trade-off between diversity and quality. This 268 insight suggests that when implementing OverRIDE, it is advantageous to base on high-probability 269 decoding methods. This allows OverRIDE to explore high-probability paths in the initial rounds, 270 and lower-probability alternatives in later rounds, thus maximizing OverRIDE’s effectiveness.

270 Table 1: Results of different sampling methods on HumanEval and MATH500.
271

272 Model	273 Method	HumanEval				MATH			
		PASS@1	PASS@5	PASS@10	CodeBLEU ↓	PASS@1	PASS@5	PASS@10	Similarity ↓
274 Qwen 2.5-7B	Greedy	64.9	64.9 / 81.2	64.9 / 86.0	1.000 / 0.626	72.5	72.5 / 84.3	72.5 / 87.0	1.000 / 0.938
	Top- p , $\tau = 0.6$	63.7	78.4 / 81.6	81.4 / 86.4	0.754 / 0.610	72.1	84.1 / 84.8	86.8 / 87.3	0.950 / 0.936
	Top- p , $\tau = 1.0$	60.3	82.1 / 83.6	85.5 / 88.5	0.666 / 0.578	71.6	84.8 / 84.9	87.2 / 87.6	0.940 / 0.929
	Top- k	62.5	80.0 / 83.2	84.4 / 87.9	0.715 / 0.603	72.2	84.2 / 84.7	86.7 / 87.5	0.948 / 0.937
	Min- p	63.2	79.7 / 83.7	83.2 / 88.1	0.736 / 0.610	71.8	84.3 / 84.3	87.2 / 87.6	0.946 / 0.932
277 Mistral 7B	Greedy	31.3	31.3 / 45.9	31.3 / 52.4	1.000 / 0.537	13.1	13.1 / 27.5	13.1 / 36.6	1.000 / 0.874
	Top- p , $\tau = 0.6$	29.9	47.4 / 49.0	54.9 / 56.9	0.563 / 0.474	11.3	27.7 / 28.4	36.4 / 37.4	0.871 / 0.862
	Top- p , $\tau = 1.0$	29.6	49.3 / 51.5	58.1 / 58.5	0.448 / 0.400	10.9	27.1 / 28.0	36.2 / 37.2	0.854 / 0.850
	Top- k	29.7	48.9 / 49.8	56.0 / 57.6	0.521 / 0.450	11.8	27.1 / 28.5	36.4 / 37.2	0.870 / 0.863
	Min- p	30.0	48.7 / 48.6	57.1 / 57.6	0.547 / 0.469	11.5	27.0 / 28.5	36.4 / 37.3	0.867 / 0.860

281
282
283
284 **OverRIDE generates human-like responses.** Table 2 shows the results on CCNews(Common
285 Crawl, 2007). Models are provided with the first 32 tokens of each news text sample, and required
286 to generate the subsequent 256 tokens. We use MAUVE (Pillutla et al., 2021) to evaluate how all 10
287 rounds of model generations match the original news. Cosine similarity is still measured pairwise
288 between generations. We present paired results where the left value represents the baseline sampling
289 method, and the right value represents the same method integrated with OverRIDE.

290 The results demonstrate that OverRIDE can
291 improve human-like generation quality while
292 enhancing generation diversity. OverRIDE
293 achieves consistent improvements in MAUVE
294 scores and reduction in cosine similarity scores
295 across all settings, indicating better alignment
296 with human writing patterns while increasing di-
297 versity between generated responses. OverRIDE
298 alleviates the typical LLM generation issues of
299 repetition (Holtzman et al., 2019). For repetitive
300 patterns generated by the original model in the
301 first round, OverRIDE avoids generating similar
302 patterns in subsequent rounds through reweight-
303 ing, thereby improving generation quality.

293 Table 2: Results of methods on CCNews.

304 Model	305 Method	306 CCNews	
		307 MAUVE ↑	308 Similarity ↓
309 Qwen 2.5-7B	Greedy	0.803 / 0.753	1.000 / 0.606
	Top- p , $\tau = 0.6$	0.901 / 0.938	0.726 / 0.678
	Top- p , $\tau = 1.0$	0.974 / 0.977	0.651 / 0.641
	Top- k	0.953 / 0.960	0.715 / 0.679
	Min- p	0.946 / 0.946	0.714 / 0.668
310 Mistral 7B	Greedy	0.883 / 0.891	1.000 / 0.675
	Top- p , $\tau = 0.6$	0.937 / 0.952	0.758 / 0.703
	Top- p , $\tau = 1.0$	0.942 / 0.947	0.697 / 0.685
	Top- k	0.904 / 0.937	0.744 / 0.713
	Min- p	0.903 / 0.934	0.759 / 0.707

311 **OverRIDE can scale up.** To validate whether OverRIDE remains effective across different model
312 sizes, we test models of varying scales. Table 3 shows the results on HumanEval and MATH500
313 across Qwen-2.5 model series ranging from 3B to 72B parameters. OverRIDE demonstrates consis-
314 tent improvements across all model sizes, enhancing both quality and diversity.

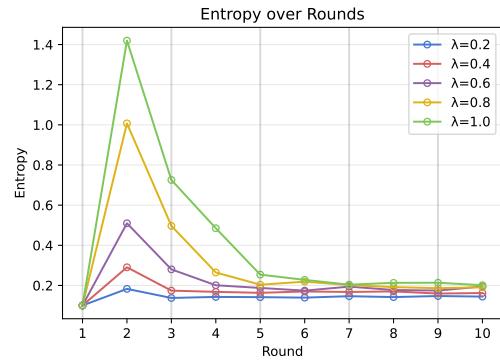
315 Table 3: Results of different size models on HumanEval and MATH500.

