MoEGen: Mixture-of-Experts for Instance Adaptive LoRA Generation

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient fine-tuning (PEFT) enables efficient adaptation of large language models, but existing MoE-based PEFT methods typically improve capacity by storing multiple full LoRA experts, causing adapter storage to grow linearly with the number of experts and restricting adaptation to a fixed expert pool. We ask whether MoE-based PEFT can produce instance-specific adaptations without explicitly storing a separate LoRA module for each expert. To address this gap, we propose **MoEGen**, an adaptation framework that shifts MoE-based PEFT from expert selection to expert-conditioned parameter generation. Instead of storing each expert as a full LoRA adapter, MoEGen represents each expert as a small learnable vector, termed an expert code. It routes each input over these vectors and uses their weighted combination to condition a lightweight hypernetwork that generates input-specific low-rank updates. This design decouples expert capacity from adapter storage while enabling instance-conditioned adaptation. Experiments on eight commonsense reasoning benchmarks show consistent improvements over strong static and MoE-based PEFT baselines across three backbones. MoEGen also performs strongly in joint medical and legal-domain adaptation.

1 Introduction

Modern pretrained LLMs demonstrate remarkable performance across a wide spectrum of natural language processing tasks Fedus et al. (2022); Touvron et al. (2023a); Chowdhery et al. (2022); OLMo et al. (2025). However, for challenging domain-specific downstream tasks, e.g., code generation and mathematical reasoning Rozière et al. (2024); Lewkowycz et al. (2022), model adaptation is a standard way to further improve task performance. Group (2024); Touvron et al. (2023a). While full supervised fine-tuning is an effective model adaption method, it is computationally expensive and requires storing separate task-specific model copies, making it less practical in resource-constrained settings Aghajanyan et al. (2021); Lialin et al. (2024).

Parameter-efficient fine-tuning (PEFT) methods provide practical alternatives for model adaptation by updating only a small subset of model parameters Li et al. (2023); Tian et al. (2024); Hu et al. (2022); Liu et al. (2024b). LoRA Hu et al. (2022), for example, learns low-rank weight updates while keeping the backbone parameters frozen. Recent MoE-LoRA methods further expand adaptation capacity by selecting or combining multiple LoRA experts through a learned router Dou et al. (2024); Liu et al. (2024a); Gao et al. (2025). However, because each expert stores a complete set of low-rank parameters, adapter storage grows linearly with the size of the expert pool, making a large and diverse adaptation space costly to maintain.

To address these limitations, we propose **MoEGen**, a lightweight plug-and-play adaptation framework that synthesizes input-specific LoRA parameters from compact latent expert codes. Instead of representing each expert as a full LoRA module, MoEGen encodes experts as small learnable vectors, termed expert codes, which are independent of the targeted module dimensions. Prompt-conditioned routing selects multiple expert codes in this low-dimensional latent space, and a shared hypernetwork generates their corresponding LoRA

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

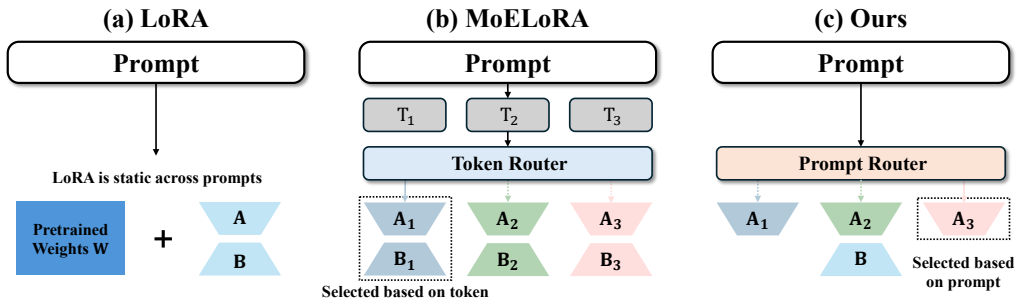


Figure 1: Comparison between traditional MoE and MoEGen. MoEGen replaces full feed-forward experts with compact style embeddings and a shared hypernetwork for input-specific LoRA generation.

parameters, which are then combined to form input-specific updates. This decouples expert capacity from explicit backbone-specific adapter storage and and enables flexible input-conditioned adaptation through combinations of generated LoRA parameters.

Overall, our contributions are as follows:

- **Lightweight expert parameterization.** MoEGen represents each expert as a low-dimensional learnable code rather than a complete LoRA adapter, substantially reducing the parameter and storage costs of maintaining multiple experts.
- **Dynamic input-conditioned LoRA generation.** MoEGen dynamically generates an input-specific low-rank matrix for each adapted component. This enables more targeted adaptation by tailoring the low-rank update to the semantic characteristics of each input.
- **Consistent empirical performance.** MoEGen improves over the strongest competing baselines by 0.6–1.1 points across eight commonsense reasoning benchmarks while updating only 0.44–0.52% of model parameters. It further achieves improvements of 4.6–6.2 points on the cross-domain joint benchmark.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning adapts pretrained models with few trainable parameters. Adapters insert bottleneck modules (Houlsby et al., 2019; Pfeiffer et al., 2021), while prefix-tuning adds learnable prefixes (Li & Liang, 2021), with potential latency and expressiveness trade-offs (Han et al., 2024). LoRA learns low-rank weight updates (Hu et al., 2022), which AdaLoRA extends through adaptive rank allocation (Zhang et al., 2023), DoRA through weight decomposition (Liu et al., 2024b), and LoftQ through quantization-aware initialization (Li et al., 2023). Alternatively, SpIEL (Ansell et al., 2024) and SMT (He et al., 2025) update selected parameters or sub-matrices. These approaches typically share learned adaptations across inputs, whereas MoEGen generates input-conditioned updates.

