Efficiently Aligning Draft Models via Parameter- and Data-Efficient Adaptation

Luxi Lin[*1], **Zhihang Lin**[*1,2], **Zhanpeng Zeng**[†1], **Yuhao Chen** [3],
Qingyu Zhang [4], **Jixiang Luo** [5], **Rongrong Ji** [1]
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing,
Ministry of Education of China, Xiamen University, 361005, P.R. China
[2]Shanghai Innovation Institute
[3]University of Science and Technology of China (USTC)
[4]University of Chinese Academy of Sciences (UCAS)
[5]Institute of Artificial Intelligence (TeleAI), China Telecom
lewuluu@gmail.com, zzeng@xmu.edu.cn

Abstract

Speculative decoding accelerates LLM inference but suffers from performance degradation when target models are fine-tuned for specific domains. A naive solution is to retrain draft models for every target model, which is costly and inefficient. To address this, we introduce a practical parameter- and data-efficient framework named Efficient Draft Adaptation, abbreviated as EDA, for efficiently adapting draft models. EDA introduces three innovations: (1) a decoupled architecture that utilizes shared and private components to model the shared and target-specific output distributions separately, enabling parameter-efficient adaptation by updating only the lightweight private component; (2) a data regeneration strategy that utilizes the fine-tuned target model to regenerate training data, thereby improving the alignment between training and speculative decoding, leading to higher average acceptance length; (3) a sample selection mechanism that prioritizes high-value data for efficient adaptation. Our experiments show that EDA effectively restores speculative performance on fine-tuned models, achieving superior average acceptance lengths with significantly reduced training costs compared to full retraining. Code is available at `https://github.com/Lyn-Lucy/Efficient-Draft-Adaptation`.

1 Introduction

Large language models (LLMs) have achieved remarkable progress in natural language understanding and generation Radford et al. [2018], Touvron et al. [2023], Bai et al. [2023]. However, their inference efficiency remains bottlenecked by the token-by-token nature of autoregressive decoding, limiting deployment in latency- and throughput-sensitive applications. Speculative decoding Leviathan et al. [2023], Chen et al. [2023] offers a lossless acceleration paradigm: a lightweight draft model proposes multiple candidate tokens ahead, and the target model verifies these candidates in parallel. By adopting an accept–reject sampling procedure that guarantees the final output distribution is exactly preserved, speculative decoding reduces target-model forward passes without sacrificing generation quality, yielding substantial speedup in practice Kwon et al. [2023], Zheng et al. [2024].

In practice, generic pre-trained LLMs are often further adapted to specific tasks or domains (e.g., math Yu et al. [2024b], Yue et al. [2024], code Rozière et al. [2023], Luo et al. [2024], and medicine Wu et al. [2023], Zhang et al. [2023]) via supervised fine-tuning (SFT), which substantially

[*]Equal contribution
[†]Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

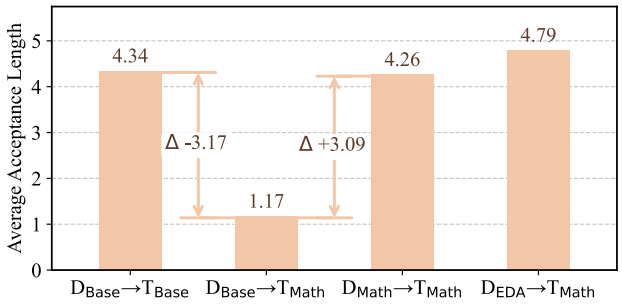

Figure 1: The comparison of average acceptance length on GSM8K under different draft-target pairings. We refer to the target models as T_{Base} (Qwen2.5-7B) and T_{Math} (Qwen2.5-Math-7B). The target model T_{Base}'s draft model D_{Base} achieves high average acceptance length when paired ($D_{Base} \rightarrow T_{Base}$), while it suffers a substantial acceptance drop when paired with T_{Math} ($D_{Base} \rightarrow T_{Math}$). Our EDA framework can restore performance through efficient, lightweight adaptation.

shifts their output distributions. However, speculative decoding is effective only when the draft model's output distribution closely aligns with that of the target model. As a result, the draft model trained before SFT no longer aligns well with the fine-tuned target model, leading to degraded average acceptance length and speedup. We illustrate this phenomenon in Figure 1. A draft model (D_{Base}) trained to match the pre-fine-tuning target model, denoted as T_{Base}, exhibits a substantial drop in average acceptance length when it is directly used as the draft model for the fine-tuned target model, denoted as T_{Math}. Thus, we must retrain a dedicated draft model for each fine-tuned target model to realign their output distributions, incurring significant training costs. There is thus a pressing need for efficient draft adaptation techniques that can quickly track target-model changes.

In this paper, we propose a novel approach to **E**fficiently align **D**raft models via parameter- and data-efficient **A**daptation, dubbed EDA. Specifically, we observe that the output distributions of target models before and after fine-tuning share a substantial overlap, as evidenced by the non-zero average acceptance length when directly applying the draft model trained for the base target model to the fine-tuned target model, as shown in Figure 1. Thus, we decouple the draft model into shared and private components to separately model the shared and task-specific output distributions. Compared to traditional methods that train a monolithic draft model for each target model, our design enables efficient transfer by reusing the shared component and updating only the lightweight private component when adapting to a new target model, showing the parameter efficiency of our approach.

Next, we improve the average acceptance length of the draft model by reducing the mismatch between the output distribution of training and speculative decoding stages. Specifically, during the draft model training stage, the draft model is optimized to predict the next token in the provided dataset. However, during speculative decoding, the draft model needs to predict the target model's next token, which may differ from the ground-truth token in the provided dataset, leading to a mismatch between training and speculative decoding. To address this issue, we regenerate the training set using the fine-tuned target model itself, thereby improving the alignment between training and drafting, leading to a higher average acceptance length.

Finally, we further improve the adaptation efficiency from the data perspective, as adapting the draft model with a large training set can still incur unnecessary cost. We found that not all training samples contribute equally to improving the average acceptance length during adaptation. Thus, we introduce a sample value selection mechanism that scores and ranks candidate training samples to construct a more compact adaptation subset. Under a fixed training budget, this enables the draft model to focus on generation segments that are most influential for improving average acceptance length during adaptation, thereby achieving higher adaptation efficiency with fewer samples.

We conduct extensive experiments on various downstream tasks to evaluate the effectiveness of EDA. For example, when adapting a draft model trained for Qwen2.5-7B to the fine-tuned target model Qwen2.5-Math-7B, EDA achieves an average acceptance length of 4.79, significantly outperforming the baseline adaptation method that achieves only 4.37, while incurring only 39.2% of the training cost of full draft model retraining.

