Conservative Inference-Time Tuning with an Improvement Guarantee

Daiki Chijiwa
NTT, inc.
Computer and Data Science Laboratories
daiki.chijiwa@ntt.com

Taku Hasegawa
NTT, inc.
Human Informatics Laboratories
taku.hasegawa@ntt.com

Abstract

Foundation models have been widely adopted as starting points for fine-tuning, leading to expert-level performance on various downstream tasks. They have been actively developed by improving the pretraining data and/or model architectures. As a result, whenever a new foundation model is released, we have to fine-tune it again on the downstream tasks, requiring non-negligible computational costs as well as data-management costs. A promising approach to this problem is inference-time model update, which modifies the prediction of the new pretrained model using those of the old pretrained/fine-tuned models. However, the existing methods have no theoretical guarantee of improvement in accuracy and may cause unexpected failure. In this paper, we establish the first method for inference-time model update with a theoretical guarantee of improvement in expected rewards, and show that our method can improve performance even when the existing methods fail theoretically. Finally, we empirically validate the effectiveness of our method in practical experiments.

1 Introduction

Foundation models (Bommasani et al., 2021) or simply base models—which are models trained on massive data for general tasks or domains—have been widely adopted as starting points for *fine-tuning*, i.e., training expert models on target tasks or domains. While modern foundation models can handle new tasks even in a few-shot or zero-shot manner (Brown et al., 2020), fine-tuning still boosts their performance on target tasks significantly.

By scaling up their pretraining data or architectures, foundation models are progressively updated with improved performance. As a result, whenever foundation models are updated, their fine-tuned counterparts must also be retrained from the new foundation models. Such retraining requires not only computational costs but also data-management costs for each downstream dataset.

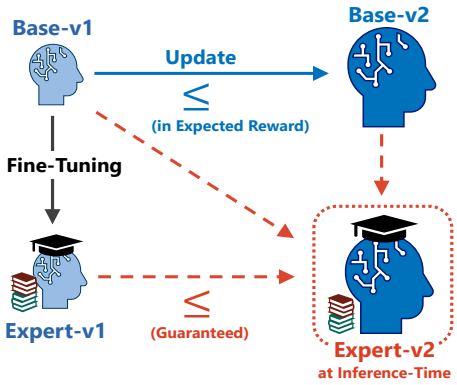


Figure 1: Our framework constructs a new expert model at inference-time, from the pair of old base/expert models and a new base model, and guarantees the new expert is "better" (see Section 3.3 for the definition) than the old expert if the new base is "better" than the old base.

To avoid such retraining costs, there are mainly two approaches that transfer the fine-tuned knowledge or ability across different base models either (i) in the parameter space (Chijiwa, 2024), or (ii) in the logit space (Mitchell et al., 2024; Liu et al., 2024). On the one hand, the former approach transforms the fine-tuning dynamics from an old base model into the one from a new base model, by exploiting symmetries in the parameter space. However, because it operates in parameter space, this approach is inherently constrained to models with compatible architectures. On the other hand, the latter approach—which we call inference-time tuning—is more promising in the near term, because it is agnostic to model architectures and does not require additional training. In particular, it can transfer the fine-tuned knowledge at inference time, from the old base model to the new base model even with a different size or architecture, as long as they have the same logit/prediction space.

Nevertheless, while the existing methods of the

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 25995–26005
October 24-29, 2026 ©2026 Association for Computational Linguistics

logit-level approach are based on the principle of KL-regularized reward maximization (Ziebart, 2010), they still lack any theoretical guarantee for performance improvement, which may lead to unexpected failures as we will see in Section 4. This motivates us to explore an alternative principle for inference-time tuning whose robustness is theoretically guaranteed without any additional training.

In this work, as an alternative principle to the prior one, we establish the framework of *conservative inference-time tuning* (Figure 1) with theoretical guarantees of improvements in token-level (implicit) rewards. To this end, we first observe that the prediction of a fine-tuned model can be decomposed into two parts: one part is inherited from the old base model, and the residual part contains the task-specific knowledge. Then we identify the inherited part and replaces it with the prediction of the new base model. Since this construction can be seen as a mixture-type ensemble, an improvement guarantee in token-level rewards can be derived. Experiments show that, even in cases the prior methods fail, the proposed method can still improve the accuracy of expert models consistently.

More related work are discussed in Appendix A.

2 Backgrounds

Let us consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ with $|\mathcal{Y}| < \infty$. For example, in language generation, $x \in \mathcal{X}$ is a token sequence of length t and $y \in \mathcal{Y}$ is a possible next token.

Predictive Models. In machine learning, a *predictive distribution*, or simply called a *model*, refers to a conditional probability distribution $p_\theta(y|x)$ with a parameter θ that assigns a probability of a possible response $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$.

