

Length Value Model: Scalable Value Pretraining for Token-Level Length Modeling

Zhen Zhang¹, Changyi Yang^{2,4}, Zijie Xia⁴, Zhen Yang⁵, Chengzhi Liu¹, Zhaotiao Weng¹, Yepeng Liu¹, Haobo Chen¹, Jin Pan^{3,4}, Chenyang Zhao⁴, Yuheng Bu¹, Alkesh Patel⁵, Zhe Gan⁵, Xin Eric Wang¹

¹University of California, Santa Barbara, ²Carnegie Mellon University, ³University of Wisconsin–Madison, ⁴LMSYS Org, ⁵Apple Inc.

Token serves as the fundamental unit of computation in modern autoregressive models, and generation length directly influences both inference cost and reasoning performance. Despite its importance, existing approaches lack fine-grained length modeling, operating primarily at the coarse-grained sequence level. In this paper, we introduce the Length Value Model (LenVM), a token-level framework that models the remaining generation length at each decoding step. By formulating length modeling as a value estimation problem and assigning a constant negative reward to each generated token, LenVM predicts a bounded, discounted return that serves as a proxy for the remaining generation horizon. This formulation enables scalable, annotation-free value pretraining. Experiments on LLMs and VLMs demonstrate that LenVM provides a highly effective signal at inference time. On the LIFEbench exact length matching task, applying LenVM to a 7B model improves the length score from 30.9 to 64.8, significantly outperforming frontier closed-source models. Furthermore, LenVM enables continuous control over the trade off between performance and efficiency. On GSM8K at a budget of 200 tokens, LenVM maintains 63 percent accuracy compared to 6 percent for token budget baseline. It also accurately predicts total generation length from the prompt boundary. Finally, LenVM’s token-level values offer an interpretable view of generation dynamics, revealing how specific tokens shift reasoning toward shorter or longer regimes. Results demonstrate that LenVM supports a broad range of applications, including length control, prediction, and interpretation of generation dynamics. They suggest that generation length can be effectively modeled as a token-level value signal, highlighting the potential of LenVM as a general framework for length modeling and as accurate token-level value estimates for RL training.

Correspondence: zhen_zhang@ucsb.edu, ericxwang@ucsb.edu
Project Page: <https://github.com/eric-ai-lab/Length-Value-Model>

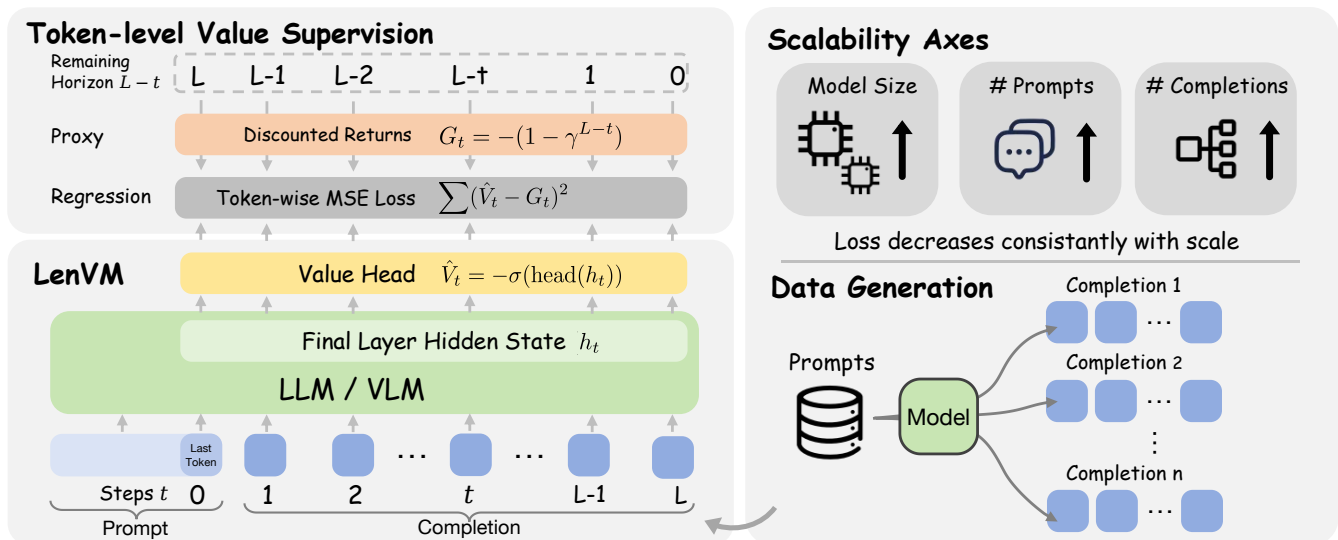


Figure 1. Length Value Model (LenVM) architecture and Pipeline. At each decoding step, LenVM attaches a scalar value head to the final-layer hidden state of an LLM or VLM to predict the value $V_t \in (-1, 0)$. LenVM enables annotation-free and scalable value pretraining across model size, prompt count, and number of completions.

Contents

1	Introduction	3
2	Length Value Model	4
2.1	Modeling Length as a Discounted Return	4
2.2	LenVM Head	5
2.3	Training Objective	5
3	Experiments: Validating LenVM as a Token-Level Length Signal	5
3.1	Experimental Setup	6
3.2	Application 1: Length-Controlled Generation	6
3.3	Performance–Efficiency Trade-off	7
3.4	Application 3: Generation Length Prediction	8
3.5	Scalability of LenVM	9
3.6	Case Study: Length Tokens as Markers of Length Shifts	10
4	Ablations	11
4.1	Length-Space Representation	12
4.2	Batch Construction Strategy	12
4.3	Discount Factor γ	12
4.4	Numerical Precision	12
5	Related Work	12
5.1	Length-Controlled Generation	12
5.2	Output Length Prediction	13
5.3	RL Framing and Reward Shaping of Generation Length	13
6	Conclusion	14
	Appendix	18
A	Additional Experimental Details	18
B	LIFEBench Evaluation Details	18
C	LenVM as an RL Reward Signal	19
C.1	Length Reward and Value Correspondence	19
C.2	Advantage Computation and Combination	20
C.3	Advantages over Sequence-Level Penalty	20
D	Precision Analysis for Length Inversion	21
D.1	Properties of $m(v)$ on $v \in (-1, 0)$	21
D.2	Visualization and asymptotics under the BF16 rounding model	21
E	Future-Dependent Weighting Changes the Regression Target	22
E.1	General Weighted Objective	22
E.2	Instantiations	23
F	Why Inverting the Transformed Horizon Underestimates Expected Remaining Length	24
G	Derivation of the Exponential Tilting Solution in the Performance–Efficiency Trade-off Experiment	25

1. Introduction

Token generation is the core computational process in modern AI systems. Large language models (LLMs), vision-language models (VLMs), and specialized models for code, mathematics, and agentic tasks all produce outputs sequentially. The number of generated tokens therefore forms a fundamental unit of inference-time computation. It influences both *performance*, since additional tokens can enable more reasoning in challenging settings (Snell et al., 2024), and *cost*, since each token incurs compute, KV-cache memory, latency, and energy (Kwon et al., 2023, Pope et al., 2022). Despite its importance, existing approaches to length modeling and control remain coarse-grained. Sequence-level penalties during training (Team et al., 2025), prompt-based instructions at inference (Xiao et al., 2026a), and pre-decode predictors (Zhang et al., 2025a) all operate at the level of whole sequences or fixed pre-generation decisions, rather than the token-level dynamics of generation itself. What is missing is a token-level model of remaining generation length.

To address this gap, we propose the **Length Value Model (LenVM)**, a token-level formulation of generation length modeling. At each decoding step, LenVM predicts a scalar value for the current decoding state. Conceptually, the quantity of interest is the remaining generation length, which depends on how much future generation is still to be realized. We model this quantity through a value formulation: by assigning a constant negative reward to each generated token and discounting future steps by γ , we define a discounted return that serves as a bounded proxy for the remaining generation horizon. LenVM is trained to predict this return, yielding a principled reduction from token-level length modeling to value estimation and placing the problem in a standard value-function framework. In this view, LenVM evaluates a decoding state by its remaining distance to completion rather than by task utility. This construction also maps highly variable raw lengths into a bounded target space, improving conditioning when generation lengths range from a few tokens to over 32k. The resulting value signal can be used at inference time for length estimation and control, and at training time as a length-aware value baseline.

A central advantage of LenVM is that it naturally supports **scalable value pretraining**. The resulting supervision signal has four favorable properties. First, it is **annotation-free**: training targets are constructed automatically from sampled completions, without any additional human labeling or reward modeling. Second, it is **dense**: every token position contributes a target, rather than only one target per completed response. Third, it is **exact** for each sampled trajectory: rewards and returns are deterministically computed from realized completion lengths, introducing no additional annotation or reward-model noise. Fourth, it is **scalable**: supervision grows naturally with both prompt count and the number of completions sampled per prompt, while remaining cheap to construct and well matched to increasing model size. Together, these properties make LenVM well suited to large-scale value pretraining, enabling high-quality token-level value supervision without the annotation bottlenecks of conventional reward and value modeling.