2 Related Work

2.1 Speculative Decoding

Speculative decoding Leviathan et al. [2023], Chen et al. [2023], Lin et al. [2025] utilizes a draft model to generate multiple candidate tokens, which are then validated by a target model, significantly speeding up LLM decoding. Its realized acceleration is largely governed by how well the draft model matches the target model's conditional distribution, commonly reflected by the average acceptance length. Prior work commonly improves speculative decoding by training or aligning a dedicated draft model for a particular target model, e.g., via distillation-style distribution matching in token space Cai et al. [2024], Zhou et al. [2024], Yan et al. [2025] or by drafting in an intermediate representation space as in EAGLE Li et al. [2024a,b]. These methods typically optimize the draft model to better follow the target model's next-token preferences, thereby increasing consecutive acceptances and translating into higher end-to-end speedup.

However, they generally assume a fixed target model and require re-training or re-adaptation when the target changes to preserve average acceptance length and speedup. In contrast, we treat cross-target transfer of the draft model as a key problem, enabling reuse and efficient adaptation via a shared private decomposition, rather than training a separate draft model for each target model.

2.2 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) adapts large pre-trained models by updating only a small set of additional parameters while keeping the backbone frozen. Representative approaches include inserting lightweight adapters into Transformer blocks Houlsby et al. [2019], learning low-rank updates as in LoRA Hu et al. [2022], and optimizing prompt-like parameters such as prefix/prompt tuning Li and Liang [2021]. More recently, PEFT has also been extended toward more modular designs that introduce multiple lightweight modules and allocate capacity more flexibly. For instance, Yu et al. [2024a] proposes mixture-of-experts adapters with lightweight routing to support specialization and reduce interference in multi-task and continual learning.

Our work is related to parameter-efficient adaptation, which updates only a small set of parameters while keeping most of the model fixed Houlsby et al. [2019], Hu et al. [2022], Li and Liang [2021]. Unlike traditional PEFT methods that adapt the target model, we apply this approach to the speculative draft model, enabling the reuse of shared structures while updating only a small, target-specific component. This ensures the draft model efficiently aligns with changes in the target model's generation patterns, preserving acceptance rates and decoding speedup.

3 Speculative Decoding

Draft Model Training Stage. Starting with a prefix $x_{<t}$ sampled from the data distribution $\mathcal{D}$, the draft model θ_d is trained to closely match the target distribution of the larger model θ_t as:

$$\theta_d = \arg\min_{\theta_d} \mathbb{E}_{x\sim\mathcal{D}} \sum_t \Big[L_{reg} + \mathrm{CE}\big(P_{\theta_t}(x_t \mid x_{<t}), P_{\theta_d}(x_t \mid x_{<t})\big)\Big], \tag{1}$$

where CE denotes the cross-entropy loss, and L_{reg} is a regularization term depending on specific implementations, such as intermediate representation matching Li et al. [2024a]. $P_{\theta_t}(x_t \mid x_{<t})$ and $P_{\theta_d}(x_t \mid x_{<t})$ denote the target and draft model's prediction probability distributions, respectively. When $P_{\theta_t}(x_t \mid x_{<t})$ shifts, as often happens in practice (e.g., after fine-tuning), the draft model $P_{\theta_d}(x_t \mid x_{<t})$ no longer aligns well with it. This misalignment necessitates retraining the entire draft model on updated datasets for each fine-tuned target model, resulting in significant training overhead.

Speculation Stage. Based on the prefix $x_{<t}$, the draft model proposes K speculative candidate tokens $\hat{x}_{t:t+K-1} = \{x_t, \cdots, x_{t+K-1}\}$ from draft distributions.

Verification Stage. The target model verifies the conditional distributions in parallel, $P_{\theta_t}(x_{t+i} \mid x_{<t}, \hat{x}_{t:t+i-1})$ for $i = 0, \ldots, K-1$, and accepts candidate tokens based on the acceptance rule Leviathan et al. [2023] until encountering the first rejection. At the rejection point, a replacement token is sampled directly from the target model's distribution. This mechanism ensures the final output distribution remains identical to standard autoregressive sampling from the target model.

4 Method

This section introduces EDA, a framework designed for efficient adaptation of draft models to align with fine-tuned target models. EDA integrates a shared–private draft architecture, domain-specific self-generation, and representation-shift-based data selection strategy to efficiently restore speculative decoding performance. Figure 2 provides an overview of our EDA framework.

4.1 Modeling Shared and Private Output Distributions Separately

As shown in Figure 1, directly using the draft model D_{Base} trained for the base target model T_{Base} with the fine-tuned target model T_{Math} leads to a significant decrease in average acceptance length. However, the non-zero acceptance length suggests that their output distributions partially overlap.

We define the overlapping portion of the output distribution shared across different target models as the shared output distribution, while the remaining part is referred to as the target-specific output distribution. Conventional speculative decoding methods Li et al. [2024a], Zhou et al. [2024], Yan et al. [2025] typically train a monolithic draft model to capture the entire target distribution, without differentiating between the shared and target-specific components. As a result, these methods often require retraining or fine-tuning the entire draft model for each fine-tuned target model, leading to substantial training overhead. This raises a key question: *Can we efficiently adapt the draft model to align with fine-tuned target models by reusing the shared distribution and focusing only on the target-specific differences?*

To address this question, we decouple the draft model into a shared expert and a private expert, enabling separate modeling of the shared and target-specific output distributions. When the target model undergoes fine-tuning, the shared expert remains fixed, while only the lightweight private expert is updated. This approach allows the draft model to efficiently adapt to the target-specific output distribution in a parameter-efficient manner.

Shared–Private Gated Draft Architecture. Based on the above decomposition, we implement the draft model as a two-expert gated architecture. Specifically, we replace the feed-forward network (FFN) in the Transformer block with a shared–private gated module, while keeping the attention layers unchanged. The module comprises a shared expert E^{s} and a private expert E^{p}, combined via a learnable gating mechanism. Given the hidden representation $h_t \in \mathbb{R}^d$ of the t-th token, the outputs of two experts are computed as:

$$\big[u_t^{\mathrm{s}},\, u_t^{\mathrm{p}}\big] = \big[E^{\mathrm{s}}(h_t),\, E^{\mathrm{p}}(h_t)\big]. \tag{2}$$

The gate values g_t^s and g_t^p are computed as:

$$\big[g_t^{\mathrm{s}},\, g_t^{\mathrm{p}}\big] = \mathrm{softmax}\big([(w^{\mathrm{s}})^\top h_t,\ (w^{\mathrm{p}})^\top h_t]\big), \tag{3}$$

where $w^{\mathrm{s}}, w^{\mathrm{p}} \in \mathbb{R}^d$ are learnable routing weights. The softmax function ensures the gate values sum to 1. The output of the shared–private experts gated module is then computed as:

$$\tilde{h}_t = g_t^{\mathrm{s}}\, u_t^{\mathrm{s}} + g_t^{\mathrm{p}}\, u_t^{\mathrm{p}}. \tag{4}$$

This design creates a lightweight gated draft model with two experts, where routing dynamically adjusts the contributions of the shared and private components.