2.2 Mixture-of-Experts

Mixture-of-Experts increases model capacity through sparse expert routing (Shazeer et al., 2017), as demonstrated by Switch Transformer (Fedus et al., 2022) and GLaM (Du et al., 2022). Extending this paradigm to PEFT, MoELoRA (Liu et al., 2024a) and LoRAMoE (Dou et al., 2024) route among multiple LoRA experts, while HydraLoRA introduces asymmetric expert structures (Tian et al., 2024). To make expert allocation

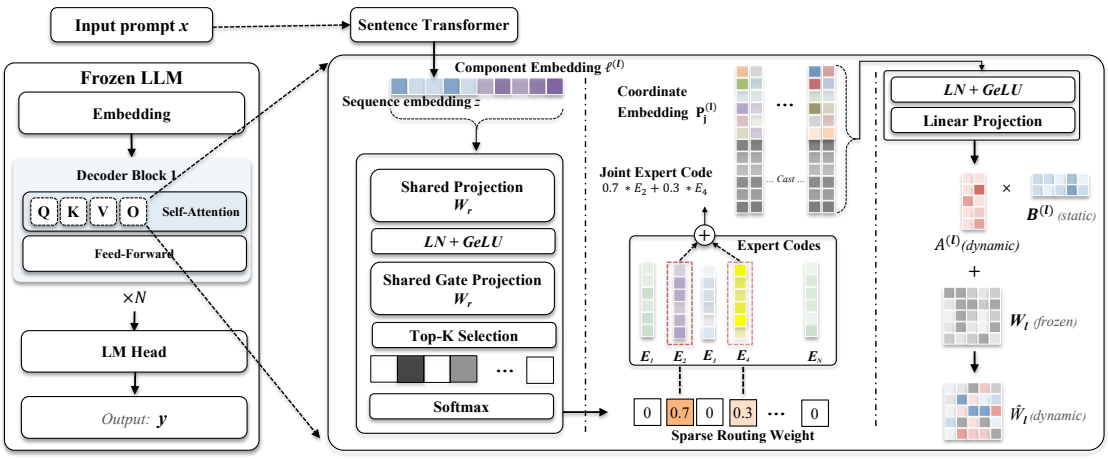


Figure 2: Overview of MoEGen. A frozen sentence encoder maps the input prompt x to a semantic embedding z. For each adapted component l, z is concatenated with a learnable component embedding $\ell^{(l)}$ and projected as $h^{(l)} = W_r[z; \ell^{(l)}]$ for sparse expert-code routing. Each selected expert code, together with component-local coordinates $p_j^{(l)}$, conditions a shared hypernetwork to generate a candidate LoRA A matrix. The candidate matrices are then fused using the corresponding routing weights to obtain the input-specific LoRA A matrix.

more flexible, LD-MoLE uses differentiable routing to adapt the number of activated experts for each token across layers (Zhuang et al., 2026). Despite these advances, maintaining expert-specific adapter parameters ties expert-pool expansion to storage growth. In contrast, MoEGen routes over compact expert codes and generates input-conditioned LoRA parameters through a shared hypernetwork as shown in Fig. 1.

2.3 Hypernetworks

Hypernetworks generate weights for another network (Ha et al., 2016). For efficient adaptation, HyperFormer generates task-conditioned adapters to share knowledge across tasks (Karimi Mahabadi et al., 2021), while Hyperdecoders extends conditioning to individual inputs (Ivison & Peters, 2022). This paradigm also supports dialect adaptation through HyperLoRA (Xiao et al., 2023), request-specific text editing through HyperEdit (Zeng et al., 2026), and context internalization through Doc-to-LoRA (Charakorn et al., 2026). Building on these advances, MoEGen combines sparse expert-code routing with shared hypernetwork-based LoRA generation, enabling input-dependent specialization while expanding the expert pool without storing a full adapter per expert.

3 Methodology

3.1 Overview

MoEGen (Fig. 2) consists of four modules: a frozen pretrained LLM, a frozen prompt encoder, a sparse MoE router, and a shared LoRA hypernetwork. For each input, the prompt encoder produces a context embedding, which is used by the router to select a small number of expert codes. These expert codes do not store LoRA parameters directly. Instead, they serve as compact conditioning vectors that guide the hypernetwork to generate dynamic LoRA matrices.

We apply MoEGen to the attention projections of each decoder block, including the query, key, value, and output projections; we refer to each adapted projection as a *component*. For a component with frozen weight $W^{(l)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, MoEGen generates a low-rank update $\Delta W^{(l)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and computes the adapted output as

$$y = W^{(l)}x + \frac{\alpha}{r}\Delta W^{(l)}x, \tag{1}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$, $y \in \mathbb{R}^{d_{\text{out}}}$, r is the LoRA rank, and α is the LoRA scaling factor. The low-rank update is factorized as $\Delta W^{(l)} = B^{(l)}(A^{(l)})^\top$ with $B^{(l)} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A^{(l)} \in \mathbb{R}^{d_{\text{in}} \times r}$, where $r \ll \min(d_{\text{in}}, d_{\text{out}})$. The original LLM parameters are never modified.

3.2 Per-Component Semantic Routing with Compact Expert Codes

MoEGen maintains a single pool of N trainable expert codes $\{E_1, \ldots, E_N\}$ with $E_i \in \mathbb{R}^{d_h}$, shared across all L components. The routing parameters remain compact because all components share a common gating projection W_g.

For each input prompt x, we use a frozen sentence encoder Zhang et al. (2025) $f_{\text{enc}} : \mathcal{X} \to \mathbb{R}^{d_z}$ to obtain a semantic context embedding $z = f_{\text{enc}}(x)$. The encoder remains fixed during training to preserve general semantic representations and reduce optimization cost. A lightweight router f_{r} then maps this shared semantic embedding into component-specific routing features:

$$f_{\text{r}}(z) = [h^{(1)}, \ldots, h^{(L)}], \qquad H = \left[(h^{(1)})^\top \quad \cdots \quad (h^{(L)})^\top\right]^\top \in \mathbb{R}^{L \times d_h}. \tag{2}$$

where L is the number of adapted LoRA components (i.e., target projections across decoder layers), and each row $h^{(l)} \in \mathbb{R}^{d_h}$ denotes the routing feature for component l.

To enable component-specific specialization, we associate each component with a learnable embedding $\ell^{(l)} \in \mathbb{R}^{d_\ell}$ and concatenate it with the shared prompt embedding:

$$h^{(l)} = W_r[z; \ell^{(l)}], \tag{3}$$

where $[\cdot;\cdot]$ denotes vector concatenation and $W_r \in \mathbb{R}^{d_h \times (d_z + d_\ell)}$ is a shared projection matrix. This formulation allows the router to model interactions between prompt-level semantics and component-specific identities, while requiring only $d_h(d_z + d_\ell) + Ld_\ell$ parameters.

Each expert code E_i is a compact conditioning vector rather than a full LoRA adapter: it encodes an adaptation style that the hypernetwork (Sec. 3.3) later expands into component-specific LoRA matrices. For each component l, the routing feature $h^{(l)}$ is first passed through a nonlinear block and then projected to expert logits:

$$\tilde{h}^{(l)} = \text{GELU}\Big(\text{LN}\Big(h^{(l)}\Big)\Big), g^{(l)} \quad = W_g\tilde{h}^{(l)} + b_g. \tag{4}$$

We then perform top-k selection and normalize the selected logits:

$$\mathcal{T}^{(l)} = \text{TopK}\Big(g^{(l)}, k\Big), w_i^{(l)} \quad = \frac{\exp\Big(g_i^{(l)}\Big)}{\sum_{j \in \mathcal{T}^{(l)}} \exp\Big(g_j^{(l)}\Big)}, \quad i \in \mathcal{T}^{(l)}. \tag{5}$$

Here, $\mathcal{T}^{(l)} \subseteq \{1, \ldots, N\}$ is the set of indices corresponding to the k largest entries of $g^{(l)}$. Experts outside $\mathcal{T}^{(l)}$ are assigned zero weight.