Particularly in deep learning, predictive models are implemented by neural networks $f_\theta(x) = (f_\theta(x)_y)_{y\in\mathcal{Y}} : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ parameterized by a high-dimensional vector $\theta \in \mathbb{R}^N$, which receives an input x and returns the corresponding $|\mathcal{Y}|$-dimensional vector called a *logit*. Then the conditional probability $p_\theta(y|x)$ is obtained by applying the softmax function $\sigma : \mathbb{R}^{|\mathcal{Y}|} \to \mathcal{P}^{|\mathcal{Y}|}$ to the logit $f_\theta(x)$, where $\mathcal{P}^{|\mathcal{Y}|} := \{(p_1, \cdots, p_n) \in \mathbb{R}^{|\mathcal{Y}|} \mid \sum_i p_i = 1\}$ denotes the space of all probabilities over $\mathcal{Y}$, and $\sigma(\mathbf{z})$ is defined by $\sigma(\mathbf{z}) := (\exp(z_i)/\sum_k \exp(z_k))_{i=1,\cdots,|\mathcal{Y}|}$ for $\mathbf{z} = (z_1, \cdots, z_{|\mathcal{Y}|}) \in \mathbb{R}^{|\mathcal{Y}|}$.

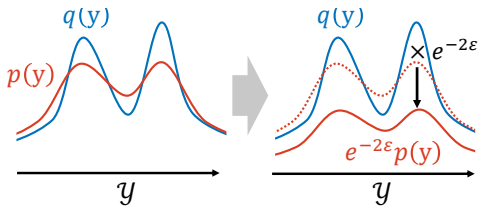


Figure 2: After scaled appropriately, the base model $p(-|x)$ can be embedded into the fine-tuned model $q(-|x)$ in the prediction space over $\mathcal{Y}$.

Logit-Level Model Update. A logit-level approach such as emulated fine-tuning (EFT; Mitchell et al. (2024)) and equivalently proxy-tuning (Liu et al., 2024) has been known as a promising approach that does not depend on model architectures. Both EFT and proxy-tuning calculate a new predictive distribution imitating the updated model at inference time:

$$\begin{aligned}\widetilde{p}_{\mathrm{ft}}(y|x) &:= \sigma(f_{\widetilde{\theta}_{\mathrm{pt}}}(y|x) + f_{\theta_{\mathrm{ft}}}(y|x) - f_{\theta_{\mathrm{pt}}}(y|x)) \\ &= \frac{1}{Z(x)} p_{\widetilde{\theta}_{\mathrm{pt}}}(y|x) \left(\frac{p_{\theta_{\mathrm{ft}}}(y|x)}{p_{\theta_{\mathrm{pt}}}(y|x)}\right), \qquad (1)\end{aligned}$$

where $f_\theta(y|x)$ is a logit before softmax and $Z(x)$ is the normalizing factor. As discussed in Mitchell et al. (2024), eq. (1) can be interpreted as a closed-form solution of KL-regularized reward maximization as follows:

$$\begin{aligned}\widetilde{p}_{\mathrm{ft}}(y|x) = \arg\max_{p(y|x)} \Big\{ &\mathbb{E}_{y\sim p(y|x)}\left[R(y|x)\right] \\ &- D_{\mathrm{KL}}\left(p(-|x) \,\|\, p_{\widetilde{\theta}_{\mathrm{pt}}}(-|x)\right)\Big\}, \qquad (2)\end{aligned}$$

where $R(y|x) := \log(p_{\theta_{\mathrm{ft}}}(y|x)/p_{\theta_{\mathrm{pt}}}(y|x))$ can be seen as an implicit reward for the target task.

3 Methods

3.1 A Motivating Observation

Let $p(y|x) = \sigma(f_\theta(x))$ be some base model and $q(y|x) = \sigma(f_{\theta'}(x))$ be its fine-tuned model, where σ is the softmax function as defined in Section 2. Let $\Delta(x) := f_{\theta'}(x) - f_\theta(x)$ be the difference in logits between these models, and we suppose each of its entries $\Delta(x) = (\Delta(x)_y)_{y\in\mathcal{Y}}$ is bounded by some small number ε_x, i.e., $|\Delta(x)_y| \le \varepsilon_x$ for all $y \in \mathcal{Y}$. Then we have

$$\begin{aligned}q(y|x) = \frac{\exp(f_{\theta'}(x)_y)}{Z'_x} &= \frac{\exp(f_\theta(x)_y + \Delta(x)_y)}{Z'_x} \\ &\ge \frac{\exp(f_\theta(x)_y)e^{-\varepsilon_x}}{e^{\varepsilon_x} Z_x} = e^{-2\varepsilon_x} p(y|x),\end{aligned}$$

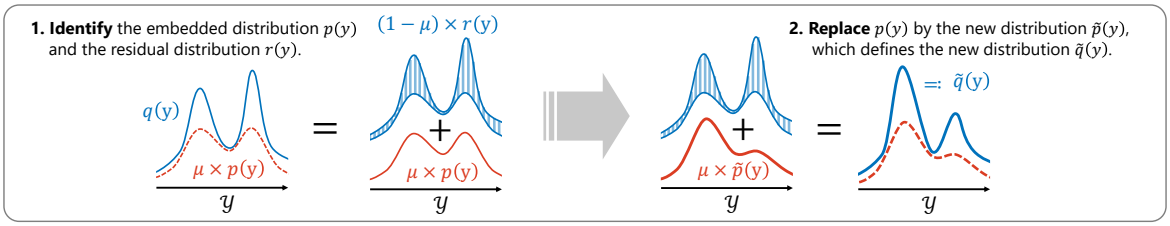


Figure 3: Contrastive Mixture Prediction (CMP) constructs the new distribution $\widetilde{q}$ by first identifying the embedded distribution p in the target distribution q, and then replacing p with the new distribution $\widetilde{p}$.

where Z_x and Z'_x are the normalization factors of softmax. Thus, it implies these two models p and q satisfy $\mu_x p(y|x) \leq q(y|x)$ for some $\mu_x \in (0, 1]$, with at least $\mu_x \geq e^{-2\varepsilon_x}$. We can interpret this relation geometrically as shown in Figure 2, where the base prediction p scaled by μ_x can be seen as embedded into the fine-tuned prediction q. Here μ_x can be close to 1 if q has not largely deviated from p during fine-tuning, and otherwise μ_x can be small if q has been drastically changed.