316 Model	HumanEval				MATH			
	PASS@1	PASS@5	PASS@10	CodeBLEU ↓	PASS@1	PASS@5	PASS@10	Similarity ↓
Qwen-2.5-3B	58.2	78.7 / 79.0	82.9 / 83.2	0.629 / 0.547	63.3	78.3 / 79.2	82.8 / 83.5	0.944 / 0.937
Qwen-2.5-7B	63.7	78.4 / 81.6	81.4 / 86.4	0.754 / 0.610	72.1	84.1 / 84.8	86.8 / 87.3	0.950 / 0.936
Qwen-2.5-14B	40.0	63.4 / 70.5	71.1 / 77.2	0.698 / 0.588	75.8	85.1 / 85.7	87.7 / 88.0	0.941 / 0.929
Qwen-2.5-32B	67.8	78.8 / 80.5	81.2 / 84.2	0.695 / 0.579	78.5	86.4 / 87.2	88.6 / 89.3	0.945 / 0.933
Qwen-2.5-72B	73.5	82.1 / 83.8	83.4 / 87.0	0.803 / 0.674	78.9	87.0 / 88.0	88.8 / 90.5	0.955 / 0.941

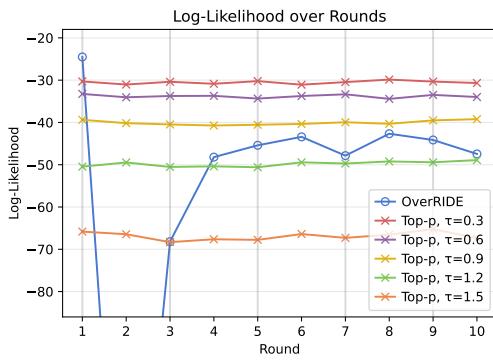
317 Furthermore, compared to smaller models (3B), larger models (72B) achieve greater performance
318 gains through OverRIDE, despite the trainable adapter parameters constituting a smaller proportion
319 of the total parameters. We attribute this to the enhanced capacity of larger models, which provides
320 more expressive hidden states that enable adapters to easily capture the differences between the
321 original next-token distribution and the distribution of the sampled tokens.

324 3.3 DECODING DYNAMICS
325

326 To better understand OverRIDE’s behavior and its impact on generation diversity, we analyze the
327 entropy of token distributions and the log-likelihood of generated sequences across decoding rounds.
328 For the following experiments, we implement OverRIDE with greedy decoding using the Qwen-2.5-
329 7B model, and evaluate on the MATH500 dataset.



343 Figure 5: Next-token distribution entropy across
344 decoding rounds for different λ values. Higher
345 entropy indicates greater diversity.
346



364 Figure 6: Log-likelihood of responses across
365 decoding rounds, under OverRIDE and top- p
366 sampling with different temperatures.
367

368 Figure 5 shows how the reweighting parameter λ controls the diversity of token selection during
369 the decoding process. We observe a similar pattern across all λ values: In round 1, the average
370 token entropy starts at a low level because of greedy decoding. In round 2, a significant spike in
371 entropy occurs as OverRIDE begins actively suppressing high-probability patterns. In later rounds,
372 the entropy gradually decreases and eventually stabilizes. Higher λ values lead to higher entropies,
373 with $\lambda = 1.0$ reaching a peak of nearly 10 times its initial value. This demonstrates that stronger
374 reweighting leads to more aggressive exploration and higher diversity in the generated responses,
375 particularly in early rounds where the model actively diverges from previous generation patterns.

376 Figure 6 shows the average log-likelihood of generated responses across decoding rounds. Over-
377 RIDE initially experiences a sharp decline in log-likelihood in early rounds as it actively suppresses
378 previously seen patterns, forcing exploration into low-probability regions of the original distribution.
379 The log-likelihood partially recovers in later rounds and eventually stabilizes at an intermediate
380 level. This reveals OverRIDE’s decoding dynamics – the interplay between OverRIDE’s suppres-
381 sion mechanism and the model’s original distribution. As the response set grows, it contains both
382 high and low likelihood generations. When suppressing patterns common to all previous responses,
383 the model discovers alternative paths that, while distinct from the initially preferred responses, still
384 represent valid solutions with higher likelihood.

385 The comparison with top- p sampling at various temperatures reveals the differences in how these
386 methods approach diversity. While sampling with higher temperatures consistently produces low
387 log-likelihood across all rounds, OverRIDE’s strategy is more dynamic: exploring between low-
388 and high-probability responses. This indicates that OverRIDE’s approach to increasing diversity is
389 fundamentally different from temperature scaling, which is only flattening the distribution.

390 3.4 EFFICIENCY ANALYSIS
391

392 To assess the efficiency impact of our parallel implementation, we apply the parallel version of
393 OverRIDE on vLLM (Kwon et al., 2023). We conduct experiments using Qwen-2.5 model series,
394 with size ranging from 3B to 72B. All models are deployed using on a single node, and distributed
395 on a single or multiple GPUs. Specifically, for 3B, 7B, 14B, 32B, 72B models, we use 1, 1, 2, 4, 8
396 GPUs respectively. More details about the setup can be found in Appendix B. Models are evaluated
397 on the MATH500 dataset with top- p sampling ($\tau = 0.6$). For each context, we sample 10 rounds
398 with parallel decoding. Throughput is measured as output tokens per second.

Table 4 shows the throughput comparison across different model sizes. As the model size increases from 3B to 72B parameters, the throughput loss gradually decreases from 8.2% to 6.4%. This validates our hypothesis in Section 2.4 that, since the adapter parameters constitute a smaller proportion of total parameters as the model scales up, OverRIDE becomes more efficient for larger models. This provides assurance for deploying OverRIDE with even larger models in production environments.

3.5 OUTPUT HEAD CONFIGURATION

To determine the optimal rank for the output head adapters in our parallel implementation, we conduct experiments with different rank settings. Experiments are conducted with Qwen-2.5-7B on the MATH500 dataset with top- p sampling ($\tau = 0.6$). The rank r refers to the rank of the adapter matrices W_A^t and W_B^t in the output heads. We compare against a "Full" setting that uses independent output head parameters W^t for each decoding round to compute q_t without adapters.

Table 5 reveals a trade-off between model capacity and computational efficiency. When the rank is too small ($r = 4$), the adapter lacks sufficient parameters to accurately model the distribution q_t , leading to suboptimal performance. When the rank becomes too large ($r = 256$), we observe severe efficiency degradation with a 30.6% throughput drop, while performance gains are minimal compared to smaller ranks. We eventually set $r = 16$, which provides sufficient modeling capacity while maintaining reasonable computational overhead.

3.6 SENSITIVITY ANALYSIS

To understand how pattern suppression in OverRIDE affects model performance, we conduct a sensitivity analysis on the hyperparameter λ , which controls the intensity of the reweighting effect. Higher values of λ lead to more aggressive suppression of patterns in previously generated responses. Figure 7 shows the performance of OverRIDE with varying values of λ . Results are evaluated on the MATH500 dataset with top- p sampling ($\tau = 0.6$).

The results show that the reweighting parameter λ is model-specific. For Qwen-2.5-7B, we observe PASS@5 and PASS@10 reaching their highest at $\lambda = 0.8$. In contrast, Mistral-7B shows an optimal performance with lower λ s, with PASS@5 and PASS@10 reaching their highest at $\lambda = 0.4$. Beyond

Table 4: Comparison of throughput across different model sizes under parallel decoding.

