POISE: Instance-Specific Prompt Tuning under Latent Mixture Target Distributions

Wenhao Zheng[1†], **Shuai Xu**[1†], **Lixing Chen**[2], **Xiu Su**[3],
Shichao Kan[1*], **Xiaoting Lyu**[4], **and Zhe Qu**[1*]
[1]School of Computer Science and Engineering, Central South University
[2]School of Computer Science, Shanghai Jiao Tong University
[3]School of Artificial Intelligence, Central South University
[4]School of Software Engineering, Xi'an Jiaotong University
{254711108, 254703052, xiusu1994, kanshichao, zhe_qu}@csu.edu.cn
lxchen@sjtu.edu.cn, xiaoting.lyu@xjtu.edu.cn

Abstract

Prompt tuning (PT) provides a parameter-efficient way to adapt frozen large language models, but its behavior becomes less clear when task contains multiple related yet distinguishable patterns. We formalize this setting as a latent mixture target distribution, where the target behavior is modeled as a finite mixture of latent task components. We first show that static prompting can retain a non-vanishing approximation gap when one shared prompt is insufficient to capture the target distribution. Then we formalize instance-specific prompting as an input-dependent prompt mapping and show that, under ideal prompt assignment, its achievable regret can approach zero. Guided by this analysis, we propose POISE (Prompt Optimization via Instance-Specific Experts), a structured and parameter-efficient approximation to instance-specific prompting. POISE represents the prompt space with learnable expert prompts, combines them through input-dependent routing, and introduces a shared low-rank residual correction to capture common structure. Extensive experiments across multiple benchmarks show that POISE achieves strong performance while maintaining high parameter efficiency.

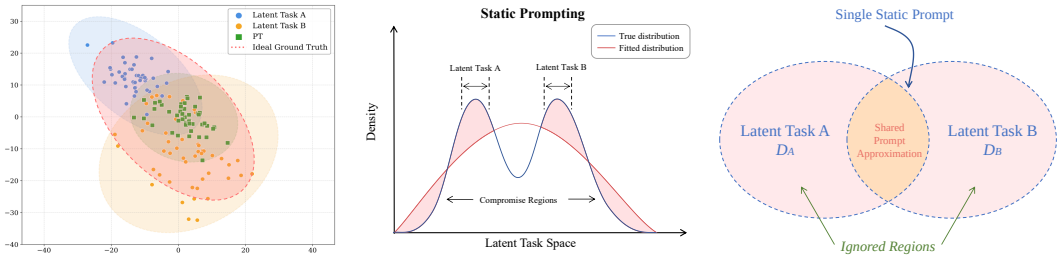
1 Introduction

Large Language Models (LLMs) have achieved remarkable progress across NLP and related domains Brown et al. [2020], Touvron et al. [2023]. However, full fine-tuning for downstream tasks remains computationally expensive and storage-intensive. Parameter-Efficient Fine-Tuning (PEFT) has thus emerged as a practical paradigm, adapting large pre-trained models with few trainable parameters Houlsby et al. [2019], Hu et al. [2022]. Among PEFT methods, Prompt Tuning (PT) is particularly attractive for its simplicity and efficiency. It prepends trainable continuous vectors to the input while keeping the backbone model frozen Lester et al. [2021], Liu et al. [2022]. By optimizing only these soft prompts, PT significantly reduces cost and supports task-level deployment.

Despite its strong empirical performance, PT's behavior under task distributions with multiple related but distinguishable patterns remains poorly understood. Existing studies have shown that PT works well when the target behavior is relatively homogeneous or closely aligned with the model's pre-training distribution He et al. [2021], Gu et al. [2022]. In practice, however, many downstream tasks are not governed by a single uniform input-output pattern. Instead, different instance groups may rely on different semantic cues, reasoning patterns, or label-generation mechanisms, while still sharing

† Co-first authors: Wenhao Zheng and Shuai Xu.
∗ Corresponding authors: Shichao Kan and Zhe Qu.

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



(a) PCA of prompt embeddings. (b) Distribution. (c) Domain Coverage.

Figure 1: Empirical and conceptual motivation for static prompt compromise under latent mixture target distributions. (a) The static prompt lies between regions of different latent task components, indicating a compromise representation. (b) A single global approximation can smooth out component-specific modes. (c) A shared representation mainly covers the common region of D_A and D_B, leaving component-specific regions insufficiently captured.

common task structure. We describe this setting as a *latent mixture target distribution*, where the overall distribution is formed by a mixture of latent task components. This raises a central question: ***can a single static prompt effectively approximate such a latent mixture target distribution?***

To investigate this issue, we conduct an observational study on BoolQ Clark et al. [2019]. In Figure 1(a), different latent task components occupy distinguishable regions in the prompt embedding space, while the static prompt lies in an intermediate region, indicating a compromise representation rather than alignment with any component-specific semantic region. The ideal ground-truth region is closer to these component-specific regions, revealing a visible gap between the static prompt and the desired target structure. Although the ground-truth region may extend beyond expert boundaries, this is expected in high-dimensional semantic spaces, where fitting multiple related components leaves unavoidable gaps between learned embeddings and the underlying target structure Yu et al. [2020], Aghajanyan et al. [2021]. Figures 1(b) and 1(c) provide conceptual explanations: static prompting fits a single global approximation over multiple latent components, smoothing out component-specific modes, and a shared representation captures the common region while leaving component-specific regions insufficiently covered. These observations reveal a key limitation: a shared prompt can model common structure but struggles to capture distinct behaviors required by different latent components.

Motivated by these observations, We intend to examine this limitation from a theoretical perspective. We first revisit static prompting, which uses one globally shared prompt for all inputs. Our analysis shows that if the latent mixture distribution lies outside the static prompt-conditioned family, static prompting retains a non-vanishing approximation gap. Recent studies have begun to exploit instance-level information for prompt construction or selection Zeng et al. [2025], Liu et al. [2025], Li et al. [2025a]. Motivated by this trend, We then study instance-specific prompting, where a learned mapping ϕ generates prompts from the input. Our analysis shows that under ideal conditions where suitable prompts can be assigned to different latent components, instance-specific prompting can drive achievable regret close to zero. This suggests that input-dependent prompt mappings provide a principled way to approximate latent mixture target distributions.

However, this ideal formulation assumes access to a flexible input-dependent prompt mapping $\phi(x)$, which requires generating a full prompt matrix for each input. Directly learning such an unrestricted mapping is difficult to optimize and would weaken the parameter-efficiency advantage of PT.

To bridge this gap between the theoretical formulation and practical prompt tuning, we propose **POISE** (Prompt Optimization via Instance-Specific Experts), a structured and parameter-efficient approximation to $\phi(x)$. Instead of generating an unconstrained prompt independently for each input, POISE represents the prompt space with a compact set of learnable expert prompts and uses input-dependent routing to combine them according to the observed instance. The expert prompts provide reusable prompt patterns for different latent task components, while the router adapts their combination across inputs. To further capture shared structure not fully covered by the expert combination, POISE introduces a shared low-rank residual correction ΔR, which improves expressiveness while maintaining parameter efficiency. We conduct extensive experiments on a range of benchmarks. The results show that POISE achieves strong performance across different settings.

2 Preliminaries

Prompt tuning (PT) Lester et al. [2021] adapts a frozen pre-trained language model by optimizing a small set of continuous prompt parameters Li and Liang [2021]. Let π_θ denote a pre-trained language model with frozen parameters θ. Given an input x, PT prepends a trainable continuous prompt matrix $s \in \mathbb{R}^{L\times D}$ to the input embeddings, where L is the prompt length and D is the embedding dimension. The frozen model then defines a prompt-conditioned distribution $\pi_\theta(y|x, s)$ over the output y. In this formulation, s is the only trainable variable, while the backbone parameters θ remain fixed. Given a target task distribution μ, the standard PT loss is the expected negative log-likelihood:

$$\mathcal{L}_{\mathrm{PT}}(s) = \mathbb{E}_{(x,y)\sim\mu}\left[-\log \pi_\theta(y|x, s)\right], \tag{1}$$

$\mathcal{L}_{\mathrm{PT}}$ is loss function optimized by s and $s^* = \arg\min_s \mathcal{L}_{\mathrm{PT}}(s)$ denotes the learned optimal prompt.

Real-world tasks are often not governed by a single uniform input-output pattern. Even within the same overall task, different groups of instances may rely on different semantic cues, reasoning patterns, or label-generation mechanisms Swayamdipta et al. [2020]. For example, one subset of inputs may be governed by lexical matching, while another may require compositional reasoning or domain-specific knowledge. These groups are not necessarily independent tasks; they may share common structure, but they can still exhibit distinguishable conditional behaviors. To capture this structure, we model the target distribution as a mixture of latent task components.