The above factor μ_x also indicates how much q inherits its behavior from p. Particularly, we can easily see that q is decomposed as follows:

$$q(y|x) = (1 - \mu_x)r(y|x) + \mu_x p(y|x), \quad (3)$$

where $r(y|x) := (q(y|x) - \mu_x p(y|x))/(1 - \mu_x)$ is the residual distribution[1]. In other words, among the outputs y sampled from $q(y|x)$, some of them are samples from $p(y|x)$ with probability μ_x, and the others are from $r(y|x)$ with probability $1 - \mu_x$. Since p does not depend on training data used for fine-tuning q, the residual distribution r successfully decouples information acquired by fine-tuning from the entire information represented by q.

3.2 Proposed Method

Given a new base model $\widetilde{p}$, an old base model p and its fine-tuned model q, our goal is to construct a new prediction $\widetilde{q}(-|x)$ by updating the prediction $q(-|x)$ from the old fine-tuned model at inference time, as if the part of the prediction inherited from p was replaced by that of the new base model $\widetilde{p}$. To this end, following the decomposition formula in (3), we decompose $q(-|x)$ into the task-specific part $r(-|x)$ and the task-independent part $p(-|x)$. The core idea of our method is to replace the task-independent part by the new base model in this decomposition. Specifically, as illustrated in Figure 3, our method consists of the following steps.

1. Identify the *embedding ratio* of p in q as the maximal $\mu_x = \mu_x^*$ with $\mu_x p(-|x) \leq q(-|x)$.
2. Replace the old distribution $p(-|x)$ embedded in $q(-|x)$ by the new distribution $\widetilde{p}(-|x)$.

By these steps we get a new prediction defined by

$$\begin{aligned}\widetilde{q}(y|x) &:= (1 - \mu_x^*)r(y|x) + \mu_x^*\widetilde{p}(y|x), \\ &= q(y|x) + \mu_x^*(\widetilde{p}(y|x) - p(y|x)), \quad (4)\end{aligned}$$

where $r(y|x) := (q(y|x) - \mu_x^* p(y|x))/(1 - \mu_x^*)$, the task-dependent part of q as discussed in Section 3.1, and $\mu_x^* := \min_{y \in \mathcal{Y}} q(y|x)/p(y|x)$ [2] is the embedding ratio given $x \in \mathcal{X}$. We refer to this new prediction by Contrastive Mixture Prediction (CMP). In contrast to the exponential-type formulation of the existing methods (1) taking the difference in the logit space, our formulation can be seen a mixture-type formulation taking the difference in the prediction space.

3.3 Theoretical Guarantees

Recall that, as explained in Section 2, the existing methods can be seen as maximization of the token-level implicit reward $R(y|x)$ with KL-regularization (2). Motivated by this perspective, we consider that a token-level oracle reward $R(y|x) : \mathcal{Y} \to \mathbb{R}_{\geq 0}$ for the target task is implicitly given. For example, when x is a multiple-choice problem, $R(y|x)$ can be set to the indicator function that returns 1 only if y is the token corresponding to the correct choice and otherwise returns 0, in which case its expected reward $\mathbb{E}_{y \sim p(y|x)}[R(y|x)]$ is nothing but the probability that p assigns to the correct choice given the problem x. Beyond this example, more general rewards can be considered as $R(y|x)$, such as an oracle reward for evaluating grammatical correctness at each token.

Hereafter, to simplify the notation, we consider the input x to be fixed, and omit it from equations,

[1] In the case $\mu_x = 1$, the residual r is undefined but then we can consider $(1 - \mu_x)r(y|x)$ to be 0 in (3).

[2] The quantity μ_x^* also plays a central role in *rejection sampling* (Von Neumann, 1963) for a target q with a proposal p.