3.3 EXPERT-CONDITIONED LORA GENERATION

For each adapted component l, MoEGen generates an input-specific LoRA matrix $A^{(l)}$ while maintaining a trainable matrix $B^{(l)}$ shared across inputs. Motivated by the output-space efficiency considerations in Text-to-LoRA (Charakorn et al., 2025), this asymmetric design reduces generation overhead while retaining input-conditioned adaptation.

Parameter generation is conditioned on compact expert codes and component-local coordinates. For each component l, we introduce a learnable coordinate embedding $P^{(l)} \in \mathbb{R}^{d_{\text{in}} \times d_h}$, whose j-th row $p_j^{(l)}$ corresponds to the j-th input dimension. For each selected expert $i \in \mathcal{T}^{(l)}$, a shared hypernetwork H_ϕ uses its code E_i and these coordinates to generate a candidate LoRA matrix $A_i^{(l)}$. The candidate matrices are then fused using the input-dependent routing weights. Generation and fusion are defined row-wise as

$$a_{j,i}^{(l)} = H_\phi\Big([p_j^{(l)}; E_i]\Big), \qquad a_j^{(l)} = \sum_{i \in \mathcal{T}^{(l)}} w_i^{(l)} a_{j,i}^{(l)}, \tag{6}$$

where $a_{j,i}^{(l)}, a_j^{(l)} \in \mathbb{R}^r$ denote the j-th rows of the candidate matrix $A_i^{(l)}$ and the final matrix $A^{(l)}$, respectively. The resulting input-specific matrix is therefore $A^{(l)} = \sum_{i \in \mathcal{T}^{(l)}} w_i^{(l)} A_i^{(l)} \in \mathbb{R}^{d_{\text{in}} \times r}$.

The hypernetwork consists of LayerNorm, GELU activation, and a linear projection from $2d_h$ to r dimensions. Shared across components and experts, this block contributes $\mathcal{O}(d_h r)$ parameters, independent of model depth, d_{in}, d_{out}, and the number of experts N.

Finally, the fused matrix $A^{(l)}$ and the trainable matrix $B^{(l)} \in \mathbb{R}^{d_{\text{out}} \times r}$ define the low-rank update $\Delta W^{(l)} = B^{(l)}(A^{(l)})^\top$. Following LoRA (Hu et al., 2022), $B^{(l)}$ is initialized to zero for stable training.

3.4 TRAINING OBJECTIVE

MoEGen is trained end-to-end with the standard causal language modeling objective. We optimize three groups of parameters: (i) the router parameters W_r, $\{\ell^{(l)}\}_{l=1}^L$, W_g, b_g; (ii) the expert codes $\{E_i\}_{i=1}^N$; and (iii) the hypernetwork parameters ϕ_{Linear}, $\{P^{(l)}, B^{(l)}\}_{l=1}^L$. The base model weights remain frozen throughout training. To avoid routing collapse and encourage balanced expert utilization, we add a load-balancing regularization loss following Fedus et al. (2022). The full training objective is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{lb}} \mathcal{L}_{\text{load}}, \tag{7}$$

where the language modeling loss is $\mathcal{L}_{\text{LM}} = -\sum_{t=1}^T \log p_\theta(y_t \mid y_{<t}, X)$. given an input-output pair (X, Y), λ_{lb} controls the strength of the load-balancing regularization. The load-balancing loss is defined as

$$\mathcal{L}_{\text{load}} = \frac{N}{L} \sum_{l=1}^{L} \sum_{i=1}^{N} \left(\frac{1}{Bk} \sum_{b=1}^{B} \mathbf{1}[i \in \mathcal{T}_b^{(l)}] \right) \quad \times \left(\frac{1}{B} \sum_{b=1}^{B} w_{b,i}^{(l)} \right). \tag{8}$$

where $\mathcal{T}_b^{(l)}$ denotes the top-k expert set selected for the b-th input at component l, and $w_{b,i}^{(l)}$ is the normalized routing weight for expert i, with $w_{b,i}^{(l)} = 0$ if $i \notin \mathcal{T}_b^{(l)}$. This regularizer couples the discrete routing decisions with the differentiable routing probabilities, encouraging more balanced expert usage across both inputs and components.

4 EXPERIMENT

4.1 EXPERIMENT SETUP

Datasets and benchmarks. Following He et al. (2025), we use COMMONSENSE-170K, which combines the training splits of eight commonsense reasoning benchmarks: BoolQ Clark et al. (2019), PIQA Bisk et al. (2019), SIQA Sap et al. (2019), HellaSwag Zellers et al. (2019), WinoGrande Sakaguchi et al. (2019), ARC-Easy Clark et al. (2018), ARC-Challenge Clark et al. (2018), and OpenBookQA Mihaylov et al. (2018). We evaluate each method on the held-out test set of each benchmark. We also use a mixed-task dataset composed of four biomedical and legal-domain tasks: MedNLI Romanov & Shivade (2018), PubMedQA Jin et al. (2019), HQS Ben Abacha & Demner-Fushman (2019), and BillSum Kornilova & Eidelman (2019). MedNLI and PubMedQA are evaluated by accuracy, while HQS and BillSum are evaluated by ROUGE-1, ROUGE-2, and ROUGE-L Lin (2004). For this setting, the *Avg* column averages one primary metric per task: accuracy for MedNLI and PubMedQA, and ROUGE-L for HQS and BillSum.

Baselines. We compare MoEGen with static and mixture-of-experts PEFT methods. The static baselines include LoRA, DoRA, SpIEL, and SMT, whose adaptation parameters remain fixed across inputs at inference time. The MoE-based baselines include MoELoRA and MoLA, which incorporate expert routing into low-rank adaptation. These comparisons assess the effectiveness of input-conditioned parameter generation relative to static adaptation and existing expert-based approaches.