We introduce a latent task variable τ to denote the underlying latent task component associated with an instance. Each component τ_k induces its own atomic conditional distribution over the output space. This provides a simple way to represent population-level heterogeneity while still allowing different components to share common task structure.

Definition 2.1 (Latent Mixture Target Distribution) *Let $\{\tau_k\}_{k=1}^K$ denote K latent task components, where each τ_k induces an atomic conditional distribution $P(\cdot|x, \tau_k)$. Let $\{w_k\}_{k=1}^K$ be mixture weights satisfying $w_k \geq 0$ and $\sum_{k=1}^K w_k = 1$, where w_k denotes the population-level proportion of the k-th component. We define the latent mixture target distribution $\xi^{\mathrm{tar}}(\cdot|x)$ as*

$$\xi^{\mathrm{tar}}(\cdot|x) = \sum_{k=1}^{K} w_k P(\cdot|x, \tau_k).$$

This finite-mixture formulation serves as the latent mixture target distribution in our theoretical analysis, more detail can see in Appendix A.1. Together with the PT formulation in (1), it allows us to view PT as an approximation problem, i.e., the prompt-conditioned distribution $\pi_\theta(\cdot|x, s)$ is optimized to match the latent mixture target distribution $\xi^{\mathrm{tar}}(\cdot|x)$.

3 From Static Prompting to Instance-Specific Prompting

3.1 Static Prompting under Latent Mixture Targets

Static prompting refers to the standard PT setting introduced in (1), where a single learnable prompt is optimized and then shared across all input instances. In other words, the conditioning signal provided by the prompt is fixed after training and does not vary with the input. We formalize this static prompting paradigm as follows.

Definition 3.1 (Static Prompting) *Consider a frozen pre-trained language model π_θ. Static prompt tuning introduces a globally shared continuous prompt matrix $s \in \mathbb{R}^{L\times D}$. For any input instance x, it induces the following static prompt-conditioned distribution:*

$$\pi_\theta(y_{1:N} \mid x, s) = \prod_{n=1}^{N} \pi_\theta(y_n \mid x, y_{<n}, s).$$

After defining the static prompt-conditioned distribution $\pi_\theta(\cdot \mid x, s)$, we now evaluate how well it can approximate the latent mixture target distribution. Let $\mathcal{D}$ denote the distribution of input instances. For a static prompt s, we define its expected sequence-level regret with respect to $\xi^{\mathrm{tar}}(\cdot \mid x)$ as

$$\mathcal{R}^{\mathrm{stat}}(s) := \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\xi^{\mathrm{tar}}(\cdot|x)}\left[-\log \pi_\theta(y \mid x, s)\right] - \mathbb{E}_{y\sim\xi^{\mathrm{tar}}(\cdot|x)}\left[-\log \xi^{\mathrm{tar}}(y \mid x)\right]\right]. \tag{2}$$

The first term measures the cross-entropy between $\xi^{\mathrm{tar}}(\cdot|x)$ and $\pi_\theta(\cdot|x,s)$. For each fixed input x, (2) compares the negative log-likelihood under $\pi_\theta(\cdot|x,s)$ with the intrinsic negative log-likelihood of $\xi^{\mathrm{tar}}(\cdot|x)$ itself. Since the latter is independent of s, their difference measures the distributional mismatch between $\xi^{\mathrm{tar}}(\cdot|x)$ and $\pi_\theta(\cdot|x,s)$. This gives the expected KL form:

$$\mathcal{R}^{\mathrm{stat}}(s) = \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x)\|\pi_\theta(\cdot \mid x,s)\right)\right]. \tag{3}$$

We next analyze the regret of static prompting under the latent mixture target distribution. The latent mixture structure alone does not necessarily imply that static prompting must fail, since the frozen model can still condition on the input x. However, when different latent task components require distinct conditioning patterns, a single globally shared prompt may be insufficient to fully capture $\xi^{\mathrm{tar}}(\cdot \mid x)$. Our analysis focuses on this challenging case, where $\xi^{\mathrm{tar}}(\cdot \mid x)$ cannot be exactly represented by any static prompt-conditioned distribution $\pi_\theta(\cdot \mid x,s)$. When $\xi^{\mathrm{tar}}(\cdot \mid x)$ is not realizable by the static prompt family, the static regret remains strictly positive, i.e., $\inf_{s\in\mathcal{S}} \mathcal{R}^{\mathrm{stat}}(s) > 0$. In other words, optimizing a single shared prompt cannot eliminate the approximation gap. The proof are provided in Appendix A.2.

This non-vanishing gap reflects the remaining mismatch between $\xi^{\mathrm{tar}}(\cdot \mid x)$ and the best attainable static prompt-conditioned distribution. Intuitively, the gap becomes larger when different latent task components require incompatible prompt-conditioned behaviors, and smaller when they share more common structure.

3.2 Instance-Specific Prompting

The preceding analysis shows that a single shared prompt can leave a non-vanishing approximation gap when the latent mixture target distribution is not realizable by the static prompt family. Recent PT methods Zeng et al. [2025], Liu et al. [2025], Li et al. [2025a] have explored using instance-level information to construct or select prompts, so that the conditioning signal can vary across inputs. We use instance-specific prompting to denote this input-dependent paradigm, in which each instance is associated with its own prompt through a learned mapping. To analyze this setting, we define instance-specific prompting as follows.

Definition 3.2 (Instance-Specific Prompting) *Consider a pre-trained language model π_θ and an input instance x. Instance-specific prompting introduces a hypothesis space Φ of instance-specific prompt mappings. Each mapping $\phi \in \Phi$ maps an input x to a prompt matrix $\phi(x) \in \mathbb{R}^{L\times D}$. Conditioned on $\phi(x)$, the model induces the following instance-specific prompt-conditioned distribution:*

$$\pi_\theta(y_{1:N} \mid x, \phi(x)) = \prod_{n=1}^{N} \pi_\theta(y_n \mid x, y_{<n}, \phi(x)).$$

To evaluate ϕ, we define its expected sequence-level regret with respect to the latent mixture target distribution $\xi^{\mathrm{tar}}(\cdot \mid x)$:

$$\mathcal{R}^{\mathrm{ins}}(\phi) = \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x)\|\pi_\theta(\cdot \mid x,\phi(x))\right)\right], \tag{4}$$

where $\mathcal{D}$ denotes the distribution of input instances.

(4) provides a criterion for evaluating ϕ. Compared with static prompting, the key difference is that ϕ can produce different prompts for different inputs, so the achievable regret depends on how well ϕ adapts to the latent mixture structure of $\xi^{\mathrm{tar}}(\cdot \mid x)$. We therefore study the minimum regret attainable within Φ, which leads to the following theorem.

Theorem 3.1 (Upper Bound for Instance-Specific Prompting) *Let $\xi^{\mathrm{tar}}(\cdot \mid x)$ be the latent mixture target distribution in Definition 2.1, and let Φ be the hypothesis space of instance-specific prompt mappings. For each latent task component τ_k, let $\mathcal{D}_k$ denote its input distribution, and let ϵ_k be the best prompt-conditioned approximation error attainable for this component:*

$$\epsilon_k := \inf_s \mathbb{E}_{x\sim\mathcal{D}_k}\left[D_{\mathrm{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s)\right)\right].$$

Let $\epsilon_{\mathrm{route}}$ denote the expected additional penalty caused by assigning an input to a prompt that is mismatched with its latent task component. If Φ contains a mapping ϕ that selects component-specific prompts up to this routing penalty, then the minimum achievable instance-specific regret satisfies

$$\inf_{\phi\in\Phi} \mathcal{R}^{\mathrm{ins}}(\phi) \le \sum_{k=1}^{K} w_k\epsilon_k + \epsilon_{\mathrm{route}}. \tag{5}$$

Theorem 3.1 shows that the regret of instance-specific prompting is controlled by the component-wise approximation errors ϵ_k and the routing penalty ϵ_{route}. If suitable prompts can approximate the latent task components, then $\epsilon_k \to 0$ for all k; if ϕ reliably assigns instances to suitable prompts, then $\epsilon_{\text{route}} \to 0$. Consequently, the upper bound in (5) approaches zero, indicating that instance-specific prompting can approximate latent mixture target distributions under reliable prompt assignment. The detailed of Theorem 3.1 are provided in Appendix A.3.

Overall, our analysis identifies both the limitation of static prompting and the potential of instance-specific prompting. When the target distribution lies outside the static prompt family, a single shared prompt will leave a non-vanishing approximation gap. In contrast, instance-specific prompting can achieve vanishing regret in the ideal case where suitable prompts are available for latent task components and inputs can be reliably assigned to them. However, directly learning an unrestricted mapping $\phi(x)$ that generates a full prompt matrix for every input is difficult in practice. This motivates a structured approximation to $\phi(x)$: we represent the prompt space with a compact set of learnable prompt experts and use the input instance to determine their combination. The next section instantiates this idea as POISE.