Model q	Model p	Model $\widetilde{p}$	Methods	MBPP	HumanEval	BigCodeBench
Qwen2.5-Coder-1.5B (Instruct)	Qwen2.5-1.5B (Base)	N/A	q	62.43	60.37	18.80
		Qwen3-1.7B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	27.51 (−34.9)	57.93 (−2.44)	13.20 (−5.60)
			$\widetilde{q}_{\mathrm{CMP}}$	65.08 (+2.65)	64.02 (+3.65)	18.80 (±0.00)
		Qwen2.5-7B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	41.27 (−21.2)	41.46 (−18.9)	8.40 (−10.4)
			$\widetilde{q}_{\mathrm{CMP}}$	71.96 (+9.53)	59.76 (−0.61)	21.20 (+2.40)
Qwen2.5-Coder-3B (Instruct)	Qwen2.5-3B (Base)	N/A	q	74.60	73.17	24.60
		Qwen3-4B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	79.37 (+4.77)	75.00 (+1.83)	21.00 (−3.60)
			$\widetilde{q}_{\mathrm{CMP}}$	78.57 (+3.97)	78.65 (+5.48)	24.60 (±0.00)
		Qwen2.5-7B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	61.11 (−13.5)	74.39 (+1.22)	20.00 (−4.60)
			$\widetilde{q}_{\mathrm{CMP}}$	79.62 (+5.02)	75.00 (+1.83)	24.40 (−0.20)

Table 1: Evaluation with coding models on standard coding benchmarks. The method q refers to the expert model itself, $\widetilde{q}_{\mathrm{EFT}}$ refers to the prior method defined by (1), and $\widetilde{q}_{\mathrm{CMP}}$ refers to ours defined by (4). The colored numbers in parentheses indicate the difference from the results of the expert model q.

like $R(y)$ instead of $R(y|x)$. Given such a task-specific reward $R(y)$, we can provide a theoretical guarantee for how the expected reward $\mathbb{E}_q[R(y)]$ changes under the new prediction (4):

Proposition 3.1. *Let $p(y), q(y), \widetilde{p}(y)$ be distributions over $\mathcal{Y}$ and $\mu \geq 0$ be a constant satisfying $\mu p(y) \leq q(y)$ for all $y \in \mathcal{Y}$. Then we have two distributions $r(y) := (q(y) - \mu p(y))/(1-\mu)$ and $\widetilde{q}(y) := (1-\mu)r(y) + \mu\widetilde{p}(y)$ as discussed above, satisfying the following:*

(i) $\mathbb{E}_{\widetilde{q}}[R(y)] \geq (1-\mu)\mathbb{E}_r[R(y)]$
$= \mathbb{E}_q[R(y)] - \mu\mathbb{E}_p[R(y)]$,

(ii) If p and $\widetilde{p}$ satisfy $\mathbb{E}_p[R(y)] \leq \mathbb{E}_{\widetilde{p}}[R(y)]$, then q and $\widetilde{q}$ satisfy $\mathbb{E}_q[R(y)] \leq \mathbb{E}_{\widetilde{q}}[R(y)]$.

Proof. See Appendix D.2. □

The first result (i) provides the **robustness guarantee** of our model (4) that holds regardless of whether the new base model $\widetilde{p}$ is better or worse than the old base model p. The second result (ii) provides the **improvement guarantee** that holds when the new $\widetilde{p}$ achieves better (or at least equal) expected reward than the old p. On the other hand, the previous logit-level methods formulated by (1) do not have such a guarantee of the reward improvement, even when $\widetilde{p}$ achieves better expected reward than p, as shown by a counterexample in Appendix D.3.

4 Experiments

In this section, we empirically validate the robustness of our mixture-type approach (4) compared to the previous exponential-type approach (1).

While the previous approach has been known to work well with instruction-following expert models (Mitchell et al., 2024; Liu et al., 2024), we first observe that it unexpectedly fails with specific combinations of models and tasks. Specifically, here we focus on expert models tuned for coding tasks, Qwen2.5-Coders (Hui et al., 2024), with either scaled-up or upgraded versions of the base model, Qwen2.5 (Yang et al., 2024a).

Results. Table 1 presents the evaluation results on standard coding benchmarks across different model configurations. In the case of the smaller expert model (Qwen2.5-Coder-1.5B), the previous method significantly degrades the performance of q with both the new base models (Qwen3-1.7B and Qwen2.5-7B). In contrast, our method consistently improves the performance of q by replacing the old base model p with the new base model $\widetilde{p}$, even when the previous method fails, as our theoretical results in Section 3.3 suggested. In the case of the larger expert model (Qwen2.5-Coder-3B), the previous method does not degrade the performance of q with the new base models (Qwen3-4B and Qwen2.5-7B), while our method also achieves comparable or better performance compared to the previous method. These results indicate the robustness of our method as expected by its theoretical guarantees, even when the pair of the base and expert models (p, q) is more vulnerable to the change of the base model in the previous method.

5 Conclusion

In this paper, we established a new principle for inference-time tuning based on the mixture formulation, instead of the exponential formulation in the prior work, and demonstrated its robustness both theoretically and empirically. We hope that this mixture-based perspective will provide a principled basis for designing inference-time tuning methods that are both effective and reliable.

Limitations

(i) Our proposed method is designed to be inherently robust to the choice of the new base models. In contrast, existing methods can modify q more aggressively, as they do not provide any guarantee. Consequently, for some combinations of the base and target models, such aggressive modification may lead to substantial performance gains. Our contribution is the first attempt to establish an alternative principle that have theoretical guarantees for performance improvement, rather than a superior one. Unifying advantages of these different approaches remains a promising direction for future research. (ii) The theoretical guarantees we have provided are based on the expected reward, which may not always correlate well with the actual performance of the model on specific tasks. Developing more refined theoretical/empirical analyses that better capture the practical performance of the models is an important direction for future research. (iii) Our experiments focus primarily on the coding models and tasks where we find that the existing methods unexpectedly fails due to the lack of theoretical guarantees. Although such failure cases may arise in other combinations of models and tasks, an exhaustive evaluation of all such cases remain for future work.

