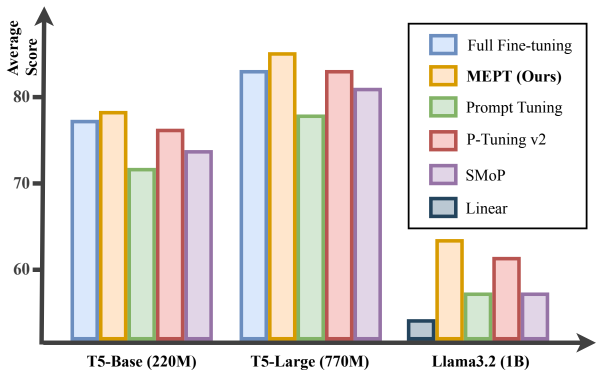
MEPT: Mixture of Expert Prompt Tuning as a Manifold Mapper

Runjia Zeng[1] **Guangyan Sun**[2] **Qifan Wang**[3] **Tong Geng**[2,4] **Sohail Dianat**[1]
Xiaotian Han[5] **Raghuveer Rao**[6] **Xueling Zhang**[1] **Cheng Han**[7] **Lifu Huang**[8] **Dongfang Liu**[1†]

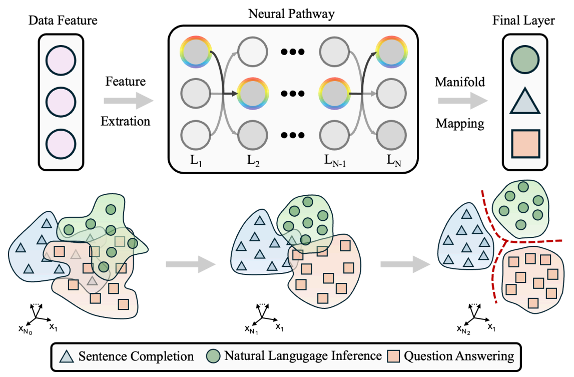
[1]Rochester Institute of Technology [2]University of Rochester [3]Meta AI [4]Rice University
[5]Case Western Reserve University [6]U.S. DEVCOM Army Research Laboratory
[7]University of Missouri - Kansas City [8]UC Davis †Corresponding author

Abstract

Considering deep neural networks as manifold mappers, the *pretrain-then-fine-tune* paradigm can be interpreted as a two-stage process: pretrain establishes a broad knowledge base, and fine-tune adjusts the model parameters to activate specific neural pathways to align with the target manifold. Although prior fine-tuning approaches demonstrate success, their rigid parameter space limits their ability to dynamically activate appropriate neural pathways, rendering them ill-equipped to adapt flexibly to the diverse and evolving data distributions. In light of this view, we propose a novel approach, Mixture of Expert Prompt Tuning **(MEPT)**, as an effective and efficient manifold-mapping framework. MEPT leverages the Mixture of Experts architecture by integrating multiple prompt experts to adaptively learn diverse and non-stationary data distributions. Empirical evaluations demonstrate that MEPT outperforms several state-of-the-art parameter efficient baselines on SuperGLUE, achieving notable improvements in mean accuracy (*e.g.*, 1.94%) while significantly reducing activated prompts by 79.25%. The effectiveness of MEPT is further supported by theoretical insights from manifold learning and validated through neural activation pathway visualization results. Our code is avaliable at runjia.tech/emnlp_mept.



(a) MEPT (ours) *vs.* Concurrent Arts



(b) Manifold Hypothesis

Figure 1: **MEPT (ours)** vs **concurrent arts under the manifold learning perspective.** (a) Average accuracy performance comparison with vanilla prompt tuning methods across varying size model structures. (b) Illustration of the manifold hypothesis, where different neural activation paths alter task manifold geometry (blue: 'Sentence completion,' green: 'Natural language inference,' orange: 'Question answering'), transforming non-linearly separable manifolds into separable ones in the final layer (separated with red dotted lines).

1 Introduction

Deep Neural Networks (DNNs) (Hinton et al., 2006) are designed to learn the intrinsic structure of data by mapping inseparable high-dimensional data distributions (*i.e.*, the left tangled feature representation in the Fig. 1 (b)) onto low-dimensional manifolds (*i.e.*, the right linearly separable manifolds in the Fig. 1 (b)). This procedure can be conceptualized as manifold mapping (Cohen et al., 2020; Zhu et al., 2018). Through this mapping, DNNs effectively match the data to its most appropriate manifold by eliciting a set of neural responses (*i.e.*, neural pathway highlighted by rainbow ring in the middle of Fig. 1 (b)) that align their learned representations with the intrinsic geometric structure of the data. This capacity to learn and represent these manifolds is central to the success of DNNs in tasks ranging from image recognition (Hinton et al., 2006; Cohen et al., 2020) to language modeling (Peters et al., 2018; Vaswani et al., 2017).

The prevailing training paradigm for large language models (LLMs) is under the *pretrain-then-finetune* (Dodge et al., 2020; Guo et al., 2020; He et al., 2022; Hu et al., 2022) paradigm, which can also be reinterpreted through the lens of manifold learning. In this perspective, pretrain serves in the role to establish a broad knowledge base by exposing the model to diverse data distributions, enabling it to capture a general representation of various manifolds. Finetune then modifies the model's parameters to activate unique neuron pathways that align with the target task's data manifold.

Traditional full fine-tuning is parameter-intensive and does not efficiently utilize the general knowledge acquired during pre-training. As LLMs continue to scale in size, researchers have introduced the *parameter-efficient fine-tuning* (PEFT) paradigm to address these limitations. Though highly successful, current PEFT techniques (*e.g.*, partial tuning (Chen et al., 2021; Jia et al., 2021; Mahajan et al., 2018), low-rank adaptation (LoRA) (Jie et al., 2023), prompt tuning (Ju et al., 2022; Dong et al., 2024; Yan et al., 2023; Zang et al., 2022)) are constrained by their reliance on fixed parameter spaces: They fall short in dynamically activating the appropriate neuron pathways necessary for effectively handling real-world data, which often entails complex, overlapping, or unknown manifolds. As a result, they are ill-equipped to flexibly adapt to the diverse and evolving data distributions in practice.

In light of this view, we propose **M**ixture of **E**xpert (MoE) **P**rompt **T**uning (**MEPT**). By leveraging the principles of MoE to activate specialized pathways within the LLMs, we are able to dynamically adapt the network's representational capacity with the requirements of the task-specific data manifolds. MEPT exhibits several compelling advantages: ❶ ***Generality.*** By employing a modular and adaptive MoE design, MEPT facilitates the handling of diverse and non-stationary data manifold mapping, significantly enhancing the model's ability to represent and process the complex structures inherent in real-world datasets (see §4.4 and §6); ❷ ***Scalability.*** The sparse MoE architecture specializes in the representation of each prompt expert, enabling a compact and efficient neural pathway activation, which imposes no additional burden when scaling up the number of experts during training and inference (see §3.2, §5 and Tab. 1); ❸ ***Interpretability.*** We find that the effectiveness of our approach is directly supported by theoretical insights from manifold learning and further validated through instinct neural activation pathway visualization results (see §6), demonstrating the capability for flexible and effective manifold mapping. In contrast, existing PEFT methods lack an intuitive manifold structure, thereby rendering their operation opaque and akin to a black box.

The remainder of the paper is structured as follows: we begin by theoretically re-examining the vanilla prompt tuning approach in §3.1, showing that it can be considered as a specialized MEPT version with only one expert. We also introduce the general notations for the mixture of experts. The formal introduction of MEPT is presented in §3.2. Comprehensive evaluation on the SuperGLUE benchmark under various LLMs in §4.4 showcases the superior performance of MEPT compared to state-of-the-art methods. Further ablation studies in §5 provide strong evidence for the effectiveness and efficiency of our proposed expert prompts. To explore the valuable interpretability offered by explicit manifold mapping, we also visualize the neural pathway activation and final manifold representation in §6. Our main contributions are summarized as the following:

- We unveil the essence of neural network learning through manifold visualization, cross-validating our motivation and effectiveness under manifold learning theory while elucidating the decision-making process.
- We further investigate the potential of mixture training using a universal prompt embedding, which imposes higher requirements for prompt generality. It highlights the importance of MoE, providing valuable insights for future PEFT architecture design.
- We conduct comprehensive experiments on various tasks and LLM sizes in the SuperGLUE benchmark, demonstrating the effectiveness of MEPT over several state-of-the-art prompt tuning and full fine-tuning methods.

2 Related works

2.1 Parameter-Efficient Fine-Tuning

Large Language Models (LLMs) push forward the state-of-the-art on natural language processing through their remarkable capabilities in various tasks (Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020). However, as the size of LLMs continue to grow, the computational cost of fine-tuning these models has increasingly become parameter-intensive, posing significant challenges for practical applications (Kaplan et al., 2020; Han et al., 2023). To address this problem, PEFT has emerged as a promising solution that significantly reduces computational and memory requirements while maintaining model performance (Houlsby et al., 2019; Hu et al., 2022; Lester et al., 2021; Pfeiffer et al., 2021). Three PEFT paradigms have gained particular prominence in recent years.

Adapter-based approaches introduce small, learnable neural networks (*i.e.*, adapters) between the layers of the pre-trained model (Houlsby et al., 2019; Pfeiffer et al., 2020b). Adapters typically consist of down-projection and up-projection layers, creating a bottleneck architecture that effectively captures task-specific information while keeping the original model parameters frozen (Houlsby et al., 2019; He et al., 2022; Pfeiffer et al., 2021; Karimi Mahabadi et al., 2021).