Model	Throughput (token/s)		Drop
	Baseline	OverRIDE	
Qwen-2.5-3B	5439.9	4992.2	-8.2%
Qwen-2.5-7B	3709.6	3435.8	-7.4%
Qwen-2.5-14B	2798.0	2600.3	-7.1%
Qwen-2.5-32B	2091.3	1947.4	-6.9%
Qwen-2.5-72B	1463.0	1369.9	-6.4%

Table 5: Comparison of performance and efficiency across different rank settings.

Rank	PASS@10	Throughput	Drop
Baseline	86.8	3709.6	/
$r = 4$	86.9	3453.1	-6.9%
$r = 16$	87.5	3435.8	-7.4%
$r = 64$	87.3	3301.0	-11.0%
$r = 256$	87.1	2569.2	-30.7%
Full	87.3	1984.1	-46.5%

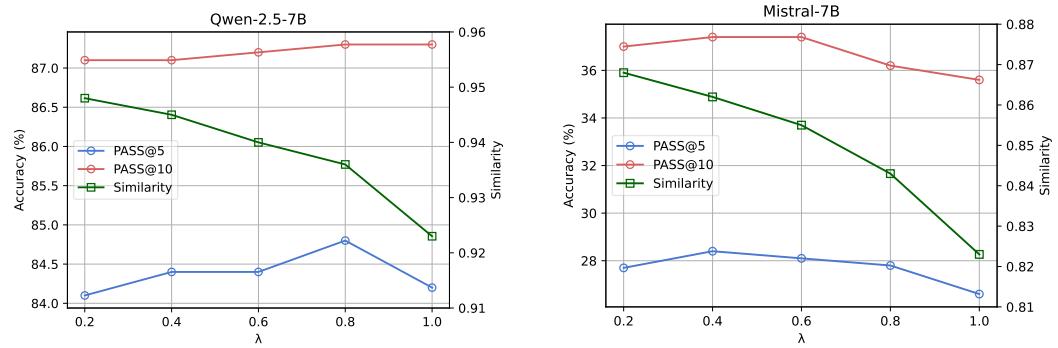


Figure 7: Effect of reweighting parameter λ on PASS@ k performance (%) and similarity. The left figure shows results for Qwen-2.5-7B, while the right figure shows results for Mistral-7B.

432 this range, performance gradually declines as λ increases. This shows that Mistral-7B is more
 433 sensitive to the reweighting mechanism. For both models, similarity scores consistently decrease
 434 with higher λ values, confirming that stronger reweighting leads to more diverse responses.
 435

436 Such sensitivity to λ can be attributed to the characteristics of output diversity for different models.
 437 From our main results in Tables 1, we observe that Mistral-7B consistently produces more diverse
 438 outputs than Qwen-2.5-7B with different sampling methods, as evidenced by their lower CodeBLRU
 439 and cosine similarity scores across all settings. This suggests that models with already sufficiently
 440 diverse responses benefit from lower λ values, since excessive pattern suppression becomes coun-
 441 terproductive and may cause degradation in performance.
 442

443 4 RELATED WORK

444 **Stochastic Decoding** Deterministic decoding approaches like greedy search and beam search
 445 (Freitag and Al-Onaizan, 2017) tend to select tokens that maximize model confidence, resulting in
 446 outputs that lack diversity, and often suffer from issues such as repetition or dull content. Stochastic
 447 decoding methods address these limitations by introducing randomness in the generation process.
 448 Temperature sampling (Ackley et al., 1985) flattens or sharpens the probability distribution to control
 449 randomness. Top- p sampling (Holtzman et al., 2019) and top- k sampling (Fan et al., 2018) truncate
 450 the distribution tail to exclude low-confidence tokens. More adaptive approaches (Basu et al., 2020;
 451 Hewitt et al., 2022; Nguyen et al., 2024) dynamically adjust sampling parameters based on distri-
 452 bution entropy or model confidence to generate creative and coherent responses. However, these
 453 methods operate independently at the token level without considering previous responses, which
 454 systematically limits their ability to produce diverse outputs across multiple generations.
 455

456 **Decoding with Guidance** Recent works have explored guiding the decoding process with addi-
 457 tional models or in-context information. A series of contrastive methods enhance generation faith-
 458 fulness and quality by contrasting between different models (Li et al., 2022a), different layers within
 459 the same model (Gera et al., 2023; Chuang et al., 2023; Das et al., 2024), or model outputs with dif-
 460 ferent contexts (Shi et al., 2024). Speculative decoding approaches (Xia et al., 2022; Leviathan et al.,
 461 2023; Chen et al., 2023) use lightweight draft models to predict multiple tokens in parallel, so as to
 462 improve decoding efficiency. Multi-token prediction methods (Fu et al., 2024; Gloeckle et al., 2024;
 463 Cai et al., 2024; Li et al., 2024; Guo et al., 2025) introduce additional decoding heads or branches to
 464 accelerate inference while maintaining output quality. Guided search methods (Yao et al., 2023; Xie
 465 et al., 2023; Zhu et al., 2024) employ verifiers or evaluators to navigate complex reasoning. While
 466 unlikelihood training (Welleck et al., 2019) and SimCTG (Su et al., 2022) also attempt to encourage
 467 diversity by addressing degeneration, they require extensive training on large corpora. In contrast,
 468 our method dynamically guides generation based on the specific patterns observed in previous re-
 469 sponses. This requires only lightweight fine-tuning during inference without depending on external
 470 knowledge or models. Additionally, our method enables context-specific diversity that adaptively
 471 evolves with each decoding round.

472 5 CONCLUSION

473 In this paper, we propose OverRIDE, an iterative decoding method that enhances the diversity of
 474 LLM outputs. OverRIDE works by reweighting the next-token distribution of autoregressive mod-
 475 els based on previously generated responses. By fine-tuning a guide model on previously generated
 476 responses, OverRIDE effectively suppresses patterns in history responses and encourages explo-
 477 ration of diverse alternatives across decoding rounds. To extend OverRIDE to parallel decoding
 478 for higher efficiency, we propose improvements in architecture and procedural design, and further
 479 implement OverRIDE on existing LLM serving systems.
 480

481 Our experimental results demonstrate that OverRIDE improves generation diversity while main-
 482 taining or improving quality across various models and sampling methods. We also provide insights
 483 into OverRIDE’s decoding dynamics, revealing how it balances exploration and exploitation during
 484 the generation process. Experiments with different size models show that OverRIDE can scale up
 485 to larger models, while maintaining performance and efficiency with parallel decoding. Our work
 486 suggests that OverRIDE is a promising approach for enhancing the diversity of LLM outputs.