Implementation details. MoEGen uses a frozen Qwen3-Embedding-0.6B Zhang et al. (2025) sentence encoder. We use $N{=}8$ expert codes with code dimension $d_h{=}32$, top-$k{=}2$ routing, and LoRA rank $r{=}64$, and apply the generated adapters to the Q/K/V/O projections of every decoder block. All methods are trained

Table 1: Performance comparison on eight commonsense reasoning benchmarks. All benchmark results are accuracy scores (%). Params (%) denotes the percentage of trainable parameters. Average denotes the macro-average across the eight benchmarks. The best score for each benchmark and the highest average within each base model are highlighted in bold.

MODEL	METHOD	Params (%)	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	**Average**
LLaMA2-7B	LoRA	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA	0.42	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	80.5
	SpIEL	0.83	70.5	80.6	80.8	85.8	83.4	81.2	65.8	81.8	78.7
	SMT	0.84	72.0	83.8	80.8	93.3	82.8	86.7	74.0	81.0	81.8
	MoELoRA	1.19	**73.8**	85.5	**81.3**	**94.8**	**85.9**	87.9	75.0	84.2	83.6
	MoLA	0.87	71.8	81.9	78.6	88.9	81.6	82.8	69.5	80.0	79.4
	MOEGEN (Ours)	0.75	73.3	**86.4**	**81.3**	**94.8**	**85.9**	**88.9**	**75.4**	**87.4**	**84.2**
LLaMA3-8B	LoRA	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	SpIEL	0.70	72.1	83.6	80.0	91.8	85.4	91.2	76.8	80.8	82.7
	SMT	0.71	75.7	88.4	81.4	96.2	88.2	92.7	83.2	88.6	86.8
	MoELoRA	1.05	74.7	88.9	81.3	96.4	**89.3**	92.3	81.4	**89.6**	86.7
	MoLA	0.73	73.9	86.8	79.2	93.9	86.0	87.6	75.9	83.2	83.3
	MOEGEN (Ours)	0.47	**76.1**	**90.9**	**83.0**	**96.6**	89.1	**93.3**	**84.4**	89.4	**87.9**
Qwen3-8B-base	LoRA	0.70	74.3	89.0	80.3	94.4	88.0	94.7	88.1	90.4	87.4
	DoRA	0.71	72.9	90.9	82.7	95.9	88.9	97.0	90.9	**93.2**	89.1
	SpIEL	1.00	72.8	88.2	79.8	93.8	88.2	95.8	87.2	87.8	86.7
	SMT	0.71	74.9	88.9	81.9	94.8	86.6	95.5	88.4	92.2	87.9
	MoELoRA	0.75	72.5	90.6	81.9	93.4	84.1	**97.9**	**93.6**	89.8	88.0
	MoLA	1.50	74.3	90.4	80.9	95.3	86.7	96.8	91.9	92.2	88.6
	MOEGEN (Ours)	0.52	**75.6**	**91.4**	**83.0**	**96.4**	**90.0**	97.7	92.4	92.6	**89.9**

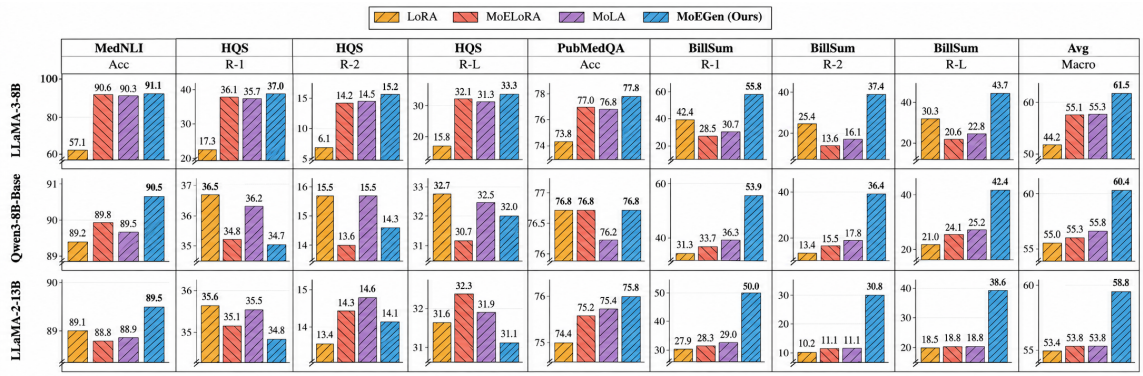


Figure 3: Comparison on the cross-domain benchmark at epoch 3. Rows correspond to LLaMA-3-8B, Qwen3-8B-Base, and LLaMA-2-13B. MedNLI and PubMedQA report accuracy; HQS and BillSum report ROUGE-1, ROUGE-2, and ROUGE-L. Avg is the macro-average of the two accuracy scores and the two ROUGE-L scores. Bold values indicate the best results within each panel. Additional numerical results are provided in the appendix.

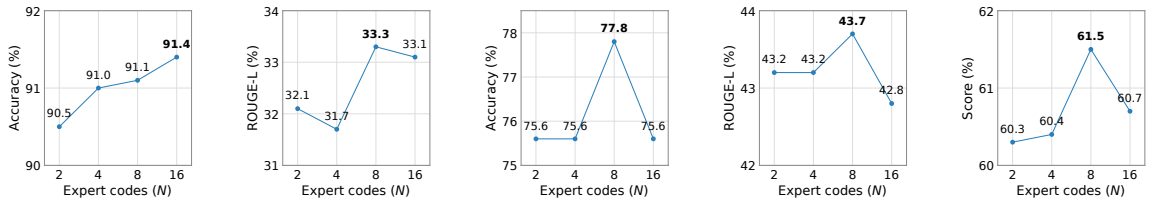
for 3 epochs with AdamW Loshchilov & Hutter (2017) and bfloat16 precision. The effective batch size is 16 for commonsense reasoning and 64 for mixed-task joint training. Additional details are provided in Appendix.

4.2 Commonsense Reasoning

Table 1 reports the results across LLaMA-2-7B Touvron et al. (2023b), LLaMA-3-8B et. al (2024), and Qwen3-8B-base Yang et al. (2025). MoEGen achieves the highest average score on all three backbones, reaching 84.2, 87.9, and 89.9, respectively. Compared with the strongest static PEFT baseline on each backbone, MoEGen improves the average score by 2.4 points over SMT on LLaMA-2-7B, by 1.1 points over SMT on LLaMA-3-8B, and by 0.8 points over DoRA on Qwen3-8B-base. MoEGen trains only 0.75%, 0.47%, and 0.52% of the backbone parameters on LLaMA2-7B, LLaMA3-8B, and Qwen3-8B-Base, respectively. These parameter budgets are smaller than those of both MoLA and MoELoRA, indicating that the improvements are not due to a larger trainable parameter budget.