Expert Parameterization and Draft Distribution. Having defined the two-expert gated architecture and its routing mechanism, we now specify the parameterization of each expert and describe how the combined hidden representation is mapped to the draft token distribution. Each expert is parameterized as a standard bottleneck MLP:

$$E(h) = U\,\phi(Vh), \tag{5}$$

where $\phi(\cdot)$ is a nonlinear activation function, $V \in \mathbb{R}^{m\times d}$ and $U \in \mathbb{R}^{d\times m}$ are the down- and up-projection matrices with m being the intermediate dimension. The draft model produces token probabilities through:

$$P_{\theta_d}(x_t \mid x_{<t}) = \mathrm{softmax}(W_o \tilde{h}_t), \tag{6}$$

where $W_o \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the output projection head and $|\mathcal{V}|$ denotes the vocabulary size.

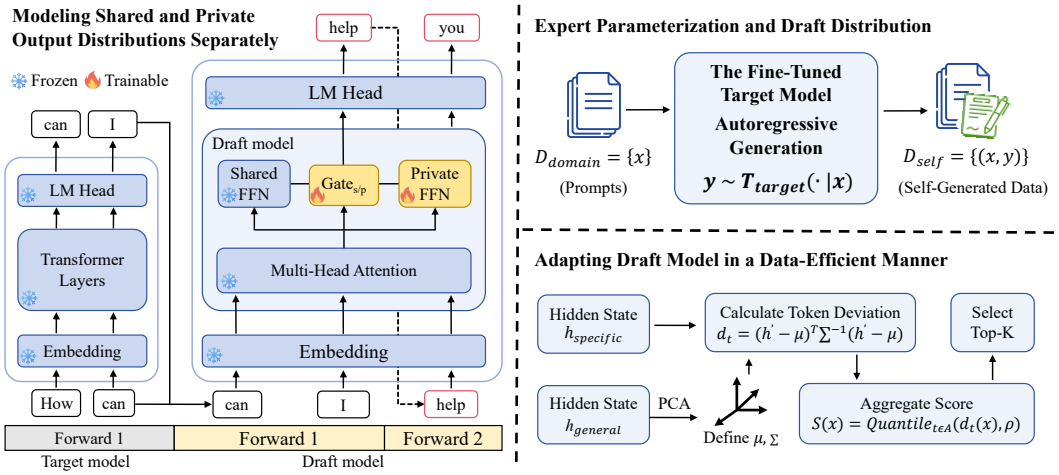

Figure 2: The EDA framework for efficient draft model adaptation, combining shared–private draft decomposition, domain-specific self-generation, and representation-shift–based data selection to restore speculative decoding performance with minimal training cost.

Draft Model Initialization and Adaptation. The shared expert E^{s} is initialized using the parameters of a draft model pre-trained on general datasets, such as D_{Base} trained on ShareGPT ShareGPT [2023], to model the shared output distribution. For adaptation to a fine-tuned target model like T_{Math}, the shared expert E^{s} remains frozen, while only the private expert E^{p} and the routing parameters w^{p} and w^{s} are updated. This allows the private expert to model target-specific output distributions, with the gating mechanism dynamically combining the contributions of both experts.

4.2 Matching Training and Drafting Objective

We further improve the average acceptance length of our EDA by addressing the mismatch between the draft model's training objectives and speculative decoding objectives. Specifically, consider two token sequences: $x = \{x_{<t}, \cdots, x_n\}$ sampled from the data distribution $\mathcal{D}$, and $x' = \{x_{<t}, \cdots, x'_n\}$ generated by the target model via autoregressive sampling conditioned on the same prefix $x_{<t}$. Since the target model's prediction distribution may not perfectly align with $\mathcal{D}$, x_n and x'_n can differ.

Traditionally, the draft model is trained on datasets like the first token sequence x using Eq. (1) Li et al. [2024a]. However, during speculative decoding, it predicts the next token in the second token sequence x' generated by the target model. This mismatch between the training and drafting objectives leads to suboptimal average acceptance length.

In domain-specific fine-tuning scenarios, such as medicine, law, or code, proprietary datasets used for fine-tuning the target model are often inaccessible due to privacy or ownership restrictions. As a result, the draft model must be trained on publicly available datasets, which may significantly differ from the proprietary data distribution, further widening the gap between the draft model's training and speculative decoding objectives, ultimately reducing the average acceptance length.

To address this mismatch, we use a domain-specific self-generation strategy, in which the fine-tuned target model provides supervision during the draft model training stage. Given a domain-specific dataset $\mathcal{D}_{\text{domain}} = \{x^i\}$, for each x^i we sample the target model's completions via conditional autoregressive generation:

$$y^i \sim P_{\theta_t}(\cdot \mid x^i). \tag{7}$$

This procedure produces the training set generated by the target model itself:

$$\mathcal{D}_{\text{self}} = \{(x^i, y^i)\}. \tag{8}$$

We then optimize the draft model on $\mathcal{D}_{\text{self}}$ via:

$$\theta_d = \arg\min_{\theta_d} \mathbb{E}_{x \sim \mathcal{D}_{\text{self}}} \sum_t \Big[L_{\text{reg}} + \text{CE}\big(x_t, P_{\theta_d}(\cdot \mid x_{<t})\big) \Big]. \tag{9}$$

This objective performs trajectory-level alignment on target-generated tokens rather than full distribution distillation. By matching training and drafting objectives, our EDA achieves a higher average acceptance length during speculative decoding, as shown in our experiments.

4.3 Adapting Draft Model in a Data-Efficient Manner

Training draft models in the entire domain-specific datasets can still incur significant costs, even when optimization is restricted to the private expert. Thus, we further improve draft model adaptation efficiency from the data perspective.

As discussed in Sec. 4.1, the shared expert captures the shared output distribution, while the private expert focuses on modeling the target-specific output distribution. To achieve efficient draft model adaptation, it is crucial to prioritize samples that exhibit significant deviations from the shared output distribution, as these provide the most valuable information for adapting the private expert, while filtering out samples that are already well-represented. Building on the observation that specialization often manifests as structured shifts in the representation space Lee et al. [2018], we introduce a training-free sample value metric based solely on the hidden states generated by the target model during self-generation. This metric is then used to identify and select a subset of the training data that is most beneficial for refining the private expert.