486 ETHICS STATEMENT
487488 OverRIDE dynamically modifies the next-token distribution of LLMs. By suppressing previously
489 patterns and encouraging exploration of low-probability regions, our method could potentially dam-
490 age the safety alignment of the original model. This might lead to the generation of harmful or biased
491 contents. The primary negative effect we observe is repetitive content generation at high reweighting
492 parameter values λ . However, there exists cases where OverRIDE can produce unexpected outputs,
493 see Table 12 for an example. We recommend using moderate reweighting parameters and imple-
494 menting additional safety filters when applying OverRIDE in production environments. We have
495 used LLM tools to polish the writing of this paper.
496497 REPRODUCIBILITY STATEMENT
498499 We provide the following materials for reproducing the experiments in our paper: source code in the
500 supplementary material; datasets and metrics information in Appendix A; implementation details in
501 Appendix B; prompts used for generation in Appendix C.
502503 REFERENCES
504

505 David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann
506 machines. *Cognitive science*, 9(1):147–169, 1985.

507 Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R Varsh-
508 ney. Mirostat: A neural text decoding algorithm that directly controls perplexity. *arXiv preprint*
509 *arXiv:2007.14966*, 2020.

510 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
511 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
512 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

513 Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri
514 Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv*
515 *preprint arXiv:2401.10774*, 2024.

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

519 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared
520 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large
521 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

522 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared
523 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large
524 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

525 Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola:
526 Decoding by contrasting layers improves factuality in large language models. *arXiv preprint*
527 *arXiv:2309.03883*, 2023.

528 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
529 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to
530 solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

531 Common Crawl. Common crawl: Open repository of web crawl data. <https://commoncrawl.org/>, 2007.

532 Souvik Das, Lifeng Jin, Linfeng Song, Haitao Mi, Baolin Peng, and Dong Yu. Entropy
533 guided extrapolative decoding to improve factuality in large language models. *arXiv preprint*
534 *arXiv:2404.09338*, 2024.

540 Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan,
 541 Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. Competitive programming
 542 with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025.

543

544 Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint*
 545 *arXiv:1805.04833*, 2018.

546

547 Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. *arXiv*
 548 *preprint arXiv:1702.01806*, 2017.

549

550 Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm infer-
 551 ence using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

552

553 Ariel Gera, Roni Friedman, Ofir Ariviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim,
 554 and Eyal Shnarch. The benefits of bad advice: Autocontrastive decoding across model layers.
 555 *arXiv preprint arXiv:2305.01628*, 2023.

556

557 Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Syn-
 558 naeve. Better & faster large language models via multi-token prediction. *arXiv preprint*
 559 *arXiv:2404.19737*, 2024.

560

561 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
 562 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd
 563 of models. *arXiv preprint arXiv:2407.21783*, 2024.

564

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

568

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

572

573 John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model
 574 desmothing. *arXiv preprint arXiv:2210.15191*, 2022.

575

576 Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text
 577 degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

578

579 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
 580 Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

581

582 Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh
 583 Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lu-
 584 cile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao,
 585 Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *ArXiv*,
 abs/2310.06825, 2023.

586

587 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph
 588 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 589 serving with pagedattention. In *Proceedings of the 29th symposium on operating systems prin-
 590 ciples*, pages 611–626, 2023.

591

592 Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative
 593 decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

594

595 Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou,
 596 Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with
 597 you! *arXiv preprint arXiv:2305.06161*, 2023.

598

599 Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke
 600 Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization.
 601 *arXiv preprint arXiv:2210.15097*, 2022a.

594 Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires
 595 rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.
 596

597 Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittweis, Rémi Leblond, Tom
 598 Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation
 599 with alphacode. *Science*, 378(6624):1092–1097, 2022b.

600 Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chat-
 601 gpt really correct? rigorous evaluation of large language models for code generation. *Advances*
 602 *in Neural Information Processing Systems*, 36:21558–21572, 2023.

603

604 Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qing-
 605 wei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning
 606 for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

607 Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-
 608 Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint*
 609 *arXiv:2407.01082*, 2024.

610

611 Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi,
 612 and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using diver-
 613 gence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828, 2021.

614

615 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea
 616 Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances*
 617 *in Neural Information Processing Systems*, 36:53728–53741, 2023.

618

619 Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou,
 620 Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis.
 621 *arXiv preprint arXiv:2009.10297*, 2020.

622

623 Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog,
 624 M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang,
 625 Omar Fawzi, et al. Mathematical discoveries from program search with large language models.
 626 *Nature*, 625(7995):468–475, 2024.

627

628 Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi
 629 Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for
 630 code. *arXiv preprint arXiv:2308.12950*, 2023.

631

632 Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih.
 633 Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024*
 634 *Conference of the North American Chapter of the Association for Computational Linguistics: Human*
 635 *Language Technologies (Volume 2: Short Papers)*, pages 783–791, 2024.

636

637 Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford,
 638 Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances*
 639 *in neural information processing systems*, 33:3008–3021, 2020.

640

641 Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. A contrastive
 642 framework for neural text generation. *Advances in Neural Information Processing Systems*, 35:
 643 21548–21561, 2022.

644

645 Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston.
 646 Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

647

648 Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative de-
 649 coding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint*
 650 *arXiv:2203.16487*, 2022.

651

652 Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael
 653 Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Process-
 654 ing Systems*, 36:41618–41650, 2023.

648 An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li,
 649 Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint*
 650 *arXiv:2412.15115*, 2024.

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

653 Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhen-
 654 guo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions
 655 for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

656 Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi
 657 Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sqlang: Efficient execution of
 658 structured language model programs. *Advances in neural information processing systems*, 37:
 659 62557–62583, 2024.

660 Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. Deductive beam search: Decoding deducible rationale
 661 for chain-of-thought reasoning. *arXiv preprint arXiv:2401.17686*, 2024.