References

AIME. Aime problems and solutions.

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

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, and 95 others. 2021. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Thomas Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Daiki Chijiwa. 2024. Transferring learning trajectories of neural networks. In The Twelfth International Conference on Learning Representations.

Daiki Chijiwa, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Susumu Takeuchi. 2024. Base-change at prediction: Inference-time update of fine-tuned models. In ICML 2024 Workshop on LLMs and Cognition.

Daiki Chijiwa, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Susumu Takeuchi. 2025. Portable reward tuning: towards reusable fine-tuning across different pretrained models. In Proceedings of the 42nd International Conference on Machine Learning, ICML'25. JMLR.org.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 11781–11791, Singapore. Association for Computational Linguistics.

Yu Fei, Yasaman Razeghi, and Sameer Singh. 2025. Nudging: Inference-time alignment of LLMs via guided decoding. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12702–12739, Vienna, Austria. Association for Computational Linguistics.

Yuanyang He, Zitong Huang, Salman Khan, Wangmeng Zuo, Ling Shao, and Chun-Mei Feng. 2026. Cpt: Consistent proxy tuning for black-box models. In ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 18007–18011.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. Advances in neural information processing systems.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2. 5-coder technical report. arXiv preprint arXiv:2409.12186.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi:

Generative discriminator guided sequence generation. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 4929–4952.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12286–12312.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. arXiv preprint arXiv:2401.08565.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 6691–6706, Online. Association for Computational Linguistics.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In Thirty-seventh Conference on Neural Information Processing Systems.

Xinyu Lu, Xueru Wen, Yaojie Lu, Bowen Yu, Hongyu Lin, Haiyang Yu, Le Sun, Xianpei Han, and Yongbin Li. 2025. Transferable post-training via inverse value learning. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4436–4447, Albuquerque, New Mexico. Association for Computational Linguistics.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, and 88 others. 2024. Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2024. An emulator for fine-tuning large language models using small language models. In The Twelfth International Conference on Learning Representations.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. CombLM: Adapting black-box language models through small fine-tuned models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 2961–2974, Singapore. Association for Computational Linguistics.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3973–3997, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alfréd Rényi. 1961. On measures of entropy and information. In Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics, volume 4, pages 547–562. University of California Press.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, and 178 others. 2024. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118.

John Von Neumann. 1963. Various techniques used in connection with random digits. John von Neumann, Collected Works, 5(768-770):1.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

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. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. arXiv preprint arXiv:2409.12122.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3511–3535, Online. Association for Computational Linguistics.

Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, Siqi Zuo, Andrea Hu, Christopher A. Choquette-Choo, Jingyue Shen, Joe Kelley, Kshitij Bansal, Luke Vilnis, Mateo Wirth, Paul Michel, Peter Choy, Pratik Joshi, Ravin Kumar, Sarmad Hashmi, Shubham Agrawal, Zhitao Gong, and 7 others. 2024. Codegemma: Open code models based on gemma. arXiv preprint arXiv:2406.11409.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, and 14 others. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. arXiv preprint arXiv:2406.15877.

Brian D Ziebart. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. Carnegie Mellon University.

Appendices

A Related Work

Logit-Level Model Update. In this paper, we have focused on the problem of updating predictions from a fine-tuned model by using ones from old and new base models at inference time. To solve this problem, most of the prior work have employed an approach that takes difference of logits between the old and new base models. Emulated fine-tuning (Mitchell et al., 2024) and proxy-tuning (Liu et al., 2024) are early prior work in this direction, where the former has its origins in KL-regularized reward maximization (Ziebart, 2010) and the latter has its origins in decoding methods for enhancing the experts' capability (Liu et al., 2021; Li et al., 2023). Chijiwa et al. (2024) proposed to swap the roles of the base model and expert model in the KL-regularized reward maximization and tune the regularization coefficient, but introducing additional training cost. Deng and Raffel (2023), Lu et al. (2025), and Chijiwa et al. (2025) proposed to replace the implicit reward in (2) with an explicit reward model, which requires training in advance. He et al. (2026) applied proxy-tuning to black-box training using only the logit information. Contrary to our mixture-type approach, the logit-level approach of these work lack the theoretical guarantees as discussed in Section 3.3.

Other Inference-Time Approaches. There have also been approaches to steer a large base model by a small expert model in which the corresponding small base model is not leveraged, contrary to our setting. Ormazabal et al. (2023) proposed a learned combination of predictions from these models, but which introduces additional training cost and thus does not meet our requirement. Fei et al. (2025) proposed a training-free method called Nudging that interrupts the inference of the large base model, when its top-1 probability is below a threshold, and then uses the small expert model instead for generating a fixed number of tokens. Since its inference is mostly carried out by the large base model, it also may fail to leverage the expert's capability in our experimental settings. See Appendix E.3 for a more detailed comparison with our method.

There is also a line of work on controlled text generation by attribute conditioning (Krause et al., 2021; Yang and Klein, 2021; Pascual et al., 2021). Such an approach requires additional training of classifier models for each attribute, and thus is not directly comparable to our method that does not require any training.