Low-Rank Adaptation (LoRA) (Hu et al., 2022) represents a reparameterization approach that decomposes weight updates into low-rank matrices. This method has sparked numerous innovations in dynamic rank selection and efficiency improvements (*e.g.*, DyLoRA (Valipour et al., 2023), AdaLoRA (Zhang et al., 2023), LoRA Dropout (Lin et al., 2024), LoRA+ (Hayou et al., 2024)).

Prompt Tuning (Lester et al., 2021; He et al., 2022) introduces learnable continuous prompts to the input sequence while keeping the pre-trained model frozen. Unlike discrete prompts that use natural language tokens, the continuous prompts are directly optimized during fine-tuning (Liu et al., 2021; Vu et al., 2021; Choi et al., 2023; Li et al., 2025). Different from prior arts that fix tunable parameter representations, MEPT activates different experts for each manifold mapping, enabling more dynamic adaptation.

2.2 Mixture of Experts

Mixture of Experts (MoE) (Jacobs et al., 1991; Eigen et al., 2014) represents a significant advancement in model architecture design. In MoE architectures, the model consists of multiple specialized sub-networks ($a.k.a.$ experts) that are selectively activated through a gating mechanism (Shazeer et al., 2017; Fedus et al., 2022). This design enables parameter scaling without proportional increases in computational cost, making it particularly valuable for large pre-trained models. Recent advancements have also extended MoE's applicability to three PEFT paradigms (Wang et al., 2022; Wu et al., 2024; Choi et al., 2023). The promising advancements of MoE inspire our design, aligning with our motivation to establish a dynamic and explainable fine-tuning pipeline.

3 Methodology

3.1 Preliminaries

Prompt Tuning. In traditional prompt tuning methods (Lester et al., 2021; Li and Liang, 2021), input prompts consist of a set of d-dimensional embedding vectors, where the dimensionality matches that of the text tokens. These prompts are inserted at the beginning of the input sequence in each Transformer encoder layer and interact with all the text tokens. They facilitate the learning of task-specific embeddings, effectively guiding the model's performance on new tasks.

Formally, these input prompts are defined as $P = \{P^1, P^2, \ldots, P^N\}$, where P^j denotes the learnable input prompts in the j_{th} Transformer encoder layer L_j, and N is the total number of layers. Then the outputs of encoder layers are represented as:

$$Z^j = L_j(P^j,\ Z^{j-1}) \quad j = 1, 2, 3, \ldots, N$$

where Z^j represents the contextual embeddings of the text tokens computed by the j_{th} encoder layer. The different colors indicate trainable and frozen parameters, respectively. Z^0 is the text token embeddings initialized from the backbone.

Mixture of Experts. In Transformer-based LLMs, each MoE layer generally consists of a set of K "expert networks" $\{f_1, \ldots, f_K\}$, alongside a "gating network" $\mathcal{G}$. Formally, the gating network $\mathcal{G}$, parameterized by $\mathbf{\Theta}$ and typically consisting of a linear-softmax network, yields the output $\mathcal{G}(\mathbf{x}; \mathbf{\Theta})$. Consequently, the output of the dense MoE layer can be formulated as:

$$\mathcal{F}(\mathbf{x}; \mathbf{\Theta}) = \textstyle\sum_{i=1}^{K} \mathcal{G}(\mathbf{x}; \mathbf{\Theta})_i f_i$$

$$\mathcal{G}(\mathbf{x}; \mathbf{\Theta})_i = \mathrm{softmax}(g(\mathbf{x}; \mathbf{\Theta}))_i = \frac{\exp(g(\mathbf{x}; \mathbf{\Theta})_i)}{\sum_{j=1}^{K} \exp(g(\mathbf{x}; \mathbf{\Theta})_j)}$$

where $g(\mathbf{x}; \mathbf{\Theta})$ represents the gating value prior to the softmax operation.

3.2 MEPT

In this paper, we present a novel prompt tuning approach, MEPT, for dynamic and general LLM adaptation (see Fig. 2(b)). The model employs a stacked MoE architecture for prompt embeddings, facilitating multi-layer representation learning to

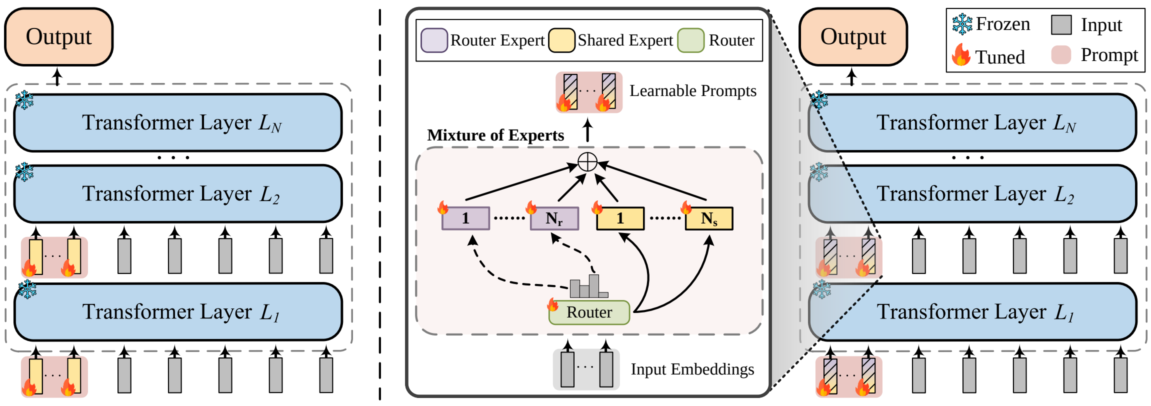


Figure 2: **Overview of PT $vs.$ MEPT (ours) frameworks.** (a) Vanilla Prompt Tuning. (c) The overall architecture of our proposed MEPT (see §3.2), including router expert, shared expert and router in each layer.

achieve universal adaptation. At each layer, prompt representations are constructed by two types of experts: *router experts* and *shared experts*.

Router Experts as Representation Specialists. While conventional prompts are effective in acquiring knowledge about the new task, they do not possess the capability to flexibly map each input to the most suitable lower-dimensional manifold. During fine-tuning on a new task with new data, the word distributions may significantly differ from each other. Consequently, it becomes imperative to enhance the model's capacity for activating different neural pathways for better manifold mapping. This entails enabling better attention among the mixture of expert prompt to flexibly learn new patterns that emerge in the task-specific context. Specifically, the j-th layer knowledge is selected from N_r router experts as:

$$\mathbf{RE^j} = Top(\mathcal{G}(\mathbf{h^{j-1}}; \mathbf{\Theta})_i) R_i^j \qquad j = 1, 2, \ldots, N$$

where $R^j \in \mathbb{R}^{N_r \times m \times d}$ denote the embeddings of the router expert prompts in the j-th transformer encoder/decoder layer, and N_r, m, and d represent the number of router experts, the length of the prompt, and the dimension of hidden state, respectively. 🟪🟨🟩 colors represent router experts, shared experts, and router, respectively. The router, denoted by $\mathcal{G}$, is implemented as a linear network followed by a softmax activation. It is responsible for directing the input h^0 (in the first layer) or the hidden state h^j ($j \in [1, N]$) from the preceding layer to the appropriate expert prompts (in subsequent layers). Finally, we sparsely activate only the i-th ($i \in [1, N_r]$) prompt expert with the highest routing probability through the Top($\cdot$).

These router experts are explicitly designed to specialize in capturing diverse aspects of input data and task-specific features. By dynamically routing inputs through different experts (*i.e.*, see the pathway visualizations in Fig. 4), the model effectively learns diverse patterns critical for task adaptation (*e.g.*, the superior performance validated in Tab. 1). Unlike traditional prompt tuning with fixed mappings, MEPT utilizes specialized prompts tailored to distinct subspaces of the input distribution.

Shared Experts as Knowledge Consolidators. Traditional MoE routing strategies often result in parameter redundancy, as multiple experts may encode overlapping common knowledge (Dai et al., 2024) (*i.e.*, vanilla prompt tuning typically requires prompt lengths of 10 to 100 to store the parameters). To address this, shared experts are introduced to centralize and integrate common knowledge across contexts (*i.e.*, MEPT only activates 10–20 prompts per layer). Formally, the shared expert prompt embeddings in j-th layer can be represented as:

$$\mathbf{SE^j} = S_1^j + \cdots + S_{N_s}^j$$

where $S^j \in \mathbb{R}^{N_s \times m \times d}$ denotes the shared expert prompts in the j-th transformer encoder layer, and N_s represent the number of shared experts. Its isolated, router-irrelevant design indicates consistent activation, facilitating the retention of common knowledge (see Appendix §F) while enhancing the specificity of expert knowledge learning for the router experts. We provide further discussion with DeepSeek (Dai et al., 2024) in Appendix G.3.

Training MEPT is to build a manifold mapper. During training, MEPT separately updates the router experts R^j, shared experts S^j and router $\mathcal{G}$ in each layer. This process enables MEPT to mine distinct data patterns through the router experts to enable a more precise manifold mapping while concurrently consolidating common knowledge into the shared experts, mitigating parameter redundancy that previously stored in the router prompt. Since the prompts in each layer share the same architectural design, the embeddings of the prompt in the j-th layer can be represented as:

$$P^j = \mathbf{RE^j} + \mathbf{SE^j}$$

Training with MEPT at each layer effectively constructs a manifold mapper, as this process progressively refines the representation space (*i.e.*, facilitate within-class convergence while keeping between-class separation, see detailed analysis in §6), causing it to "collapse" around manifold-relevant information (class-specific prototype hidden vector). This system has two merits:

- *Flexible Mapping:* MEPT exhibits an emergent capability to dynamically activate distinct expert prompts tailored to the input, thereby mapping them uniquely onto the corresponding manifolds (see analysis in §6), bringing consistently superior performance across various tasks, backbones and training schemes (see Tab. 1 and 6).