662

663 A DATASETS AND METRICS

664

665 **Datasets** We consider four datasets to evaluate OverIDE’s effectiveness across different do-
 666 mains, including code generation, mathematical reasoning, and story generation:

- 667 • **HumanEval** (Chen et al., 2021b): a code generation dataset consisting of 164 program-
 668 ming problems that require to generate Python functions based on function descriptions;
- 669 • **MATH500** (Hendrycks et al., 2021): a standard 500-sample subset of the original MATH
 670 dataset with 12,500 competition mathematics problems;
- 671 • **GSM8K** (Cobbe et al., 2021): a dataset of 8,500 grade school math word problems;
- 672 • **CCNews**: a subset of the Common Crawl (Common Crawl, 2007) corpus that contains
 673 news articles from news sites all over the world.

674 **Evaluation metrics** To evaluate the quality and diversity of multiple generated responses, we
 675 employ the following metrics:

- 676 • **PASS@k**: Following previous work (Chen et al., 2021a; Grattafiori et al., 2024), we use
 677 the PASS@k accuracy to evaluate the correctness of multiple generated solutions. PASS@k
 678 estimates the pass rate of a problem if allowed to sample k times.
- 679 • **CodeBLEU** (Ren et al., 2020): For code generation tasks, we use CodeBLEU to evaluate
 680 the similarity of generated code snippets. CodeBLEU is a weighted combination of BLEU,
 681 BLEU-weighted, AST match and data-flow match scores. We calculate the pairwise Code-
 682 BLEU scores between all generated code snippets for each context:

$$683 \text{CodeBLEU}_i = \frac{1}{\binom{n}{2}} \sum_{j=1}^n \sum_{k=j+1}^n \text{CodeBLEU}(\mathbf{r}_{i,j}, \mathbf{r}_{i,k}), \quad (4)$$

684 where $\mathbf{r}_{i,j}$ is the j -th code snippet for the i -th context, and n is the number of code snippets.

- 685 • **Cosine similarity**: To measure output diversity, we compute pairwise cosine similarities
 686 between the embeddings of generated responses. For each context \mathbf{c}_i , we obtain embed-
 687 dings $\phi(\mathbf{r}_{i,j})$ for each response $\mathbf{r}_{i,j}$ using the OpenAI text-embedding-3-small¹ embedding
 688 model, and calculate the average pairwise similarity:

$$689 \text{Similarity}_i = \frac{1}{\binom{n}{2}} \sum_{j=1}^n \sum_{k=j+1}^n \frac{\langle \phi(\mathbf{r}_{i,j}), \phi(\mathbf{r}_{i,k}) \rangle}{\|\phi(\mathbf{r}_{i,j})\|_2 \|\phi(\mathbf{r}_{i,k})\|_2}, \quad (5)$$

690 where $\mathbf{r}_{i,j}$ is the j -th response for the i -th context, and n is the number of responses.

691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701

¹<https://platform.openai.com/docs/models/text-embedding-3-small>

702 • **MAUVE** (Pillutla et al., 2021): We use MAUVE to measure the similarity between model
 703 generations and human answers. MAUVE is obtained by computing the KL divergence
 704 between the two distributions in a quantized embedding space of a foundation model.
 705

706 **B IMPLEMENTATION DETAILS**
 707

708 **OverRIDE Configuration** We conduct all experiments on 8 NVIDIA L40S GPUs. For the
 709 reweighting parameter λ , we use $\lambda = 0.8$ for Qwen-2.5-7B (Yang et al., 2024) and $\lambda = 0.4$ for
 710 Mistral-7B (Jiang et al., 2023), based on our sensitivity analysis in Section 3.6. When fine-tuning
 711 the output head adapters, we use a learning rate of 1e-3. W_A is initialized with normal distribution
 712 $N(0, 0.02)$, and matrix W_B is initialized with zeros.
 713

714 **vLLM Configuration** We use the default settings of vLLM (Kwon et al., 2023) except for the
 715 following: gpu memory utilization is set to 0.8 for all the experiments; we use a tensor parallel size
 716 of 1 for 3B and 7B models, 2 for 14B models, 4 for 32B models, and 8 for 72B models.
 717

718 **Sampling Methods** We implement sampling methods with the following parameters on all the
 719 datasets: (1) Top- p sampling (Holtzman et al., 2019) with top- $p = 0.9$; (2) Top- k sampling (Fan
 720 et al., 2018) with $k = 20$ and temperature $\tau = 0.6$; (3) Min- p sampling (Nguyen et al., 2024) with
 721 min- $p = 0.05$ and temperature $\tau = 0.6$.
 722

723 **C PROMPTS**
 724

725 Table 6 shows the prompts used in our experiments. The prompt for HumanEval (Chen et al., 2021b)
 726 is adjusted from EvalPlus (Liu et al., 2023).
 727

728 Table 6: Prompts used in our experiments on HumanEval, MATH, GSM8K, and CCNews.
 729

730 Dataset	731 Prompt
732 HumanEval	733 Please provide a self-contained Python script that solves 734 the following problem in a markdown code block: 735 ```python 736 ⟨ Problem ⟩ 737 ```
738 MATH	739 ⟨ Problem ⟩ 740 Let's think step by step and output the final answer within 741 \boxed{ }.
742 GSM8K	743 ⟨ Problem ⟩ 744 Let's think step by step and output the final answer within 745 \boxed{ }.
746 CCNews	747 ⟨ First 32 tokens of the news ⟩

756 **D RESULTS ON GSM8K**
757758 Table 7 shows the results on GSM8K (Cobbe et al., 2021). We report PASS@k accuracy and pair-
759 wise cosine similarity. For PASS@5 accuracy, PASS@10 accuracy, and the similarity score, we
760 present paired results where the left value represents the baseline sampling method, and the right
761 value represents the same method integrated with OverRIDE.
762763 Table 7: Results of different sampling methods on GSM8K.
764

765 Model	766 Method	767 GSM8K			
		768 PASS@1	769 PASS@5	770 PASS@10	771 Similarity \downarrow
772 Qwen 773 2.5-7B	Greedy	774 89.8	775 89.8 / 94.7	776 89.8 / 95.6	777 1.000 / 0.951
	Top- p , $\tau = 0.6$	778 89.6	779 94.8 / 95.2	780 95.8 / 96.3	781 0.960 / 0.947
	Top- p , $\tau = 1.0$	782 88.8	783 95.1 / 95.1	784 96.0 / 96.5	785 0.947 / 0.940
	Top- k	786 89.6	787 94.9 / 95.2	788 96.0 / 96.3	789 0.955 / 0.945
	Min- p	790 89.7	791 95.0 / 95.0	792 95.8 / 96.3	793 0.957 / 0.947
794 Mistral 795 7B	Greedy	796 39.5	797 39.5 / 65.6	798 39.5 / 74.2	799 1.000 / 0.938
	Top- p , $\tau = 0.6$	800 39.2	801 68.5 / 69.4	802 77.4 / 78.2	803 0.927 / 0.920
	Top- p , $\tau = 1.0$	804 36.9	805 69.6 / 69.9	806 79.6 / 80.2	807 0.910 / 0.907
	Top- k	808 37.9	809 69.1 / 69.6	810 78.0 / 80.0	811 0.921 / 0.915
	Min- p	812 38.5	813 68.9 / 69.6	814 77.8 / 79.1	815 0.924 / 0.918

