

Bridging SLMs and LLMs: A Lightweight Framework for Knowledge Injection

Zihao Li¹, Xuekong Xu², Yuqing Yu², Lixin Zou², Chenliang Li^{2*}, Wei Lu³

¹Laboratory for Artificial Intelligence and New Forms of Education, Faculty of Artificial Intelligence in Education, Central China Normal University

²School of Cyber Science and Engineering, Wuhan University

³School of Information Management, Wuhan University

zihao.li@ccnu.edu.cn, {xuekongxu, yuyuqing, zoulixin, cllee, weilu}@whu.edu.cn

Abstract

While large language models (LLMs) show immense potential in specialized domains, their adaptation is often hindered by limited access to domain-specific data and the high computational cost of model training. To address these challenges, we propose a lightweight knowledge injection framework. Instead of sharing raw data directly, our approach fine-tunes a small model and transfers its knowledge vectors to the target LLM for knowledge injection, thereby reducing data exposure and computational overhead. To mitigate the knowledge loss incurred during the injection process, we further introduce an iterative knowledge calibration strategy. We theoretically bound the error of the transformation matrix and prove the stability and linear convergence of the calibration process. Comprehensive experiments demonstrate that our method significantly outperforms parameter-efficient baselines and achieves performance comparable to full-parameter fine-tuning, while consuming only 28% of the GPU memory and 3% of the raw training data. Code is available at <https://github.com/Yuhanleeee/LISKI>.

1 Introduction

Large language models (LLMs) struggle to empower specialized domains. The challenges primarily stem from (1) the *restricted access to raw data* and (2) the *high demand for computational resources*, as shown in Figure 1. In particular, developers often cannot directly access and use domain-specific data, preventing LLMs from grasping professional knowledge and effectively supporting the corresponding downstream application. On the other hand, private users struggle to develop domain-specific LLMs with proprietary data, given the requirement of substantial computational cost.

To overcome these challenges, recent efforts have explored two primary directions. (1)



(a) Private User Limited by Computational Resources



(b) LLM Developer Limited by Raw Data Access



(c) Efficient and Lightweight Knowledge Injection

Figure 1: (a) Users in specialized domains possess valuable domain-specific data but often face limited computational resources, (b) whereas LLM developers have abundant computational resources but lack access to the raw data. This gap between data access and computational resources hinders the adaptation of LLMs to specialized domains. (c) Our method addresses this challenge by injecting domain-specific knowledge into LLMs while bypassing direct raw-data sharing and extensive computational requirements, thereby enhancing their capabilities in specialized domains.

Parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019) emerged as a prominent solution for specific domain knowledge injection under the constraints of computational resources. The representative methods include Low-Rank adaptation (LoRA) (Hu et al., 2021) and its variants (Dettmers et al., 2023; Zhang et al., 2023b; Liu et al., 2024), tailored adapters (Houlsby et al., 2019; Pfeiffer et al., 2021), and other approaches such as prefix tuning and prompt tuning (Li and Liang, 2021; Lester et al., 2021). (2) Privacy-preserving fine-tuning (Sun et al., 2024; Edemac and Wu, 2025) aims to adapt LLMs to downstream tasks while preventing the exposure of sensitive training data through mechanisms such as differen-

*Corresponding author.

tial privacy (Abadi et al., 2016), encrypted optimization (Aono et al., 2017), or federated learning (Konečný et al., 2016; McMahan et al., 2017; Kairouz and McMahan, 2021). Although these methods avoid direct raw-data sharing by retaining data locally, it typically requires users to store and optimize the entire model, resulting in a big concern regarding the substantial computational and memory overhead in practical deployments. None of the above approaches can address these two challenges simultaneously.

Bridging this gap, we propose a lightweight framework, in which **users only require fine-tuning a small-scale source language model (SLM) on the raw data**. The resulting parameters are shared with developers for *knowledge injection* into the target language model (LLM). Specifically, we learn a closed-form bilinear transformation via Alternating Least Squares (ALS) (Jain et al., 2013; Michel et al., 2019) to align the parameter spaces of the SLM and LLM, enabling the domain knowledge encoded in the fine-tuned SLM parameters to be efficiently transferred to the target LLM. This approach equips LLMs with knowledge from specialized domains while circumventing direct access to raw data and extensive computational costs.