4 POISE

Following the theoretical motivation above, we propose **POISE** (Prompt Optimization via Instance-Specific Experts), a structured approximation to the instance-specific prompt mapping $\phi(x)$. Rather than generating an unrestricted full prompt matrix for each input, POISE constructs $\phi(x)$ from a compact set of learnable expert prompts and determines their combination according to the observed instance. This design preserves the input-dependent flexibility of instance-specific prompting while keeping the prompt construction parameter-efficient and stable to optimize.

Specifically, POISE consists of three components: **expert prompt basis**, **input-dependent expert routing**, and **low-rank residual correction**. The expert prompts provide reusable prompt patterns for different latent task components, the router adapts their combination for each input, and the low-rank residual captures common structure not fully covered by the expert combination. Together, these components instantiate instance-specific prompting in a tractable form.

Expert Prompt Basis. To construct a compact prompt basis, we obtain a set of expert prompts through a lightweight initialization phase. We optimize expert prompts to capture distinct modes of the mixture latent task distribution for a limited number of steps while keeping the backbone model frozen. This process yields E expert prompts, denoted as

$$\mathcal{S}_{\text{exp}} = \{s^{(e)}\}_{e=1}^{E}, \qquad s^{(e)} \in \mathbb{R}^{L \times D},$$

where E is the number of experts, L is the prompt length, and D is the embedding dimension.

Each expert captures a distinct pattern relevant to latent task components, collectively forming a basis for instance-specific prompts. Detailed procedures for this initialization are provided in Appendix B.2. Since these experts are only briefly tuned, this stage introduces minimal computational overhead, while the primary adaptation is deferred to the dynamic routing mechanism.

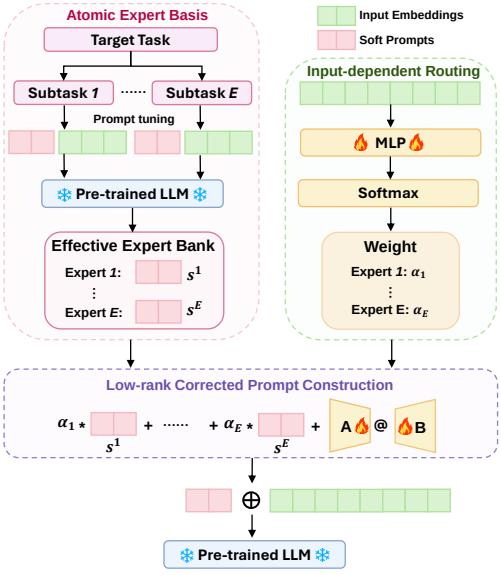


Figure 2: Overview of POISE.

Input-dependent Expert Routing. Given the expert prompt basis, POISE needs to determine how to combine expert prompts for each input. Since different inputs correspond to different latent task, a fixed expert combination is insufficient. We therefore introduce a lightweight router that produces input-dependent expert weights. For an input sequence x with embedding representations

$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T \times D}$, we first apply mean pooling to obtain an instance-level representation:

$$h(x) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t \in \mathbb{R}^D.$$

The routing weights are computed by a bottleneck projection followed by softmax normalization:

$$\alpha(x) = \mathrm{Softmax}(W_{\mathrm{up}} \, \mathrm{ReLU}(W_{\mathrm{down}} h(x) + b_{\mathrm{down}}) + b_{\mathrm{up}}) \,,$$

where $W_{\mathrm{down}} \in \mathbb{R}^{d_r \times D}$, $W_{\mathrm{up}} \in \mathbb{R}^{E \times d_r}$, and $d_r \ll D$ is the router bottleneck dimension. The output $\alpha(x) \in \mathbb{R}^E$ satisfies $\alpha_e(x) \in [0, 1]$ and $\sum_{e=1}^{E} \alpha_e(x) = 1$.

This router provides the input-dependent assignment required by instance-specific prompting. Instead of selecting one fixed prompt for all inputs, POISE softly combines expert prompts according to $\alpha(x)$, allowing different inputs to activate different prompt conditions.

Low-rank Residual Correction. Although expert routing provides input-dependent adaptation, a convex combination of expert prompts may still miss shared residual structure across the latent mixture target distribution. To address this, POISE adds a globally shared low-rank correction

$$\Delta R = AB,$$

where $A \in \mathbb{R}^{L \times r}$, $B \in \mathbb{R}^{r \times D}$, and $r \ll D$. This residual term refines the expert-based prompt approximation while keeping the number of additional parameters small. It captures common structure shared across inputs and complements the input-dependent expert combination.

Final Prompt Construction. Combining the expert prompt basis, input-dependent routing, and low-rank residual correction, POISE defines the final instance-specific prompt for input x as

$$\phi(x) = \sum_{e=1}^{E} \alpha_e(x) s^{(e)} + \Delta R. \tag{6}$$

The resulting prompt $\phi(x) \in \mathbb{R}^{L \times D}$ is prepended to the input embeddings and fed into the frozen language model. This construction gives a structured approximation to the instance-specific prompt mapping studied in Section 3.2. The expert prompts provide candidate prompt patterns for different latent task components, the router adapts these patterns to each input, and the low-rank residual captures shared structure. In this way, POISE implements instance-specific prompting without directly generating a full unconstrained prompt matrix for every instance.

5 Experiments

5.1 Implementation Details.

For the preparation of the expert prompts, we avoid computationally heavy pre-training to maintain parameter efficiency. To obtain consistent task-specific semantic clusters, we utilize the Llama-3.1-8B-Instruct model to generate pseudo-labels for expert initialization. The foundational experts are subsequently optimized for only 100 to 500 steps depending on the dataset size. For our POISE method, we set the prompt length of 10, 20, 40, 60, 100 and the low-rank bottleneck dimension r of 10, 20, 30, 40. During the fine-tuning phase on the SuperGLUE Wang et al. [2019],we follow the setup from prior papers(Shi and Lipani [2024]). We determine the optimal hyperparameters through a grid search: the routing learning rate is searched within 5e-4, 1e-3, 5e-3, and the low-rank learning rate is searched within 1e-3, 5e-3, 1e-2.

For the MBPP Austin et al. [2021], we using the learning rates of 1e-3 for the PT, 2e-4 for the structurally decoupled and offset-based prompt tuning methods and the routing learning rate of 2e-4 and the low-rank learning rate of 1e-4 for POISE. Our model is trained on NVIDIA L40S-48GB GPUs. More detailed implementation of our experiments see Appendix B and Appendix C.

5.2 Main Results

Performance on SuperGLUE. As shown in Table 1, POISE achieves 77.80% on T5-base using 0.06% trainable parameters, comparing favorably to static baselines like PT (72.13%) and DePT (75.32%).

Table 1: The best performance except full fine-tuning is in **bold**, and the second best is shown in underlined."*"and† indicate the results reported from Liu et al. [2025] or its corresponding paper, respectively. All scores are averaged over 3 runs to avoid the influence of the random seeds.

Method	# Para	BoolQ Acc	CB Acc	COPA Acc	MRC F1a	RTE Acc	WiC Acc	Average Score
T5-Base (220M)								
Fine-Tuning† Aribandi et al. [2021]	100%	82.30	91.30	60.00	79.70	84.50	69.30	77.85
Prompt-Tuning* Lester et al. [2021]	0.06%	78.12	84.42	54.37	78.30	75.27	62.29	72.13
P-Tuning v2* Liu et al. [2022]	0.53%	80.81	90.23	61.28	79.83	**81.98**	67.56	76.94
XPrompt* Ma et al. [2022]	0.04%	79.67	86.72	56.95	78.57	78.29	64.31	74.09
ResPrompt* Razdaibiedina et al. [2023]	0.21%	79.25	85.33	58.64	78.42	77.14	62.36	73.52
SMoP† Choi et al. [2023]	8e-3%	79.40	86.42	58.30	79.60	77.50	65.20	74.40
SuperPos-Prompt† SadraeiJavaeri et al. [2024]	-	74.00	80.20	62.00	72.90	70.40	67.60	71.18
VFPT* Zeng et al. [2024]	0.21%	78.38	90.92	61.76	78.73	76.90	65.36	75.34
LoPA* Jain et al. [2024]	1.07%	77.37	86.38	68.00	76.61	73.52	64.26	74.36
DePT Shi and Lipani [2024]	0.05%	79.30	92.90	58.00	74.30	78.73	68.70	75.32
EPT* Lan et al. [2025]	0.06%	79.14	90.18	56.33	73.43	78.99	67.71	74.30
CaPT* Liu et al. [2025]	4e-3%	79.54	**94.16**	64.33	**80.46**	79.78	66.77	77.51
POISE(Ours)	0.06%	**82.62**	90.46	**65.00**	77.36	81.15	**70.22**	**77.80**
T5-Large (770M)								
Fine-Tuning Aribandi et al. [2021]	100%	85.75	95.26	76.00	84.41	88.05	72.11	83.60
Prompt-Tuning* Lester et al. [2021]	0.04%	83.20	90.32	57.50	83.10	86.11	68.74	78.16
P-Tuning v2* Liu et al. [2022]	0.52%	**85.82**	95.56	77.00	84.07	89.25	71.03	83.79
XPrompt* Ma et al. [2022]	0.02%	83.82	91.39	82.05	81.26	87.72	**73.51**	83.29
ResPrompt* Razdaibiedina et al. [2023]	0.15%	83.51	90.64	**82.79**	84.02	86.97	71.13	83.18
SMoP* Choi et al. [2023]	3e-3%	83.45	92.37	71.00	83.92	87.70	68.60	81.17
VFPT* Zeng et al. [2024]	0.18%	83.89	93.71	75.63	83.24	88.10	71.00	82.56
LoPA* Jain et al. [2024]	0.57%	81.86	92.15	73.00	82.82	83.47	69.28	80.43
EPT* Lan et al. [2025]	0.04%	84.77	93.40	54.00	80.03	86.33	71.79	78.39
CaPT* Liu et al. [2025]	3e-3%	84.56	**97.22**	80.00	**84.43**	88.45	69.44	84.03
POISE(Ours)	0.04%	84.89	**99.27**	79.00	83.34	**91.30**	70.53	**84.72**