778 The results on GSM8K further validate our main findings. OverRIDE consistently improves both
779 quality (PASS@5 and PASS@10) and diversity (lower similarity scores) across all sampling
780 methods and temperature settings for both models. These results confirm OverRIDE’s effectiveness in
781 mathematical reasoning tasks, suggesting that the dynamic reweighting mechanism prevents over-
782 confidence and encourages exploration of diverse yet accurate responses.
783784 **E COMPARISON WITH OTHER DIVERSE DECODING METHODS**
785786 We compare OverRIDE with two additional diversity-enhancement decoding baselines: Contrastive
787 Decoding (CD) (Li et al., 2022a) and SimCTG (Su et al., 2022). For Contrastive Decoding, follow-
788 ing the original paper, we conduct experiments on GPT2-XL. We set the amateur model as GPT2,
789 amateur model temperature $\tau_{\text{amateur}} = 0.5$, and plausibility threshold $\alpha = 0.1$. We set $\lambda = 0.4$
790 OverRIDE. For SimCTG, we set degeneration penalty $\alpha = 0.6$, and top- k prediction $k = 4$.
791792 Table 8 shows the comparison results with Contrastive Decoding on GPT2-XL on CCNews. Table 9
793 shows the comparison results with SimCTG on Qwen-2.5-7B-Instruct and Mistral-7B-Instruct-v0.3.
794 OverRIDE achieves better diversity with better quality compared to the baselines.
795796 Table 8: Comparison with Contrastive Decoding on GPT2-XL.
797

798 Model	799 Method	800 MAUVE \uparrow	801 Similarity \downarrow
		802 CD	803 0.752
804 GPT2-XL	805 OverRIDE	806 0.846	807 0.664

808 Table 9: Comparison with SimCTG on Qwen-2.5-7B and Mistral-7B
809

810 Model	811 Method	812 HumanEval		813 MATH500		814 CCNews	
		815 PASS@10	816 CodeBLEU \downarrow	817 PASS@10	818 Similarity \downarrow	819 MAUVE \uparrow	820 Similarity \downarrow
821 Qwen 822 2.5-7B	823 SimCTG	824 77.2%	825 0.747	826 86.8%	827 0.954	828 0.976	829 0.756
	823 OverRIDE	824 86.4%	825 0.610	826 87.3%	827 0.936	828 0.938	829 0.678
830 Mistral 831 7B	832 SimCTG	833 56.1%	834 0.528	835 36.8%	836 0.868	837 0.915	838 0.758
	832 OverRIDE	833 56.9%	834 0.474	835 37.4%	836 0.862	837 0.952	838 0.703

810 F TUNING AND SAFEGUARD IMPLEMENTATION FOR λ
811812 We recommend to tune λ on a small validation set, starting from 0 and gradually increasing it until
813 the desired diversity is achieved. Although not presented in our experiments, in extreme cases when
814 the optimal λ is too high, the reweighting ratio $(p(x_i | x_{<i})/q_t(x_i | x_{<i}))^\lambda$ may explode. Thus, we
815 additionally provide a safeguard to prevent the reweighting ratio from exploding. Specifically, we
816 can clip the reweighting ratio with a clipping threshold β :
817

818
$$p_t(x_i | x_{<i}) = \frac{1}{Z} \text{clip} \left(\left(\frac{p(x_i | x_{<i})}{q_t(x_i | x_{<i})} \right)^\lambda, \frac{1}{\beta}, \beta \right) p(x_i | x_{<i}), \quad (6)$$

819
820

821 where $p(x_i | x_{<i})$ is the distribution from the original model, $q_t(x_i | x_{<i})$ is the distribution from
822 the guide model at round t , λ is the reweighting factor, and Z is the normalization term.
823824 G CASE STUDY
825826 To intuitively illustrate how OverRIDE works, we provide a qualitative analysis on the token prob-
827 ability changes across decoding rounds in two distinct scenarios in Table 10. These examples high-
828 light how our method balances between improving diversity and preserving quality.
829830 Table 10: Next-token probability (%) comparison across decoding rounds for OverRIDE.
831

(a) Case 1: Diversity				(b) Case 2: Quality			
Prompt:	To calculate the area of a triangle, the first step is to _____			Prompt:	The Fibonacci sequence can be defined recursively: $F(n) = F(n-2) + F(n-1)$		
Token	Round 1	Round 2	Round 10	Token	Round 1	Round 2	Round 10
measure	21.3	17.2	18.7	1	100.0	100.0	100.0
determine	18.0	19.2	19.4	2	0.0	0.0	0.0
calculate	17.0	11.8	15.4	$\langle \text{space} \rangle$	0.0	0.0	0.0
find	15.8	19.7	14.6	3	0.0	0.0	0.0
identify	14.7	11.7	15.2)\$	0.0	0.0	0.0
know	1.6	2.6	1.5	0	0.0	0.0	0.0

844 Case 1 illustrates OverRIDE’s effectiveness in encouraging diversity when multiple valid continu-
845 ations exist. For the same mathematical problem with multiple valid choices as the first step, the
846 model’s token preferences evolve over the rounds. This dynamic reweighting illustrates how Over-
847 RIDE continuously adapts based on previously generated responses, encouraging exploration of
848 equally valid but less-traveled paths in the generation space.
849850 Case 2 illustrates OverRIDE’s ability to preserve quality when there is only a single correct answer.
851 For the definition of Fibonacci sequence, $F(n) = F(n-2) + F(n-1)$ is the only mathematically
852 correct completion in this case. OverRIDE maintains 100% probability for the token ”1” across all
853 rounds, demonstrating that our method does not force diversity at the expense of correctness.
854855 These cases highlight OverRIDE’s core strength: the ability to distinguish between scenarios where
856 diversity should be encouraged and where precision should be maintained. This ability derives from
857 the reweighting mechanism. When a token is originally assigned with high probability, reweighting
858 does not dramatically increase the probability of other tokens. When multiple tokens are assigned
859 with similar probabilities, reweighting encourages the model to explore less likely options.
860