- *Efficient Representing*: Its intrinsically efficient parameter space leverages a leaner approach (*e.g.*, the parameter count for the prompt-based method is mh (see analysis in §3.2)), enabling more scalable growth compared with the traditional MoE (*i.e.*, the parameter count of the MLP component in a transformer is approximately $8h^2$ (Korthikanti et al., 2023), where h denotes the hidden layer dimension).

4 Experiments

4.1 Datasets

Following previous works on prompt tuning (Lester et al., 2021; Choi et al., 2023), we use six NLU tasks from the SuperGLUE benchmark to evaluate the performance of the language model (Raffel et al., 2020; Aribandi et al., 2022), including BoolQ (Clark et al., 2019), CB (Jiang and de Marneffe, 2019), COPA (Roemmele et al., 2011), MRC (Khashabi et al., 2018), RTE (Giampiccolo et al., 2007), WiC (Pilehvar and Camacho-Collados, 2019) and WSC (Levesque et al., 2012). More details are provided in the Appendix §A.

4.2 Baselines

Our model is compared with full fine-tuning, linear adaptation (*i.e.*, only tuning the final linear layer in Llama-3.2 1B) and multiple shallow and deep prompt tuning methods. More PEFT comparisons are provided in the Appendix §C.

4.3 Implementation Details

MEPT is implemented with the Huggingface PEFT library (Mangrulkar et al., 2022), which is a unified and extensible toolkit for PEFT research. Our model is trained on NVIDIA A100-40GB GPUs. We translate each SuperGLUE dataset into a text-to-text format following (Choi et al., 2023). Three scales pre-trained models are used: T5-Base, T5-Large and Llama3.2 with 200M, 770M and 1B parameters, respectively. Specifically, we train our prompts for 70 epochs using a batch size ranging from 16 to 32 (*i.e.*, to avoid out-of-memory issues on a single A100 GPU) under different learning rate settings. There are two hyperparameters in our model: the lengthes of prompt and number of router experts. For our method, we linearly search the best prompt length from {10, 15, 20} and number of router experts from {4, 10, 20}. Notably, we only utilize one single shared expert per layer to improve efficiency. To guarantee reproducibility, our full implementation shall be publicly released upon paper acceptance. See more details in §B.

4.4 Main Results

The main performance comparison results are shown in Tab. 1, leading to three key observations. **First,** MEPT not only outperforms all vanilla prompt tuning baselines but also surpasses full fine-tuning, achieving superior accuracy. On T5-Base, it improves Acc by 0.18% and 1.09% over full fine-tuning and prior best models, respectively. This highlights the effectiveness of the prompt expert in enhancing performance. In contrast, existing less effective prompt tuning methods focus solely on updating input prompt tokens, neglecting the importance of activating distinct neural pathways based on the target data manifold. **Second,** MEPT generalizes well across different model sizes and architectures while maintaining consistently superior performance. Even on the decoder-only Llama model, which is pretrained with a causal language modeling (CLM) objective, MEPT outperforms linear adaptation and prior best methods by 8.75% and 1.94%, respectively, highlighting its broad applicability. **Third,** MEPT benefits from an efficient and specialized prompt representation enabled by its sparse MoE routing design. Compared to traditional deep prompt tuning, P-Tuning v2, which also prepend prompts at each layer, MEPT reduces

Method	Para	Boolq Acc	CB F1/Acc	COPA Acc	MRC F1a	RTE Acc	WiC Acc	Average Score
T5-Base (220M)								
Fine-Tuning† (Aribandi et al., 2022)	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., 2022b)	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)	0.01%	79.40	86.42	58.30	79.60	77.50	65.20	74.40
DePT† (Shi and Lipani, 2024)	-	79.30	-	-	-	79.10	68.70	-
SuperPos-Prompt† (Ali Sadraei Javaheri 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
EPT (Lan et al., 2024)	0.06%	79.14	90.18	56.33	73.43	78.99	67.71	74.30
MEPT (Ours)	0.13%	**81.13**	**90.92**	**62.33**	**80.70**	**83.52**	69.60	**78.03**
T5-Large (770M)								
Fine-Tuning (Aribandi et al., 2022)	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., 2022b)	0.52%	85.82	95.56	77.00	84.07	89.25	71.03	83.79
XPrompt* (Ma et al., 2022)	0.02%	85.54	91.39	**85.05**	84.36	87.30	**73.22**	84.48
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)	0.04%	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
EPT (Lan et al., 2024)	0.04%	84.77	93.40	54.00	80.03	86.33	71.79	78.39
MEPT (Ours)	0.12%	**85.96**	**97.61**	79.67	**85.29**	**90.01**	73.09	**85.27**
Llama-3.2 (1B)								
Linear Head (Dubey et al., 2024)	3e-4%	59.85	51.69	56.33	48.94	55.23	53.45	54.25
Prompt-Tuning (Lester et al., 2021)	0.06%	60.95	61.61	57.67	57.73	55.96	54.70	56.94
P-Tuning v2 (Liu et al., 2022b)	0.53%	62.48	64.29	61.00	60.34	58.12	**60.15**	61.06
SMoP (Choi et al., 2023)	0.04%	61.13	62.50	59.33	57.46	57.40	54.23	57.51
VFPT (Zeng et al., 2024)	0.17%	62.44	61.72	59.67	58.41	**64.35**	57.60	60.70
EPT (Lan et al., 2024)	0.06%	61.56	65.22	56.00	60.18	63.90	59.45	61.05
MEPT (Ours)	0.11%	**64.56**	**67.86**	**63.33**	**62.22**	60.29	59.72	**63.00**

Table 1: Performance comparison result (%) on SuperGLUE development set. '*' and † indicates the results reported from (Wang et al., 2023) or their corresponding paper, respectively. 'Para' is the number of trainable parameters. The best results except fine-tuning are in bold with underline representing the second best ones. For tasks with two metrics, the average score is reported. All scores are averaged over 3 runs. Results are statistically significant with respect to all baselines on each PLM (all p-value < 0.005).

trainable parameters by up to 79.25% (*i.e.*, 0.53% *vs*. 0.11%). Another interesting finding is the performance gap between LLaMA 3.2 and T5-Base on SuperGLUE under prompt tuning, which may be attributed to differences in pretraining objectives and architecture (decoder-only *vs*. encoder-decoder, see detailed analysis in §B). This observation is consistent with (Hu et al., 2024a).

5 Ablation Study

Prompt Depth. We investigate the performance of MEPT based on the specific layers where prompt experts are deployed, as shown in Tab. 2. The results indicate that incorporating prompt experts at a single layer (*i.e.*, shallow) can also enhances accuracy. Furthermore, extending prompt experts to deeper layers (*i.e.*, default *vs*. deep) leads to the highest overall performance.

MEPT		CB	COPA	WiC	MRC
(Default)	1-12	**90.9**	**62.3**	**69.6**	**80.7**
(Deep)	1-6	89.8	62.0	67.0	80.1
	7-12	89.9	61.3	66.1	80.0
(Shallow)	1	86.4	58.3	65.2	79.6
	12	85.9	58.0	65.1	79.8

Table 2: Performance comparison with different prompt positions on CB, COPA, WiC and MRC for T5-Base.

Routing Method	CB	COPA	WiC	MRC
MEPT (Default)	**90.9**	**62.3**	**69.6**	**80.7**
- Stochastic	83.2	57.7	64.9	69.6
- Dense	85.7	59.0	66.0	79.7
- Gumbel-Softmax	88.3	61.3	65.1	80.0
- Perturbation	90.1	62.0	67.4	80.3
- w/o Shared Experts	89.3	60.7	67.0	80.1
- Replace Shared Experts	89.9	62.0	67.9	80.2

Table 3: Different router variants of MEPT in T5-Base.

Routing Methods. We conduct ablation studies on alternative routing methods in Tab. 3 to as-

Learning Space		CB	COPA	WiC	MRC
Router	Prompt				
✓	✓	**90.9**	**62.3**	**69.6**	**80.7**
	✓	89.1	61.3	66.3	79.4
✓		88.5	58.0	65.0	78.3

Table 4: Ablative study of router and expert prompt.

L \ N_r	4	6	10	14	20
10	83.0	82.0	81.8	82.1	82.7
15	**83.5**	83.0	82.1	82.0	82.0
20	81.0	81.8	81.6	81.8	82.3

Table 5: Sensitivity of the number of router experts (N_r) and prompt lengths (L) on SuperGlue RTE for T5-Base.

sess the impact of the router, as discussed in § 3.2. When applying stochastic operations within the router, a dramatic performance decline is observed. This degradation likely arises from the model's inability to consistently activate the most appropriate experts. Similarly, adding perturbations, enabling dense activation, or incorporating the Gumbel operation in the router also reduces effectiveness, as these modifications constrain the model's specialization. Lastly, we examine the influence of shared experts by directly removing them from the default version of MEPT and replacing them with the router experts (comparable parameters with the default), respectively. Notably, without shared experts, the limited prompt embeddings may struggle to capture sufficient information, leading to performance degradation. However, increasing parameter storage helps alleviate this issue. This highlights the critical role of shared experts in efficiently storing and leveraging "common knowledge."

Variants of MEPT. A fundamental difference between MEPT and other vanilla prompt tuning approaches lies in the integration the MoE into prompt embeddings (see §3.2). In our standard implementation, the router and prompt are both updated at each training step. To investigate the impact of this design, we analyze the contributions of these components individually in Tab. 4. As seen, the separate updating strategy significantly restricts the dynamic learning of manifold mappings. In contrast, simultaneously updating the router and prompt components exhibits a synergistic effect, leading to substantial performance improvements.