B Experimental Details

Models.

- Qwen2.5 (Base): Qwen2.5 is a family of pre-trained language models with various model sizes (Yang et al., 2024a). We used the 3B and 7B parameter versions as the base models in our experiments.
- Qwen3 (Base): Qwen3 is an upgraded version of Qwen2.5 with improved architecture, datasets and training recipes (Yang et al., 2025). We used the 1.7B and 4B parameter versions as the new base models that are similar sizes to the old base models in our experiments.
- Qwen2.5-Coder (Instruct): Qwen2.5-Coder is a family of instruction-tuned models based on Qwen2.5, specialized in coding tasks (Hui et al., 2024). We used the 1.5B and 3B parameter versions as the expert models in our experiments.

Datasets.

- HumanEval (Chen et al., 2021): HumanEval is a benchmark for evaluating the functional correctness of Python code generation, publicly available under the MIT License. We used its variant called HumanEval+ (Liu et al., 2023) which consists of 164 Python programming problems from the HumanEval dataset, with more test cases.
- MBPP (Austin et al., 2021): MBPP (Mostly Basic Python Problems) is a benchmark for evaluating the ability of language models to synthesize short Python programs from natural-language descriptions, publicly available under the CC-BY-4.0 License. We used its variant called MBPP+ (Liu et al., 2023), which consists of 399 Python programming tasks from the original MBPP dataset, with more test cases.
- BigCodeBench (Zhuo et al., 2024): BigCodeBench is a benchmark for evaluating code generation on practical and challenging programming tasks involving diverse function calls and complex instructions. It consists of

1,140 fine-grained Python programming tasks with 139 libraries and 7 domains. It is publicly available under the Apache 2.0 License.

We used these datasets for the research purpose only, which is allowed by their licenses.

Evaluation settings. The generation is performed with greedy decoding and the maximum generation length is set to 2048 tokens. The evaluation is performed by extracting the generated code wrapped by a code block, executing it and checking whether it passes the test cases provided in each dataset. To ensure the code is generated within a code block, we provide the following system prompt to each coding model: "You are a helpful assistant. Please reason step by step, and put your final answer within a code block."

Computing resources. All the experiments were conducted on two NVIDIA L40S GPUs.

C Implementation Details

Truncate relatively-negligible probabilities in p. In the case of language models, the output space $\mathcal{Y}$ is the set of all tokens shared among models (p, q and $\widetilde{p}$), whose size is typically ranging from tens of thousands to hundreds of thousands. As a result, most of $y \in \mathcal{Y}$ is irrelevant to the input x. However, due to the positive nature of the softmax function, such irrelevant tokens also have positive (almost meaningless) probabilities, which results in unreasonably small $\mu_x^* = \min_y q(y)/p(y)$. Thus, in our implementation, we regard $p_{x,\varepsilon} := \varepsilon \max_y p(y|x)$ with $\varepsilon = 0.1$ as a relatively-negligible threshold, and set $p(y|x)$ to zero if it is less than $p_{x,\varepsilon}$.

Pseudocode. The pseudocode for our method is described in Algorithm 1. Here are a few remarks: (i) In the specific case of language models, the input x is typically the concatenation of initial prompt tokens and past output tokens, and the output y corresponds to the next token. (ii) Line 4: The `truncate` function refers to the truncation operation described above using the negligible threshold $p_{x,\varepsilon}$. However, it can be replaced by other natural truncation techniques such as top-p and top-k truncations. (iii) Line 5: Theoretically, we can assume $p(y|x) > 0$ and $q(y|x) > 0$ due to the softmax function as explained in Section 2. Nevertheless, even if there is some y with $p(y|x) \ll q(y|x)$ and thus $p(y|x) \approx 0$ relatively, which may lead to the zero-division problem, we can safely ignore such y.

Algorithm 1 Contrastive Mixture Prediction (CMP)

Require: Models $p, q, \widetilde{p}$; input $x \in \mathcal{X}$
1: Set $\widetilde{\mathsf{q}}[y] \leftarrow 0$ for each $y \in \mathcal{Y}$
2: Get $q(-|x), p(-|x)$ by forward computations
3: $\mathsf{q}[y] \leftarrow q(y|x),\ \mathsf{p}[y] \leftarrow p(y|x)$ for all $y \in \mathcal{Y}$
4: $\mathsf{p} \leftarrow \texttt{truncate}(\mathsf{p})$
5: $\mu \leftarrow \min_{y \in \mathcal{Y}}\{\mathsf{q}[y]/\mathsf{p}[y]\}$
6: Get $\widetilde{p}(-|x)$ by forward computation
7: $\widetilde{\mathsf{p}}[y] \leftarrow \widetilde{p}(y|x)$ for all $y \in \mathcal{Y}$
8: $\widetilde{\mathsf{q}}[y] \leftarrow \mathsf{q}[y] + \mu(\widetilde{\mathsf{p}}[y] - \mathsf{p}[y])$ for all $y \in \mathcal{Y}$
9: **return** $\widetilde{\mathsf{q}}$

This is because there must always be another y satisfying $p(y|x) > q(y|x)$ which means $p(y|x) \not\approx 0$ relatively.