Given a self-generated sample $x \sim \mathcal{D}_{\text{self}}$ of length L, let $\mathbf{H} = \{h_t\}_{t=1}^{L}$ denote the sequence of hidden representations produced by the target model. We define $\mathcal{A} \subset \{1, \ldots, L\}$ as the set of indices corresponding to the answer tokens.

We first apply Principal Component Analysis (PCA) to reduce the dimensionality of the answer token's hidden representations as:

$$h'_t \leftarrow \mathrm{PCA}(h_t), \quad \forall t \in \mathcal{A}, \tag{10}$$

This reduction ensures that only the most relevant features of the data are retained, improving both computational efficiency and stability for subsequent statistical calculations.

After reducing dimensionality, we calculate the mean μ and covariance matrix Σ of the reduced representations h'_t using a general-generation dataset $\mathcal{D}_{\text{general}}$:

$$\mu = \mathbb{E}_{h' \sim \mathcal{D}_{\text{general}}}[h'], \quad \Sigma = \mathrm{Cov}_{h' \sim \mathcal{D}_{\text{general}}}(h'). \tag{11}$$

Here, $\mathcal{D}_{\text{general}}$ is the 68k ShareGPT samples used for base draft training; PCA, μ, and Σ are fitted once on the final-layer answer-token hidden states of T_{Base}, and each fine-tuned target's representations are projected into this fixed space. We then compute the Mahalanobis score Mahalanobis [1936] $d_t(x)$ to quantify each answer-token representation's deviation from the reference distribution $\mathcal{D}_{\text{general}}$. The score is computed on the PCA-reduced representations $h_t'^{(x)}$. Specifically, for each $t \in \mathcal{A}$, we define

$$d_t(x) = \big(h_t'^{(x)} - \mu\big)^{\top} \Sigma^{-1} \big(h_t'^{(x)} - \mu\big). \tag{12}$$

Since domain- and task-specific behaviors often concentrate in a small number of critical steps or local segments, we aggregate token-level deviations with a tail-sensitive statistic to obtain a sample-level value score $s(x)$. In particular, we use the high-quantile aggregation

$$s(x) = \mathrm{Quantile}_{t \in \mathcal{A}}\big(d_t(x), \rho\big), \tag{13}$$

where ρ controls the quantile level.

Given the whole dataset $\mathcal{D}_{\text{self}}$, we rank samples by $s(x)$ and select the subset as:

$$\mathcal{D}_{\text{compact}} = \big\{x_i \in \mathcal{D}_{\text{self}} \mid s(x_i) \text{ ranks among the top } \lfloor \alpha |\mathcal{D}_{\text{self}}| \rfloor \text{ scores}\big\}, \tag{14}$$

where $\alpha \in (0, 1]$ denotes the selection ratio. The selected subset is used to update the private expert while keeping the shared expert frozen. This strategy shifts training effort from representation regions already covered by the shared expert to answer segments that exhibit larger deviations from the general reference, improving adaptation efficiency under a limited sample budget. Importantly, the selection process incurs no extra forward passes or auxiliary components, as the hidden representations h_t are directly reused from the self-generation stage.

Table 1: Main results on five math tasks (GSM8K, AIME_2024, SVAMP, H-MATH, MathQA) on Qwen2.5-Math-7B target model Yang et al. [2024]. We report average acceptance length τ and decoding speedup, where Speed. denotes decoding speedup.

	GSM8K		**AIME_2024**		**SVAMP**		**H-MATH**		**MathQA**		**Avg**	
Method	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.
Temperature $T = 0$												
Training-Free	1.17	0.85×	1.18	0.76×	1.19	0.85×	1.20	0.82×	1.23	0.91×	1.19	0.84×
Full-FT	4.37	2.88×	5.05	3.14×	4.40	2.91×	4.98	3.23×	4.66	3.19×	4.69	3.07×
LoRA	4.32	2.84×	4.90	2.93×	4.36	2.75×	4.92	3.18×	4.55	3.00×	4.61	2.94×
EDA	**4.79**	**3.06**×	**5.41**	**3.20**×	**4.96**	**3.09**×	**5.60**	**3.59**×	**5.16**	**3.43**×	**5.18**	**3.27**×
Temperature $T = 1$												
Training-Free	1.02	0.65×	1.15	0.72×	1.08	0.68×	1.21	0.75×	1.04	0.66×	1.10	0.69×
Full-FT	2.65	1.72×	2.89	1.84×	2.74	1.76×	3.12	1.95×	3.01	1.90×	2.88	1.83×
LoRA	2.60	1.70×	2.80	1.78×	2.70	1.74×	3.05	1.92×	2.93	1.86×	2.82	1.80×
EDA	**3.15**	**1.92**×	**3.38**	**2.05**×	**3.26**	**1.98**×	**3.75**	**2.24**×	**3.52**	**2.12**×	**3.41**	**2.06**×

Table 2: Main results on five code tasks (HumanEval, APPS, BigCodeBench, HumanEval+, MBPP) on Qwen2.5-Coder-7B target model Hui et al. [2024].

	HumanEval		**APPS**		**BigCodeBench**		**HumanEval+**		**MBPP**		**Avg**	
Method	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.
Temperature $T = 0$												
Training-Free	1.75	1.21×	1.69	1.14×	1.74	1.24×	1.74	1.08×	1.85	1.21×	1.75	1.18×
Full-FT	4.79	3.10×	4.87	2.91×	3.66	2.38×	4.70	2.62×	4.93	2.82×	4.59	2.77×
LoRA	4.70	3.05×	4.80	2.88×	3.60	2.35×	4.62	2.58×	4.78	2.76×	4.50	2.72×
EDA	**5.35**	**3.36**×	**5.65**	**3.34**×	**4.18**	**2.67**×	**5.31**	**2.98**×	**5.43**	**3.18**×	**5.18**	**3.11**×
Temperature $T = 1$												
Training-Free	1.28	0.78×	1.35	0.81×	1.22	0.75×	1.41	0.85×	1.30	0.79×	1.31	0.80×
Full-FT	3.72	2.00×	4.05	2.13×	3.35	1.80×	3.88	2.05×	4.32	2.23×	3.86	2.04×
LoRA	3.60	1.95×	3.95	2.08×	3.30	1.78×	3.80	2.00×	4.25	2.18×	3.78	2.00×
EDA	**4.38**	**2.28**×	**4.62**	**2.38**×	**3.95**	**2.06**×	**4.55**	**2.32**×	**4.92**	**2.48**×	**4.48**	**2.30**×

Table 3: Main results on five medical tasks (MedMCQA, MedQA-USMLE, PubMedQA, MedQA, MMLU) on Meditron3-Qwen2.5-7B target model OpenMeditron [2025].