Table 2: Performance comparison on MBPP benchmark. We report average *pass@1* scores on Llama3-8B-Instruct.

Model	Method	#Para	Acc
Llama3-8B-Instruct	PT	0.41M	59.92
	DePT	0.60M	63.32
	ADePT	0.60M	63.81
	POISE (Ours)	0.71M	**64.98**

Static prompt often face challenges adapting to latent mixture distributions. Furthermore, while dynamic methods like CaPT provide instance-awareness, they typically lack explicit decomposition of shared syntax and task-specific semantics. POISE alleviates these issues: dynamic routing directs instances to appropriate experts for semantic disambiguation, while the shared low-rank correction captures task-agnostic structures, easing optimization conflicts.

This design scales robustly to T5-large, where POISE obtains 84.72% with 0.04% parameters. This trend suggests that as model capacity increases, the approximation penalty of fitting mixture distributions with static prompts becomes more pronounced. By delegating semantic tasks to foundational experts and maintaining a global syntactic framework via low-rank parameters, POISE offers a balanced and effective approach to complex multi-task environments. We further investigate the robustness and computational efficiency of POISE through additional experiments. Specifically, we evaluate the stability of POISE across five random seeds under controlled training configurations and analyze its end-to-end training cost, including expert initialization and pseudo-label generation. Detailed results are provided in Appendix D.

Performance on MBPP. To evaluate generalization on decoder-only architectures and complex generative tasks, we assess POISE on the MBPP using Llama-3.1-8B-Instruct. Code generation necessitates a precise alignment between rigid, task-agnostic programming syntax and diverse, task-specific algorithmic logic. As detailed in Table 2, POISE achieves a pass@1 score of 64.98%, surpassing vanilla PT (59.92%) and decoupled baselines such as DePT (63.32%) and ADEPT (63.81%). This outcome validates our architectural design: instance-aware routing dynamically adapts

Table 3: Performance comparison across different backbone models and benchmarks. The best results are in bold, and the second-best results are underlined.

Model	Benchmark	Method	Acc.
Llama3-8B-Instruct	**GSM8K**	LoPA	66.64
		CaPT	<u>67.48</u>
		POISE (Ours)	**68.30**
Mistral-7B	**GSM8K**	PT	47.27
		LoPA	<u>51.63</u>
		CaPT	51.02
		POISE (Ours)	**52.72**
Mistral-7B	**MMLU-Pro**	PT	24.18
		LoPA	<u>29.37</u>
		CaPT	26.78
		POISE (Ours)	**31.33**

to diverse algorithmic logic within the semantic subspace, while the global low-rank perturbation ΔR preserves strict syntactic integrity.

Generalization across Models and Benchmarks. To further investigate the generalizability of POISE beyond conventional language understanding and code generation tasks, we extend our evaluation to GSM8K and MMLU-Pro using two decoder-only backbones, Llama3-8B-Instruct and Mistral-7B. These benchmarks introduce distinct challenges for prompt adaptation. GSM8K focuses on mathematical reasoning, where solving problems requires interpreting diverse problem structures and performing multi-step numerical reasoning. In contrast, MMLU-Pro evaluates knowledge-intensive reasoning across a broad range of disciplines, requiring models to integrate domain-specific knowledge with complex reasoning processes. Such diversity in semantic contexts and reasoning requirements provides a challenging setting for evaluating whether a prompt tuning method can adapt effectively to heterogeneous input patterns.

As shown in Table 3, POISE consistently outperforms competing methods across all evaluated model–benchmark combinations. On GSM8K, POISE achieves 68.30% accuracy with Llama3-8B, surpassing CaPT by 0.82 percentage points. When applied to Mistral-7B, POISE achieves 52.72% on GSM8K and 31.33% on MMLU-Pro, exceeding the strongest baselines by 1.09 and 1.96 percentage points, respectively. Notably, the consistent improvements across mathematical and knowledge-intensive reasoning tasks suggest that POISE is not restricted to a particular task structure or backbone model. These findings are consistent with our motivation that input-dependent expert composition can provide more flexible adaptation to diverse instance-level requirements, while the shared low-rank residual offers a complementary mechanism for capturing common prompt structure.

Visualization Analysis. Figure 3 illustrates the PCA distribution of prompt embeddings. While static PT tends to cluster in a compromise region between different latent task components, POISE prompts spread across component-specific subspaces. In particular, the embeddings partially extend beyond the immediate regions covered by individual latent tasks. This observation suggests that the fusion of experts in high-dimensional space naturally encounters a structural gap between discrete task domains. Although shared low-rank correction ΔR is introduced to bridge this discrepancy, a residual gap persists.

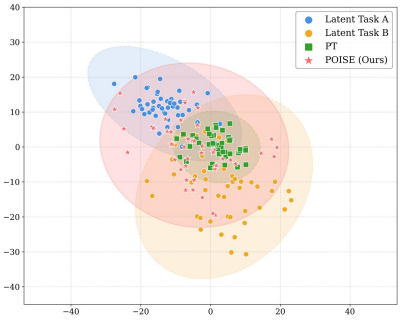
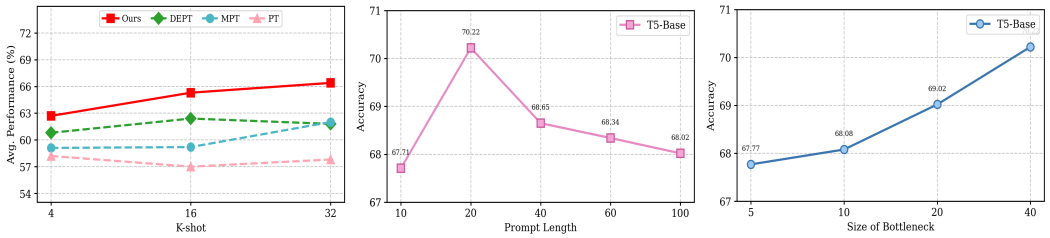


Figure 3: PCA of prompt embeddings.

5.3 Ablation Experiments

Effect of Architectural Components. Table 4 ablates the core components of our method: the global low-rank matrix (ΔR), the expert prompt, and the dynamic Expert Routing. Integrating static experts into a ΔR-only baseline improves performance via semantic initialization. Conversely, discarding ΔR while retaining dynamic routing degrades adaptation, confirming ΔR's necessity for capturing



(a) K-Shot (b) Prompt Length (c) Bottleneck Size

Figure 4: **More ablation experimentis.** (a)The performance changes of POISE(Ours), DEPT, MPT, and PT at different K-shot. (b)The impact of prompt length. (c)The impact of bottleneck size.

Table 4: Ablation study on the components of our proposed method. We compare our complete method with a baseline utilizing only the low-rank matrix.

ΔR	Expert Prompt	Expert Routing	**WiC**	**MRC**	**RTE**
✓	✗	✗	66.27	73.93	77.68
✓	✓	✗	67.72	75.33	78.71
✗	✓	✓	69.09	76.41	80.05
✓	✓	✓	**70.22**	**77.36**	**81.15**

task-agnostic structures. The full POISE framework yields the lowest approximation error across all datasets. This indicates that while static combinations force a compromise on latent mixture inputs, instance-specific routing effectively manages localized semantic assignments.