We evaluate whether LenVM learns a meaningful token-level signal of remaining generation length. At inference time, LenVM exposes a smooth **performance–efficiency trade-off**: exponentially tilting the token distribution with predicted values traces a Pareto frontier between response quality and generation length. The same signal also supports explicit **length control**, where LenVM-guided decoding improves adherence to hard constraints such as Equal To, At Most, and At Least on LIFEbench (Zhang et al., 2025b) (§ 3.2, § 3.3). In addition, LenVM provides useful **length prediction**, including from the very first generated token, across math, code, and instruction-following domains (§ 3.4). Its token-level values further offer an interpretable view of generation dynamics, revealing where trajectories shift between shorter and longer reasoning regimes (§ 3.6). Beyond inference, we show that LenVM admits a natural interpretation as a **length-specific value baseline** in PPO-style RL (Schulman et al., 2017), and we provide scaling evidence that its value-learning objective improves consistently with scale (§ 3.5). Together, these results establish LenVM as a principled

and scalable framework for token-level length modeling, while leaving full RL fine-tuning to future work.

2. Length Value Model

We propose the **Length Value Model** (LenVM), a token-level value function over decoding states for modeling generation length. As shown in Figure 1, LenVM treats autoregressive generation as an episodic process and models remaining generation length through a discounted return induced by a constant per-token cost. Rather than predicting raw remaining length directly, it predicts a bounded and monotone proxy of the remaining generation horizon. Starting from the last prompt token, LenVM outputs a scalar value at each decoding step. We first introduce the return formulation (§2.1), then describe the prediction head (§2.2) and training objective (§2.3).

2.1. Modeling Length as a Discounted Return

Let s_t denote the decoding state after the prompt and the first t generated tokens, with s_0 corresponding to the last prompt token. For a sampled completion of generated length L , decoding steps are indexed by $t \in \{0, \dots, L\}$, where $t = L$ is the EOS step.

We assign a constant negative reward at each non-terminal decoding step:

$$r_t = -(1 - \gamma), \quad t = 0, \dots, L - 1, \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor, and define $r_L = 0$. The factor $(1 - \gamma)$ is introduced purely for normalization, so the resulting return lies in a fixed range compatible with a sigmoid-bounded output.

The discounted return from step t is

$$G_t \triangleq \sum_{i=0}^{L-t} \gamma^i r_{t+i} = -(1 - \gamma^{L-t}). \tag{2}$$

Thus, for each non-terminal step, $G_t \in (-1, 0)$ and is a strictly monotone function of the remaining generation length $L - t$: states closer to termination have values nearer to 0, while states with longer continuations have values closer to -1 . Compared with raw length, this bounded transform preserves ordering while improving target scale and conditioning. It also satisfies the Bellman recursion

$$G_t = r_t + \gamma G_{t+1}, \tag{3}$$

placing token-level length modeling in a standard value-learning framework.

For a fixed model-induced decoding policy π , the corresponding length value function is

$$V^\pi(s_t) = \mathbb{E}_\pi[G_t \mid s_t]. \tag{4}$$

In practice, we use sampled trajectories and regress to their realized returns as Monte Carlo supervision. When an explicit estimate of remaining token count is needed, the transform can be inverted as

$$l = \frac{\log(1 + G_t)}{\log \gamma}. \tag{5}$$

This formulation is (1) **annotation-free**, since targets are computed directly from observed completion lengths; (2) **dense**, since every non-terminal step contributes a regression target; (3) **exact**, since the per-step

rewards and returns are deterministically computed from the realized trajectory, avoiding additional noise from human annotation, heuristic labeling, or learned reward models; and (4) **scalable**, since multiple completions can be sampled per prompt to automatically construct many supervised trajectories.

2.2. LenVM Head

LenVM is built on top of LLMs or VLMs by attaching a scalar value head to the final-layer hidden state at each decoding step:

$$z_t = \text{head}(h_t). \tag{6}$$

In this work, we instantiate $\text{head}(\cdot)$ as a two-layer MLP with SiLU activation,

$$z_t = W_2 \text{SiLU}(W_1 h_t + b_1) + b_2, \tag{7}$$

followed by

$$V_\theta(s_t) = -\sigma(z_t), \tag{8}$$

which ensures $V_\theta(s_t) \in (-1, 0)$.

2.3. Training Objective

For a minibatch of N prompt-completion trajectories, where the n -th trajectory has generated length $L^{(n)}$, we optimize the token-averaged mean squared error

$$\mathcal{L}_{\text{len}} = \frac{\sum_{n=1}^N \sum_{t=0}^{L^{(n)}-1} (V_\theta(s_t^{(n)}) - G_t^{(n)})^2}{\sum_{n=1}^N L^{(n)}}, \tag{9}$$

where

$$G_t^{(n)} = -(1 - \gamma^{L^{(n)}-t}). \tag{10}$$

Since $G_t^{(n)}$ is computed exactly from the observed trajectory, this objective corresponds to Monte Carlo regression with dense token-level supervision.

More generally, one may consider weighted variants of the same objective, where each token-level squared error is multiplied by a weight. In our setting, we avoid weights that depend on future rollout outcomes relative to the current decoding step, since such weights change the effective regression objective. Trajectory-level averaging (token-mean then sequence-mean) is one such case, as it induces a per-token weight inversely proportional to the realized completion length. We discuss this issue in Appendix E.

Relation to GAE. LenVM also admits GAE (Schulman et al., 2015). With $\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$, one can construct bootstrapped targets from temporal-difference residuals. In our experiments, however, $\lambda = 1$ performs best. This is consistent with the task structure: supervision is dense and directly recoverable from completed sequences.

3. Experiments: Validating LenVM as a Token-Level Length Signal

In this section, we evaluate at inference time how well LenVM captures the token-level signal of remaining generation length. Our experiments ask three questions: (i) Can LenVM support inference-time control

Table 1. Datasets used to train general LenVMs.

Domain	Dataset	Scale
Code	OpenCodeReasoning-2 (Ahmad et al., 2025) (Python)	1.42M
Instruction Following	WildChat (Zhao et al., 2024)	529k
Math	DeepMath-103K (He et al., 2025)	103k

without modifying the base generator? **(ii)** Can it predict generation length from the prompt boundary? **(iii)** Does its value-learning objective scale with model and data? We answer these questions through four evaluations: **(1) length-controlled generation** (§3.2), **(2) performance–efficiency trade-off** (§3.3), **(3) generation length prediction** (§3.4), and **(4) scalability of LenVM** (§3.5).

We note that applying LenVM during decoding requires additional forward passes and therefore introduces **extra inference latency**. The end-to-end latency is also affected by many engineering factors. Accordingly, our experiments are intended primarily to validate LenVM’s ability to model and control generation length, rather than to optimize wall-clock latency.

3.1. Experimental Setup

General LenVM. We train LenVMs on a multi-domain mixture of math, code, and instruction-following data rather than training separate domain-specific models. Table 1 summarizes the training mixture. For each prompt, we sample multiple completions and convert them into dense per-token regression targets using the discounted return in §2.

Models. For Qwen2.5 experiments (both LLMs and VLMs), we initialize LenVM from the **Qwen2.5-Instruct** family (Yang et al., 2024). For Qwen3 experiments, we initialize LenVM from **Qwen3-Base** models (Yang et al., 2025).

More experimental detail are elaborated in Appendix A.

3.2. Application 1: Length-Controlled Generation

We first evaluate whether LenVM can provide inference-time control signals for explicit length constraints. We evaluate on LIFE Bench (Zhang et al., 2025b), which contains 360 instances across question answering, summarization, reasoning, and creative generation, with 180 Chinese and 180 English examples. LIFE Bench defines three settings: *Equal To*, *At Most*, and *At Least*. Following LIFE Bench, we report **Length Score (LS↑)** for all settings and **Length Deviation (LD↓)** for *Equal To*. We consider target lengths from 32 to 1024 tokens. More details of LIFE Bench and our modification are given in Appendix B.

Hard-constraint decoding with LenVM. At each step, we first restrict the vocabulary to a candidate set \mathcal{V}_t using standard truncation strategies, then score each candidate token with LenVM on the next state $\hat{v}(x) = \hat{v}_\phi(s_t \oplus x)$. We then select tokens using:

- **Equal To:** First convert the remaining target length $L - t$ into value space v_t^* and select $\arg \min_{x \in \mathcal{V}_t} |\hat{v}(x) - v_t^*|$, favoring candidates whose predicted value is closest to the target v_t^* .