	MedMCQA		**MedQA-USMLE**		**PubMedQA**		**MedQA**		**MMLU**		**Avg**	
Method	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.
Temperature $T = 0$												
Training-Free	2.20	1.47×	2.17	1.40×	1.92	1.29×	2.14	1.29×	2.33	1.49×	2.15	1.39×
Full-FT	3.97	2.35×	3.97	2.28×	3.86	2.34×	3.97	2.35×	3.67	3.03×	3.89	2.47×
LoRA	3.85	2.30×	3.90	2.25×	3.78	2.18×	3.92	2.28×	3.50	2.80×	3.79	2.36×
EDA	**4.30**	**2.55**×	**4.34**	**2.55**×	**4.03**	**2.40**×	**4.32**	**2.51**×	**4.06**	**3.29**×	**4.21**	**2.66**×
Temperature $T = 1$												
Training-Free	1.42	0.88×	1.55	0.95×	1.38	0.82×	1.62	1.02×	1.48	0.92×	1.49	0.92×
Full-FT	2.42	1.42×	2.65	1.52×	2.28	1.35×	2.72	1.56×	2.55	1.48×	2.52	1.47×
LoRA	2.35	1.38×	2.60	1.50×	2.25	1.33×	2.68	1.52×	2.50	1.46×	2.48	1.44×
EDA	**2.85**	**1.62**×	**3.12**	**1.75**×	**2.72**	**1.56**×	**3.25**	**1.82**×	**3.08**	**1.72**×	**3.00**	**1.69**×

5 Experiments

5.1 Experimental Configurations

Dataset. We evaluate EDA across three downstream domains, including mathematical reasoning, code generation, and medical question answering. For each domain, we use five representative

Table 4: Component-wise ablation on Qwen2.5-Math-7B. We report average acceptance length τ and decoding speedup on representative math benchmarks.

Method	Trainable Params	Samples	GSM8K	AIME	MathQA
Full-FT	462MB	68k	4.37 / 2.88×	5.05 / 3.14×	4.66 / 3.19×
+ Shared/Private FFN	127MB	68k	4.40 / 2.89×	5.02 / 3.12×	4.70 / 3.20×
+ Self-generated Data	127MB	68k	4.83 / 3.08×	5.46 / 3.23×	5.20 / 3.46×
+ Data Selection	127MB	34k	4.79 / 3.06×	5.41 / 3.20×	5.16 / 3.43×

benchmarks and follow the standard evaluation protocols of each benchmark to construct prompts. Detailed descriptions of the evaluation benchmarks and training datasets are provided in Appendix B.

Model Setup. We evaluate EDA in a practical setting where a general-purpose base model is fine-tuned into multiple domain-specific target models. To avoid the prohibitive cost of retraining target models ourselves, we use off-the-shelf models from the Qwen2.5 family Qwen Team [2025]: Qwen2.5-7B as the base target, and Qwen2.5-Math-7B Yang et al. [2024], Qwen2.5-Coder-7B Hui et al. [2024], and Meditron3-Qwen2.5-7B OpenMeditron [2025] as domain-specific fine-tuned targets. This setup preserves architectural consistency while simulating real-world target-model specialization. Additional experiments on another model architecture are provided in Appendix C. The draft model architecture and implementation details are described in Appendix A.

Baseline. We compare EDA with three representative baselines that share the same pre-trained backbone model but differ in their adaptation strategies, training cost, and parameter efficiency. (1) **Training-Free** directly uses a pre-trained model as the draft model without any additional training or adaptation to the fine-tuned target model. (2) **Full Fine-Tuning (Full-FT)** adapts the same backbone model by updating all parameters on the available adaptation set, representing the strongest but most expensive transfer baseline. (3) **Low-Rank Adaptation (LoRA)** Hu et al. [2022] performs parameter-efficient transfer by training only low-rank update matrices while freezing the backbone. We match the number of trainable parameters to EDA to ensure a fair comparison in terms of parameter efficiency. For Full-FT and LoRA, we use the same publicly available domain adaptation datasets under an identical data budget. For EDA, we use the same domain data as prompts to construct the target-model self-generated adaptation set described in Section 4.2, and apply the proposed representation-shift-based data selection strategy to retain 50% of the adaptation samples.

Evaluation Metrics. We evaluate speculative decoding using average acceptance length τ and decoding speedup. Average acceptance length, formally defined in Appendix A, measures the expected number of accepted draft tokens before rejection, and speedup is the latency ratio between standard autoregressive decoding and speculative decoding. Since speculative decoding is lossless, we focus on efficiency metrics and do not separately report task accuracy.

5.2 Main Results

Tables 1–3 summarize the main results across math, code, and medical benchmarks under both greedy ($T = 0$) and stochastic ($T = 1$) decoding. Across all three domains, EDA consistently improves both average acceptance length τ and decoding speedup over Training-Free, Full-FT, and LoRA baselines.

The Training-Free baseline performs poorly after target-model fine-tuning, confirming that directly reusing the original draft model cannot handle the distribution shifts introduced by domain specialization. Compared with Full-FT and parameter-matched LoRA, EDA achieves stronger speculative decoding performance in most settings while updating only a lightweight target-specific component and using a reduced adaptation set. These results show that combining shared–private draft adaptation, target-model self-generation, and representation-shift-based data selection effectively restores draft–target alignment under limited adaptation cost. Unless otherwise specified, subsequent evaluations follow the math transfer setting with $T = 0$ on a single H200 GPU.

5.3 Component-wise Ablation Study

To understand how each component contributes to EDA, we conduct a cumulative ablation on the Qwen2.5-Math-7B target model. As shown in Table 4, the shared/private FFN architecture

Table 5: Comparison of adaptation structures and training overhead. The left part reports performance on math and general benchmarks, where each cell denotes τ / speedup. The right part reports the adaptation overhead on math transfer, following the same experimental setup as in Sec. 5.5.