Compared with standard LoRA, MoEGen achieves a higher score on every evaluated benchmark across all three backbones. On LLaMA-2-7B, it improves the average score from 77.6 to 84.2. On LLaMA-3-8B, the average score increases from 80.8 to 87.9. On Qwen3-8B-base, where LoRA achieves an average score of 87.4, MoEGen reaches 89.9. These results show that input-conditioned adaptation provides consistent benefits across model families with different baseline accuracies. Notably, the improvement remains positive even on Qwen3-8B-Base, where standard LoRA already performs strongly, suggesting that MoEGen captures useful input-dependent specialization beyond a single static low-rank update.

MoEGen also achieves higher average scores than the MoE-based PEFT methods MoELoRA and MoLA. MoELoRA and MoLA increase adapter capacity by introducing multiple LoRA experts, with each expert corresponding to a fixed set of learned parameters. In contrast, MoEGen routes among compact expert codes and uses the resulting conditioning signal to synthesize input-specific LoRA matrices. MoEGen improves over MoELoRA by 0.6, 1.2, and 1.9 points on LLaMA-2-7B, LLaMA-3-8B, and Qwen3-8B-base, respectively. Compared with MoLA, the corresponding improvements are 4.8, 4.6, and 1.3 points. The results indicate that generating input-specific adapter parameters yields higher average accuracy than selecting or mixing a fixed set of adapter experts in these experiments.



(a) MedNLI (b) HQS (c) PubMedQA (d) BillSum (e) Average

Figure 4: Effect of the expert-code pool size on the cross-domain joint benchmark with LLaMA-3-8B. Panels report accuracy for MedNLI and PubMedQA, ROUGE-L for HQS and BillSum, and the average score across the four tasks. All scores are shown on a percentage scale, with the best value in each panel bolded. Each panel uses its own vertical-axis range.

4.3 Cross-Domain Joint Training

To evaluate MoEGen under heterogeneous multi-task adaptation, we jointly train on MedNLI, PubMedQA, HQS, and BillSum following Lu et al. (2024). These tasks cover natural language inference, question answering, and clinical and legal summarization. All methods use the same training data and optimization protocol and are evaluated separately on each task. Figure 3 reports the results.

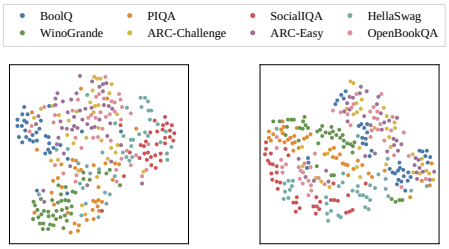
On LLaMA-3-8B, MoEGen achieves the highest score on every metric and improves the average score by 6.2 points over MoLA, from 55.3% to 61.5%. On BillSum, MoEGen reaches 43.7% ROUGE-L, compared with 30.3% for the best baseline. Plain LoRA performs worst: it drops to 57.1% on MedNLI and 15.8% on HQS, which indicates that a single static update struggles to accommodate the competing adaptation requirements of all four tasks. The advantage holds on Qwen3-8B-Base. MoEGen reaches the highest average of 60.4%, compared with 55.8% for the strongest baseline (MoLA), obtains the best MedNLI accuracy, and matches the best PubMedQA accuracy. On BillSum it reaches 42.4% ROUGE-L against 25.2% for the best baseline, while on HQS it stays within 0.7 points of the best result.

MoEGen also achieves the highest average score on LLaMA-2-13B, reaching 58.8% compared with 53.8% for MoELoRA and MoLA. It obtains the highest MedNLI and PubMedQA accuracy and improves BillSum ROUGE-L from 18.8% to 38.6%. On HQS, its scores remain within 1.2 points of the best baseline across all three ROUGE metrics. Together, the results on all three backbones show that MoEGen consistently improves overall performance under cross-domain joint training.

4.4 Ablation Study

Number of Expert. We study the effect of the expert-code pool size $N \in \{2, 4, 8, 16\}$ on the cross-domain joint benchmark using LLaMA-3-8B. All variants follow the setup in Section 4.3, with r=64, d_h=32, top-2 routing, an effective batch size of 64, and 3 training epochs. As shown in Figure 4, N=8 achieves the highest average score and performs best on three of the four tasks. Smaller pools may provide insufficient capacity for heterogeneous domains, whereas a larger pool does not yield further improvements. Notably, when N=2 under top-2 routing, both codes are always selected, reducing the router to weighting a fixed code pair. Its lower performance suggests that input-dependent code selection contributes to the effectiveness of MoEGen. We therefore use N=8 for the cross-domain benchmark. An ablation study on the routing design and shared expert is also provided in Appendix A.2.

Context encoding and MoE routing. We study the contributions of the context encoder and MoE routing in Table 2 (a). By default, MoEGen uses the frozen `Qwen3-Embedding-0.6B` encoder to obtain the routing context z. Replacing it with the final-token representation of the frozen 8B backbone (`w/o sentence`



(a) Routed fused style (b) Router latent representation

Figure 5: Case study of MoEGen routing on eight commonsense reasoning test sets. **(a)** t-SNE of the routed fused style produced by the hypernetwork. **(b)** t-SNE of the router latent representation used by the gate.

Table 2(a): Ablation of the context encoder and MoE routing using LLaMA3-8B as the base model. Without the frozen encoder, routing uses the backbone's last-token representation. Without MoE routing, the model uses a single hypernetwork.

Method	MedNLI Acc	BillSum R-L	HQS R-L	PubMedQA Acc	Avg
MoEGen	**91.1%**	**43.7%**	**33.3%**	**77.8%**	**61.5%**
w/o sentence encoder	90.8%	43.6%	32.4%	76.4%	60.8%
w/o MoE	90.7%	40.5%	30.9%	77.0%	59.8%

Table 2(b): Comparison of the context encoder with respect to training time, inference latency, and memory usage. Bold indicates lower time, latency, or memory usage.

Method	Train Time ↓ (min)	Latency ↓ (ms/ex.)	Memory ↑ (GB)	Frozen Params.
MoEGen	**13.1**	**24.6**	19.3	0.6B
w/o sentence encoder	14.2	25.2	**17.1**	**N/A**

`encoder`) reduces the average score from 61.5% to 60.8%, with the largest drops on PubMedQA (77.8% to 76.4%) and HQS (33.3% to 32.4%). This suggests that the external encoder provides more effective routing context under the evaluated configuration. We also remove the MoE routing module and retain only the input-conditioned HyperNetwork (w/o MoE). The average score drops to 59.8%, with larger decreases on BillSum (43.7% to 40.5%) and HQS (33.3% to 30.9%). This suggests that expert routing provides additional gains beyond input-conditioned HyperNetwork generation alone.