Table 2. Results on LIFEbench. We compare LenVM-guided hard-constraint decoding against frontier closed-source models. Following the LIFEbench reporting format, we present Equal To (Length Deviation↓, Length Score↑), At Most (Length Score↑), and At Least (Length Score↑). We report target lengths from 32 to 1024 tokens. LenVM equips open models to significantly outperform frontier proprietary models in exact length matching.

Model	Equal To		At Most Score↑	At Least Score↑
	Deviation↓	Score↑		
<i>Closed-source frontier models</i>				
GPT-4o	74%	35.5	77.9	98.5
GPT-5.4	135%	37.4	65.4	98.9
GPT-5.4-thinking	131%	47.8	72.7	98.9
Claude-Sonnet-4-6	105%	34.1	62.9	100.0
Claude-Sonnet-4-6-thinking	124%	51.3	69.3	100.0
Claude-Opus-4-6	66%	35.5	51.5	100.0
Claude-Opus-4-6-thinking	87%	<u>53.2</u>	67.4	100.0
Gemini-3-Flash-Preview	123%	40.3	57.3	99.6
Gemini-3.1-Pro-Preview	91%	49.3	70.7	100.0
<i>Qwen2.5-3B-Instruct</i>				
w/o LenVM	83%	25.6	92.1	94.6
w/ LenVM(1.5B)	56% (↓27pp)	62.6 (↑37.0)	93.0	93.1
<i>Qwen2.5-7B-Instruct</i>				
w/o LenVM	71%	30.9	98.5	89.1
w/ LenVM(1.5B)	44% (↓27pp)	64.8 (↑33.9)	96.1	99.5
<i>Qwen3-30B-A3B-Instruct</i>				
w/o LenVM	90%	36.8	87.0	99.3
w/ LenVM(1.7B)	57% (↓33pp)	67.2 (↑30.4)	99.4	99.8

- **At Least:** $\arg \min_{x \in \mathcal{V}_t} \hat{v}(x)$, favoring more negative values and hence longer continuations.
- **At Most:** $\arg \max_{x \in \mathcal{V}_t} \hat{v}(x)$, favoring values closer to 0 and hence earlier termination.

Results. Table 2 shows that LenVM substantially improves constraint following, especially in the *Equal To* setting. For example, on Qwen2.5-7B-Instruct, LenVM improves Length Score from 30.9 to 64.8 and reduces deviation from 71% to 44%. The comparison to closed-source models is also informative: those models rely on prompt-based control, which is inherently coarse-grained, and they struggle on *Equal To*, with deviations between 66% and 135% and scores below 54. By contrast, LenVM provides an explicit token-level control signal over the remaining horizon, allowing open models to achieve substantially stronger precision.

3.3. Performance–Efficiency Trade-off

We next test whether LenVM supports *continuous* control of the performance–efficiency trade-off without modifying the base generator. Instead of enforcing a hard target length, we reweight the next-token distribution so that tokens predicted to lead to shorter continuations become more likely.

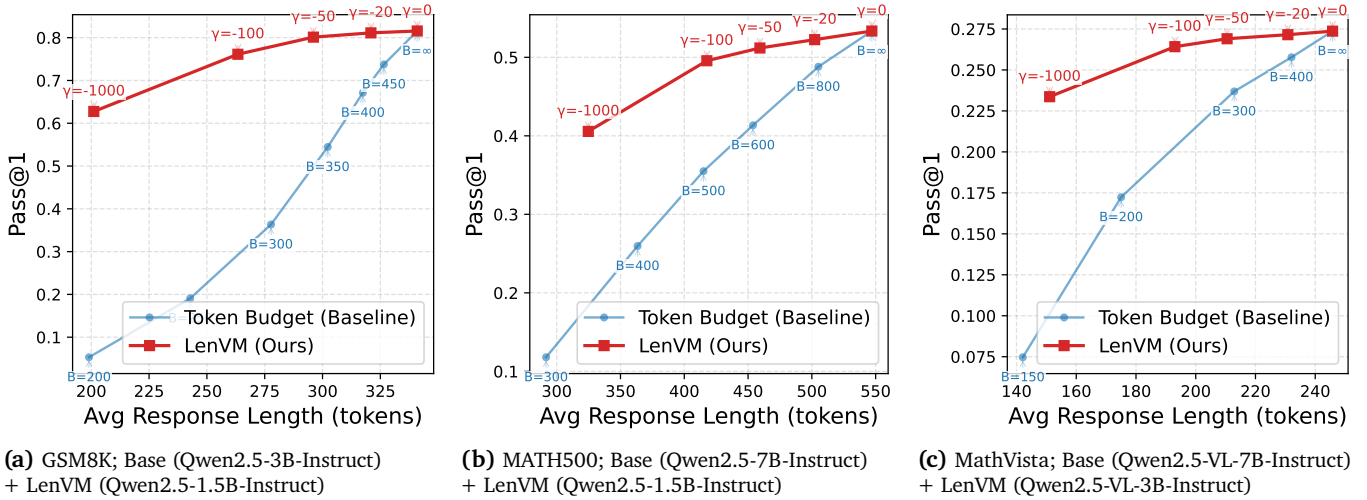


Figure 2. Performance (Pass@1) versus average generation length. LenVM-guided exponential tilting (red) consistently outperforms a hard token-budget baseline (blue). The x-axis reports the average *truncated* response length. Since the base generator is unchanged, the result shows that the model already has the ability to solve problems with shorter generations, and LenVM helps uncover those trajectories by reweighting token choices. Models in parentheses represent the initialization checkpoints.

Value-guided exponential tilting. Let p be the base model distribution over candidate set \mathcal{V}_i . We define

$$\min_{p'} \mathbb{E}_{p'}[\hat{v}(x)] - \frac{1}{\beta} D_{\text{KL}}(p' \| p), \quad (11)$$

whose solution is

$$p'(x) = \frac{p(x) \exp(\beta \hat{v}(x))}{\sum_{x' \in \mathcal{V}_i} p(x') \exp(\beta \hat{v}(x'))}. \quad (12)$$

The derivation is provided in Appendix G. Here $\beta < 0$ controls the steering strength. When $\beta = 0$, decoding reduces to the original model. As β becomes more negative, generation is increasingly biased toward shorter trajectories.

We evaluate on GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), and MathVista (Lu et al., 2023). As a baseline, we use a hard token budget that truncates generation once the token count exceeds threshold B and marks the example as incorrect. For fair comparison, the x-axis reports the average *truncated* generation length for both methods.

Figure 2 shows that LenVM-guided decoding consistently achieves a better performance–efficiency frontier than hard truncation. At the same average truncated length, LenVM retains much higher Pass@1. For example, on GSM8K with Qwen2.5-3B-Instruct, at roughly 200 tokens the hard budget baseline achieves about 6% Pass@1, whereas LenVM maintains about 63%. Because the base generator is never modified, this result indicates that the model itself already contains shorter successful reasoning paths. LenVM simply biases decoding toward them. Moreover, varying β changes the frontier smoothly, confirming that LenVM provides a continuous knob for balancing performance and token budget.

3.4. Application 3: Generation Length Prediction

Beyond control, we test whether LenVM can predict generation length from the prompt boundary, i.e., from state s_0 before any response token is emitted. This is useful for scheduling, batching, and memory planning

Table 3. Prompt-boundary length prediction accuracy. Mean Relative Error (MRE, lower is better) on three domains.

Model Size	MRE (\downarrow)		
	Math	Code	IF
1.5B	17.0%	29.0%	33.0%
3B	13.6%	24.0%	27.2%
7B	11.0%	19.5%	23.0%
14B	10.4%	17.0%	19.8%
32B	9.8%	14.9%	17.1%

at inference time.

To evaluate prediction accuracy, we use held-out prompts from math, code, and instruction-following data. For each prompt, we sample $N = 64$ completions with lengths L_1, \dots, L_N . Because LenVM predicts discounted remaining horizon rather than raw token count, we evaluate in the same transformed space:

$$u(L_i) = 1 - \gamma^{L_i}, \quad \mu_u = \frac{1}{N} \sum_{i=1}^N u(L_i), \tag{13}$$

and define the return-consistent ground-truth horizon as

$$L_{GT} = \frac{\ln(1 - \mu_u)}{\ln \gamma}. \tag{14}$$

We then invert LenVM’s prediction at s_0 into a length estimate \hat{L} and report Mean Relative Error (MRE):

$$\text{MRE}(\hat{L}, L_{GT}) \triangleq \mathbb{E} \left[\frac{|\hat{L} - L_{GT}|}{L_{GT}} \right]. \tag{15}$$

Further discussion of this target appears in Appendix F.

As shown in Table 3, LenVM predicts generation horizon from the prompt boundary with low relative error across all domains, and accuracy improves consistently with scale. At 32B, MRE reaches 9.8% on math, 14.9% on code, and 17.1% on instruction following. This shows that LenVM is not only useful for online control, but also captures predictive information about expected generation length before decoding begins.