Power of Model Scale. Figure 5 illustrates the performance of POISE, DePT, and PT on larger model scales (T5-3B and T5-11B). POISE achieves the highest average scores across both scales, surpassing the PT baseline by 2.6% and 1.6% respectively, demonstrating its robust scalability. Notably, as the backbone parameter count increases to the 11B scale, all evaluated prompting methods perform strongly, rendering the performance differences among them less distinct. This phenomenon aligns with previous findings(Lester et al. [2021]).

Few-shot adaptation. We evaluate POISE against representative baselines in k-shot ($k \in \{4, 16, 32\}$) settings. As shown in the top of Figure 4a, POISE maintains a consistent advantage across all low-resource configurations. This performance gain stems from the structural priors embedded within our architecture. Specifically, the pre-initialized experts serve as semantic anchors that regularize the optimization trajectory. In data-scarce regimes, where static prompts are highly susceptible to overfitting, POISE's dynamic routing confines instance representations within well-defined expert sub-spaces.

Sensitivity to Prompt Length. Fig 4b illustrates the impact of the soft prompt length on the WiC dataset. The performance peaks at a prompt length of 20 and gradually declines as the length increases to 100. We hypothesize that overly long continuous prompts might exacerbate optimization difficulties and potentially dilute the semantic focus of the dynamic expert combination.

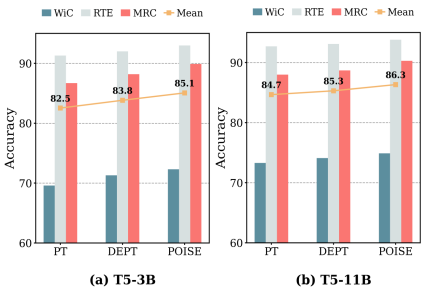


Figure 5: The performance changes.

Impact of the Low-Rank Bottleneck. We further analyze the impact of the low-rank bottleneck in Fig 4c. As the bottleneck size increases from 5 to 40, the performance on WiC steadily improves from 67.77% to 70.22%. This empirical evidence demonstrates the synergistic effectiveness of our design: the dynamic fusion mechanism optimally leverages both the shared foundation and the specialized experts to achieve superior overall performance on latent mixture tasks.

6 Related Works

Prompt Tuning and Theoretical Foundations. PEFT has become a widely used approach for adapting LLMs Li et al. [2025b]. Among these methods, continuous PT is important due to its simplicity and efficiency, as it introduces a small number of trainable vectors while keeping the backbone model frozen Lester et al. [2021], Liu et al. [2022], Vu et al. [2022]. Despite its strong empirical performance, the underlying mechanisms and limitations are still not fully understood. Recent studies have investigated its fitness landscapes and operational boundaries, which reveal that soft prompts can behave unpredictably and may struggle to acquire new skills Petrov et al. [2023]. From a theoretical perspective, viewing PT through the lens of memory-based meta-learning Ortega et al. [2019] provides further insight. In particular, Genewein et al. [2025] models pre-trained networks as Bayesian sequential predictors. It shows that while static prompting works well for unimodal target distributions, it can suffer from posterior collapse and unavoidable approximation errors when dealing with more complex, multi-task settings Yang et al. [2019].

Dynamic Prompting and Decoupled Architectures. To address the representation limits of static global prompts, a growing body of work has explored more flexible prompting strategies, including dynamic, instance-dependent, and decomposed prompting structures. For example, DePT Shi and Lipani [2024] decouples prompts into different components to isolate task-specific knowledge and ADePT Tang et al. [2025] as an variant which produces adaptive input token embedding offsets via shallow token-shared networks, addressing the limitations of DePT. At the same time, inspired by Mixture-of-Experts, several approaches use routing networks to dynamically compose soft prompts based on input contexts, such as ATTEMPT Asai et al. [2022], IDPG Wu et al. [2022], SMoP Choi et al. [2023], MEPT Zeng et al. [2025], and PT-MoE Li et al. [2025a]. These methods improve the model's representational flexibility and help reduce negative transfer across diverse task modes. In addition, in the broad fields, MOE has also demonstrated strong capabilities Wang et al. [2022], Liu et al. [2021], Wu et al. [2024], Zhao et al. [2024], Zhang et al. [2025]. However, a key challenge remains: how to balance task-specific semantic distinctions with shared, task-agnostic structures, especially in latent mixture task settings. Our proposed POISE is an improved variant of dynamic prompting,which an better resolve latent mixture target distribution.

7 Conclusion

In this work, we study prompt tuning under latent mixture target distributions, where the target behavior consists of multiple related yet distinguishable latent task components. We show that static PT can leave a non-vanishing approximation gap when a single shared prompt is insufficient to represent the target distribution. We further analyze instance-specific prompting and show that its regret can vanish when suitable prompts can be reliably assigned to different inputs. Motivated by this analysis, we propose POISE, a structured instance-specific prompt tuning method that combines input-dependent routing over component-specific prompts with a shared low-rank correction. This design enables the model to adapt prompts across instances while maintaining parameter efficiency. Extensive experiments demonstrate that POISE consistently improves over static and structurally decoupled prompt tuning baselines, providing an effective and theoretically motivated approach for prompt adaptation under latent mixture target distributions.

8 Acknowledgments

This work was supported in part by the National Science Foundation of China 62302525 and 62673325, the China Postdoctoral Science Foundation 2025M781515.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328, 2021.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. Ext5: Towards extreme multi-task scaling for transfer learning. *arXiv preprint arXiv:2111.10952*, 2021.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 6655–6672, 2022.

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.

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

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936, 2019.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124, 2019.

Tim Genewein, Li Kevin Wenliang, Jordi Grau-Moya, Anian Ruoss, Laurent Orseau, and Marcus Hutter. Understanding prompt tuning and in-context learning via meta-learning. *arXiv preprint arXiv:2505.17010*, 2025.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

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

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 8410–8423, 2022.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

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

Abhinav Jain, Swarat Chaudhuri, Thomas Reps, and Chris Jermaine. Prompt tuning strikes back: Customizing foundation models with low-rank prompt adaptation. *Advances in Neural Information Processing Systems*, 37:47297–47316, 2024.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, 2018.

Pengxiang Lan, Enneng Yang, Yuting Liu, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. Ept: efficient prompt tuning by multi-space projection and prompt fusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24366–24374, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, 2021.

Zongqian Li, Yixuan Su, and Nigel Collier. Pt-moe: An efficient finetuning framework for integrating mixture-of-experts into prompt tuning. *arXiv preprint arXiv:2505.09519*, 2025a.

Zongqian Li, Yixuan Su, and Nigel Collier. A survey on prompt tuning. *arXiv preprint arXiv:2507.06085*, 2025b.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

Yiyang Liu, James C Liang, Heng Fan, Wenhao Yang, Yiming Cui, Xiaotian Han, Lifu Huang, Dongfang Liu, Qifan Wang, and Cheng Han. All you need is one: Capsule prompt tuning with a single vector. *arXiv preprint arXiv:2510.16670*, 2025.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. Xprompt: Exploring the extreme of prompt tuning. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 11033–11047, 2022.

Pedro A Ortega, Jane X Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

Aleksandar Petrov, Philip HS Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. *arXiv preprint arXiv:2310.19698*, 2023.

Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Anastasiia Razdaibiedina, Yuning Mao, Madian Khabsa, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi. Residual prompt tuning: Improving prompt tuning with residual reparameterization. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6740–6757, 2023.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95, 2011.

MohammadAli SadraeiJavaeri, Ehsaneddin Asgari, Alice Carolyn McHardy, and Hamid Reza Rabiee. Superpos-prompt: Enhancing soft prompt tuning of language models with superposition of multi token embeddings. *arXiv preprint arXiv:2406.05279*, 2024.

Zhengxiang Shi and Aldo Lipani. DePT: Decomposed prompt tuning for parameter-efficient fine-tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KjegfPGRde.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, 2020.

Pengwei Tang, Xiaolin Hu, and Yong Liu. Adept: Adaptive decomposed prompt tuning for parameter-efficient fine-tuning. *arXiv preprint arXiv:2501.03291*, 2025.

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.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, 2022.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, 2022.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*, 2023.

Haoyuan Wu, Haisheng Zheng, Zhuolun He, and Bei Yu. Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 737–749, 2024.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vinod Vydiswaran, and Hao Ma. Idpg: An instance-dependent prompt generation method. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5507–5521, 2022.

Zhiyong Yang, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. Generalized block-diagonal structure pursuit: Learning soft latent task assignment against negative transfer. *Advances in Neural Information Processing Systems*, 32, 2019.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

Runjia Zeng, Cheng Han, Qifan Wang, Chunshu Wu, Tong Geng, Lifu Huang, Ying N Wu, and Dongfang Liu. Visual fourier prompt tuning. *Advances in Neural Information Processing Systems*, 37:5552–5585, 2024.

Runjia Zeng, Guangyan Sun, Qifan Wang, Tong Geng, Sohail Dianat, Xiaotian Han, Raghuveer Rao, Xueling Zhang, Cheng Han, Lifu Huang, et al. Mept: Mixture of expert prompt tuning as a manifold mapper. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16271–16291, 2025.