D Theoretical Discussions

D.1 A remark on μ_x^* and Rényi divergence

Let us write

$$D_\alpha\big(p \,\big\|\, q\big) := \frac{1}{\alpha - 1} \log \mathbb{E}_{y \sim p(y)} \left[\left(\frac{p(y)}{q(y)} \right)^{\alpha - 1} \right],$$

the Rényi divergence of order α of p from q (Rényi, 1961). The Rényi divergence of order $\alpha \to \infty$ can be written as $D_\infty\big(p \,\big\|\, q\big) = \log \sup_y p(y)/q(y)$. Thus, we can easily see that the quantity μ_x^* can be interpreted as

$$\mu_x^* = e^{-D_\infty\left(p(-|x) \,\middle\|\, q(-|x)\right)},$$

In particular, μ_x^* gets close to 1 when p and q become similar distributions to each other.

D.2 Proof of Proposition 3.1

Proof. The point of the proof is that the decomposition of q and $\widetilde{q}$ as mixture distributions implies the decomposition of the expected reward as follows:

$$\begin{aligned}
\mathbb{E}_q[R(y)] &= (1-\mu)\mathbb{E}_r[R(y)] + \mu\mathbb{E}_p[R(y)], \\
\mathbb{E}_{\widetilde{q}}[R(y)] &= (1-\mu)\mathbb{E}_r[R(y)] + \mu\mathbb{E}_{\widetilde{p}}[R(y)].
\end{aligned}$$

Then we can show (i) by the non-negativity of the reward $R(y)$:

$$\begin{aligned}
\mathbb{E}_{\widetilde{q}}[R(y)] &= (1-\mu)\mathbb{E}_r[R(y)] + \mu\mathbb{E}_{\widetilde{p}}[R(y)] \\
&\geq (1-\mu)\mathbb{E}_r[R(y)].
\end{aligned}$$

Similarly, if p and $\widetilde{p}$ satisfy $\mathbb{E}_p[R(y)] \leq \mathbb{E}_{\widetilde{p}}[R(y)]$, we can show (ii):

$$\begin{aligned}
\mathbb{E}_{\widetilde{q}}[R(y)] &= (1-\mu)\mathbb{E}_r[R(y)] + \mu\mathbb{E}_{\widetilde{p}}[R(y)] \\
&\geq (1-\mu)\mathbb{E}_r[R(y)] + \mu\mathbb{E}_p[R(y)] \\
&= \mathbb{E}_q[R(y)].
\end{aligned}$$

□

Model q	Model p	Model $\widetilde{p}$	Methods	MBPP	HumanEval	BigCodeBench
CodeGemma-7B (Instruct)	Gemma-7B (Base)	N/A	q	69.05	47.56	19.80
		Gemma2-27B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	3.44 (−65.61)	0.00 (−47.56)	0.00 (−19.80)
			$\widetilde{q}_{\mathrm{CMP}}$	70.37 (+1.32)	42.07 (−5.49)	21.00 (+1.20)

Table 2: Results with Gemma models.

Model q	Model p	Model $\widetilde{p}$	Methods	MATH	AIME'25	AIME'26
Qwen2.5-Math-1.5B (Instruct)	Qwen2.5-1.5B (Base)	N/A	q	61.42	12.00	10.60
		Qwen3-1.7B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	58.09 (−3.33)	6.00 (−6.00)	6.00 (−4.6)
			$\widetilde{q}_{\mathrm{CMP}}$	60.00 (−1.42)	6.00 (−6.00)	5.30 (−5.30)
		Qwen2.5-7B (Base)	$\widetilde{q}_{\mathrm{EFT}}$	52.80 (−8.62)	6.00 (−6.00)	2.00 (−8.60)
			$\widetilde{q}_{\mathrm{CMP}}$	61.40 (−0.02)	12.60 (+0.60)	8.00 (−2.60)

Table 3: Results on math benchmarks.

D.3 Counterexample for Logit-Level Methods

Here we show that the logit-level prediction model (1), introduced in Section 2 as previous methods, do not satisfy the improvement guarantee (Proposition 3.1 (ii)) in contrast to our prediction model 4.

Let $\mathcal{Y} = \{y_1, y_2, y_3\}$ be the output space and $p(y), q(y), \widetilde{p}(y)$ be probabilistic distributions over $\mathcal{Y}$. Let us consider the following examples:

$$R(y) := \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \; p(y) := \begin{pmatrix} 6/8 \\ 1/8 \\ 1/8 \end{pmatrix},$$
$$\widetilde{p}(y) := \begin{pmatrix} 1/8 \\ 5/8 \\ 2/8 \end{pmatrix}, \; q(y) := \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}.$$

In this case, we have $\mathbb{E}_p[R(y)] = 1/8 < 2/8 = \mathbb{E}_{\widetilde{p}}[R(y)]$ and thus they satisfy the assumption of (ii) in Proposition 3.1. On the other hand, $\widetilde{q}(y)$ is given by

$$\widetilde{q}(y) = \frac{1}{Z} q(y) \frac{\widetilde{p}(y)}{p(y)} = \begin{pmatrix} 1/43 \\ 30/43 \\ 12/43 \end{pmatrix},$$

where $Z := \sum_y q(y)\widetilde{p}(y)/p(y)$ is the normalization factor. Then we have $\mathbb{E}_{\widetilde{q}}[R(y)] = 12/43 < 1/3 = \mathbb{E}_q[R(y)]$, and thus these examples provide a counterexample to Proposition 3.1 (ii).