Lossless knowledge injection is unattainable, given the approximation error introduced by the bilinear transformation. To alleviate this issue, we further propose a *knowledge calibration* stage to refine the injected knowledge. Specifically, the user randomly samples a small subset of raw data as calibration data, which is shared with the developer to fine-tune both the SLM and LLM. We then compute the discrepancy between the bilinearly transformed SLM parameters and the corresponding fine-tuned LLM weights, using the resulting residual as a calibration matrix to correct knowledge transfer deviations during injection.

Theoretical analysis bounds transformation errors and establishes the convergence and stability of knowledge calibration. Extensive experiments demonstrate that our method outperforms LoRA by up to $4\times$ in BLEU, while achieving performance comparable to full-parameter fine-tuning with only 28% peak GPU memory and 3% of the raw data for calibration. To sum up, the main contributions of this work are threefold:

- We propose a knowledge injection framework that injects domain-specific knowledge into LLMs through parameter merging rather than

raw data fine-tuning, enabling efficient adaptation under limited computational resources.

- Mathematical derivation provides the closed form and the error bound of the bilinear transformation. Theoretical analysis further ensures the convergence and stability of the knowledge calibration process.
- Experiments demonstrate that our method outperforms PEFT approaches by a large margin while maintaining strong scalability and transferability.

2 Related Work

Parameter-Efficient Fine-tuning. Parameter-efficient fine-tuning (PEFT) enables domain-specific adaptation of LLMs under limited computational budgets. Representative approaches include prompt tuning, adapters, and low-rank methods (Han et al., 2024), among which LoRA (Hu et al., 2021) is widely adopted. LoRA approximates full-parameter updates with two trainable low-rank matrices, reducing training costs but potentially limiting knowledge capacity and downstream performance. Subsequent works increase LoRA capacity through module stacking or adaptive rank allocation (Lialin et al., 2024; Xia et al., 2024; Zhang et al., 2023a,b). However, they still require loading the entire LLM, with computational and memory costs scaling with model size. In contrast, our method fine-tunes only an SLM and injects the acquired knowledge into LLMs of arbitrary scale under a fixed computational budget.

Privacy-Preserving Fine-tuning. Privacy-preserving fine-tuning adapts LLMs while reducing raw-data sharing. Representative paradigms include differential privacy (DP-SGD) (Abadi et al., 2016) and federated learning (FL) (Konečný et al., 2016; McMahan et al., 2017; Kairouz and McMahan, 2021). FedLLM enables collaborative adaptation without centralizing proprietary data (Kuang et al., 2024). FedPEFT reduces local training costs by updating lightweight modules such as LoRA. Offsite-Tuning (Xiao et al., 2023) and soft prompt transfer (Wang et al., 2025) shift most model capacity to the server. However, these approaches still require clients to load the LLM backbone and suffer from inconsistent local updates (Bian et al., 2025). In contrast, our framework acquires knowledge locally using an SLM and transfers it to a server-side LLM through knowledge injection, without requiring clients to host the backbone.

3 Methodology

3.1 Problem Formulation

Formally, let $\mathcal{D}_s$ denote the dataset from the specialized domain s , and let θ_s be an SLM, with $\hat{\theta}_s$ representing θ_s fine-tuned on $\mathcal{D}_s$. We defined the parameter difference between the fine-tuned $\hat{\theta}_s$ and the SLM θ_s as a knowledge vector δ_s :

$$\delta_s = \hat{\theta}_s - \theta_s, \quad (1)$$

where δ_s encapsulates the parametrized knowledge and task-specific capabilities from the domain s . Further, we denote θ_g as a target LLM. Typically, the scale of θ_g is much larger than that of θ_s , i.e., $N_{\theta_s} \ll N_{\theta_g}$, where N denotes the number of parameters. Given a knowledge vector δ_s , knowledge injection updates the parameters of θ_g using δ_s , endowing the target LLM with the specialized knowledge from domain s . This process can be formalized as:

$$\theta'_g \leftarrow f(\theta_g, \delta_s), \quad (2)$$

where $f(\cdot)$ denotes the knowledge injection function, and θ'_g represents the updated parameters.

3.2 Formulation of the Injection Framework

Overall, the framework of our proposed knowledge injection is shown in Figure 2. It includes two processes: **knowledge injection** and **knowledge calibration**. In the knowledge injection stage, we first design a knowledge injection function $\phi(\cdot)$, which projects the parameters of the knowledge vector δ_s into the parameter space of θ_g for parameter fusion:

$$\theta'_g = \theta_g + \lambda \cdot \phi(\delta_s). \quad (3)$$

where λ is a hyperparameter that regulates the strength of injected knowledge derived from δ_s .