3.5. Scalability of LenVM

A key advantage of LenVM is that its supervision is annotation-free, dense, and easy to scale. We therefore study scaling along three axes: **(i) model size**, **(ii) number of training questions**, and **(iii) number of samples per question**.

Figure 3 shows consistent improvements along all three axes. Larger models achieve lower validation loss, matching the gains in prompt-boundary prediction accuracy from Table 3. Increasing the number of training prompts also improves performance, indicating that LenVM benefits from broader coverage of tasks and trajectory types. Finally, increasing the number of sampled completions per prompt further reduces validation loss, showing that supervision can be scaled not only through more prompts but also through more sampled trajectories from the same prompts. Overall, these results support LenVM as a scalable value-pretraining objective.

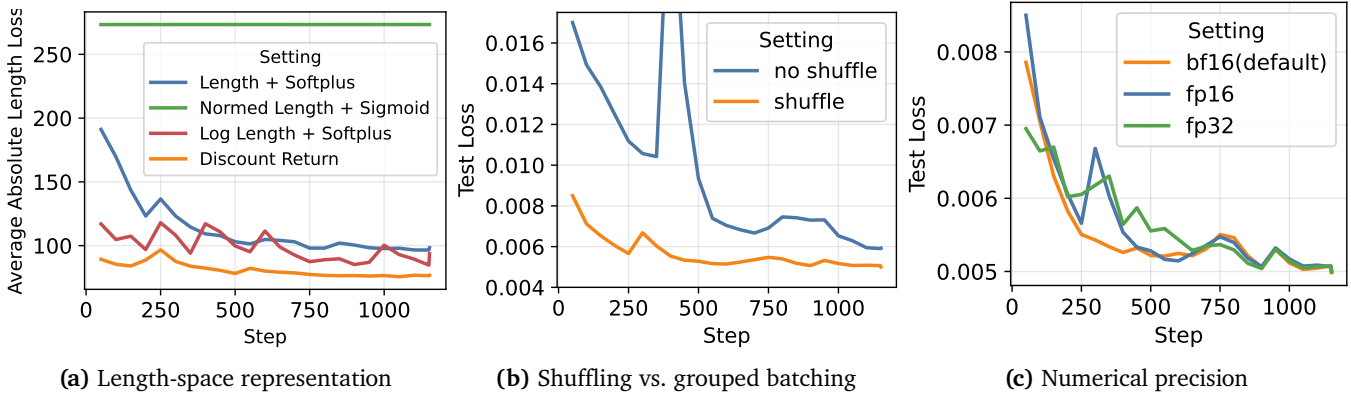


Figure 5. Ablations on LenVM design choices. Comparison of (a) target representation, (b) batch construction strategy, and (c) numerical precision.

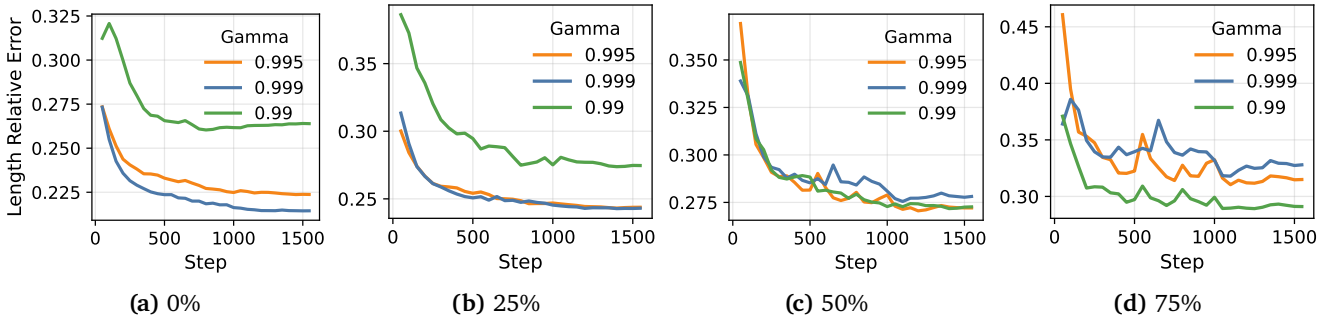


Figure 6. Ablation on the discount factor γ . Prediction error at different relative decoding positions. Larger γ performs better earlier in generation, while smaller γ performs better closer to termination.

cloud according to their frequency of occurrence. We use the completion from **Qwen3-30B-A3B-instruct** and 8k data from **DeepMath-103K** dataset.

Figure 4a shows that positive length tokens often resemble local reasoning pivots, including `ah`, `but`, `now`, `wait`, `let`, `think`, `try`, and `consider`. These tokens often appear when the model revises an earlier step, introduces a new subcase, or extends the current line of reasoning. In particular, `ah` frequently occurs in contexts such as “Ah! I see the mistake” or “Ah! Here’s an idea,” which are reminiscent of **Aha Moment** where the model has an epiphany.

By contrast, in Figure 4b, negative length tokens such as `therefore`, `clearly`, `perfect`, `\n\n`, and celebratory emoji markers such as `check-mark emoji` or `party emoji` are more often associated with closure, confirmation, or answer finalization. This analysis is descriptive rather than causal, but it suggests that LenVM provides not only a control signal, but also a simple token-level lens on generation dynamics.

4. Ablations

We ablate four design choices in LenVM: (i) **length-space representation**, (ii) **batch construction strategy**, (iii) **discount factor γ** , and (iv) **numerical precision (FP16/BF16/FP32)**. Unless otherwise specified, all experiments keep the model, optimizer, and evaluation protocol fixed.

4.1. Length-Space Representation

We first compare four target parameterizations in Figure 5a: (1) **Length + Softplus**, (2) **Normalized Length + Sigmoid**, (3) **Log Length + Softplus**, and our proposed (4) **Discount Return + Sigmoid**. Among them, **Discount Return + Sigmoid** consistently achieves the lowest average absolute length error throughout training. Raw-length and normalized-length regression perform worse, partly because generation lengths are highly long-tailed. In particular, normalizing by the maximum length compresses the dense short-length region into a narrow interval near zero, where discrimination becomes difficult. Log-length regression is a stronger baseline and substantially improves over raw or normalized length, but still underperforms discount return. Unlike a static scale transform such as log length, discount return is aligned with the stepwise Bellman recursion of autoregressive decoding, yielding a more coherent token-level training signal. This result shows that for token-level length modeling, directly regressing raw length or normalized length is ineffective, while *the discounted-return formulation provides a better-conditioned and more effective target*.

4.2. Batch Construction Strategy

We compare fully shuffled training against grouped batching that keeps multiple completions from the same prompt together, while holding the total number of data per update fixed. As shown in Figure 5b, shuffling consistently improves evaluation loss. This suggests that, unlike the reward model, grouping multiple samples from the same prompt within a batch is unnecessary and can mildly hurt generalization.

4.3. Discount Factor γ

The discount factor γ determines how remaining horizon is distributed in the target space, and therefore where prediction resolution is concentrated along the trajectory. Larger γ compresses long horizons more aggressively, which tends to improve prediction earlier in generation. Smaller γ allocates more resolution near termination, improving prediction later in the trajectory. To evaluate this trade-off, we measure prediction error at relative decoding positions 0%, 25%, 50%, and 75%. Figure 6 shows the expected pattern: larger γ performs better at earlier positions, while smaller γ performs better closer to the end. Thus, γ acts as a simple knob for balancing early-horizon compression against late-horizon discrimination. In practice, we choose an intermediate γ to balance early-horizon compression against late-horizon discrimination. Specifically, we set γ such that the 99th-percentile generation length $L_{0.99}$ satisfies $1 - \gamma^{L_{0.99}} = 0.99$, so that nearly all observed horizons are mapped into the high-resolution region of the target space.

4.4. Numerical Precision

We compare FP16, BF16, and FP32 under the same training and evaluation setup. Figure 5c shows nearly identical loss curves across all three formats, with no substantial difference in convergence or final performance. This indicates that LenVM is numerically stable under common floating-point precisions.

5. Related Work

5.1. Length-Controlled Generation

Length control has been explored through prompting, fine-tuning, and decoding-time constraints. Prompt-based methods can be attractive for black-box models: for example, Xie and Lee propose a one-shot countdown prompt that improves strict compliance without changing model weights (Xie and yi Lee, 2025).