E Additional Experiments

E.1 Experiments with Gemma models

In Table 2, we conducted experiments with the Gemma models; we used CodeGemma-7B (Zhao et al., 2024) as the expert model, Gemma-7B (Mesnard et al., 2024) as its base model, and Gemma2-27B (Riviere et al., 2024) as an improved base model. It can be seen that, while the EFT tends to generate collapsed codes that are invalid as Python scripts, our method successfully improves the coding performance on MBPP and BigCodeBench.

E.2 Experiments on Math Benchmarks

In Table 3, we conducted experiments with a math expert model of Qwen2.5-Math-1.5B (Yang et al., 2024b) on challenging math benchmarks: MATH (Hendrycks et al., 2021), AIME'25 and AIME'26 (AIME). While our method slightly underperforms the expert model, it keeps degradation in accuracy to a lower level compared to EFT.

E.3 Comparison with Nudging (Fei et al., 2025)

In Table 4, we compare our method with Nudging (Fei et al., 2025), which is a training-free method that interrupts the inference of the large base model when its top-1 probability is below a threshold and then uses the small expert model instead for generating a fixed number of tokens. The results show that, while Nudging can achieve significant performance gains in some cases, but it also suffers from significant performance degradation in other cases, particularly with the smaller expert model (Qwen2.5-Coder-1.5B). This indicates that Nudging is not robust in leveraging the expert's capability in our experimental settings as well as the exponential-type previous methods. Nevertheless, it is noteworthy that Nudging can be complementary to our method, as it can further improve the performance of the base model in some cases where our method does not achieve significant im-

Model q	Model p	Model $\widetilde{p}$	Methods	MBPP	HumanEval	BigCodeBench
Qwen2.5-Coder-1.5B (Instruct)	Qwen2.5-1.5B (Base)	N/A	q	62.43	60.37	18.80
		Qwen3-1.7B (Base)	$\widetilde{q}_{\text{Nud}}$	65.34 (+2.91)	57.31 (−3.06)	17.00 (−1.80)
			$\widetilde{q}_{\text{CMP}}$	65.08 (+2.65)	64.02 (+3.65)	18.80 (±0.00)
		Qwen2.5-7B (Base)	$\widetilde{q}_{\text{Nud}}$	73.28 (+10.9)	50.00 (−10.3)	22.10 (+2.40)
			$\widetilde{q}_{\text{CMP}}$	71.96 (+9.53)	59.76 (−0.61)	21.20 (+2.40)
Qwen2.5-Coder-3B (Instruct)	Qwen2.5-3B (Base)	N/A	q	74.60	73.17	24.60
		Qwen3-4B (Base)	$\widetilde{q}_{\text{Nud}}$	82.27 (+7.67)	75.00 (+1.83)	23.50 (−1.10)
			$\widetilde{q}_{\text{CMP}}$	78.57 (+3.97)	78.65 (+5.48)	24.60 (±0.00)
		Qwen2.5-7B (Base)	$\widetilde{q}_{\text{Nud}}$	78.04 (+3.44)	49.39 (−23.78)	22.60 (−2.00)
			$\widetilde{q}_{\text{CMP}}$	79.62 (+5.02)	75.00 (+1.83)	24.40 (−0.20)

Table 4: Comparison between Nudging (Fei et al., 2025) and our method. The method q refers to the expert model itself, $\widetilde{q}_{\text{Nud}}$ refers to the method called Nudging proposed in Fei et al. (2025), and $\widetilde{q}_{\text{CMP}}$ refers to ours defined by (4). The colored numbers in parentheses indicate the difference from the results of the expert model q.

threshold ε	0.0	0.001	0.01	0.1	0.2
Avg μ^*	0.0020	0.5723	0.7026	0.8208	0.8572
ACC (%)	74.34	76.46	78.31	79.63	78.57

Table 5: Results on MBPP. (q: Qwen2.5-Coder-3B, p: Qwen2.5-3B, $\tilde{p}$: Qwen2.5-7B)

threshold ε	0.0	0.001	0.01	0.1	0.2
Avg μ^*	0.0005	0.5685	0.7012	0.8231	0.8604
ACC (%)	73.78	75.61	74.39	75.00	70.12

Table 6: Results on HumanEval. (q: Qwen2.5-Coder-3B, p: Qwen2.5-3B, $\tilde{p}$: Qwen2.5-7B)

provement, which implies a promising direction for future work.

E.4 Ablation Study

Varying the truncation factor ε. Here we observe how the truncation factor ε affects the embedding ratios μ_x^* and the resulting accuracy on benchmarks. Tables 5 and 6 are the results with ε from 0.0 to 0.2. These results show that even relatively small thresholds (from 0.001 to 0.01) may yield sufficiently large embedding ratios μ_x^* compared to the no truncation setting, i.e., $\varepsilon = 0.0$, and the increased μ^* overall leads to the improved accuracy.