864 **H EXAMPLES OF GENERATED OUTPUTS**
865866 Table 11: Examples of Qwen-2.5-7B generated responses on MATH500.
867

868 Method	869 Top-p, $\tau = 0.3$	870 Top-p, $\tau = 0.3 + \text{OverRIDE}$
871 Context	872 Determine if the graph of the equation below is a parabola, circle, ellipse, hyperbola, point, line, two lines, or empty. 873 $(\frac{x}{2} - 3)^2 + y^2 = 10$	874 The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. This equation is 875 in the standard form of a circle, which is $(x - h)^2 + (y - k)^2 = r^2$, 876 where (h, k) is the center of the circle and r is the radius. 877 To see this more clearly, we can rewrite the given equation as: 878 $(\frac{x}{2} - 3)^2 + (y - 0)^2 = 10$. Here, we can identify $h = 6$ 879 (since $\frac{x}{2} - 3 = \frac{x-6}{2}$), $k = 0$, and $r^2 = 10$. Therefore, the 880 center of the circle is $(6, 0)$ and the radius is $\sqrt{10}$. Since the 881 equation matches the standard form of a circle, the graph of the 882 given equation is a circle. Thus, the answer is circle ×
883 Response 1	884 The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. To determine 885 the type of conic section this represents, we can compare it to the 886 standard forms of conic sections. The general form for a circle is 887 $(x - h)^2 + (y - k)^2 = r^2$, where (h, k) is the center and r is the radius. 888 We can rewrite the given equation to match this form: 889 $(\frac{x}{2} - 3)^2 + y^2 = 10$. Notice that $(\frac{x}{2} - 3)^2$ can be rewritten 890 as $(\frac{x-6}{2})^2$. So the equation becomes: $(\frac{x-6}{2})^2 + y^2 = 10$. 891 This is in the form $(x - h)^2 + (y - k)^2 = r^2$ where $h = 6$, 892 $k = 0$, and $r^2 = 10$. Therefore, $r = \sqrt{10}$. Since the equation 893 matches the standard form of a circle, the graph of the equation 894 is a circle. Thus, the answer is circle ×	895 To determine the type of conic section represented by the equation 896 $(\frac{x}{2} - 3)^2 + y^2 = 10$, we can compare it to the standard 897 forms of conic sections. First, let's rewrite the equation 898 in a more recognizable form: $(\frac{x}{2} - 3)^2 + y^2 = 10$. This 899 equation is in the form $(X - h)^2 + (Y - k)^2 = r^2$, which 900 is the standard form of a circle. Here, $X = \frac{x}{2}$ and $Y = y$, and 901 the center of the circle is at $(h, k) = (6, 0)$ because $\frac{x}{2} - 3 = X$ 902 implies $X = \frac{x}{2} - 3$ and thus $x = 2(X + 3)$, so the center 903 in terms of x and y is $(6, 0)$. The radius r is $\sqrt{10}$. Therefore, 904 the given equation represents a circle. The answer is circle ×
905 Response 2	906 The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. To determine 907 the type of conic section this represents, we can compare it to the 908 standard forms of conic sections. First, let's rewrite the equation 909 in a more recognizable form: $(\frac{x}{2} - 3)^2 + y^2 = 10$. This 910 equation is in the form $(X - h)^2 + (Y - k)^2 = r^2$, which 911 is the standard form of a circle. Here, $X = \frac{x}{2}$, $h = 3$, $Y = y$, 912 $k = 0$, and $r^2 = 10$. Therefore, $r = \sqrt{10}$. So, the equation 913 $(\frac{x}{2} - 3)^2 + y^2 = 10$ represents a circle with center $(6, 0)$ 914 and radius $\sqrt{10}$. Thus, the answer is circle ×	915 To determine the type of conic section represented by the equation 916 $(\frac{x}{2} - 3)^2 + y^2 = 10$, we can compare it to the standard 917 forms of conic sections. First, let's rewrite the equation 918 in a more recognizable form by expanding and simplifying it: 919 1. The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. 2. Expand 920 $(\frac{x}{2} - 3)^2$: $(\frac{x}{2} - 3)^2 = (\frac{x}{2})^2 - 2 \cdot \frac{x}{2} \cdot 3 + 3^2 =$ 921 $\frac{x^2}{4} - 3x + 9$. 3. Substitute this back into the original equation: 922 $\frac{x^2}{4} - 3x + 9 + y^2 = 10$. 4. Simplify by moving 923 10 to the left side: $\frac{x^2}{4} - 3x + 9 + y^2 - 10 = 0$. 5. To better 924 understand the form, we can complete the square for the x 925 terms: $\frac{x^2}{4} - 3x = \frac{1}{4}(x^2 - 12x)$. Completing the square 926 inside the parentheses: $x^2 - 12x = (x - 6)^2 - 36$. Thus, 927 $\frac{1}{4}(x^2 - 12x) = \frac{1}{4}((x - 6)^2 - 36) = \frac{(x - 6)^2}{4} - 9$. Substitute 928 this back into the equation: $\frac{(x - 6)^2}{4} - 9 + y^2 - 10 = 0$. Simplify 929 the constants: $\frac{(x - 6)^2}{4} + y^2 = 10$. Divide through by 10 to get 930 the equation in standard form: $\frac{(x - 6)^2}{40} + \frac{y^2}{10} = 1$. This is the 931 standard form of an ellipse, $\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$, where 932 (h, k) is the center of the ellipse, $a^2 = 40$, and $b^2 = 10$. 933 Therefore, the graph of the equation $(\frac{x}{2} - 3)^2 + y^2 = 10$ is 934 an ellipse. The final answer is: ellipse ✓
935 Response 3	936 The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. To determine 937 the type of conic section this represents, we can compare it to the 938 standard forms of conic sections. First, let's rewrite the equation 939 in a more recognizable form: $(\frac{x}{2} - 3)^2 + y^2 = 10$. This 940 equation is in the form $(X - h)^2 + (Y - k)^2 = r^2$, which 941 is the standard form of a circle. Here, $X = \frac{x}{2}$, $h = 3$, $Y = y$, 942 $k = 0$, and $r^2 = 10$. Therefore, $r = \sqrt{10}$. So, the equation 943 $(\frac{x}{2} - 3)^2 + y^2 = 10$ represents a circle with center $(6, 0)$ 944 and radius $\sqrt{10}$. Thus, the answer is circle ×	945 The given equation is $(\frac{x}{2} - 3)^2 + y^2 = 10$. To determine 946 the type of conic section this equation represents, we can rewrite it 947 in a more recognizable form. Let's start by simplifying and 948 comparing it to the standard forms of conic sections. 1. **Rewrite 949 the equation:** $(\frac{x}{2} - 3)^2 + y^2 = 10$. 2. **Isolate the squared 950 terms:** This equation is already in a form that resembles the 951 standard form of an ellipse, $\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$, but we 952 need to manipulate it to match this form. 3. **Divide both sides 953 by 10:** $\frac{(\frac{x}{2} - 3)^2}{10} + \frac{y^2}{10} = 1$. 4. **Simplify the denominators:** 954 $\frac{(\frac{x}{2} - 3)^2}{10} = \frac{(\frac{x-6}{2})^2}{10} = \frac{(x-6)^2}{40}$. So the equation 955 becomes: $\frac{(x-6)^2}{40} + \frac{y^2}{10} = 1$. 5. **Compare with the standard 956 form of an ellipse:** The equation $\frac{(x-6)^2}{40} + \frac{y^2}{10} = 1$ matches 957 the standard form of an ellipse $\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$, where 958 $a^2 = 40$ and $b^2 = 10$. Since the equation matches the standard 959 form of an ellipse, the graph of the given equation is an 960 ellipse. ellipse ✓
961 Similarity	962 0.983	963 0.795