Recent prompt-only methods also incorporate explicit structure and counting to regulate length without retraining; Plan-and-Write uses a plan-first scaffold and word-counting guidance to improve adherence in summarization tasks (Akinfaderin et al., 2025). For black-box settings where fine-tuning is not possible, Gu et al. (2024) propose an iterative sampling framework that combines Metropolis–Hastings with importance-sampling acceleration to achieve length-constrained generation without modifying model parameters. More broadly, constrained generation has a long history of MCMC-style inference-time methods; CGMH applies Metropolis–Hastings moves to satisfy lexical constraints without parallel training data (Miao et al., 2018), and recent work revisits MCMC constrained sampling with stronger distributional guarantees (Gonzalez et al., 2025). Training-based approaches can yield more precise control by modifying the model’s internal length representation; Butcher et al. (2024) introduce length-difference positional encoding (LDPE) and fine-tuning to encourage coherent termination near the target length, while Xie et al. (2026) propose Hansel, which uses periodically outputted hidden special tokens to track the remaining target length during finetuning. In contrast to approaches that embed length information into the generator, our method trains a *separate* token-level horizon estimator and uses it as a decoding-time control signal, enabling length control without retraining the base LLM.

5.2. Output Length Prediction

Predicting completion length is important for both serving efficiency and stochastic sampling, and can be approached either statically (before decoding) or progressively (during decoding). Recent work shows that hidden representations and entropy signals can provide accurate, low-overhead length prediction; in particular, Xie et al. (2026) propose entropy-guided token pooling for static prediction and progressive length prediction (PLP) for online remaining-length estimation. Beyond entropy-guided methods, Piotrowski et al. (2025) study forecasting remaining output length from frozen layerwise hidden states, exploring aggregation and graph-based regressors. In code generation, horizon-length prediction has also been used to improve fill-in-the-middle planning: Ding et al. (2025) propose a horizon-length prediction objective to better align infills with distant right context. Relatedly, Xiao et al. (2026b) examine whether LLMs can self-track output length and propose a dynamic feedback mechanism that adjusts generation online to meet length targets without retraining. Our LVM is also an online predictor, but is trained as a standalone value model with bounded regression targets and is designed to provide a stable per-token signal across diverse task families and length scales. Variable output lengths complicate batching and can waste compute through padding and fragmentation; length prediction can improve schedulers and load balancing. Zheng et al. (2023) study response length perception and sequence scheduling to improve inference throughput.

5.3. RL Framing and Reward Shaping of Generation Length

Our formulation views length regularization as a dense per-step penalty (i.e., negative reward) with an associated discounted return, making remaining horizon a value-function quantity. This makes LVM naturally compatible with PPO-style training as a length-specific value baseline and advantage component (Section C), while in this paper we focus on inference-time decoding control and analysis. More broadly, recent RLHF work argues for value-centric formulations: DVPO pretrains a global value model on preference data and uses it as a frozen critic for policy optimization (Huang et al., 2026), and V_0 learns a generalist value estimator at the initial prompt for policy-agnostic routing and resource allocation (Zhang et al., 2026). Recent work on efficient reasoning also studies how to adapt length penalties during RL to balance accuracy and conciseness (Xiang et al., 2025, Liu et al., 2025, Li et al., 2025). VAPO further targets stable and efficient value-based RL for long-CoT reasoning by addressing value bias, heterogeneous sequence lengths, and sparse rewards (Yue et al., 2025).

6. Conclusion

We proposed **LenVM**, a token-level value model for remaining generation length. By assigning a constant per-token reward and predicting the resulting discounted return, LenVM places length modeling in a standard value-learning framework with bounded, dense, and annotation-free supervision. Our experiments show that this simple formulation is both useful and scalable. LenVM supports precise length-controlled generation, exposes a smooth performance–efficiency trade-off without modifying the base generator, predicts expected generation horizon, and improves consistently with model scale and automatically collected supervision. These results suggest that discounted length value is an effective target for token-level length modeling and a promising substrate for scalable value pretraining.

References

- Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning-ii: A simple test time scaling approach via self-critique. *arXiv preprint arXiv:2507.09075*, 2025.
- Adewale Akinfaderin, Shreyas Subramanian, and Akarsha Sehwal. Plan-and-write: Structure-guided length control for llms without model retraining. *ArXiv*, abs/2511.01807, 2025. URL <https://api.semanticscholar.org/CorpusID:282739780>.
- Bradley Butcher, Michael O’Keefe, and James Titchener. Precise length control for large language models. *Nat. Lang. Process. J.*, 11:100143, 2024. URL <https://api.semanticscholar.org/CorpusID:274788732>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Yifeng Ding, Hantian Ding, Shiqi Wang, Qing Sun, Varun Kumar, and Zijian Wang. Planning-aware code infilling via horizon-length prediction, 2025. URL <https://arxiv.org/abs/2410.03103>.
- Emmanuel Anaya Gonzalez, Sairam Vaidya, Kanghee Park, Ruyi Ji, Taylor Berg-Kirkpatrick, and Loris D’antoni. Constrained sampling for language models should be easy: An mcmc perspective. *ArXiv*, abs/2506.05754, 2025. URL <https://api.semanticscholar.org/CorpusID:279245064>.
- Yuxuan Gu, Wenjie Wang, Xiaocheng Feng, Weihong Zhong, Kun Zhu, Lei Huang, Tat-Seng Chua, and Bing Qin. Length controlled generation for black-box llms. *ArXiv*, abs/2412.14656, 2024. URL <https://api.semanticscholar.org/CorpusID:274859461>.
- Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.
- Chenghua Huang, Lu Wang, Fangkai Yang, Pu Zhao, Zhixu Li, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Pretrain value, not reward: Decoupled value policy optimization, 2026. URL <https://arxiv.org/abs/2502.16944>.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Haotong Zhang, and Ion Stoica. Efficient memory management for large language model serving with

- pagedattention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL <https://api.semanticscholar.org/CorpusID:261697361>.
- Yanhao Li, Lu Ma, Jiaran Zhang, Lexiang Tang, Wentao Zhang, and Guibo Luo. Leash: Adaptive length penalty and reward shaping for efficient large reasoning model, 2025. URL <https://arxiv.org/abs/2512.21540>.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The twelfth international conference on learning representations*, 2023.
- Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning, 2025. URL <https://arxiv.org/abs/2510.15110>.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.
- Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling, 2018. URL <https://arxiv.org/abs/1811.10996>.
- Grzegorz Piotrowski, Mateusz Bystronki, Mikołaj Hołysz, Jakub Binkowski, Grzegorz Chodak, and Tomasz Jan Kajdanowicz. When will the tokens end? graph-based forecasting for LLMs output length. In Jin Zhao, Mingyang Wang, and Zhu Liu, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 843–848, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-254-1. doi: 10.18653/v1/2025.acl-srw.61. URL <https://aclanthology.org/2025.acl-srw.61/>.
- Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *ArXiv*, abs/2211.05102, 2022. URL <https://api.semanticscholar.org/CorpusID:253420623>.
- John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015. URL <https://api.semanticscholar.org/CorpusID:3075448>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL <https://api.semanticscholar.org/CorpusID:28695052>.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024. URL <https://api.semanticscholar.org/CorpusID:271719990>.
- Kimi Team, Angang Du, Bofei Gao, Bofei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Feng Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Haochen Ding, Hao-Xing Hu, Haoming Yang, Hao Zhang, Haotian Yao, Hao-Dong Zhao, Haoyu Lu, Haoze Li, Hao Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jia-Xing Guo, Jianling Su, Jianzhou Wang, Jie

- Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Li-Na Shi, Li tao Ye, Long Yu, Meng xiao Dong, Neo Y. Zhang, Ning Ma, Qi Pan, Qucheng Gong, Shaowei Liu, Shen Ma, Shu-Yan Wei, Sihan Cao, Si-Da Huang, Tao Jiang, Wei-Wei Gao, Weiming Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wen He, Xian sen Wei, Xian-Xian Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xue biao Pan, Y. Charles, Yang Li, Yan-Ling Hu, Yangyang Liu, Yanru Chen, Ye-Jia Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Yingbo Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhengxin Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziya Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599, 2025. URL <https://api.semanticscholar.org/CorpusID:275789974>.
- Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning, 2025. URL <https://arxiv.org/abs/2506.05256>.
- Meiman Xiao, Ante Wang, Qingguo Hu, Zhongjian Miao, Huangjun Shen, Longyue Wang, Weihua Luo, and Jinsong Su. Can llms track their output length? a dynamic feedback mechanism for precise length regulation. *ArXiv*, abs/2601.01768, 2026a. URL <https://api.semanticscholar.org/CorpusID:284486970>.
- Meiman Xiao, Ante Wang, Qingguo Hu, Zhongjian Miao, Huangjun Shen, Longyue Wang, Weihua Luo, and Jinsong Su. Can llms track their output length? a dynamic feedback mechanism for precise length regulation, 2026b. URL <https://arxiv.org/abs/2601.01768>.
- Huanyi Xie, Yubin Chen, Liangyu Wang, Lijie Hu, and Di Wang. Predicting llm output length via entropy-guided representations. *ArXiv*, abs/2602.11812, 2026. URL <https://api.semanticscholar.org/CorpusID:285540500>.
- Juncheng Xie and Hung yi Lee. Prompt-based one-shot exact length-controlled generation with llms. *ArXiv*, abs/2508.13805, 2025. URL <https://api.semanticscholar.org/CorpusID:280686321>.
- 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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiabin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shi-Qiang Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. 2025. URL <https://api.semanticscholar.org/CorpusID:278602855>.
- Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiabin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL <https://api.semanticscholar.org/CorpusID:274859421>.
- Yu Yue, Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, Wenyan Xu, Jiase Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang,