		Math Benchmarks		General Benchmarks	
Method	Params	GSM8K	MathQA	Alpaca	MT-Bench
Full-FT	462MB	**4.82 / 3.08**×	**5.20 / 3.46**×	5.17 / 3.02×	4.19 / 2.67×
LoRA	**127MB**	4.66 / 2.99×	5.07 / 3.38×	4.89 / 2.76×	4.04 / 2.40×
EDA	**127MB**	4.79 / 3.06×	5.16 / 3.43×	**5.21 / 3.04**×	**4.22 / 2.68**×

Metric	Re-Train	EDA
Trainable Size	462 MB	**127 MB** (27.5%)
Training Time	5.1 h	**2.0 h** (39.2%)
τ	4.22	**4.79** (+13.5%)

Table 6: Routing analysis and branch ablation of the Math-adapted EDA draft model on answer tokens. Entropy is the base-2 routing entropy (1 for uniform routing). Shared-/Private-only zero out the other expert; entries report CE to the target distribution (↓) / top-1 agreement with the target (↑).

	Routing Analysis			Branch Ablation (CE / Top-1)	
Domain	Shared Gate	Private Gate	Entropy	Shared-only	Private-only
General / Alpaca	**58.64%**	41.36%	0.98	**5.63 / 25.54%**	8.71 / 7.11%
Math / GSM8K	29.57%	**70.43%**	0.88	10.18 / 11.76%	**4.60 / 39.96%**

matches the performance of Full-FT while reducing the trainable parameters from 462MB to 127MB. Replacing the original adaptation data with target-model self-generated data provides the largest improvement, confirming that aligning the training objective with the speculative decoding trajectory is crucial for draft adaptation. Finally, the representation-shift-based data selection reduces the adaptation samples by 50% with only negligible performance loss. These results show that the three components contribute in complementary ways: the architecture improves parameter efficiency, self-generation improves target alignment, and data selection improves data efficiency.

5.4 Effectiveness of Separately Modeling Shared and Private Distributions

We compare EDA with Full-FT and parameter-matched LoRA, all trained on the same self-generated and selected data. As shown in the left of Table 5, EDA matches Full-FT with only 27.5% of its trainable parameters and outperforms LoRA, with a larger margin on the general-domain benchmarks (Alpaca and MT-Bench), suggesting that LoRA interferes with reusable drafting capability while the frozen shared expert preserves it.

To verify that the experts learn their intended roles, we analyze the Math-adapted EDA checkpoint on Alpaca and GSM8K samples (Table 6). The router favors the shared expert on Alpaca and the private expert on GSM8K, and branch ablation shows the same reversal: shared-only aligns better with the target model on Alpaca, whereas private-only aligns better on GSM8K. This confirms that the shared expert retains general drafting behavior, while the private expert captures the domain fine-tuning shift.

5.5 Effectiveness of Adapting Draft Model in a Data-Efficient Manner

We evaluate the effect of our representation-shift-based data selection under different data budgets. As shown in Figure 3, the proposed selection strategy achieves near full-data performance with only 50% of the adaptation data. This indicates that representation shift is an effective proxy for identifying samples that are more useful for adapting the private expert. Detailed comparisons with alternative selection metrics and additional analysis are provided in Appendix D.

5.6 Adaptation Overhead Analysis

As shown in the right part of Table 5, on the math transfer task from Qwen2.5-7B to Qwen2.5-Math-7B, EDA substantially outperforms training a draft model from scratch in both efficiency and performance. Re-Train updates 462 MB of parameters and takes 5.1 hours, whereas EDA adapts only 127 MB of parameters (27.5%) and finishes in 2.0 hours (39.2%). Despite the reduced training cost, EDA achieves a higher average acceptance length, improving τ from 4.22 to 4.79. These results demonstrate that EDA delivers better alignment while markedly reducing adaptation time.

Table 7: Results on five math tasks with a larger target model, OpenMath-Nemotron-14B, which is fine-tuned from Qwen2.5-14B. We report average acceptance length τ and decoding speedup.

	GSM8K		**AIME_2024**		**SVAMP**		**H-MATH**		**MathQA**		**Avg**	
Method	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.	τ	Speed.
Temperature $T = 0$												
Training-Free	1.72	1.25×	1.80	1.31×	1.69	1.22×	1.91	1.34×	1.84	1.28×	1.79	1.28×
Full-FT	4.31	2.95×	4.52	3.08×	4.35	3.04×	4.72	3.13×	4.73	3.20×	4.53	3.08×
LoRA	4.24	2.88×	4.43	3.01×	4.26	2.96×	4.66	3.07×	4.61	3.14×	4.44	3.01×
EDA	**4.69**	**3.14**×	**5.02**	**3.39**×	**4.68**	**3.17**×	**5.30**	**3.48**×	**5.03**	**3.44**×	**4.94**	**3.32**×
Temperature $T = 1$												
Training-Free	1.58	1.14×	1.66	1.20×	1.61	1.16×	1.74	1.25×	1.70	1.19×	1.66	1.19×
Full-FT	4.10	2.74×	4.37	2.80×	4.01	2.65×	4.59	2.96×	4.44	2.89×	4.30	2.81×
LoRA	4.02	2.68×	4.28	2.73×	3.95	2.58×	4.52	2.89×	4.35	2.82×	4.22	2.74×
EDA	**4.51**	**2.95**×	**4.85**	**3.15**×	**4.56**	**2.88**×	**5.16**	**3.36**×	**4.83**	**3.18**×	**4.78**	**3.10**×

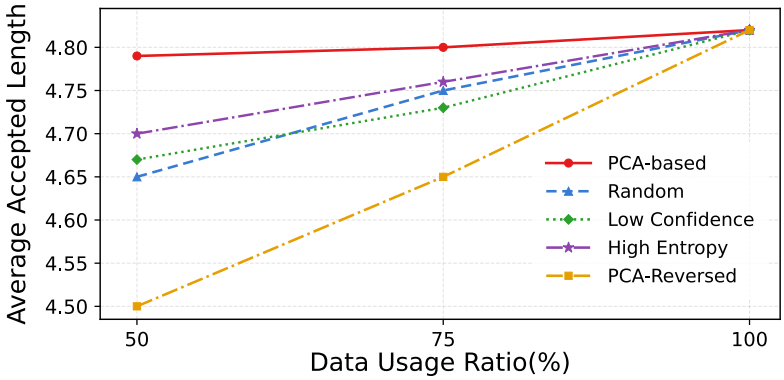

Figure 3: Comparison of data selection strategies under varying data budgets. Draft adaptation is evaluated from Qwen2.5-7B to Qwen2.5-Math-7B on GSM8K using average acceptance length τ.

5.7 Scaling to Larger Target Models

To examine whether EDA scales to larger target models, we adapt the draft model of Qwen2.5-14B to OpenMath-Nemotron-14B, a math-specialized model fine-tuned from Qwen2.5-14B. As shown in Table 7, EDA achieves the highest τ and speedup on all five benchmarks under both temperatures, improving the average τ over Full-FT from 4.53 to 4.94 at $T = 0$ and from 4.30 to 4.78 at $T = 1$.