918
919
920
921
922
923
924
925
926
927
928

Table 12: Examples of Qwen-2.5-7B generated responses on GSM8K.

Method	Top- <i>k</i>	Top- <i>k</i> + OverIDE
Context	How much does it cost you for lunch today at Subway if you pay \$40 for a foot-long fish sub and thrice as much for a six-inch cold-cut combo sub?	
Response 1	<p>To determine the total cost of lunch today at Subway, we need to add the cost of the foot-long fish sub to the cost of the six-inch cold-cut combo sub. First, we know that the cost of the foot-long fish sub is \$40. Next, we need to calculate the cost of the six-inch cold-cut combo sub. According to the problem, this cost is three times the cost of the foot-long fish sub. So, we multiply the cost of the foot-long fish sub by 3: $40 \times 3 = 120$ Now, we add the cost of the foot-long fish sub to the cost of the six-inch cold-cut combo sub to find the total cost of lunch: $40 + 120 = 160$ Therefore, the total cost for lunch today at Subway is 160. ✓</p>	<p>To determine the total cost for lunch today at Subway, we need to calculate the cost of the items mentioned and then sum them up. 1. The cost of the foot-long fish sub is given as \$40. 2. The cost of the six-inch cold-cut combo sub is three times the cost of the foot-long fish sub. Therefore, we calculate the cost of the six-inch cold-cut combo sub as follows: $3 \times 40 = 120$ 3. Now, we add the cost of the foot-long fish sub and the six-inch cold-cut combo sub to find the total cost: $40 + 120 = 160$ Thus, the total cost for lunch today is 160. ✓</p>
Response 2	<p>To determine the total cost of lunch for the day, we need to calculate the cost of both the foot-long fish sub and the six-inch cold-cut combo sub.</p> <p>First, we know the cost of the foot-long fish sub is \$40.</p> <p>Next, we are told that the six-inch cold-cut combo sub costs three times as much as the foot-long fish sub. Therefore, we calculate the cost of the six-inch cold-cut combo sub as follows: $3 \times 40 = 120$</p> <p>Now, we add the cost of the foot-long fish sub and the six-inch cold-cut combo sub to find the total cost of lunch: $40 + 120 = 160$</p> <p>Thus, the total cost for lunch today is 160. ✓</p>	<p>It appears there's a misunderstanding in your question. As an AI, I don't consume food or have a cost for meals. The information you've provided about the prices of the foot-long fish sub and the six-inch cold-cut combo sub at Subway is unrelated to my functioning or costs. These prices are for illustrative purposes and are not related to any actual expenses for me.</p> <p>If you're asking about the cost of a meal for an AI, it would be zero because AI entities do not consume food or require payment for functionality. If you have any other questions or need further assistance with calculations or information related to Subway's menu prices, feel free to ask! ✗</p>
Response 3	<p>To determine the total cost of the lunch, we need to calculate the cost of each item and then sum them up. First, we know the cost of the foot-long fish sub is \$40. Next, we need to find out the cost of the six-inch cold-cut combo sub. The problem states that the six-inch cold-cut combo sub costs three times as much as the foot-long fish sub. Therefore, we calculate: $3 \times 40 = 120$. So, the six-inch cold-cut combo sub costs \$120. Now, we add the cost of the foot-long fish sub and the six-inch cold-cut combo sub to get the total cost of the lunch: $40 + 120 = 160$. Thus, the total cost of the lunch is 160. ✓</p>	<p>To calculate the total cost of the lunch, we start by identifying the cost of each item and then summing them up.</p> <p>1. The cost of the foot-long fish sub is given as \$40.</p> <p>2. The cost of the six-inch cold-cut combo sub is three times the cost of the foot-long fish sub. Therefore, we calculate: Cost of six-inch cold-cut combo sub = $3 \times \\$40 = \\120</p> <p>3. Adding the two costs together gives the total cost of the lunch: Total cost = $\\$40 + \\$120 = \\$160$ So, the total cost for the lunch today at Subway is 160. ✓</p>
Response 4	<p>To determine the total cost of lunch for Subway, we need to calculate the cost of both the foot-long fish sub and the six-inch cold-cut combo sub.</p> <p>1. The cost of the foot-long fish sub is given as \$40.</p> <p>2. The cost of the six-inch cold-cut combo sub is three times the cost of the foot-long fish sub. Therefore, we calculate: $3 \times 40 = 120$ So, the six-inch cold-cut combo sub costs \$120.</p> <p>3. To find the total cost of lunch, we add the cost of the foot-long fish sub and the six-inch cold-cut combo sub: $40 + 120 = 160$ Thus, the total cost for lunch today at Subway is 160. ✓</p>	<p>It seems there might be a bit of a misunderstanding. As an artificial intelligence, I don't consume food or require meals, so the concept of paying for lunch doesn't apply to me. The scenario you've presented involves costs for food, but since I'm not a physical entity, I don't have personal expenses or the need to purchase lunch. Therefore, the question about how much it costs me for lunch today at Subway is not applicable. Not Applicable</p> <p>✗</p>
Similarity	0.927	0.626

964
965
966
967
968
969
970
971