- Yonghui Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025. URL <https://arxiv.org/abs/2504.05118>.
- Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *ArXiv*, abs/2505.13417, 2025a. URL <https://api.semanticscholar.org/CorpusID:278769267>.
- Wei Zhang, Zhenhong Zhou, Kun Wang, Junfeng Fang, Yuanhe Zhang, Rui Wang, Ge Zhang, Xavier Li, Li Sun, Lingjuan Lyu, et al. Lifebench: Evaluating length instruction following in large language models. *arXiv preprint arXiv:2505.16234*, 2025b.
- Yi-Kai Zhang, Zhiyuan Yao, Hongyan Hao, Yueqing Sun, Qi Gu, Hui Su, Xunliang Cai, De-Chuan Zhan, and Han-Jia Ye. v_0 : A generalist value model for any policy at state zero, 2026. URL <https://arxiv.org/abs/2602.03584>.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.
- Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline, 2023. URL <https://arxiv.org/abs/2305.13144>.

Appendix

A. Additional Experimental Details

We train LenVM using LlamaFactory. The discount factor γ is set to 0.997 for **Qwen2.5-Instruct** and **Qwen2.5-VL-Instruct**, and to 0.9998 for **Qwen3-Instruct**. We set $\lambda = 1$ for GAE.

For the ablation and scalability experiments, we sample 100k examples from each dataset, except for math, where we use all 95k available examples. For the Length-Controlled Generation and Performance–Efficiency Trade-off experiments, we use the full training sets of all three datasets. We randomly sample 8k examples for validation during training. Unless otherwise specified, data sampling uses temperature 1.0 and top-p 1.0, and we sample up to 16 completions per prompt.

We train LenVM for 2 epochs with learning rate 2×10^{-5} , batch size 1024, and BF16 precision.

In **Length-Controlled Generation**, we use temperature 1.0, top-p 0.999, and top-k 15. All baseline settings follow the original LIFEbench paper, except that constraints are applied to token count in our experiments. We use the official tokenizer of each model, or the token count returned by the corresponding API response, to measure generation length. Although tokenization schemes differ across models, all methods are evaluated under the same token-based constraint setting.

In the **Performance–Efficiency Trade-off** experiments, we use temperature 1.0 and top-p 1.0. For LenVM guided decoding, we set min-p to 0.01 to reduce the candidate set size and avoid excessively large LenVM forward batches. We sample 64 completions for each question.

B. LIFEbench Evaluation Details

Benchmark overview. LIFEbench (Zhang et al., 2025b) is a comprehensive benchmark designed to evaluate the ability of large language models to follow explicit length instructions across diverse tasks and length scales. It comprises 10,800 (360 data, 3 length constraints, 10 length target) instances spanning four task categories:

- **Question Answering (QA):** Open-ended questions drawn from Quora, Stack Exchange, WikiHow, and Yahoo Answers (English), and from Zhihu and WikiHow (Chinese).
- **Summarization:** Long documents (10,000–15,000 words) sourced from scientific papers, government reports, book chapters, and Wikipedia (English), and from corporate annual reports, Wikipedia, and XueXiQiangGuo (Chinese).
- **Reasoning:** Eighty problems (40 per language) generated via GPT-4o, covering six reasoning types: deductive, inductive, abductive, analogical, causal, and probabilistic.
- **Creative Generation:** Writing prompts derived from movie scripts, story corpora, and news datasets (English), and from web novels, social commentary, and WeChat public articles (Chinese).

Length constraints. The original benchmark focuses on word and character lengths. Each instance is paired with one of three constraint types (*Equal To*, *At Most*, and *At Least*) and one of ten target lengths: 16, 32, 64, 128, 256, 512, 1,024, 2,048, 4,096, and 8,192 words for English or characters for Chinese. In our experiments, we modify these constraints to operate at the token level. Specifically, we prompt the model to generate a response where the total token count is equal to, at most, or at least the specified target length.

Evaluation metrics. Two metrics are used:

- **Length Deviation (LD):**

$$\text{LD} = \frac{L_{\text{output}} - L_{\text{constraint}}}{L_{\text{constraint}}}, \quad (17)$$

where L_{output} is the actual output length and $L_{\text{constraint}}$ is the target length. Positive values indicate over-generation; negative values indicate under-generation.

- **Length Score (LS):** A bounded score in $[0, 100]$ derived from LD via an exponential penalty function that is adapted to each constraint type:

$$\text{LS} = \begin{cases} 100 \cdot e^{k_1 \cdot \text{LD}}, & \text{LD} < 0 \quad (\text{Equal To / At Least}), \\ 100 \cdot e^{-k_2 \cdot \text{LD}}, & \text{LD} > 0 \quad (\text{Equal To / At Most}), \\ 100, & \text{otherwise,} \end{cases} \quad (18)$$

with $k_1 = 5$ and $k_2 = 2$. A score of 100 indicates perfect adherence; lower scores reflect larger deviations.

Implementation details. Each prompt follows the template [Instruction] + [Requirement], where the instruction specifies the task and input content, and the requirement states the length constraint in natural language (e.g., “Your answer must be equal to 512 tokens.”).

C. LenVM as an RL Reward Signal

In this section, we discuss how to integrate the Length Value Model (LenVM) as a learned signal for length-aware Reinforcement Learning (RL). The per-step constant penalty introduced in Section 2 induces a shaped return that monotonically reflects the remaining horizon. The LenVM provides an accurate, learned estimate of this discounted return at each step, making it a natural fit as a value baseline for length penalties. By decomposing the optimization objective into a task-specific component and a length-specific penalty, one can leverage the LenVM to improve credit assignment and training stability. **Importantly, this section is conceptual:** in this paper we do not perform RL fine-tuning with the LenVM; instead, we use the LenVM purely at **inference time** to guide length-controlled decoding (Section 3.2 and Section 3.3).

C.1. Length Reward and Value Correspondence

In standard RL fine-tuning (e.g., PPO), the agent receives a task reward r_t^{task} , which is often sparse and given only at the end of the sequence (or combined with a per-step KL penalty). To discourage unnecessarily long generations, we introduce an additional length reward. Following the formulation in Section 2, we assign a constant, dense penalty at each step:

$$r_t^{\text{length}} = -(1 - \gamma). \quad (19)$$

Consequently, the true expected return for this length reward is exactly the discounted return $G_t = -(1 - \gamma^{L-t})$. The LenVM prediction, $\hat{v}_\phi(s_t) \in (-1, 0)$, directly estimates this expected return. Thus, the LenVM naturally serves as a pre-trained or co-trained value function for the length reward:

$$V^{\text{length}}(s_t) \triangleq \hat{v}_\phi(s_t). \quad (20)$$

This establishes a strict mathematical correspondence between the per-step length reward and the LenVM value baseline.

C.2. Advantage Computation and Combination

With the decoupled rewards and value functions, one can compute the advantages for the task and the length independently using Generalized Advantage Estimation (GAE).

For the task reward, one maintains a standard task value network $V^{\text{task}}(s_t)$ and computes the task temporal difference (TD) errors and advantages:

$$\delta_t^{\text{task}} = r_t^{\text{task}} + \gamma_{\text{task}} V^{\text{task}}(s_{t+1}) - V^{\text{task}}(s_t), \quad (21)$$

$$A_t^{\text{task}} = \sum_{i=0}^{L-1-t} (\gamma_{\text{task}} \lambda_{\text{task}})^i \delta_{t+i}^{\text{task}}. \quad (22)$$

Similarly, for the length penalty, one can use the LenVM as the value baseline to compute the length TD errors and advantages:

$$\delta_t^{\text{length}} = r_t^{\text{length}} + \gamma \hat{v}_\phi(s_{t+1}) - \hat{v}_\phi(s_t), \quad (23)$$

$$A_t^{\text{length}} = \sum_{i=0}^{L-1-t} (\gamma \lambda)^i \delta_{t+i}^{\text{length}}. \quad (24)$$

The final advantage used for the policy gradient update can be written in a general form as a combination of the two components:

$$A_t^{\text{total}} = f\left(A_t^{\text{task}}, A_t^{\text{length}}\right). \quad (25)$$

A simple linear combination can be written as:

$$A_t^{\text{total}} = A_t^{\text{task}} + s \cdot A_t^{\text{length}}, \quad (26)$$

where $s \in \mathbb{R}$ is a scaling coefficient that controls the strength and direction of the length term: $s > 0$ penalizes longer generations, while $s < 0$ encourages longer generations. This combined advantage directly integrates the length term into the policy update.