6 Conclusion

In this paper, we propose EDA, a parameter- and data-efficient framework for adapting draft models to fine-tuned target models in speculative decoding. By decoupling the draft model into shared and private components, EDA reuses shared knowledge while updating only lightweight target-specific parameters. Combined with domain-specific self-generation and representation-shift–based data selection, EDA effectively aligns draft models with evolving target distributions under limited training budgets. Extensive experiments across math, code, and medical domains show that EDA restores average acceptance length and decoding speedup while reducing adaptation cost compared to full retraining. Our results highlight the practicality of efficient draft adaptation and suggest a promising direction for scalable speculative decoding in continually evolving LLM systems.

Acknowledgments and Disclosure of Funding

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0122703), the National Science Fund for Distinguished Young Scholars (No. 62525605), the National Natural Science Foundation of China (No. U25B2066, No. U22B2051, No. 62506313), and the Xiamen Municipal Science and Technology Bureau, China (3502ZC-QXT2024009).

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *NAACL*, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*, 2024.

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

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

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.

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.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *NeurIPS Datasets and Benchmarks Track*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021c.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

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

Hugging Face H4. AIME 2024. `https://huggingface.co/datasets/HuggingFaceH4/aime_2024`, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *EMNLP*, pages 2567–2577, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *EMNLP*, 2024b.

Luxi Lin, Zhihang Lin, Zhanpeng Zeng, and Rongrong Ji. Speculative decoding reimagined for multimodal large language models. *arXiv preprint arXiv:2505.14260*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with Evol-Instruct. In *ICLR*, 2024.

Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

OpenMeditron. Meditron3-Qwen2.5-7B. `https://huggingface.co/OpenMeditron/Meditron3-Qwen2.5-7B`, 2025.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *CHIL*, 2022.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *NAACL*, 2021.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI technical report, 2018.

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

ShareGPT. ShareGPT_Vicuna_unfiltered. `https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered`, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-Instruct. In *ICML*, 2024.

Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. PMC-LLaMA: Further finetuning LLaMA on medical papers. *arXiv preprint arXiv:2304.14454*, 2023.

Ming Xu. Medical: A Chinese and English medical dataset. `https://huggingface.co/datasets/shibing624/medical`, 2023.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In *NAACL*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *CVPR*, 2024a.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024b.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *ICLR*, 2024.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Guiming Chen, Jianquan Li, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, et al. HuatuoGPT, towards taming language model to be a doctor. In *Findings of EMNLP*, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *NeurIPS*, 2024.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *ICLR*, 2024.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *ICLR*, 2025.

A Implementation Details

Hyperparameters. All methods are trained with identical optimization settings for fair comparison. We use a batch size of 16, a learning rate of 4×10^{-5}, and train for 20 epochs. The maximum training sequence length is 2048 tokens, and the loss is computed only on assistant responses. For LoRA, we apply low-rank updates to all layers involved in draft adaptation and match its trainable parameter size with EDA. In EDA, the shared expert is initialized from the first-stage general draft model and frozen during adaptation, while the private expert and routers are randomly initialized and updated. For representation-shift-based selection, we reduce hidden representations to a 256-dimensional PCA space, add diagonal regularization to the covariance matrix, and select the top α fraction of samples; we set $\alpha = 0.5$ in the main experiments. During speculative decoding, we follow the default EAGLE-2 inference configuration with a draft-tree budget of 60 tokens, maximum tree depth of 5, and 10 retained candidates per step. We use batch size 1 and a maximum generation length of 512 tokens. We evaluate decoding under $T = 0$ and $T = 1$. All training experiments are conducted on 4 NVIDIA H200 GPUs, while inference latency and decoding speedup are measured on a single NVIDIA H200 GPU.

Draft Model Architecture. Our implementation starts from an EAGLE2-style draft model architecture and replaces the standard FFN module in each decoder layer with the proposed shared-private MoE module. The self-attention layers, layer normalization, embedding layer, and output prediction head follow the original draft model design. In each shared-private MoE block, a shared-private softmax router computes the gating weights for the shared and private experts jointly. During adaptation, the shared experts are initialized from the base draft model and kept frozen, while the private experts and routing parameters are updated for the fine-tuned target model. This implementation follows the design principle in Section 4.1: preserving reusable drafting behavior in the shared component while assigning lightweight trainable capacity to target-specific shifts.

Average Acceptance Length. Average acceptance length τ measures the expected number of consecutive draft tokens accepted by the target model before a rejection occurs:

$$\tau = \mathbb{E}_{x_{<t} \sim \mathcal{D}_{\text{test}}} \left[\sum_{i=1}^{K} \prod_{j=0}^{i-1} a_{t+j} \right], \tag{15}$$

where K is the maximum number of candidate tokens proposed, $a_{t+j} \in \{0, 1\}$ indicates whether the draft token $\hat{x}_{t+j}$ is accepted ($a_{t+j} = 1$), and $\mathcal{D}_{\text{test}}$ denotes the test data distribution. A higher τ indicates better alignment between the draft and target distributions, enabling more tokens to be generated per target-model forward pass and thereby enhancing inference efficiency.

B Datasets

B.1 Evaluation Benchmarks

We evaluate EDA on three downstream domains: mathematical reasoning, code generation, and medical question answering.

For mathematical reasoning, we use GSM8K Cobbe et al. [2021], AIME2024 Hugging Face H4 [2024], SVAMP Patel et al. [2021], H-MATH Hendrycks et al. [2021c], and MathQA Amini et al. [2019].

For code generation, we use HumanEval Chen et al. [2021], APPS Hendrycks et al. [2021a], BigCodeBench Zhuo et al. [2025], HumanEval+ Liu et al. [2023], and MBPP Austin et al. [2021].

For medical question answering, we use MedMCQA Pal et al. [2022], MedQA-USMLE Jin et al. [2021], PubMedQA Jin et al. [2019], MedQA Jin et al. [2021], and MMLU Hendrycks et al. [2021b].

For all benchmarks, prompts are constructed according to their standard evaluation protocols.

B.2 Training Data

We use a unified data budget of **68,000 training samples** for each training stage and domain to ensure consistent settings and comparable training cost.

For the base draft model with shared experts, we use **ShareGPT** ShareGPT [2023]. For mathematical reasoning, we use **DeepMath-103K** He et al. [2025]. For code generation, we use **Magicoder-Evol-Instruct-110K** Wei et al. [2024]. For medical question answering, we use the **Medical** dataset Xu [2023] and extract English medical QA samples.