Prompt Length and Number of Experts. To investigate the optimal length and number of prompt experts, we present sensitivity analyses for MEPT across varying configurations, as detailed in Tab. 5. We have two observations: *First*, MEPT consistently demonstrates performance improvements in prompt tuning across diverse settings, although the optimal configuration of soft prompt length and number may vary depending on the specific task. *Second*, memory overhead during training and inference is solely dependent on the prompt length and irrelevant to the number of experts, as we only tune one set of router experts at a time due to the sparse MoE design, distinguishing us from vanilla prompt tuning methods. This provides a strong foundation for scaling up the number of router experts to tackle more complex and diverse contexts.

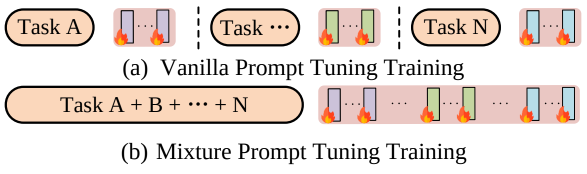

Task A
Task ···
Task N
(a) Vanilla Prompt Tuning Training
Task A + B + ··· + N
(b) Mixture Prompt Tuning Training

Figure 3: Overview of the two distinct training schemes employed in our study.

Method	Boolq	CB	COPA	MRC	RTE	WiC	Ave.
Mixture Training on T5-Base (220M)							
Prompt-Tuning	78.0	85.7	53.0	79.3	70.7	63.7	71.7
P-Tuning v2	78.8	85.7	**60.0**	79.5	70.0	65.0	73.2
SMoP	78.6	85.7	58.0	79.7	71.1	64.8	73.0
MEPT (Ours)	**80.1**	**91.0**	56.0	**80.2**	**79.7**	**67.8**	**75.8**

Table 6: Performance comparison under mixture training scheme for T5-Base. See more results in Tab. 11.

6 Analytic Findings

Results on Mixture Training Scenario. As shown in Fig. 3, the vanilla prompt tuning training scheme trains separate prompts (*i.e.*, different prompt colors represent distinct trained prompt embeddings) for each task, as distinct tasks require different prompt cues. In contrast, combining all subdatasets into a single dataset and training a unified prompt embedding for inference across all tasks imposes a higher demand on prompt generality (see Tab. 6).

Compared to separate training, mixture training exhibits a significant performance degradation (*e.g.*, 73.2 $vs.$ 76.9 average score for P-Tuning v2) due to the use of a single prompt across all tasks. Nevertheless, even under these constraints, MEPT consistently outperforms prior approaches through dynamic expert selection (see Fig. 4), yielding a greater performance improvement (+1.1 $vs.$ +2.6).

Manifold Mapping Pathway. To the best of our knowledge, research on the understanding of prompt tuning remains rare (Han et al., 2024; Zeng et al., 2024). Consequently, our research seeks to both quantitatively and qualitatively examine

Text-to-text templates and verbalizers of selected dataset

Category	Dataset	Text-to-text Template	Verbalizer
Question Answering	MultiRC	multirc question: **question** answer: **answer**. paragraph: **paragraph**	False, True
	BoolQ	boolq passage: **passage** question: **question**	False, True
Sentence Completion	COPA	copa choice1: **choice1** choice2: **choice2** premise: **premise** question: **question**	choice1, choice2

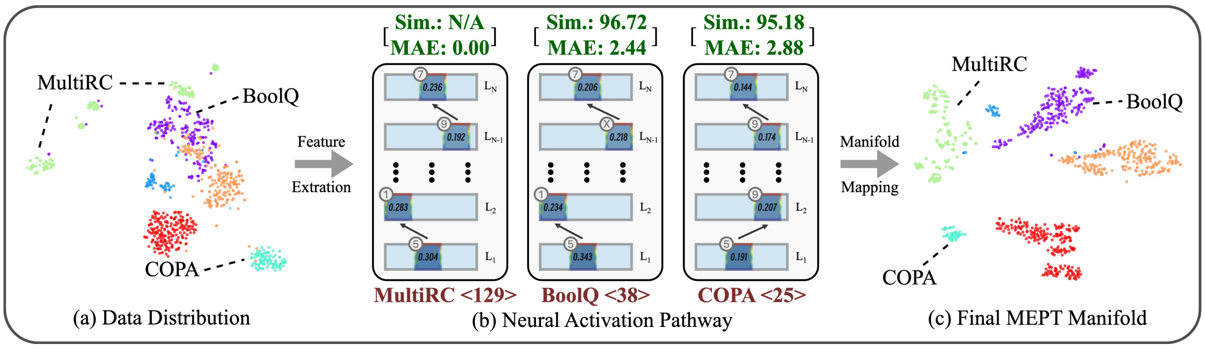


Figure 4: **Neural activation pathway for various tasks in the manifold mapping.** The attached table reveals the differences in SuperGLUE. (a) t-SNE visualizations of data distribution using BERT (Devlin et al., 2018). (b) The neural activation pathway for each task, where the highest-probability expert in each layer is highlighted. The Mean Absolute Error (MAE) and Cosine Similarity (Sim.) with MultiRC are also computed respectively. The "Semantic Similarity Score of Each Task with MultiRC" is reported in <·>. (c) t-SNE visualizations of the final manifold representation from the SuperGlue development set in Tab. 6. See more results and details in §D and §H.

the impact of neural activation pathway on the enhancement of manifold mapping. Leveraging our intuitive MoE architecture, we can effectively visualize and analyze the activated expert that has the highest routing probability in each layer of MEPT, thereby constructing a pathway tailored to each task manifold mapping. There are two key observations supporting the enhanced generality of MEPT.

Observation I: Neural activation pathways vary across distinct manifolds. Observations from three different tasks neural pathways (*i.e.*, MultiRC, BoolQ and COPA) in Fig. 4(b) reveal that Neural activation pathways vary across distinct manifolds, reflecting task-specific requirements. Similar tasks (*i.e.*, with higher semantic similarity score in <·>) tend to loyally activate overlapping pathways due to shared knowledge and the engagement of similar experts weight. For instance, the question answering datasets MultiRC and BoolQ demonstrate lower MAE (*i.e.*, 2.88 $vs.$ 2.44) and higher cosine similarity (*i.e.*, 95.18 $vs.$ 96.72) compared to COPA, a sentence completion task. More specifically, MultiRC and BoolQ tend to exhibit greater similarity in activated weights than COPA (*i.e.*, 0.343 $vs.$ 0.191 in L_1) and are more likely to activate the same expert in each layer (*i.e.*, MultiRC and BoolQ both activate expert 1 in L_2, while COPA prefers expert 9). This observation highlights the heterogeneous learning preferences and capabilities of the experts.

Observation II: The improved linear separability of manifold mappings is facilitated by MEPT. We observe significantly improved separability in manifold mappings in the final model outputs, as illustrated in Fig. 4(a) and Fig. 4(c). Specifically, the data distribution of combined dataset exhibits overlapping and poorly separable task representations (*i.e.*, orange and purple groups tend to entangle closely together, whereas the green group is scattered throughout the space). In contrast, training with MEPT facilitates linear separability of manifolds (*i.e.*, each color group maintains a distinct margin across different classes) and extraordinary manifold mappings (*e.g.*, green group forms a more compact cluster). This separability aligns with our superior performance presented earlier in Tab. 6.

7 Conclusion

We present **M**ixture of **E**xpert **P**rompt **T**uning, a manifold mapper that leverages the MoE framework to flexibly activate different neural pathways, resulting in a dynamic and general fine-tuning. It has merits in: **i)** demonstrating generality across various model sizes through expert prompts learning; **ii)** further compacting the number of parameters, thereby improving scalability; **iii)** thoroughly investigating the essence of manifold learning to elucidate the extraordinary MoE architecture. The outcomes impart essential understandings of MoE design from a novel manifold mapper perspective, facilitating further exploration within this realm.

8 Acknowledgements

This research was supported by the National Science Foundation under Grant No.2242243, 2348468 & 2450068.

Limitations

For potential limitations, MEPT introduces an additional hyper-parameter—the number of router experts (*i.e.*, N_r in §3.2). However, due to the sparse MoE structure (see analysis in §5), scaling up the number of router experts incurs minimal additional training and inference overhead. Furthermore, in practical applications, we observe in §5 that, in general, a larger number of router experts tends to yield greater performance improvements, which serves as a guideline for selecting an appropriate number of router experts. Moreover, the efficiency of hyper-parameter selection can be further enhanced by incorporating a lightweight automatic expert number search mechanism.

Regarding the performance degradation under the mixture training scheme, it may stem from the imbalance in the dataset across the six SuperGLUE tasks (see Tab. 7), leading to unequal updates of expert prompts and consequently hindering the specialization of certain expert subsets. We plan to further explore its potential on larger and more diverse NLU tasks.

References

Mohammad Ali Sadraei Javaheri, Ehsaneddin Asgari, Alice C. McHardy, and Hamid R. Rabiee. 2024. SuperPos-Prompt: Enhancing soft prompt tuning of language models with superposition of multi token embeddings. In *NeurIPS Workshop*.

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. 2022. Ext5: Towards extreme multi-task scaling for transfer learning. In *ICLR*.

Martin Bjerke, Lukas Schott, Kristopher T. Jensen, Claudia Battistin, David A. Klindt, and Benjamin A. Dunn. 2023. Understanding neural coding on latent manifolds by sharing features and dividing ensembles. *Preprint*, arXiv:2210.03155.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Xinlei Chen, Saining Xie, and Kaiming He. 2021. An empirical study of training self-supervised vision transformers. In *ICCV*.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *EMNLP*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. 2020. Separability and geometry of object manifolds in deep neural networks. *Nature communications*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Shaohua Dong, Yunhe Feng, Qing Yang, Yan Huang, Dongfang Liu, and Heng Fan. 2024. Efficient multimodal semantic segmentation via dual-prompt learning. In *IROS*.