As shown in Table 2(b), the external encoder also offers a favorable time–memory trade-off. Compared with obtaining context from the 8B backbone, it reduces training time from 14.2 to 13.1 minutes (approximately 8%) and inference latency from 25.2 to 24.6 ms per example by avoiding an additional forward pass through the larger backbone. This requires 0.6B additional frozen parameters and increases GPU memory usage from 17.1 to 19.3 GB. We therefore retain the external encoder and MoE routing in our default configuration.

4.5 Qualitative Study

We visualize the routing behavior of MoEGen on eight commonsense reasoning benchmarks. For each task, we randomly sample 50 test prompts and apply t-SNE to the router inputs and the fused expert representations.

As shown in Figure 5, the router inputs already exhibit task-dependent structure. Distinctive tasks such as SocialIQA, WinoGrande, and BoolQ form relatively compact regions, whereas science QA benchmarks such as ARC-Challenge, ARC-Easy, and OpenBookQA are more closely distributed. After expert mixing, the fused representations preserve this organization while connecting related tasks more smoothly. This suggests that the router captures both task-specific adaptation patterns and shared structures across similar reasoning tasks. Further experimental analyses, including additional qualitative examples and an evaluation of computational efficiency, are provided in Appendix B.

5 Conclusion

This paper proposes an instance-adaptive framework for parameter-efficient LLM adaptation. It aligns LoRA updates with input semantics through MoE-guided routing and dynamic parameter generation driven by a lightweight hypernetwork. By replacing full LoRA experts with compact expert codes, MoEGen expands the adaptation space while keeping the parameter cost low. Experimental results show that MoEGen consistently outperforms static LoRA variants and existing MoE-based PEFT methods across commonsense, biomedical, and legal-domain benchmarks, demonstrating its advantages in accuracy and adaptation flexibility.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing. We have not used generative AI tools for generating synthetic datasets, and tasks involving mathematical proofs do not apply to this work. We have reviewed all AI-assisted outputs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our training data include existing biomedical datasets, namely MedNLI, PubMedQA, and HQS, which have been released for research use. We acknowledge the sensitive nature of medical applications and the potential risks of inaccurate model outputs. Our experiments evaluate model adaptation on these benchmarks and do not establish suitability for clinical use.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the proposed method in Section 3 and provide the datasets, evaluation metrics, and experimental settings in Section 4.1. Appendix A.1 further specifies the implementation details and baseline configurations. Unless otherwise specified, all methods are trained for 3 epochs using AdamW with a learning rate of 2×10^{-4}, cosine learning-rate decay, and the Alpaca prompt format, with the backbone frozen and only adapter-related parameters updated. We will publicly release our code upon acceptance.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *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)*, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568/.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M. Ponti. Scaling sparse fine-tuning to large language models, 2024. URL https://arxiv.org/abs/2401.16405.

Asma Ben Abacha and Dina Demner-Fushman. On the summarization of consumer health questions. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2228–2234, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1215. URL https://aclanthology.org/P19-1215/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-loRA: Instant transformer adaption. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=zWskCdu3QA.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-LoRA: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026. URL https://arxiv.org/abs/2602.15902.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha

Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.106. URL https://aclanthology.org/2024.acl-long.106/.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022. URL https://arxiv.org/abs/2112.06905.

Aaron Grattafiori et. al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and Vs Subrahmanian. MoLA: MoE LoRA with layer-wise expert allocation. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5112–5127, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.284. URL https://aclanthology.org/2025.findings-naacl.284/.

OpenAI Group. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. *arXiv preprint arXiv:1609.09106*, 2016. URL https://arxiv.org/abs/1609.09106.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=lIsCS8b6zj.

Haoze He, Juncheng B Li, Xuan Jiang, and Heather Miller. SMT: Fine-tuning large language models with sparse matrices. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=GbgCRJedQ7.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hamish Ivison and Matthew Peters. Hyperdecoders: Instance-specific decoders for multi-task NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1715–1730, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.124. URL https://aclanthology.org/2022.findings-emnlp.124/.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering, 2019. URL https://arxiv.org/abs/1909.06146.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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)*, pp. 565–576, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.47. URL https://aclanthology.org/2021.acl-long.47/.

Anastassia Kornilova and Vladimir Eidelman. BillSum: A corpus for automatic summarization of US legislation. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (eds.), *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 48–56, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5406. URL https://aclanthology.org/D19-5406/.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models, 2023. URL https://arxiv.org/abs/2310.08659.

Vladislav Lialin, Vijeta Deshpande, Xiaowei Yao, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2024. URL https://arxiv.org/abs/2303.15647.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications, 2024a. URL https://arxiv.org/abs/2310.18339.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024b. URL https://arxiv.org/abs/2402.09353.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lei Lu, Zhepeng Wang, Runxue Bao, Mengbing Wang, Fangyi Li, Yawen Wu, Weiwen Jiang, Jie Xu, Yanzhi Wang, and Shangqian Gao. All-in-one tuning and structural pruning for domain-specific llms, 2024. URL https://arxiv.org/abs/2412.14426.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021. URL https://arxiv.org/abs/2005.00247.

Alexey Romanov and Chaitanya Shivade. Lessons from natural language inference in the clinical domain. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1586–1596, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1187. URL https://aclanthology.org/D18-1187/.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning, 2024. URL https://arxiv.org/abs/2404.19245.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023b. URL https://arxiv.org/abs/2307.09288.

Zedian Xiao, William Held, Yanchen Liu, and Diyi Yang. Task-agnostic low-rank adapters for unseen English dialects. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7857–7870, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.487. URL https://aclanthology.org/2023.emnlp-main.487/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Yiming Zeng, Jinghan Cao, Zexin Li, Wanhao Yu, Zhankai Ye, Dawei Xiang, Ting Hua, Xin Liu, Shangqian Gao, and Tingting Yu. HyperEdit: Unlocking instruction-based text editing in LLMs via hypernetworks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 466–480, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.22. URL https://aclanthology.org/2026.findings-acl.22/.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.10512.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Yuan Zhuang, Yi Shen, Yuexin Bian, Qing Su, Shihao Ji, Yuanyuan Shi, and Fei Miao. LD-moLE: Learnable dynamic routing for mixture of loRA experts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4ST2YyTjI7.

A Appendix

A.1 Implementation Details

We keep the backbone LLM frozen for all PEFT methods and update only the adapter-related parameters. Unless otherwise specified, all methods are trained for 3 epochs with AdamW Loshchilov & Hutter (2017), a learning rate of 2×10^{-4}, cosine learning-rate decay, and the Alpaca prompt format.