C Architecture-Level Generalization

To examine whether EDA depends on a specific model family, we evaluate it on LLaMA3.1-8B-Math. This setting differs from the Qwen2.5 family used in our main experiments and provides a test of whether the shared/private draft adaptation principle can transfer across model families.

As shown in Table 8, EDA consistently outperforms Full-FT and parameter-matched LoRA across all evaluated math benchmarks. These results suggest that EDA is not merely a Qwen-specific adaptation recipe, but can also transfer to another Transformer backbone.

Table 8: Generalization to LLaMA3.1-8B-Math. We report average acceptance length τ and decoding speedup.

Method	GSM8K	MathQA	SVAMP	AIME24	Avg.
LoRA	4.12 / 2.18×	4.75 / 2.07×	3.69 / 2.26×	3.58 / 2.16×	4.04 / 2.17×
Full-FT	4.20 / 2.36×	4.84 / 2.12×	3.75 / 2.30×	3.66 / 2.19×	4.11 / 2.24×
EDA	**4.64 / 2.48**×	**5.22 / 2.44**×	**4.12 / 2.52**×	**3.91 / 2.41**×	**4.47 / 2.46**×

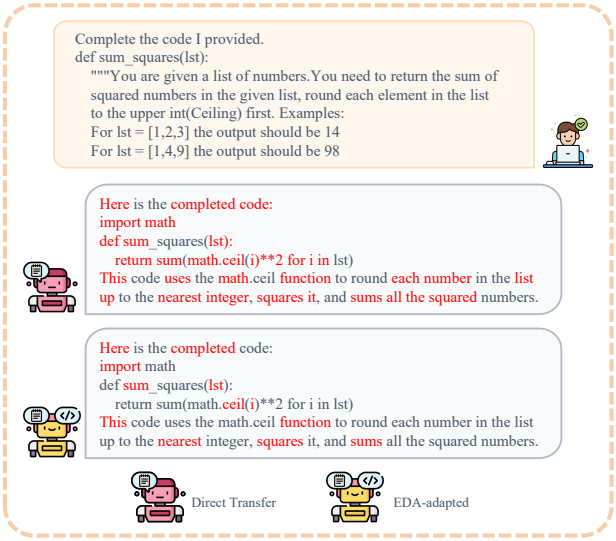

Figure 4: Qualitative example comparing direct transfer and EDA-adapted drafts on Qwen2.5-Coder-7B. Red tokens indicate incorrect predictions, while black tokens indicate correct predictions.

D Detailed Analysis of Data Selection

Figure 3 compares our PCA-based representation-shift metric with several training-free baselines. Random sampling uniformly selects adaptation samples. Low-confidence selection ranks samples by the negative average top-1 probability of the target model over answer tokens, favoring samples where the target model is less confident. High-entropy selection ranks samples by the average entropy of the target model's next-token distributions, favoring samples with more uncertain predictions. PCA-reversed selects samples with the smallest representation shifts, serving as the opposite ordering of our method.

As shown in Figure 3, our method reaches $\tau = 4.79$ using only 50% of the adaptation data, which is close to the full-data performance of 4.83. In contrast, random sampling obtains 4.65 at 50% and requires 75% of the data to reach 4.75. Low-confidence and high-entropy selection also underperform our method under the same data budgets, while PCA-reversed performs worst, reaching only 4.50 at 50%.

These results suggest that uncertainty alone is not the best indicator of sample value for draft adaptation. Low-confidence and high-entropy selection capture local prediction difficulty, whereas our representation-shift metric directly measures deviations from the general reference distribution, better matching the goal of adapting the private expert to target-specific shifts. Moreover, our method is training-free and reuses hidden states collected during self-generation, introducing negligible additional overhead.

E Qualitative Analysis

To better understand where EDA improves draft-target alignment, we conduct a qualitative case study by comparing direct transfer without adaptation and EDA-adapted drafts.

As shown in Figure 4, direct transfer often diverges from the target's code-generation trajectory and triggers early rejections. After EDA adaptation, the draft produces more code-consistent tokens, such as `def` and `return`, leading to longer consecutive acceptances and more accurate outputs.

We attribute this gain to EDA's efficient realignment: it reuses transferable regularities while lightly adapting target-specific capacity to capture fine-tuning-induced domain conventions. This suggests that EDA mainly corrects preference shifts caused by fine-tuning and formatting conventions, rather than re-learning general language patterns.

F Limitations

Our method is mainly designed for models within the same or closely related model family. Its effectiveness may be limited when applied across substantially different model families or architectures, where the learned representations and optimization behaviors can differ significantly. Extending the proposed method to more diverse model families remains an interesting direction for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions and scope. The experimental results support the proposed innovations and demonstrate their effectiveness.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the proposed method, including the scope of the experimental evaluation and potential constraints in practical applications. These limitations are briefly addressed to clarify the applicability of the method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include formal theoretical results, theorems, or mathematical proofs. Therefore, there are no theoretical assumptions or proofs to provide.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the experimental settings and implementation details needed to reproduce the main results. The code is also released to further support reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides open access to the code and data used in the experiments. Instructions are included to help reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental settings, including the datasets, training and testing procedures, hyperparameters, and evaluation protocols. These details are provided to make the experimental results understandable and reproducible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our evaluation focuses on speculative decoding efficiency metrics, including average acceptance length and decoding speedup, measured under fixed models, datasets, training budgets, and decoding configurations. Since speculative decoding is lossless, we do not repeat task-accuracy evaluations.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources used in the experiments are described in the experimental settings. This includes the hardware and runtime information needed to understand the cost of reproducing the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms to the NeurIPS Code of Ethics. The work does not involve procedures or applications that violate the ethical requirements of the conference.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper briefly discusses the broader impact of the proposed work, including its potential positive contribution to the related research community. Since the work is primarily methodological and not tied to a high-risk deployment scenario, we do not identify direct negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release data or models that pose a high risk of misuse, such as large pretrained generative models or scraped sensitive datasets. Therefore, specific safeguards for high-risk release are not applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Existing datasets, models, and code assets used in the paper are properly cited. The paper acknowledges the original sources and follows the corresponding usage terms where applicable.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code is documented with instructions for use and reproduction. No additional new dataset or high-risk asset is introduced beyond the resources described in the paper.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, human-subject studies, or manual annotation by participants. Therefore, participant instructions, screenshots, and compensation details are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research or crowdsourcing experiments. Therefore, IRB approval or equivalent ethical review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as an important, original, or non-standard component of the core methodology. They were only used for writing, editing, or polishing purposes and did not affect the scientific content or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.