C.3. Advantages over Sequence-Level Penalty

This formulation offers several distinct benefits over a naive sequence-level length penalty (e.g., $R_{\text{total}} = R_{\text{task}} - \alpha L$):

- **Dense Credit Assignment:** A sequence-level length penalty is only observed at the end of the episode, making it difficult for the policy to attribute the penalty to specific tokens generated early in the sequence. In contrast, our per-step penalty $-(1 - \gamma)$ provides a dense signal at every token, enabling the policy to immediately recognize the incremental cost of continuing generation.
- **Accurate Value Baseline:** The total sequence length L can vary drastically across rollouts, leading to high variance in sequence-level advantage estimation. By using the bounded, accurate per-step baseline provided by the LenVM ($\hat{v}_\phi(s_t) \in (-1, 0)$), we significantly reduce the variance of the length advantage A_t^{length} .

- **Decoupled Value Functions:** Modeling the task return and the length return with a single value network often leads to instability, as the length penalty can dominate or conflict with the sparse task reward. Decoupling them allows the LenVM to focus purely on horizon estimation, while the task value network focuses solely on generation quality.
- **Dynamic Tuning without Retraining:** Because the advantages are computed separately and combined via $A_t^{\text{total}} = A_t^{\text{task}} + s \cdot A_t^{\text{length}}$, the penalty weight s can be dynamically adjusted or scheduled during RL training. This flexibility allows us to tune the length penalty without invalidating the value networks.

D. Precision Analysis for Length Inversion

The estimated length \hat{l} is obtained by inverting $1 + \hat{v} = \gamma^{\hat{l}}$, which depends on the numerical precision of the value prediction. Let z denote the pre-sigmoid logit and $\hat{v} = -\sigma(z)$. For small perturbations δz , a first-order approximation yields a relative length error of the form

$$\left| \frac{\delta l}{l} \right| \approx m(v) |\delta z|, \quad m(v) = \frac{|v|}{|\ln(1+v)|}, \quad (27)$$

where $v \in (-1, 0)$ is the (predicted or true) discounted return.

BF16 rounding model (proxy). As a simple proxy for BF16 rounding in the logit space, we model the perturbation magnitude as

$$|\delta z| \approx \frac{|z|}{256}. \quad (28)$$

Combining Eqs. (27)–(28) provides a lightweight predictor of how numerical errors translate into length errors across the decoding horizon.

D.1. Properties of $m(v)$ on $v \in (-1, 0)$

Let $x \triangleq 1 + v \in (0, 1)$. Then

$$m(v) = \frac{1-x}{-\ln x}. \quad (29)$$

Using the standard inequality $1 - x < -\ln x$ for $x \in (0, 1)$, we have

$$0 < m(v) < 1 \quad \text{for all } v \in (-1, 0), \quad (30)$$

and the limiting behavior

$$\lim_{v \rightarrow 0^-} m(v) = 1, \quad \lim_{v \rightarrow -1^+} m(v) = 0. \quad (31)$$

D.2. Visualization and asymptotics under the BF16 rounding model

Asymptotically,

$$v \rightarrow -1^+ (z \rightarrow +\infty): \quad \ln(1+v) \approx -z, \quad |v| \approx 1, \quad m(v) \approx \frac{1}{|z|}, \quad \left| \frac{\delta l}{l} \right| \approx \frac{1}{256}, \quad (32)$$

$$v \rightarrow 0^- (z \rightarrow -\infty): \quad m(v) \rightarrow 1, \quad \left| \frac{\delta l}{l} \right| \approx \frac{|z|}{256} \approx \frac{|\ln(-v)|}{256}. \quad (33)$$

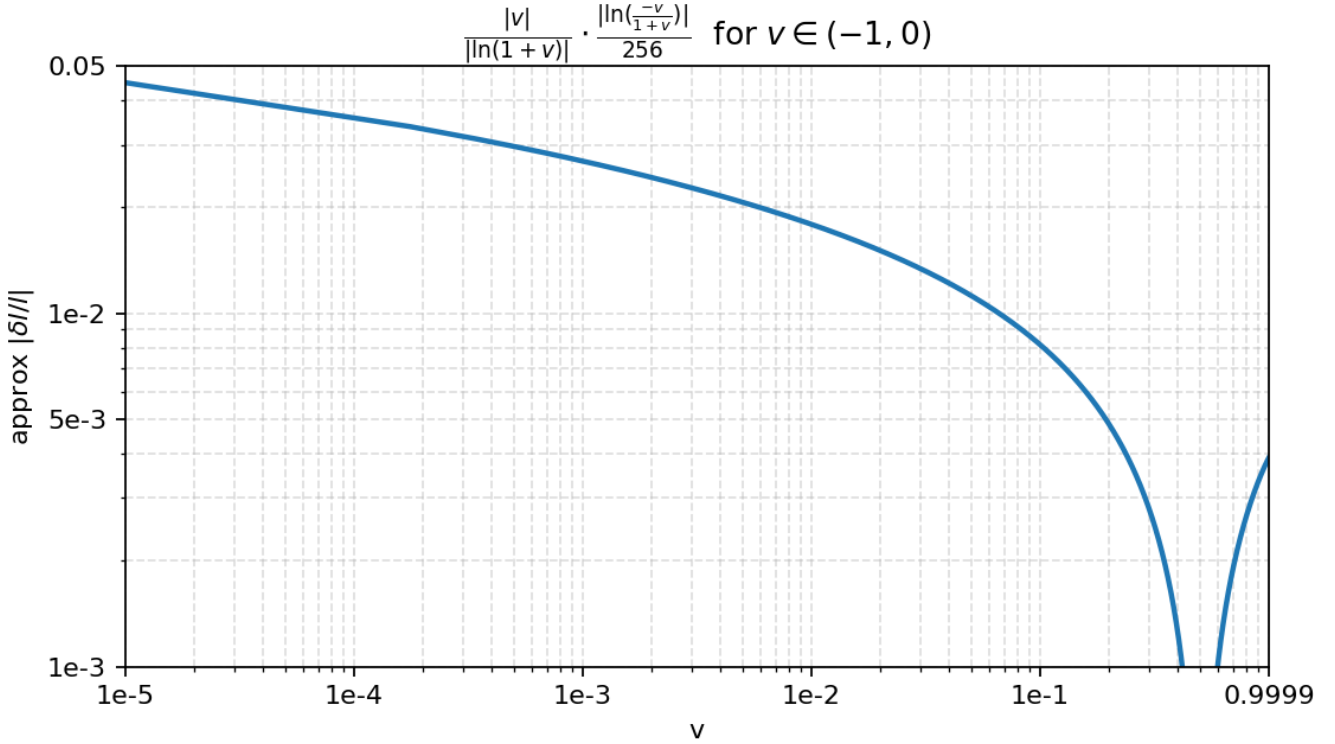


Figure 7. Illustration of the error proxy in Eq. (27) under the BF16 rounding model in Eq. (28).

E. Future-Dependent Weighting Changes the Regression Target

We clarify the population objective underlying LenVM training and show that weights depending on future rollout outcomes shift the regression target away from the state-conditional mean of the return proxy.

Setup. Let τ denote a sampled prompt-completion trajectory with generated length $L(\tau)$. For each non-terminal step $t \in \{0, \dots, L(\tau) - 1\}$, define the per-token squared error

$$\ell_{\theta}(\tau, t) \triangleq (V_{\theta}(s_t^{\tau}) - G_t^{\tau})^2, \quad G_t^{\tau} = -(1 - \gamma^{L(\tau)-t}). \quad (34)$$

E.1. General Weighted Objective

Consider the general weighted population objective

$$\mathcal{J}_w(\theta) = \frac{\mathbb{E}_{\tau, t} \left[\sum_{t=0}^{L(\tau)-1} w_t(\tau) \ell_{\theta}(\tau, t) \right]}{\mathbb{E}_{\tau, t} \left[\sum_{t=0}^{L(\tau)-1} w_t(\tau) \right]}, \quad w_t(\tau) \geq 0, \quad (35)$$

with empirical estimator

$$\mathcal{L}_w = \frac{\sum_{n=1}^N \sum_{t=0}^{L^{(n)}-1} w_t(\tau^{(n)}) \ell_\theta(\tau^{(n)}, t)}{\sum_{n=1}^N \sum_{t=0}^{L^{(n)}-1} w_t(\tau^{(n)})}. \quad (36)$$

Optimal predictor. Introducing the indicator $\mathbb{I}(s_t^\tau = s)$ and optimizing $V(s)$ pointwise, the first-order condition gives

$$\mathbb{E}_{\tau,t} \left[\sum_{t=0}^{L(\tau)-1} w_t(\tau) \mathbb{I}(s_t^\tau = s) \cdot (V(s) - G_t^\tau) \right] = 0. \quad (37)$$

Solving for $V(s)$ yields the minimizer of \mathcal{J}_w :

$$V_w^*(s) = \frac{\mathbb{E}_{\tau,t}[w_t(\tau) G_t^\tau \mid s_t^\tau = s]}{\mathbb{E}_{\tau,t}[w_t(\tau) \mid s_t^\tau = s]}, \quad (38)$$

i.e., a w -weighted conditional mean of G_t^τ at state s . Whether this coincides with the unweighted conditional mean $\mathbb{E}_{\tau,t}[G_t^\tau \mid s_t^\tau = s]$ depends on the structure of $w_t(\tau)$.