Liang Zhang, Ziyao Lu, Fandong Meng, Hui Li, Jie Zhou, and Jinsong Su. Advancing smoe for continuous domain adaptation of mllms: Adaptive router and domain-specific loss. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26584–26602, 2025.

Wangbo Zhao, Jiasheng Tang, Yizeng Han, Yibing Song, Kai Wang, Gao Huang, Fan Wang, and Yang You. Dynamic tuning towards parameter and inference efficiency for vit adaptation. *Advances in Neural Information Processing Systems*, 37:114765–114796, 2024.

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: Driven by our theoretical characterization of the "non-vanishing approximation gap," we propose POISE, a structured and parameter-efficient framework that facilitates instance-specific prompt adaptation via expert routing and low-rank residuals. The main contributions of our work (i.e., formal theoretical analysis, the POISE architecture, and empirical efficiency under heterogeneity) are claimed in both the abstract and introduction accurately.

   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: We discuss the limitation in D in Appendix.

   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: [Yes]

Justification: We have stated our theoretical assumptions and results in sections 2 and 3. A complete proof is provided in Appendix A.

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: We claim reproducibility in both section 5.1 and Appendix B. Our code will be publicly available after acceptance.

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: We claim reproducibility in both section 5.1 and Appendix B. All the datasets included in our study are publicly available (SuperGLUE and MBPP). Our code will be publicly available after acceptance. The publicly available code should be adequate to replicate the primary 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: We specify all the training and test details in section 5.1 and Appendix B.

   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: [Yes]

   Justification: The paper provides relevant information (Section Results).

   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 paper provides relevant information (Section Appendix).

   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 paper follows code of ethics.

   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: [N/A]

    Justification: No societal impact.

    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: No such risks.

    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: The paper follows the instructions.

    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: [N/A]

    Justification: We do not release new assets.

    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: Our paper does not involve crowdsourcing nor research with human subjects.

    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: Our paper does not involve crowdsourcing nor research with human subjects.

    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: Our paper does not use LLMs as any components of the core methods.

    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.

A More Theoretical Details

A.1 Derivation of the Latent Mixture Target Distribution in Definition 2.1

We derive the finite-mixture form of the latent mixture target distribution in Definition 2.1. As discussed in Section 2, real-world tasks may contain multiple related but distinguishable latent task components. To model such population-level heterogeneity, we introduce a latent task variable τ, whose prior distribution is a discrete mixture over K latent task components:

$$P_{\mathrm{tar}}(\tau) = \sum_{k=1}^{K} w_k \delta(\tau - \tau_k), \qquad w_k \geq 0, \quad \sum_{k=1}^{K} w_k = 1, \tag{7}$$

where τ_k denotes the k-th latent task component, w_k denotes its population-level proportion, and $\delta(\cdot)$ denotes the Dirac delta distribution.

For each latent task component τ_k, let $P(\cdot \mid x, \tau_k)$ denote its latent conditional distribution over the output space. The latent mixture target distribution is obtained by marginalizing the conditional distribution over the discrete latent task prior:

$$\begin{aligned}
\xi^{\mathrm{tar}}(\cdot \mid x) &= \int_{\mathcal{T}} P(\cdot \mid x, \tau) P_{\mathrm{tar}}(\tau)\, d\tau \\
&= \int_{\mathcal{T}} P(\cdot \mid x, \tau) \sum_{k=1}^{K} w_k \delta(\tau - \tau_k)\, d\tau \\
&= \sum_{k=1}^{K} w_k \int_{\mathcal{T}} P(\cdot \mid x, \tau) \delta(\tau - \tau_k)\, d\tau \\
&= \sum_{k=1}^{K} w_k P(\cdot \mid x, \tau_k).
\end{aligned} \tag{8}$$

The last equality follows from the sifting property of the Dirac delta distribution:

$$\int f(\tau)\delta(\tau - \tau_k)\, d\tau = f(\tau_k).$$

Therefore, we obtain the latent mixture target distribution in Definition 2.1:

$$\xi^{\mathrm{tar}}(\cdot \mid x) = \sum_{k=1}^{K} w_k P(\cdot \mid x, \tau_k).$$

Equivalently, for any concrete output y, the same conditional distribution can be written as

$$\xi^{\mathrm{tar}}(y \mid x) = \sum_{k=1}^{K} w_k P(y \mid x, \tau_k). \tag{9}$$

This derivation shows that the mixture weights w_k in Definition 2.1 correspond to the population-level proportions of different latent task components. Thus, the latent mixture target distribution $\xi^{\mathrm{tar}}(\cdot \mid x)$ is a unified conditional distribution formed by aggregating multiple related but distinguishable latent conditional distributions.

A.2 Proof of Approximation Gap

By Eq. (3), the expected sequence-level regret of a static prompt s is

$$\mathcal{R}^{\mathrm{stat}}(s) = \mathbb{E}_{x \sim \mathcal{D}} \left[D_{\mathrm{KL}} \left(\xi^{\mathrm{tar}}(\cdot \mid x) \| \pi_\theta(\cdot \mid x, s) \right) \right], \tag{10}$$

where $\mathcal{D}$ denotes the distribution of input instances.

We first formalize the non-realizability condition. We assume that the latent mixture target distribution $\xi^{\mathrm{tar}}(\cdot \mid x)$ cannot be exactly represented by any static prompt-conditioned distribution $\pi_\theta(\cdot \mid x, s)$. That is, for every $s \in \mathcal{S}$,

$$\pi_\theta(\cdot \mid x, s) \neq \xi^{\mathrm{tar}}(\cdot \mid x) \tag{11}$$

On a set of positive $\mathcal{D}$-measure. Equivalently, there does not exist a static prompt $s \in \mathcal{S}$ such that

$$\pi_\theta(\cdot \mid x, s) = \xi^{\mathrm{tar}}(\cdot \mid x)$$

We next show that the optimal static regret is attained. Assume that the static prompt space $\mathcal{S} \subset \mathbb{R}^{L \times D}$ is bounded and closed, and that $\mathcal{R}^{\mathrm{stat}}(s)$ is continuous on $\mathcal{S}$. Since $\mathcal{S}$ is compact, the Extreme Value Theorem guarantees the existence of an optimal static prompt $s^* \in \mathcal{S}$ such that

$$\mathcal{R}^{\mathrm{stat}}(s^*) = \min_{s \in \mathcal{S}} \mathcal{R}^{\mathrm{stat}}(s) = \inf_{s \in \mathcal{S}} \mathcal{R}^{\mathrm{stat}}(s). \tag{12}$$

It remains to show that this optimal regret is strictly positive. For every input x, Gibbs' inequality gives

$$D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x) \| \pi_\theta(\cdot \mid x, s^*)\right) \geq 0, \tag{13}$$

with equality if and only if

$$\pi_\theta(\cdot \mid x, s^*) = \xi^{\mathrm{tar}}(\cdot \mid x). \tag{14}$$

Suppose, for contradiction, that static prompting can achieve vanishing regret, i.e.,

$$\inf_{s \in \mathcal{S}} \mathcal{R}^{\mathrm{stat}}(s) = 0. \tag{15}$$

By Eq. (12), this implies

$$\mathcal{R}^{\mathrm{stat}}(s^*) = 0. \tag{16}$$

Therefore,

$$\mathbb{E}_{x \sim \mathcal{D}}\left[D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x) \| \pi_\theta(\cdot \mid x, s^*)\right)\right] = 0. \tag{17}$$

Since the KL divergence is non-negative for every x, the expectation can be zero only if

$$D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x) \| \pi_\theta(\cdot \mid x, s^*)\right) = 0 \quad \mathcal{D}\text{-a.e.} \tag{18}$$

Thus,

$$\pi_\theta(\cdot \mid x, s^*) = \xi^{\mathrm{tar}}(\cdot \mid x) \quad \mathcal{D}\text{-a.e.} \tag{19}$$

This means that the latent mixture target distribution is exactly represented by the static prompt-conditioned distribution induced by s^*, except on a set of $\mathcal{D}$-measure zero.

This contradicts the non-realizability condition above, which states that no static prompt $s \in \mathcal{S}$ can exactly represent $\xi^{\mathrm{tar}}(\cdot \mid x)$ almost everywhere. Therefore,

$$\inf_{s \in \mathcal{S}} \mathcal{R}^{\mathrm{stat}}(s) > 0. \tag{20}$$

A.3 Proof of Theorem 3.1

We prove the upper bound for instance-specific prompting by constructing an idealized instance-specific prompt mapping and bounding its regret.