For static PEFT baselines, including LoRA, DoRA, and SMT, we follow the experimental setup of SMT for training-related hyperparameters, including the learning rate, number of epochs, warmup steps, scheduler, and prompt format. We keep each method's adaptation modules consistent with the SMT setting whenever applicable: LoRA and DoRA are applied to the Q/K/V/O attention projections and the Gate/Up/Down feed-forward projections, while SMT is applied to the Q/K/V attention projections.

For the MoE-LoRA and MoLA baselines, we generally follow the hyperparameter settings used in their original papers and keep the implementation consistent with their routing designs. Specifically, MoE-LoRA uses $N{=}4$ LoRA experts with rank $r{=}32$ and a task-conditioned soft mixture, while MoLA uses $N{=}8$ LoRA experts with rank $r{=}32$ and token-level top-k routing with $k{=}2$. Both baselines are applied to the Q/K/V/O attention projections and the Gate/Up/Down feed-forward projections.

A.2 Ablation Study on Routing Design Choices

We ablate two routing design choices in MoEGen: the number of activated experts and the use of an always-on shared expert. All variants are evaluated on the cross-domain joint benchmark with LLaMA-2-13B as the backbone, following the same training setup as Section 4.3. They use LoRA rank $r{=}64$, scaling factor $\alpha{=}128$, expert dimension 32, and 3 epochs of supervised fine-tuning. The variants differ only in the routing strategy.

Table 6 shows that top-2 routing achieves the best overall performance, improving the average score from 0.5844 to 0.5877 over top-1 routing. The gains on MedNLI, BillSum, and PubMedQA suggest that combining two routed expert codes provides a more flexible adaptation space than selecting a single expert. HQS is the only exception, where top-1 routing gives a slightly higher ROUGE-L score, possibly because its summarization format is more constrained and benefits from stronger routing sparsity.

Adding an always-on shared expert does not improve performance. Compared with standard top-2 routing, the shared-expert variant reduces the average score from 0.5877 to 0.5827, with drops on BillSum, HQS, and PubMedQA. This suggests that, under a fixed parameter budget, the shared expert may absorb part of the learning signal and weaken the specialization of routed experts. Therefore, we use top-2 routing without a shared expert as the default configuration.

Method	MedNLI Acc.	BillSum R-L	HQS R-L	PubMedQA Acc.	Avg.
Top-1	0.8924	0.3838	**0.3154**	0.7460	0.5844
Top-2 + shared	0.8924	0.3824	0.3138	0.7420	0.5827
Top-2	**0.8952**	**0.3861**	0.3113	**0.7580**	**0.5877**

Table 6: Ablation study on routing design choices.

Table 3: Additional BillSum qualitative examples illustrating two distinct failure modes of the static LoRA-based baselines: a repetitive-loop failure on the Comprehensive Immunosuppressive Drug Coverage Act (top) and a mid-clause-boilerplate failure on the Federal Election Law Reform Commission Act (bottom). All methods are trained jointly on the cross-domain mixture (MedNLI, PubMedQA, HQS, BillSum) with Llama-2-13B.

Method	**R-L**	**Predicted summary**
Example 1: Comprehensive Immunosuppressive Drug Coverage Act (repetitive-loop failure).		
Reference	–	Comprehensive Immunosuppressive Drug Coverage for Kidney Transplant Patients Act of 2016. This bill amends titles II and XVIII (Medicare) of the Social Security Act to indefinitely extend Medicare coverage of immunosuppressive drugs for kidney transplant recipients. Under current law, such coverage is limited to 36 months following a transplant.
LoRA	0.182	*(c) of this section shall apply to such an individual in the same manner as they apply to an individual who is entitled to benefits under part A.*
MoE-LoRA	0.047	*(c) of this section shall not apply to such an individual. The provisions of subsection (d) of this section shall not apply to such an individual... The provisions of subsection (e) of this section shall not apply to such an individual...* (the pattern repeats for subsections (f), (g), ...)
MoLA	0.069	*(c) shall apply to such individual's coverage period in the same manner as they apply to the coverage period of an individual who is entitled to benefits under part A. The provisions of subsection (d) shall apply to such individual's coverage period... The provisions of subsection (e) shall apply...* (the pattern repeats for subsections (f), (g), ...)
MoEGen (Ours)	**0.571**	Comprehensive Immunosuppressive Drug Coverage for Kidney Transplant Patients Act of 2016. This bill amends title XVIII (Medicare) of the Social Security Act to require the Medicare program to provide immunosuppressive drug coverage for kidney transplant recipients. The bill also requires the program to provide immunosuppressive drug coverage for individuals whose entitlement to Medicare Part A hospital insurance benefits has ended by reason of the 36-month limit on such benefits.
Example 2: Federal Election Law Reform Commission Act (mid-clause-boilerplate failure).		
Reference	–	Federal Election Law Reform Commission Act - Establishes the Federal Election Law Reform Commission to recommend Federal election law reform. Requires a Federal election bill to be introduced in the House and the Senate within a specified time of the Commission's submittal of draft legislation. Authorizes appropriations.
LoRA	0.135	*... be considered as part of the rules of each House, respectively, and such rules shall supersede other rules only to the extent that they are inconsistent therewith.*
MoE-LoRA	0.135	*... be considered as part of the rules of each House, respectively, and such rules shall supersede other rules only to the extent that they are inconsistent therewith.*
MoLA	0.139	*... be considered a part of the rules of each House, respectively, and shall supersede other rules only to the extent that they are inconsistent therewith.*
MoEGen (Ours)	**0.522**	Federal Election Law Reform Commission Act - Establishes the Federal Election Law Reform Commission to study and recommend reforms in Federal election laws.

Table 4: Qualitative example on BillSum (*Heavy Truck Tax Fairness Act of 2010*).

Method	**Predicted summary**
Reference	Heavy Truck Tax Fairness Act of 2010 - Amends the Internal Revenue Code to: (1) repeal the excise tax on heavy trucks and trailers sold at retail; (2) increase the rate of tax on diesel fuel or kerosene to 31.6 cents per gallon; and (3) impose a floor stocks tax with respect to certain taxable fuels.
LoRA	*und of such tax is not allowable under section 6427(e) of the Internal Revenue Code of 1986.*
MoE-LoRA	*und of such tax is allowable under section 6427 of the Internal Revenue Code of 1986.*
MoLA	*und of such tax is allowable under section 6427 of the Internal Revenue Code of 1986.*
MoEGen (Ours)	Heavy Truck Tax Fairness Act of 2010 - Amends the Internal Revenue Code to repeal the excise tax on heavy trucks and trailers. Increases the tax on diesel fuel and kerosene. Imposes a floor stocks tax on any taxable liquid held on the enactment date by any person.