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

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2014. Learning factored representations in a deep mixture of experts. In *ICLR (Workshop Poster)*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL Workshop*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. In *ACL*.

Cheng Han, Qifan Wang, Yiming Cui, Zhiwen Cao, Wenguan Wang, Siyuan Qi, and Dongfang Liu. 2023. E2vpt: An effective and efficient approach for visual prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Cheng Han, Qifan Wang, Yiming Cui, Wenguan Wang, Lifu Huang, Siyuan Qi, and Dongfang Liu. 2024. Facing the elephant in the room: Visual prompt tuning or full finetuning? In *ICLR*.

Soufiane Hayou, Niladri Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*.

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

Bokai Hu, Sai Ashish Somayajula, Xin Pan, Zihan Huang, and Pengtao Xie. 2024a. Improving the language understanding capabilities of large language models using reinforcement learning. *arXiv preprint arXiv:2410.11020*.

Edward J. Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. 2024b. Amortizing intractable inference in large language models. *Preprint*, arXiv:2310.04363.

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

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *ECCV*.

Menglin Jia, Zuxuan Wu, Austin Reiter, Claire Cardie, Serge Belongie, and Ser-Nam Lim. 2021. Exploring visual engagement signals for representation learning. In *ICCV*.

Nanjiang Jiang and Marie-Catherine de Marneffe. 2019. Evaluating bert for natural language inference: A case study on the commitmentbank. In *EMNLP*.

Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. 2023. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *ICCV*.

Feihu Jin, Jiajun Zhang, and Chengqing Zong. 2023. Parameter-efficient tuning for large language model without calculating its gradients. In *EMNLP*.

Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. 2022. Prompting visual-language models for efficient video understanding. In *ECCV*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Artem Kirsanov, Chi-Ning Chou, Kyunghyun Cho, and SueYeon Chung. 2025. The geometry of prompting: Unveiling distinct mechanisms of task adaptation in language models. *Preprint*, arXiv:2502.08009.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Reducing activation recomputation in large transformer models. In *MLSys*.

Pengxiang Lan, Enneng Yang, Yuting Liu, Guibing Guo, Linying Jiang, Jianzhe Zhao, and Xingwei Wang. 2024. Efficient prompt tuning by multi-space projection and prompt fusion. *arXiv preprint arXiv:2405.11464*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *KR*.

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

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

Yuxin Lin, Xiaodong Ma, Yejin Chu, Zihan Jin, Yiwen Yang, Yue Wang, and Hao Mei. 2024. Lora dropout as a sparsity regularizer for overfitting control. *arXiv preprint arXiv:2404.09610*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeurIPS*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *ACL*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yiyang Liu, James Chenhao Liang, Ruixiang Tang, Yugyung Lee, Majid Rabbani, Sohail Dianat, Raghuveer Rao, Lifu Huang, Dongfang Liu, Qifan Wang, and Cheng Han. 2025. Re-imagining multimodal instruction tuning: A representation view. In *ICLR*.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *ACL*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *ECCV*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Vardan Papyan, X. Y. Han, and David L. Donoho. 2020. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. *EMNLP*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *ACL*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *EMNLP*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: Improving prompt tuning with residual reparameterization. In *ACL*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICML*.

Zhengxiang Shi and Aldo Lipani. 2024. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. In *ICLR*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Parsa Kavehzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. 2024. Sortednet: A scalable and generalized framework for training modular deep neural networks. *Preprint*, arXiv:2309.00255.

Moslem Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low rank adaptation. In *EACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. In *ACL*.

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

Qifan Wang, Yuning Mao, Jingang Wang, Hanchao Yu, Shaoliang Nie, Sinong Wang, Fuli Feng, Lifu Huang, Xiaojun Quan, Zenglin Xu, and Dongfang Liu. 2023. Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models. In *EMNLP*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *EMNLP*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of loRA experts. In *ICLR*.

Liqi Yan, Cheng Han, Zenglin Xu, Dongfang Liu, and Qifan Wang. 2023. Prompt learns prompt: Exploring knowledge-aware generative prompt collaboration for video captioning. In *IJCAI*.

Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. 2022. Unified vision and language prompt learning. *arXiv preprint arXiv:2210.07225*.

Runjia Zeng, Cheng Han, Qifan Wang, Chunshu Wu, Tong Geng, Lifu Huang, Ying Nian Wu, and Dongfang Liu. 2024. Visual fourier prompt tuning. In *NeurIPS*.

Qingru Zhang, Minkai Chen, Alexander Bukharin, Peiyuan He, Yang Cheng, Wei Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*.

Wei Zhu, Qiang Qiu, Jiaji Huang, Robert Calderbank, Guillermo Sapiro, and Ingrid Daubechies. 2018. Ldmnet: Low dimensional manifold regularized neural networks. In *CVPR*.

SUMMARY OF THE APPENDIX

This appendix contains additional details for the 63rd Annual Meeting of the Association for Computational Linguistics submission, titled *"Mixture of Expert Prompt Tuning as a Manifold Mapper"*. The appendix is organized as follows:

- §A provides an additional **introduction of the datasets** used, including the number of examples and task categories.
- §B explains **more implementation details** on backbone settings, baseline settings, training time, tokenization, and prepocessing.
- §C presents further **performance comparison with other PEFT methods**.
- §D includes supplemented **t-SNE, metric calculation and visualization details** in §6. Overall **expert utilization** is also provided and discussed.
- §E incorporates **extra ablative and pilot studies**.
- §F provides two representative **case studies** for understanding the impact of shared experts.
- §G provides discussions on **licenses, social impact, potential risks and directions of our future work**.
- §H further includes more **detailed neural pathway visualizations and analysis**.

A Dataset Statistics

Dataset	Examples	Task	Domain	Metric
Boolq	9,427	QA	Wikipedia	Acc
CB	250	NLI	various	F1/Acc
COPA	400	SC	blogs, encyclop	Acc
MRC	27,243	QA	various	F1a
RTE	2,490	NLI	news, Wiki	Acc
WiC	5,428	WSD	lexical databases	Acc

Table 7: The details of 6 SuperGLUE tasks used in our experiments. Datasets are classified into different task categories by (Raffel et al., 2020). NLI denotes natural language inference, QA denotes questions and answers task, SC denotes sentence completion, WSD denotes word sense disambiguation, Acc denotes accuracy, F1a denotes the macro F1 score.

Tab. 7 shows details of the six datasets from SuperGLUE benchmark (Wang et al., 2019) that we used for our experiments, along with their training sizes and evaluation metrics. Following (Raffel et al., 2020) and (Lester et al., 2021), for tasks that have two evaluation metrics we use the average of both scores as the final performance metric.

B Training Details

B.1 Backbone Settings

This paper employs two distinct model architectures (*i.e.*, decoder-only and encoder-decoder) and three different backbone sizes: T5-Base, T5-Large, and Llama3.2, with parameter counts of 200M, 770M, and 1B, respectively.

The performance gap between LLaMA 3.2 and T5 on SuperGLUE under prompt tuning stems from differences in pretraining objectives and architecture. LLaMA, trained with causal language modeling, excels at autoregressive generation but struggles with structured NLU tasks due to its mask strategy and decoder-only architecture, which limits bidirectional context integration. In contrast, T5's span-corruption objective fosters bidirectional reasoning, and its encoder-decoder architecture captures richer contextual dependencies. Notably, prior prompt tuning methods (Wang et al., 2023; Liu et al., 2022a) for T5 have exclusively prepended prompts to the encoder layer while leaving the decoder layer unchanged, as the latter must adapt to various downstream tasks. In our study, we explored extending prompt embeddings to both the encoder and decoder layers. However, preliminary results revealed significant performance degradation, corroborating the poor outcomes observed in LLaMA, a decoder-only model. These factors make T5 inherently more suited for prompt tuning, as its training paradigm aligns well with structured task formulations.

B.2 Baseline Settings

Fine-Tuning (Aribandi et al., 2022) refers to the standard full fine-tuning approach for T5, wherein all parameters are updated during training.
Prompt Tuning (Lester et al., 2021; Liu et al., 2025) is a foundational approach that introduces trainable prompts at the first input layer. XPrompt (Ma et al., 2022) enhances efficiency by pruning less informative token-level and piece-level prompts. P-Tuning v2 (Liu et al., 2022b) extends Prompt-Tuning by injecting distinct prompt embeddings into each Transformer layer. ResPrompt (Razdaibiedina et al., 2023) further improves performance and stability by incorporating residual

connections into the prompt tuning framework. DePT (Shi and Lipani, 2024) decomposes the soft prompt into a shorter soft prompt and a pair of low-rank matrices, which are optimized using two distinct learning rates. SuperPos-Prompt (Ali Sadraei Javaheri et al., 2024) introduces a novel reparameterization technique that leverages the superposition of multiple pretrained vocabulary embeddings to enhance soft prompt learning. SMoP (Choi et al., 2023) utilizes a gating mechanism in the first layer to train multiple short soft prompts, each specialized for different data subsets, offering an alternative to using a single long soft prompt for the entire dataset. EPT (Lan et al., 2024) leverages multi-space projection and prompt fusion. Specifically, it decomposes a given soft prompt into a shorter prompt and two low-rank matrices, optimizing them to enhance efficiency and performance. Notably, we linearly search the best learning rate from {1e-5, 5e-5, 1e-4} for MEPT and {0.1, 0.3, 0.5} for other prompt tuning methods following (Choi et al., 2023).