Recall from Eq. (4) that the expected sequence-level regret of an instance-specific prompt mapping $\phi \in \Phi$ is

$$\mathcal{R}^{\mathrm{ins}}(\phi) = \mathbb{E}_{x \sim \mathcal{D}}\left[D_{\mathrm{KL}}\left(\xi^{\mathrm{tar}}(\cdot \mid x) \| \pi_\theta(\cdot \mid x, \phi(x))\right)\right], \tag{21}$$

where $\mathcal{D}$ denotes the distribution of input instances.

To derive a component-wise upper bound, we make explicit the latent-component structure of the latent mixture target distribution. We consider the process where a latent component index k is first sampled from $\mathrm{Cat}(w_1, \ldots, w_K)$. Conditioned on component τ_k, the input is sampled from its input distribution $x \sim \mathcal{D}_k$, and the output distribution is given by the latent conditional distribution $P(\cdot \mid x, \tau_k)$. Under this construction, the distribution of input instances can be written as the mixture

$$\mathcal{D} = \sum_{k=1}^{K} w_k \mathcal{D}_k. \tag{22}$$

By the convexity of KL divergence in its first argument, the mixture-level regret can be upper-bounded by the following component-wise regret:

$$\mathcal{R}^{\mathrm{ins}}(\phi) \leq \overline{\mathcal{R}}^{\mathrm{ins}}(\phi) := \sum_{k=1}^{K} w_k \mathbb{E}_{x \sim \mathcal{D}_k}\left[D_{\mathrm{KL}}\left(P(\cdot \mid x, \tau_k) \| \pi_\theta(\cdot \mid x, \phi(x))\right)\right]. \tag{23}$$

Therefore, it suffices to upper-bound $\overline{\mathcal{R}}^{\text{ins}}(\phi)$.

For each component τ_k, the component-wise approximation error is defined as

$$\epsilon_k := \inf_s \mathbb{E}_{x\sim\mathcal{D}_k}\left[D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s)\right)\right]. \tag{24}$$

For clarity, assume that this infimum is attained by a component-specific prompt s_k^*. If the infimum is not exactly attained, the same argument applies to an arbitrarily close prompt and yields the same bound up to a vanishing approximation term.

We now construct an idealized instance-specific prompt mapping. Let $\hat{k}(x)$ denote the component assignment predicted for input x. The constructed mapping outputs the corresponding component-specific prompt:

$$\phi^*(x) = s^*_{\hat{k}(x)}. \tag{25}$$

This mapping represents the ideal case where ϕ^* selects a prompt according to the latent component structure of the input.

Substituting ϕ^* into the component-wise regret gives

$$\overline{\mathcal{R}}^{\text{ins}}(\phi^*) = \sum_{k=1}^K w_k \mathbb{E}_{x\sim\mathcal{D}_k}\left[D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s^*_{\hat{k}(x)})\right)\right]. \tag{26}$$

For inputs from component τ_k, we decompose the expectation according to whether the assignment is correct:

$$\begin{aligned}\overline{\mathcal{R}}^{\text{ins}}(\phi^*) = \sum_{k=1}^K w_k \mathbb{E}_{x\sim\mathcal{D}_k}\Big[&\mathbb{I}\Big(\hat{k}(x)=k\Big) D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s_k^*)\right)\\ &+\mathbb{I}\Big(\hat{k}(x)\neq k\Big) D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s^*_{\hat{k}(x)})\right)\Big].\end{aligned} \tag{27}$$

For the correctly assigned part, since the indicator function is bounded by one, we have

$$\begin{aligned}&\mathbb{E}_{x\sim\mathcal{D}_k}\left[\mathbb{I}\Big(\hat{k}(x)=k\Big) D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s_k^*)\right)\right]\\ &\le \mathbb{E}_{x\sim\mathcal{D}_k}\left[D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s_k^*)\right)\right] = \epsilon_k.\end{aligned} \tag{28}$$

For the incorrectly assigned part, assume there exists a finite constant M such that the KL divergence induced by any mismatched prompt is bounded:

$$D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s^*_{\hat{k}(x)})\right) \le M. \tag{29}$$

Then

$$\begin{aligned}&\mathbb{E}_{x\sim\mathcal{D}_k}\left[\mathbb{I}\Big(\hat{k}(x)\neq k\Big) D_{\text{KL}}\left(P(\cdot \mid x,\tau_k)\|\pi_\theta(\cdot \mid x,s^*_{\hat{k}(x)})\right)\right]\\ &\le M \Pr_{x\sim\mathcal{D}_k}\Big(\hat{k}(x)\neq k\Big).\end{aligned} \tag{30}$$

Combining Eq. (27), Eq. (28), and Eq. (30), we obtain

$$\overline{\mathcal{R}}^{\text{ins}}(\phi^*) \le \sum_{k=1}^K w_k\epsilon_k + M\sum_{k=1}^K w_k \Pr_{x\sim\mathcal{D}_k}\Big(\hat{k}(x)\neq k\Big). \tag{31}$$

Define the expected routing penalty as

$$\epsilon_{\text{route}} := M\sum_{k=1}^K w_k \Pr_{x\sim\mathcal{D}_k}\Big(\hat{k}(x)\neq k\Big). \tag{32}$$

Thus,

$$\overline{\mathcal{R}}^{\text{ins}}(\phi^*) \le \sum_{k=1}^K w_k\epsilon_k + \epsilon_{\text{route}}. \tag{33}$$

If the hypothesis space Φ contains the constructed instance-specific prompt mapping ϕ^*, or can approximate it arbitrarily well, then

$$\begin{aligned}\inf_{\phi\in\Phi} \mathcal{R}^{\text{ins}}(\phi) &\leq \mathcal{R}^{\text{ins}}(\phi^*) \\ &\leq \overline{\mathcal{R}}^{\text{ins}}(\phi^*) \\ &\leq \sum_{k=1}^{K} w_k \epsilon_k + \epsilon_{\text{route}}. \end{aligned} \tag{34}$$

This proves the upper bound in Eq. (5).

Finally, consider the ideal case where each component can be well approximated by its component-specific prompt, so that $\epsilon_k \to 0$ for all k. If the instance-specific prompt mapping can reliably assign inputs to suitable component-specific prompts, then

$$\Pr_{x\sim\mathcal{D}_k}\left(\hat{k}(x) \neq k\right) \to 0 \quad \text{for all } k,$$

which implies $\epsilon_{\text{route}} \to 0$. Since KL divergence is non-negative, Eq. (34) gives

$$\inf_{\phi\in\Phi} \mathcal{R}^{\text{ins}}(\phi) \to 0. \tag{35}$$

This completes the proof.

B More detailed implementation of our experiments.

We implement our experiments using PyTorch, HuggingFace Transformers, and the HuggingFace PEFT library. We evaluate our proposed POISE framework across three pre-trained language models: T5-base (220M), T5-large (770M), and Llama-3.1-8B-Instruct. To ensure complete reproducibility, we detail the specific implementation protocols and hyperparameter configurations below.

B.1 Detailed Experimental Setup and Baselines

We conduct extensive experiments to validate our proposed method. We consider two widely adopted benchmarks: (1) SuperGLUE benchmark (Wang et al. [2019]), which includes BoolQ (Clark et al. [2019]), CB (De Marneffe et al. [2019]), COPA (Roemmele et al. [2011]), MultiRC (Khashabi et al. [2018]), RTE (Giampiccolo et al. [2007]), and WiC (Pilehvar and Camacho-Collados [2019]); and (2) MBPP benchmark (Austin et al. [2021]), which is designed for code generation tasks. Following prior works (Shi and Lipani [2024];Tang et al. [2025]), we evaluate our method on the selected SuperGLUE datasets using the T5-base (220M) and T5-large (770M) models (Raffel et al. [2020]) to assess its performance on natural language understanding. To further demonstrate the versatility and effectiveness of our approach on generative tasks and modern decoder-only architectures, we evaluate it on the MBPP benchmark using the Llama-3.1-8B-Instruct model (Grattafiori et al. [2024]).

To evaluate our proposed method, we compare it against a comprehensive set of representative tuning approaches: (1) full fine-tuning (FT) (Aribandi et al. [2021]), which serves as the upper-bound reference by optimizing all model parameters; (2) vanilla Prompt-Tuning (PT) (Lester et al. [2021]), the foundational continuous prompting approach; (3) advanced prompt optimization and structural variants, including P-Tuning v2 (Liu et al. [2022]), XPrompt (Ma et al. [2022]), ResPrompt (Razdaibiedina et al. [2023]), SMoP (Choi et al. [2023]), MPT(Wang et al. [2023]), SuperPos-Prompt (SadraeiJavaeri et al. [2024]), VFPT (Zeng et al. [2024]), EPT (Lan et al. [2025]) and CaPT (Liu et al. [2025]); (4) structurally decoupled and offset-based prompt tuning methods, specifically DePT (Shi and Lipani [2024]).