Table 5: Comparison on the cross-domain benchmark with LLaMA-3-8B, Qwen3-8B-Base, and LLaMA-2-13B at epoch 3. All scores are percentages, and best results for each model are in **bold**. Avg is the unweighted mean of MedNLI/PubMedQA accuracy and HQS/BillSum ROUGE-L.

Model	Method	MedNLI	HQS			PubMedQA	BillSum			Avg
		Acc	R-1	R-2	R-L	Acc	R-1	R-2	R-L	
LLaMA-3-8B	LoRA	57.1%	17.3%	6.1%	15.8%	73.8%	42.4%	25.4%	30.3%	44.2%
	MoELoRA	90.6%	36.1%	14.2%	32.1%	77.0%	28.5%	13.6%	20.6%	55.1%
	MoLA	90.3%	35.7%	14.5%	31.3%	76.8%	30.7%	16.1%	22.8%	55.3%
	MoEGen	**91.1%**	**37.0%**	**15.2%**	**33.3%**	**77.8%**	**55.8%**	**37.4%**	**43.7%**	**61.5%**
Qwen3-8B-Base	LoRA	89.2%	**36.5%**	**15.5%**	**32.7%**	**76.8%**	31.3%	13.4%	21.0%	55.0%
	MoELoRA	89.8%	34.8%	13.6%	30.7%	**76.8%**	33.7%	15.5%	24.1%	55.3%
	MoLA	89.5%	36.2%	**15.5%**	32.5%	76.2%	36.3%	17.8%	25.2%	55.8%
	MoEGen	**90.5%**	34.7%	14.3%	32.0%	**76.8%**	**53.9%**	**36.4%**	**42.4%**	**60.4%**
LLaMA-2-13B	LoRA	89.1%	**35.6%**	13.4%	31.6%	74.4%	27.9%	10.2%	18.5%	53.4%
	MoELoRA	88.8%	35.1%	14.3%	**32.3%**	75.2%	28.3%	11.1%	18.8%	53.8%
	MoLA	88.9%	35.5%	**14.6%**	31.9%	75.4%	29.0%	11.1%	18.8%	53.8%
	MoEGen	**89.5%**	34.8%	14.1%	31.1%	**75.8%**	**50.0%**	**30.8%**	**38.6%**	**58.8%**

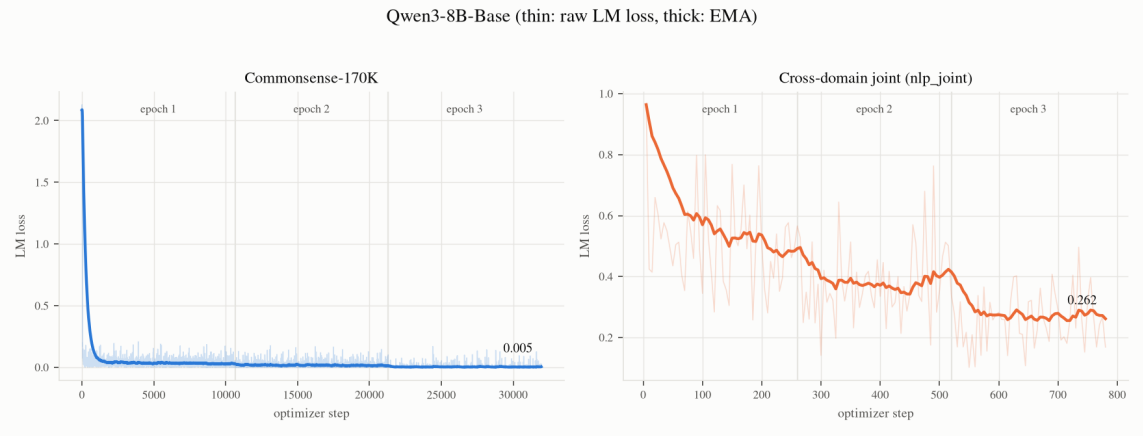


Figure 6: Training Loss Example.

Table 7: Per-sample inference latency (s). Max Tok. denotes the maximum number of generated tokens.

Task	Max Tok.	LoRA	MoE-LoRA	MoLA	Ours
MedNLI	8	0.058	0.062	0.074	0.090
HQS	64	0.309	0.336	0.437	0.337
PubMedQA	256	0.283	0.324	0.798	0.306
BillSum	384	0.966	1.025	2.259	1.006

B Additional Qualitative Study

We provide two additional BillSum predictions in Table 3, drawn from the same joint cross-domain training run as Table 4. Table 4 presents a representative example from BillSum. The reference summary describes the bill by naming the act and outlining its main provisions. In contrast, LoRA, MoE-LoRA, and MoLA generate short legal fragments copied from the bill text, failing to capture the main actions of the legislation. The outputs of MoE-LoRA and MoLA are also highly similar, suggesting that simply adding MoE-style adapters may still struggle to model diverse task behaviors under joint multi-task adaptation. MoEGen produces a more complete summary. It correctly identifies the bill title and recovers the key legislative actions, including the repeal of the excise tax and changes to fuel-related taxes. As a result, its prediction is much closer to the reference summary and achieves a substantially higher ROUGE-L score.

The two examples illustrate distinct failure modes of the static LoRA-based baselines on legal long-document inputs: collapse into a repetitive "subsection X does not apply" loop in Example 1, and emission of mid-clause rules-of-the-House boilerplate in Example 2. In both cases MoEGen opens with the act name followed by "Amends … to…", the canonical BillSum register, and enumerates the substantive actions present in the reference. None of these behaviors require extra parameters: MoEGen uses fewer trainable parameters than LoRA and MoLA and roughly the same as MoE-LoRA in Table 5, so the difference is attributable to the input-conditioned parameterization rather than to capacity. In particular, MoLA uses more than twice as many trainable parameters as MoEGen yet exhibits the same qualitative failures as the other two static baselines on these inputs.

C INFERENCE COST

We further evaluate the inference cost of MoEGen on the cross-domain joint benchmark. Table 7 reports the per-sample latency of each method using LLaMA-2-13B on a single B200 GPU with batch size 32. MoEGen introduces an input-dependent routing and generation cost, but the overhead remains small on longer-generation tasks. The largest relative gap appears on MedNLI, where the output is very short, and the fixed routing cost is less amortized. As the generation length increases, this gap becomes much smaller: MoEGen is only 9.1% slower than LoRA on HQS, 8.1% slower on PubMedQA, and 4.1% slower on BillSum. It also remains substantially faster than MoLA on the longer-generation tasks.