Linear serves as the final tunable score network in LlamaForSequenceClassification and has been evaluated exclusively within the Llama backbone framework. Notably, in all prompt tuning methods based on the Llama backbone, both the prompt and the final score network are fine-tuning.

B.3 Tokenization Preprocessing

Following common practice (Lester et al., 2021), we set the maximum input length, including the prompt, to 512 tokens for all experiments. Inputs are padded to this length, with padded tokens masked out, and truncated if they exceed 512 tokens. No additional preprocessing (e.g., punctuation removal) is applied; instead, raw text from SuperGLUE datasets is directly tokenized using the respective model's tokenizer. All experiments adhere to the SMoP (Choi et al., 2023) formatting, where classification tasks are reformulated into text-to-text format in T5 model. For instance, in BoolQ, labels '0' and '1' are mapped to 'True' and 'False,' respectively. In T5, we translate each SuperGLUE dataset into a text-to-text format following (Choi et al., 2023). In Llama, we continue to use the previously established text-to-text template while preserving the original labels, aligning the task with LlamaForSequenceClassification.

B.4 Router Details

- **Stochastic:** Use `torch.randint` to randomly select a router expert instead of always choosing the top-1 expert.
- **Dense:** Use all router experts simultaneously rather than selecting only the top-1.
- **Gumbel-Softmax:** Add Gumbel noise (temperature set to 1) to the logits, then apply a softmax function before selecting the router expert.
- **Perturbation:** Add Gaussian noise to the router with standard deviation $\sigma = 1$.

B.5 Text-to-text templates and verbalizers of SuperGLUE

BoolQ

Template: boolq passage: **passage** question: **question** **Verbalizer:** False, True

CB

Template: cb hypothesis: **hypothesis**. premise: **premise** **Verbalizer:** entailment, contradiction, neutral

COPA

Template: copa choice1: **choice1** choice2: **choice2** premise: **premise** question: **question** **Verbalizer:** choice1, choice2

MultiRC

Template: multirc question: **question** answer: **answer**. paragraph: **paragraph** **Verbalizer:** False, True

RTE

Template: rte sentence1: **premise** sentence2: **hypothesis** **Verbalizer:** entailment, not_entailment

WiC

Template: wic sentence1: **sentence1** sentence2: **sentence2** word: **word** **Verbalizer:** False, True

Methods	Boolq	CB	COPA	RTE	WiC
Prompt-Tuning*	2h38m	8m	11m	57m	51m
P-Tuning v2*	3h36m	10m	14m	1h28m	1h14m
XPrompt*	4h47m	14m	29m	2h26m	2h19m
ResPrompt*	3h47m	10m	23m	1h51m	1h34m
MEPT	3h23m	6m	8m	50m	35m

Table 8: Training time of MEPT on SuperGLUE. '*' indicates the results reported from (Wang et al., 2023).

B.6 Training Time

We further analyze and report the training time of different methods across all SuperGLUE tasks in T5-Base in Tab. 8. In Prompt-Tuning, the trainable parameters are limited to the prompts introduced at the input layer. For P-Tuning V2, trainable parameters extend across all layers, encompassing prompts throughout the model. XPrompt, following a pruning process, retains only a subset of prompts as trainable parameters, specifically at the input layer. In contrast, ResPrompt includes both input prompts and additional residual network components as trainable parameters. The total number of trainable parameters for each approach is summarized in Tab. 1. Due to the sparsity inherent in the MoE design, MEPT benefits from a relatively shorter training time, as only a subset of prompts is updated in each iteration (*i.e.*, 10 prompts in MEPT *vs.* 100 prompts in P-Tuning v2).

C Comparison with Other Parameter Efficient Methods

Methods	Boolq	CB	RTE	WiC
Fine-Tuning (Aribandi et al., 2022)	85.75	95.26	88.05	72.11
Adapter* (Pfeiffer et al., 2020a)	85.30	92.30	87.30	69.80
LoRA* (Hu et al., 2022)	85.10	92.60	87.30	70.20
MEPT (Ours)	**85.96**	**97.61**	**90.01**	**73.09**

Table 9: Performance comparison with other non-prompt tuning based parameter efficient methods on Boolq, CB, RTE and WiC for T5-Large. '*' indicates the results reported from (Jin et al., 2023).

To comprehensively evaluate performance, we compare our approach against other parameter-efficient methods that do not rely on prompt tuning, including Adapter (Pfeiffer et al., 2020a) and LoRA (Hu et al., 2022). More recent LoRA-based and adapter-based methods have been developed on different backbones and datasets, but their unpublished code makes reproduction under the same settings infeasible. The performance comparison results are presented in Tab. 9, showing that MEPT surpasses other parameter-efficient baselines by a significant margin. Notably, existing prompt tuning approaches also demonstrate superior performance compared to these methods.

D Visualization Details

Manifold Analysis Details. To analyze the learned feature representations across different tasks, we extract features from the encoder's last hidden layer by averaging over the sequence length dimension. For balanced comparison, we sample a maximum of 200 examples from each dataset's validation set. The features are then projected into a 2D space using t-SNE with a perplexity of 30 and random seed 42. We average the sequence-level representations ($h \in \mathbb{R}^{L\times d}$ where L is the sequence length and d is the hidden dimension) to obtain a fixed-size vector ($\bar{h} \in \mathbb{R}^{d}$) for each input, enabling direct comparison across examples of varying lengths. These embeddings are then visualized in a 2D scatter plot where points are colored according to their source dataset, allowing us to examine the natural clustering of different tasks in the model's learned representation manifold.

Metric Details. For analyzing model behavior, we employ different similarity metrics. To compare semantic similarities between different tasks, we utilize BERT Similarity Score which computes the dot product of L2-normalized BERT features ($BERTSim(b_1, b_2) = \frac{b_1}{||b_1||_2} \cdot \frac{b_2}{||b_2||_2}$). For neural pathway analysis to compare expert routing patterns, we use two complementary metrics: Mean Absolute Error (MAE) which measures the average magnitude of differences between expert utilization patterns ($MAE(x, y) = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$), and Cosine Similarity which quantifies the directional similarity between patterns ($CosSim(x, y) = \frac{x\cdot y}{||x||\cdot||y||}$).

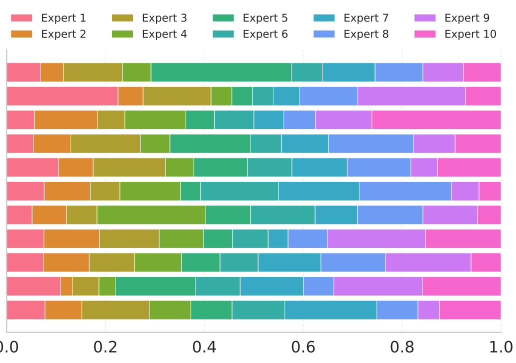


Figure 5: The expert utilization in each layer under the mixture training scheme.

Expert utilization. We present a visualization of the expert utilization across six tasks under the mixture training in Fig. 5. For each layer, the bar length in different colors corresponds to the activated frequency assigned to individual experts. As the total expert utilization are normalized to sum to 1, the total bar length for each layer remains constant. At a macro level, the visualization clearly illustrates significant variation in the contributions of each expert (*i.e.*, experts do not contribute equally across all layers), reinforcing the notion that different experts are specialized in distinct facets of natural language processing (*i.e.*, different layers exhibit preferences for distinct experts).

E Ablative and Pilot Studies

MEPT	CB	COPA	WiC	MRC
Xavier Normal	90.9	62.3	69.6	**80.7**
He Normal	90.7	62.7	69.0	80.2
Previous Knowledge	**91.0**	**63.7**	**69.9**	80.4

Table 10: Performance comparison with different prompt initializations on CB, COPA, WiC and MRC for T5-Base.

Prompt Initialization. As parameter initialization significantly influences the learning process (Glorot and Bengio, 2010), we further conduct experiments under three different initialization strategies. These include commonly used methods such as Xavier normal (Glorot and Bengio, 2010) and He normal (He et al., 2015), and a previous-knowledge-based initialization that leverages stored parameters from prior layer training to facilitate a quicker start. Our results reveal that traditional initialization methods are stable in MEPT, with Xavier normal showing a slight advantage across various tasks. Notably, the previous-knowledge-based initialization demonstrates the best performance, consistent with the intuition that leveraging prior training can provide better local minima. However, this approach requires a two-stage training process, doubling the training time and rendering it less cost-effective for practical applications. As a result, we apply xavier normal as the default initialization strategy in our framework.

Mixture Training. To further investigate the performance of MEPT under different mixture training settings, we additionally experiment with combining 2 to 5 datasets (*i.e.*, settings #2 to #5) for training. A greater number of combined datasets introduces higher diversity, thereby increasing the

#	CB	COPA	WiC	RTE	Boolq	MRC	Gain
1	✓						+0.7
2	✓	✓					+1.0
3	✓	✓	✓				+1.3
4	✓	✓	✓	✓			+1.6
5	✓	✓	✓	✓	✓		+2.1
6	✓	✓	✓	✓	✓	✓	+2.6

Table 11: Performance comparison under different mixture training settings. "Gain" denotes the absolute performance improvement over the previous best prompt tuning method, P-Tuning v2.

requirement for prompt generalization. In the non-mixture training setting #1, MEPT demonstrates the lowest performance gain, potentially due to the relatively homogeneous distribution within CB, which leads the MoE-based prompt tuning to select nearly identical neural pathways. As the number of combined datasets increases, performance improvements become more pronounced (*i.e.*, from +1.0 to +2.6), highlighting the potential of MEPT for handling more complex and diverse large-scale datasets.