B.2 Expert Initialization via LLM Pseudo-Labeling

As formalized in Section 4, the expert prompts serve as a foundational semantic basis $\mathcal{S} = \{s^{(e)}\}_{e=1}^{E}$. To ensure the precision and stability of this semantic partitioning, we utilize Llama-3.1-8B-Instruct as a zero-shot annotator to generate task-specific pseudo-labels. We select this relatively larger, instruction-tuned model primarily for its superior semantic abstraction and zero-shot categorization

capabilities compared to the smaller backbone models (e.g., T5-base). High-quality pseudo-labeling is critical for the "cold start" of expert prompts, as it ensures that the initialized experts are anchored to distinct and meaningful task manifolds. It is important to emphasize that this pseudo-labeling process is a one-time, offline pre-processing step. The labeler model is not part of the final inference architecture and does not contribute to the trainable parameter count or the inference latency of the POISE framework.

Following this labeling process, we partition the training dataset into E subsets. For each subset, we initialize a standalone continuous prompt $s^{(e)} \in \mathbb{R}^{L \times D}$ and optimize it using the standard vanilla Prompt Tuning (PT) paradigm. During this lightweight initialization phase, we search the prompt learning rate from $\{3\mathrm{e}{-1}, 4\mathrm{e}{-1}, 5\mathrm{e}{-1}\}$. Depending on the dataset size, experts are optimized for merely 100 to 500 steps with a batch size of 32. Once initialized, these foundational experts are frozen to strictly isolate the architectural contribution of the subsequent instance-aware routing.

Regarding the data processing and structural constraints across different benchmarks, we apply specific maximum sequence lengths and batch sizes to balance computational efficiency and distinct task requirements. For the majority of the SuperGLUE tasks (i.e., BoolQ, CB, COPA, RTE, and WiC), we set the maximum input sequence length to 256 and utilize a batch size of 32. Due to the inherently longer context required for reading comprehension, we increase the maximum sequence length to 512 and reduce the batch size to 16 exclusively for the SuperGLUE-MultiRC dataset. Similarly, for the generative tasks within the MBPP benchmark, we uniformly apply a maximum sequence length of 512 and a batch size of 16 to accommodate the extended context of algorithmic logic and natural language instructions.

B.3 Hyperparameter Optimization and Evaluation Protocol

During the main adaptation phase, the frozen expert prompts are dynamically routed and integrated with the learnable low-rank correction matrix. To determine the optimal configuration for the SuperGLUE benchmark, we perform a systematic grid search over the architectural and optimization hyperparameters. Specifically, the prompt length L is searched from the set $\{10, 20, 40, 60, 100\}$, and the low-rank bottleneck dimension r is searched from $\{10, 20, 30, 40\}$. For the optimization process, the learning rate of the routing network is tuned within $\{5\mathrm{e}{-4}, 1\mathrm{e}{-3}, 5\mathrm{e}{-3}\}$, while the learning rate for the low-rank correction matrix is evaluated within $\{1\mathrm{e}{-3}, 5\mathrm{e}{-3}, 1\mathrm{e}{-2}\}$.

To maintain a rigorous and consistent evaluation protocol, we apply a uniform training budget of 30,000 steps for all datasets across the SuperGLUE benchmark, regardless of their varying sample scales. Throughout the optimization process, we strictly execute an evaluation phase on the validation set every 1,000 steps, ultimately selecting the checkpoint that achieves the optimal performance metric. For the MBPP, we establish a fixed configuration to evaluate the generalizability of our framework without extensive dataset-specific tuning. The routing learning rate is fixed at $2\mathrm{e}{-4}$ and the low-rank learning rate at $1\mathrm{e}{-4}$. The model is optimized for a total of 10 epochs. It is noteworthy that our overall training time and storage overhead are comparable to PT, and no additional high costs have been introduced.

C Datasets

C.1 Original Datasets

Table 5: The structural and statistical profiles of the employed datasets are summarized below. We define Source Length and Target Length as the mean token lengths for the input premises and output predictions, respectively, derived exclusively from the training corpus. The partition sizes—representing the volume of instances utilized for training, validation, and evaluation—are designated as Train, Valid, and Test. The Type attribute specifies the cognitive paradigm (e.g., classification, reasoning) associated with each dataset.

C.2 Expert Datasets

As detailed in Appendix B.2, the training corpus for each downstream task is partitioned into $E = 2$ disjoint subsets based on the semantic pseudo-labels generated by the zero-shot LLM annotator. These subsets serve as the distinct task modes for initializing the expert prompts. The follow table

Table 5: Statistics of the datasets assessed in this study.

Dataset	Source Length	Target Length	Train	Valid	Test	Type
			SuperGLUE			
BoolQ	108.3	1.0	9,427	1,635	1,635	Question Answering
CB	64.6	1.0	250	28	28	NLI
COPA	15.2	1.0	400	100	500	Causal Reasoning
MultiRC	286.1	1.0	27,243	2,424	2,424	Question Answering
RTE	54.4	1.0	2,490	277	3,000	NLI
WiC	18.4	1.0	5,428	319	319	Word Sense Disambiguation
			Code Generation Benchmark			
MBPP	45.5	62.8	374	90	500	Code Generation

Table 6: Data statistics for the expert prompts initialization. The validation set size is kept strictly consistent with the test set size for each respective expert subset.

Dataset	Expert Semantic Domain	Train	Valid	Test
	SuperGLUE			
BoolQ	culture	2,017	526	526
	nature	2,017	526	526
CB	news	180	45	45
	dialogue	180	45	45
COPA	psy	159	40	40
	phy	159	40	40
MultiRC	multi	1,374	344	344
	single	1,374	344	344
RTE	rtenews	716	179	179
	rtewiki	716	179	179
WiC	daily	2,384	596	596
	formal	2,384	596	596
	Code Generation Benchmark			
MBPP	math	72	19	19
	string	72	19	19

summarizes the specific data distribution including the training, validation, and test splits for each semantic expert across all evaluated benchmarks.

D Additional Experimental Analyses

D.1 Robustness Across Random Seeds

Table 7 presents the performance comparison between POISE and CaPT across six SuperGLUE tasks over five random seeds, with results reported as mean $\pm$ standard deviation. POISE achieves an average score of 77.88 ± 0.54, surpassing CaPT (76.62 ± 0.51) by 1.26 percentage points. Notably, POISE outperforms CaPT on four of the six tasks, with particularly substantial improvements on BoolQ and WiC. POISE exhibits comparable variability across random seeds, indicating consistent performance without increased instability. These findings further support the robustness of POISE and its effectiveness in adapting to heterogeneous task instances.

Table 7: Performance comparison between POISE and CaPT on SuperGLUE using T5-base over five random seeds. The best mean results are highlighted in bold.

Method	BoolQ	CB	COPA	MRC	RTE	WiC	Average
				T5-Base (220M)			
CaPT	79.05 ± 1.37	$\mathbf{91.08} \pm 1.03$	64.50 ± 1.83	$\mathbf{79.33} \pm 0.91$	79.54 ± 0.96	66.21 ± 1.21	76.62 ± 0.51
POISE (Ours)	$\mathbf{82.25} \pm 0.50$	90.58 ± 1.36	$\mathbf{65.40} \pm 2.61$	77.54 ± 1.18	$\mathbf{81.19} \pm 1.71$	$\mathbf{70.31} \pm 1.26$	$\mathbf{77.88} \pm 0.54$

Table 8: Comparison of end-to-end training time on T5-large, averaged across CB and MultiRC. Training time is normalized to PT ($1\times$).

Model	Method	Training Time
T5-Large (770M)	PT	$1.00\times$
	LoPA	$1.12\times$
	CaPT	$1.10\times$
	POISE (Ours)	$1.29\times$

D.2 End-to-End Training Efficiency

Table 8 compares the end-to-end training time of different prompt tuning methods on T5-large, averaged across CB and MultiRC, which represent the smallest and largest training sets among the six SuperGLUE tasks. All methods follow the same training protocol with a maximum of 30,000 steps and early stopping. Notably, the reported training time of POISE includes the complete pipeline: pseudo-label generation, expert prompt initialization, and subsequent optimization. POISE requires $1.29\times$ the training time of PT, compared with $1.12\times$ for LoPA and $1.10\times$ for CaPT. Although POISE introduces moderate additional computational overhead, its overall training cost remains within a comparable range to existing methods. These results indicate that POISE achieves its performance improvements without incurring excessive end-to-end training costs.

E Limitations

Practically, our framework has two primary limitations. First, POISE relies on an external LLM to generate semantic pseudo-labels for expert initialization. Its performance is therefore contingent on the quality and granularity of the initial task partitioning, which may also inherit inherent biases from the external annotator. Second, the pre-training strategies for the expert prompts remain under-explored. The current initialization scheme may not fully exploit the representation potential of instance-specific prompts in capturing the complex structures of latent mixture task distributions.