E.2. Instantiations

Case 1: Token-uniform weighting ($w_t(\tau) = 1$). Setting $w_t(\tau) = 1$ recovers the token-averaged objective

$$\mathcal{J}_{\text{tok}}(\theta) = \frac{\mathbb{E}_\tau \left[\sum_{t=0}^{L(\tau)-1} \ell_\theta(\tau, t) \right]}{\mathbb{E}_\tau[L(\tau)]}, \quad (39)$$

where every non-terminal decoding step receives equal weight. Since $w_t(\tau) = 1$ is constant, it cancels in Eq. (38), and the optimal predictor is

$$V_{\text{tok}}^*(s) = \mathbb{E}_{\tau,t}[G_t^\tau \mid s_t^\tau = s]. \quad (40)$$

This is the state-conditional mean of the return proxy, which is exactly the quantity LenVM aims to estimate.

Case 2: State-dependent weighting ($w_t(\tau) = c(s_t^\tau)$, $c(s) > 0$). If $w_t(\tau)$ depends only on the current state s_t^τ , then conditioning on $s_t^\tau = s$ makes $w_t(\tau)$ a deterministic constant $c(s)$. By the linearity of conditional expectation, $c(s)$ factors out of both the numerator and denominator in Eq. (38) and cancels:

$$V_w^*(s) = \frac{c(s) \mathbb{E}_{\tau,t}[G_t^\tau \mid s_t^\tau = s]}{c(s)} = \mathbb{E}_{\tau,t}[G_t^\tau \mid s_t^\tau = s]. \quad (41)$$

Such weights only rebalance the relative importance of different states in the global objective; they do not alter the per-state regression target.

Case 3: Future-dependent weighting. If $w_t(\tau)$ depends on outcomes beyond s_t^τ , then conditioned on $s_t^\tau = s$, the weight $w_t(\tau)$ remains a non-trivial random variable that may be correlated with G_t^τ . In general,

$$V_w^*(s) \neq \mathbb{E}_{\tau,t}[G_t^\tau \mid s_t^\tau = s], \quad (42)$$

meaning the regression target itself is shifted. A concrete example is trajectory-level averaging, which sets $w_t(\tau) = 1/L(\tau)$:

$$\mathcal{J}_{\text{traj}}(\theta) = \mathbb{E}_\tau \left[\frac{1}{L(\tau)} \sum_{t=0}^{L(\tau)-1} \ell_\theta(\tau, t) \right]. \quad (43)$$

Because $L(\tau)$ is determined only after the full rollout, $1/L(\tau)$ is correlated with G_t^τ conditioned on $s_t^\tau = s$: trajectories with larger $L(\tau)$ receive smaller weight yet produce more negative G_t^τ . Consequently, $\mathcal{J}_{\text{traj}}$ does not optimize the token-uniform objective in Eq. (39); it optimizes a future-length-reweighted variant with a shifted per-state target.

For LenVM, whose goal is dense token-level Monte Carlo regression over observed decoding states, we therefore adopt the token-averaged objective in Eq. (39).

F. Why Inverting the Transformed Horizon Underestimates Expected Remaining Length

In Section 3.4, we evaluate length prediction in the transformed space rather than by directly inverting the predicted horizon into a raw length estimate. This appendix explains why such inversion systematically underestimates the true expected remaining length.

Let L denote the random remaining generation length from a given decoding state, with $L \geq 0$. Define the transformed horizon

$$u(L) = 1 - \gamma^L, \quad (44)$$

where $\gamma \in (0, 1)$.

Suppose the model predicts the conditional expectation in this transformed space exactly:

$$\hat{u} = \mathbb{E}[u(L)]. \quad (45)$$

A natural raw length estimate is then obtained by inversion:

$$\hat{L} = u^{-1}(\hat{u}) = \frac{\ln(1 - \hat{u})}{\ln \gamma}. \quad (46)$$

We now show that this estimate is always less than or equal to the true expected remaining length $\mathbb{E}[L]$.

The first and second derivatives of $u(L)$ are

$$u'(L) = -\gamma^L \ln \gamma, \quad (47)$$

and

$$u''(L) = -\gamma^L (\ln \gamma)^2. \quad (48)$$

Because $\gamma \in (0, 1)$, we have $\ln \gamma < 0$, so $u'(L) > 0$. Thus $u(L)$ is strictly increasing. Also, since $\gamma^L > 0$ and $(\ln \gamma)^2 > 0$, we have $u''(L) < 0$ for all L . Therefore, $u(L)$ is strictly concave.

By Jensen’s inequality for concave functions,

$$\mathbb{E}[u(L)] \leq u(\mathbb{E}[L]). \quad (49)$$

Since $u(\cdot)$ is strictly increasing, its inverse $u^{-1}(\cdot)$ is also strictly increasing. Applying u^{-1} to both sides of Equation 49 preserves the inequality:

$$u^{-1}(\mathbb{E}[u(L)]) \leq u^{-1}(u(\mathbb{E}[L])) = \mathbb{E}[L]. \quad (50)$$

Therefore,

$$\hat{L} \leq \mathbb{E}[L]. \quad (51)$$

As a result, the strict convexity of the discounted return mapping, combined with Jensen’s inequality, guarantees that the inverted length \hat{L} is bounded above by the true expected length $\mathbb{E}[L]$. The equality $\hat{L} = \mathbb{E}[L]$ holds if and only if the variance of L is zero (i.e., the remaining length is completely deterministic). In all stochastic sequence generation scenarios where the remaining length has variance, the inversion will systematically underestimate the true expected length. In other words, although the expectation is unbiased and accurate within the value space, directly mapping it back to the length space via u^{-1} results in an inaccurate underestimation of the true expected length. The underestimation issue is strictly a consequence of converting to the length space; there is no inherent problem when operating directly within the value space.

G. Derivation of the Exponential Tilting Solution in the Performance–Efficiency Trade-off Experiment

We derive the closed-form solution to the KL-regularised length-steering objective in Eq. (11). At each decoding step t , let $p(x)$ be the base model’s distribution over a finite candidate set \mathcal{V}_t and let $\hat{v}(x)$ be the LenVM’s value prediction for token x . The objective has two goals: (1) minimise the expected LenVM value to steer generation toward shorter completions, and (2) penalise large deviations from the base model via KL divergence to preserve generation quality. We seek the distribution p' that solves, for $\beta < 0$:

$$\min_{p'} \mathbb{E}_{p'}[\hat{v}(x)] - \frac{1}{\beta} D_{\text{KL}}(p' \| p) \quad \text{subject to} \quad \sum_x p'(x) = 1, \quad p'(x) \geq 0. \quad (52)$$

Since $\beta < 0$, the term $-\frac{1}{\beta} > 0$, so the KL divergence enters with a positive coefficient, making the objective strictly convex in p' and guaranteeing a unique global minimiser. (If $\beta > 0$ were used instead, the KL term would be subtracted, rendering the objective unbounded below and the problem ill-posed.)

Expanding the KL term and introducing a Lagrange multiplier λ for the normalisation constraint, the Lagrangian is

$$\mathcal{L}(p', \lambda) = \sum_x p'(x) \hat{v}(x) - \frac{1}{\beta} \sum_x p'(x) \ln \frac{p'(x)}{p(x)} - \lambda \left(\sum_x p'(x) - 1 \right). \quad (53)$$

Taking the derivative with respect to $p'(x)$ and setting it to zero:

$$\hat{v}(x) - \frac{1}{\beta} \left(\ln \frac{p'(x)}{p(x)} + 1 \right) - \lambda = 0. \quad (54)$$

Solving for $p'(x)$:

$$p'(x) = p(x) \exp(\beta \hat{v}(x) - 1 - \beta\lambda). \quad (55)$$

The normalisation condition $\sum_x p'(x) = 1$ fixes the constant, giving the final result:

$$p'(x) = \frac{p(x) \exp(\beta \cdot \hat{v}(x))}{\sum_{x' \in \mathcal{V}_i} p(x') \exp(\beta \cdot \hat{v}(x'))}, \quad \beta < 0. \quad (56)$$

This is the Gibbs / softmax form, and it is the unique global minimiser. With $\beta < 0$, tokens with lower predicted values (shorter expected horizons) receive higher probability mass, steering generation toward shorter completions while $|\beta|$ controls how far the distribution drifts from the base model.