	1	1→3	1→6	1→9	1→12
CB	86.4	88.5	89.8	89.9	**90.9**
COPA	58.3	60.7	61.3	62.0	**62.3**

Table 12: Additional ablation on prompt depth. $i \rightarrow j$ indicates the layer indices where prompts are inserted, with the 1st layer being closest to the input. T5-base consists of 12 layers in total.

Additional Ablation on Prompt Depth. we further conduct additional prompt depth experiments on CB and COPA and present the results in Tab. 12. Our findings reveal two key observations. First, we observe a consistent performance improvement as depth increases, demonstrating that deeper MoE prompts indeed contribute to learning better representations. Second, the performance gain is most significant when increasing the depth from 1 to 1-3 (*e.g.*, +2.4 on COPA), while the improvement becomes marginal from 1-9 to 1-12. This saturation trend aligns with previous research (Jia et al., 2022), indicating that prompt depth beyond a certain point yields diminishing returns.

	4:0	4:1	4:2	4:3	4:4
RTE	82.3	83.5	83.3	83.5	**83.6**

Table 13: Ablation Study on Routed-to-Shared Expert Ratio. Each configuration is denoted as a:b, representing the ratio between the routed experts (a) and the shared experts (b).

Routed-to-Shared Expert Ratio. To further investigate the fine-grained impact of the expert ratio, we conducted an additional experiment on the RTE dataset using T5-Base. The results are summarized in Tab. 13. As shown, increasing the number of shared experts leads to only marginal performance improvements, which is consistent with previous research (Liu et al., 2024). We hypothesize that this is primarily due to the relatively small scale and simplicity of the RTE task. In future work, we plan to extend this analysis to more complex and challenging datasets..

	T5-3B	LLaMA2-7B	T5-11B
Prompt Tuning	82.6	84.1	83.3
DePT	84.0	84.8	84.5
EPT	85.5	85.3	85.9
MEPT	**86.6**	**86.5**	**87.0**

Table 14: Preliminary RTE results (%) with quantized large foundation models.

Preliminary Results on Large Foundation Models. We explore the potential of prompt tuning on larger foundation models and have conducted preliminary experiments on the RTE task using T5-3B, LLaMA2-7B, and T5-11B. To reduce GPU memory usage, we applied quantization to these models. MEPT consistently achieves superior performance.

	SQuAD 1.1 Dev	SQuAD 2.0 Dev
Fine-Tuning	**91.0**	**82.0**
P-Tuning v2	89.5	77.8
MEPT	90.2	78.6

Table 15: Results on SQuAD 1.1 and 2.0 development sets. MEPT is competitive with prompt-tuning baselines but lags behind full fine-tuning.

Extension of MEPT to Question Answering. We further extend MEPT to more diverse task settings. Specifically, we apply MEPT to the SQuAD dataset (Rajpurkar et al., 2016), including both versions 1.1 and 2.0. This requires the model to parse passages and answer questions with a higher level of comprehension, beyond classification-style tasks. As shown in Tab. 15, MEPT consistently achieves competitive performance. However, both prompt tuning methods underperform compared to full fine-tuning, highlighting the limitations of prompt tuning for more complex and fine-grained tasks.

Mixture Training for Cross-Task Generalization. We include mixture training settings to evaluate prompt generality. We further conducted experiments by training on MultiRC and testing on BoolQ using T5-Base during rebuttal. As shown in Tab. 16, MEPT achieves notably better cross-task generalization.

	BoolQ ↓ BoolQ	MRC ↓ BoolQ	All ↓ BoolQ
Prompt-Tuning	78.1	78.0	78.0
P-Tuning v2	80.8	79.4	78.8
MEPT	**81.1**	**80.8**	**80.1**

Table 16: Results of cross-task generalization on BoolQ using T5-Base. $A \to B$ indicates training on dataset A and evaluating on dataset B. “All” refers to training on all six SuperGLUE datasets (see Tab. 6 for additional results).

F Case Study

Our design of shared experts is inspired by DeepSeek (Liu et al., 2024), which emphasizes capturing common knowledge and minimizing redundancy among routed experts. In our study, we find that within the SuperGLUE benchmark, such “common knowledge” is often aligned with semantic understanding across tasks. To illustrate this, we provide two representative case studies:

BoolQ (Boolean Questions). BoolQ is a classic question-answering dataset grounded in general knowledge. In the example below, the model is asked whether Iran and Afghanistan share the same language based on a provided passage. Without shared experts, the model fails to answer correctly (predicting 0 instead of the ground-truth label 1). This failure suggests that shared experts play a key role in encoding factual knowledge that spans across inputs.

Question: do iran and afghanistan speak the same language
Passage: Persian language – Persian, also known by its endonym Farsi, is one of the Western Iranian languages within the Indo-Iranian branch of the Indo-European language family. It is primarily spoken in Iran, Afghanistan (officially known as Dari since 1958), and Tajikistan (officially known as Tajiki since the Soviet era), and some other regions which historically were Persianate societies and considered part of Greater Iran. It is written in the Persian alphabet, a modified variant of the Arabic script, which itself evolved from the Aramaic

alphabet.
Label: 1

WiC (Word-in-Context). WiC is designed to evaluate models' ability to capture context-sensitive word meanings. In the following case, the word "place" carries different meanings: a physical location in the first sentence and a metaphorical position in the second. Without shared experts, the model incorrectly predicts the meanings to be the same, indicating its failure to distinguish subtle semantic nuances. This supports the role of shared prompts in improving semantic representation learning.

Word: place
Sentence 1: Do you want to come over to my place later?
Sentence 2: A political system with no place for the less prominent groups.
Label: 0

These qualitative findings are consistent with the quantitative results presented in Tab. 3, where the absence of shared experts leads to a noticeable performance drop on WiC (from 69.6 to 67.0). Together, these observations underscore the importance of shared experts in enhancing semantic understanding across tasks.

G Discussion

G.1 Manifold View of Layer-wise Abstraction

We analyze the network through the lens of manifold learning (Papyan et al., 2020; Kirsanov et al., 2025). As depth increases, transformer representations become more abstract and discard nuisance variation irrelevant to the task.

For a network with L layers, each layer l transforms the representation as follows:

$$h^{(l)} = f\left(h^{(l-1)}\right)$$

The abstraction process can be illustrated by:

$$x \to h^{(1)} \to h^{(2)} \to \cdots \to s$$

(from concrete to abstract)

The representation space progressively collapses around class-relevant information:

$$\dim\left(\mathrm{Var}(h^{(l)} \mid y)\right) \downarrow \quad \text{as } l \uparrow$$

This collapse exhibits several properties:

1. **Within-Class Convergence:** As l increases,

$$\||h^{(l)}(x_1) - h^{(l)}(x_2)\|| \to 0$$

for x_1, x_2 in the same class

2. **Between-Class Separation:** Simultaneously,

$$\||h^{(l)}(x_1) - h^{(l)}(x_2)\|| \to d > 0$$

for x_1, x_2 in different classes

3. **Final Collapse:** At the top layer,

$$h^{(L-1)}(x) \approx v_c \quad \text{for all } x \text{ in class } c,$$

where v_c is the prototype vector for class c.

These phenomena motivate our method. Fig. 4 shows within-class convergence (*e.g.*, the green cluster tightens), between-class separation (distinct inter-class margins), and final collapse (compare Fig. 4(a) and Fig. 4(c)).

G.2 Key Distinctions from SMoP

To highlight the advantages of MEPT, we summarize its key distinctions from SMoP (Choi et al., 2023) as follows. First, MEPT employs a deep multi-layer MoE architecture, incorporating multiple experts at every Transformer layer, while SMoP restricts MoE application to the input layer, thereby limiting its representational capacity. Second, MEPT introduces a hybrid design of shared and non-shared experts, enabling it to capture both generalizable knowledge and instance-specific features. In contrast, SMoP utilizes only non-shared experts, adhering to a traditional design. Third, MEPT integrates manifold learning with MoE through visualization, providing deeper insights into the internal dynamics of MoE-based prompt tuning.

G.3 Key Distinctions from DeepSeek MoE

Our design of routed and shared experts is inspired by (Liu et al., 2024; Valipour et al., 2024; Hu et al., 2024b; Bjerke et al., 2023; Cohen et al., 2020). However, there are two major differences between MEPT and the MoE approach used in DeepSeek (Dai et al., 2024):

- *MoE Placement and Structure*: Unlike DeepSeek and other traditional MoE methods, which apply MoE to the FFN module, MEPT incorporates the MoE structure into soft prompts that dynamically select prompt experts to enhance task adaptability.

- *Efficiency and Scalability*: MEPT benefits from a more efficient representation, making it easier to scale. Specifically, its inherently compact parameter space follows a leaner design (*e.g.*, the parameter count for the prompt-based method is mh, where m and h represents the prompt length and hidden layer dimension, respectively), allowing for more scalable expansion compared to DeepSeek-MoE. In contrast, the parameter count for the MLP component in a transformer is approximately $8h^2$.

We further in detailed discuss the differences in MoE structures below:

- *Expert Design*. DeepSeekMoE introduces Fine-Grained Expert Segmentation to facilitate more focused knowledge distribution across experts, resulting in a structure of 1 shared expert and up to 63 routed experts (*i.e.*, 1 + 63). In contrast, MEPT operates within a much smaller parameter space (*i.e.*, mh $vs.$ $8h^2$), and thus does not adopt fine-grained segmentation. Instead, we limit the number of router experts to between 10 and 20.

- *Load Balancing*. DeepSeekMoE employs both Expert-Level Balance Loss and Device-Level Balance Loss to mitigate the risk of routing collapse. In MEPT, all training is performed on a single GPU, eliminating the need for Device-Level Balance Loss. However, we were inspired by DeepSeekMoE's use of Expert-Level Balance Loss and explored whether similar strategies could help prevent the model from over-relying on a small subset of experts. As shown in Tab. 3, row six (Perturbation), we applied Gaussian noise to the router outputs.

- *Router Selection*. Both DeepSeekMoE and MEPT adopt Top-K routing; however, they differ in activation density. DeepSeekMoE uses a denser activation with K=7, resulting in 1 shared expert plus 7 routed experts (*i.e.*, 1+7 out of 63), whereas MEPT employs a much sparser activation with K=1, activating 1 shared expert and 1 routed expert (i.e., 1+1 out of 10–20). We also explored whether MEPT could benefit from denser activation. However, increasing the activation density (*e.g.*, 1+all) did not significantly improve the overall performance.

G.4 Asset License and Consent

The majority of prompt tuning (Liu et al., 2022b; Lester et al., 2021) and T5 (Aribandi et al., 2022), are licensed under Apache-2.0; Llama-3.2 1B (Dubey et al., 2024) is licensed under Llama 3.2 Community License Agreement; SuperGLUE is licensed under MIT;

G.5 Artifact Consistent With Intended Use

Our work ensures that the use of existing artifacts aligns with their intended purpose when specified. For the artifacts we create, it remains compatible with the original access conditions. In particular, we ensure that derivatives of data accessed for research purposes are confined to research contexts.

G.6 Social Impact

This work introduces MEPT, which demonstrates significant performance improvements over state-of-the-art baselines, as shown in Tab. 1, while requiring substantially fewer tunable parameters for prompt tuning. Our approach enhances model accuracy and is particularly beneficial for parameter-sensitive training scenarios, such as prompt tuning on resource-constrained devices and rapid adaptation with limited computational overhead.

G.7 Future Work

Extending to More Backbones and Datasets: Future research can investigate the generalizability of our method by applying it to a broader range of backbone architectures (*i.e.*, Deepseek and Qwen) and datasets (*i.e.*, MMLU and GSM-8k). Evaluating performance across diverse model families and domains will provide deeper insights into the robustness and adaptability of the proposed approach.
Scaling to a Larger and More Fine-Grained Expert System: Inspired by recent advancements such as DeepSeek, we aim to explore increasing the number of experts while incorporating more fine-grained specialization across layers. This could improve both efficiency and performance by dynamically selecting more targeted expert pathways for different types of inputs.
Applying to Causal Language Modeling: While our current study focuses on sequence classification or sequence-to-sequence models, extending the approach to causal language models presents an exciting avenue for research. Adapting MoE-based prompt tuning to autoregressive settings could unlock new capabilities for large-scale generative models.

G.8 Potential Risks

Consider the tuning process of LLM, which has potential risks for energy usage. Finetuning requires additional computational power, leading to energy use and increased environmental impact.

G.9 AI Disclosure

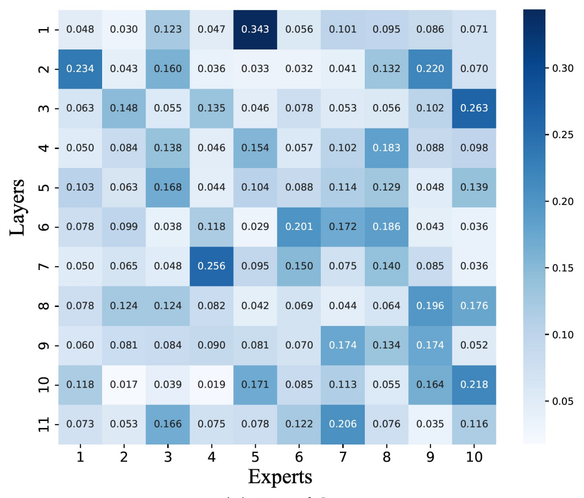
We acknowledge the use of GPT-4o for grammar correction and sentence-level refinement. The model was employed to enhance clarity, coherence, and fluency while ensuring the original meaning and intent of the text remained unchanged.

H Neural Pathways

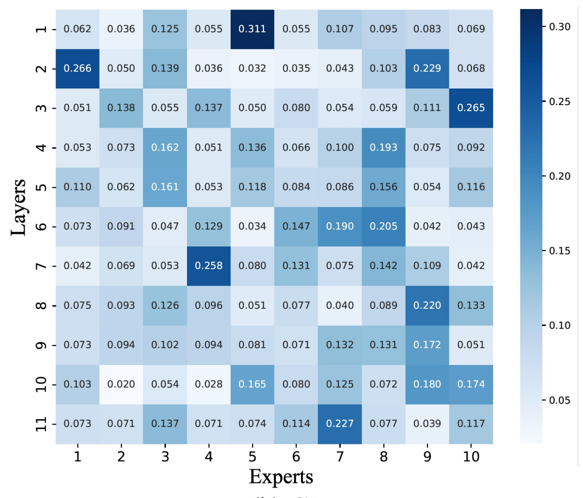
As shown in Fig. 6, we conduct a detailed analysis and visualization of expert weights across all layers and tasks, where higher probabilities are represented by deeper colors. From this visualization, we derive two key observations.

First, the neural pathways across all six tasks exhibit substantial similarity, with only a few key differences in expert selection. This is likely attributable to the intrinsic similarities present in natural language. Additionally, the relatively small dataset size for each task may contribute to this overlap. Utilizing larger and more diverse datasets could potentially enhance the differentiation of neural pathways, leading to more distinct expert specializations.

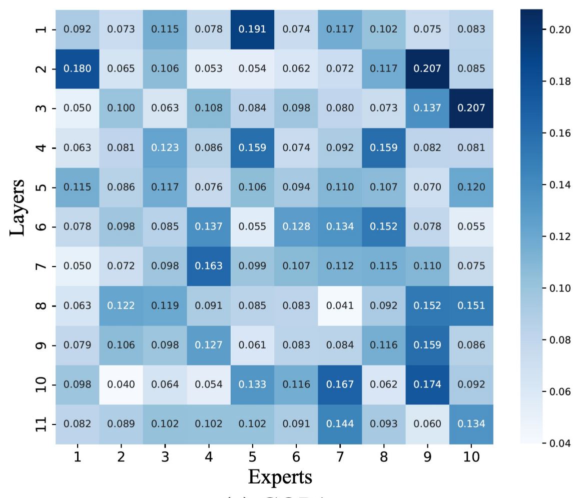
Second, tasks within the same category exhibit greater similarity in neural pathway activation compared to those from different categories, as discussed in §6. This is primarily reflected in the closer distribution of expert weights, suggesting that related tasks share common feature representations and utilize similar expert specializations.



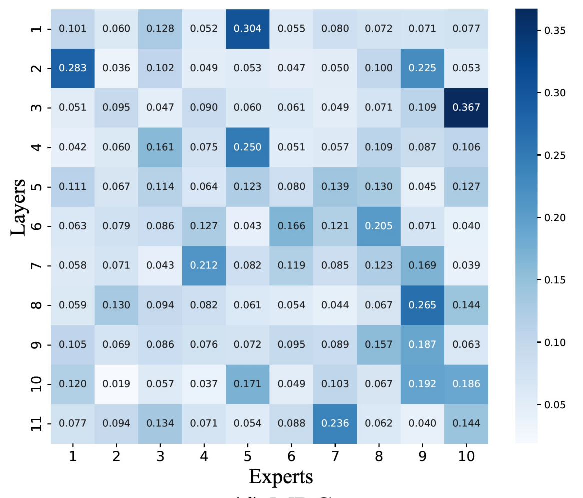
(a) BoolQ



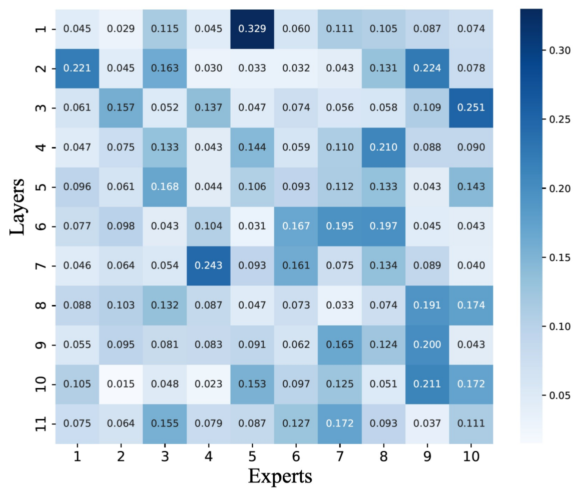
(b) CB



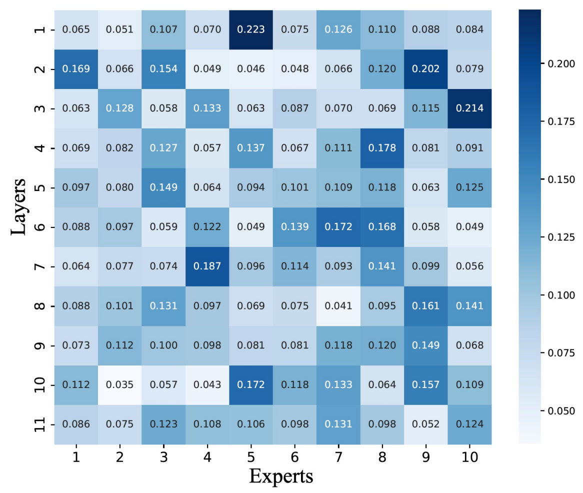
(c) COPA



(d) MRC



(e) RTE



(f) WiC

Figure 6: **Detailed neural activation pathway for six SuperGLUE tasks.** Each row is normalized to sum to 1, with the activated experts in each layer represented by the deepest color, indicating the probability of selecting a given expert.