Considering the information loss during the knowledge injection, we thereby propose a calibration strategy based on the discrepancy between the fused parameters $\phi(\delta_s)$ and the knowledge vector δ_g to compensate for the injected knowledge. Specifically, we randomly sample a very small calibration subset $\tilde{\mathcal{D}}_c$ from the dataset $\mathcal{D}_s$ (i.e., $\tilde{\mathcal{D}}_c \subset \mathcal{D}_s$) for the target LLM θ_g and the source SLM θ_s fine-tuning. The knowledge vectors δ_g^c and δ_s^c corresponding to θ_g and θ_s , respectively, are then derived as follows:

$$\delta_g^c = \hat{\theta}_g^c - \theta_g, \quad \delta_s^c = \hat{\theta}_s^c - \theta_s, \quad (4)$$

where $\hat{\theta}_g^c$ and $\hat{\theta}_s^c$ are the parameters of the target LLM and SLM after fine-tuning on $\tilde{\mathcal{D}}_c$. Therefore, the calibration knowledge vector Δ^c is defined as:

$$\Delta^c = \delta_g^c - \lambda \cdot \phi(\delta_s^c). \quad (5)$$

Substituting Eqs. (3) and (5) into Eq. (2) yields the overall knowledge injection formulation:

$$\theta'_g \leftarrow f(\theta_g, \delta_s, \delta_g^c, \delta_s^c) = \underbrace{\theta_g + \lambda \phi(\delta_s)}_{\text{Injection}} + \underbrace{\delta_g^c - \lambda \phi(\delta_s^c)}_{\text{Calibration}}. \quad (6)$$

3.3 Knowledge Injection

Bilinear Transformation Matrices. As the scale and semantic discrepancy between the source SLM θ_s and the target LLM θ_g , we propose a training-free knowledge injection module that projects the parameters from θ_s into the parameter space of θ_g for parameter fusion. Specifically, denote $\mathbf{W}_s \in \mathbb{R}^{m \times n}$ as the learnable parameter in θ_s (e.g., the weight of linear transformation), and denote $\mathbf{W}_g \in \mathbb{R}^{p \times q}$ as the corresponding parameters in θ_g , where typically $m < p$ and $n < q$. We define two transfer matrices $\mathbf{W}_\ell \in \mathbb{R}^{m \times p}$ and $\mathbf{W}_r \in \mathbb{R}^{n \times q}$, which transform $\mathbf{W}_s$ into $\mathbf{W}_g$. This process can be formalized as:

$$\mathbf{W}_g = \phi(\mathbf{W}_s) = \mathbf{W}_\ell^\top \mathbf{W}_s \mathbf{W}_r, \quad (7)$$

where $\mathbf{W}_\ell$ and $\mathbf{W}_r$ are the matrices to be estimated. Given $\mathbf{W}_g$ and $\mathbf{W}_s$, joint solving $\mathbf{W}_\ell$ and $\mathbf{W}_r$ is computationally intractable. Following the Alternating Least Squares (ALS) strategy, we deliver one variable while keeping the other fixed, thus, the two closed-form formulations can be delivered under distinct initialization constraints. As each subproblem is convex, a unique solution can be obtained (ref. Appendices A.1 and A.2 for details):

$$\begin{cases} \mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{W}_r^\top (\mathbf{W}_r \mathbf{W}_r^\top)^\dagger \mathbf{W}_s^\top (\mathbf{W}_s \mathbf{W}_s^\top)^\dagger, \\ \mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top (\mathbf{W}_\ell \mathbf{W}_\ell^\top)^\dagger \mathbf{W}_\ell \mathbf{W}_g. \end{cases} \quad (8)$$

We initialize one of the transfer matrices, $\mathbf{W}_\ell$ or $\mathbf{W}_r$, as a diagonal matrix $\mathbf{I}$ based on the dimensionality, to reduce computation and storage costs:

$$\begin{cases} \mathbf{W}_\ell := \mathbf{I}_{m \times p} & \text{if } n \leq m, \\ \mathbf{W}_r := \mathbf{I}_{n \times q} & \text{otherwise.} \end{cases} \quad (9)$$

The knowledge transfer matrices can be derived from Eq. (8) as follows:

$$\begin{cases} \mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{I}_{m \times p} \mathbf{W}_g & \text{if } n \leq m, \\ \mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{I}_{n \times q}^\top \mathbf{W}_s^\top (\mathbf{W}_s \mathbf{W}_s^\top)^\dagger & \text{otherwise.} \end{cases} \quad (10)$$

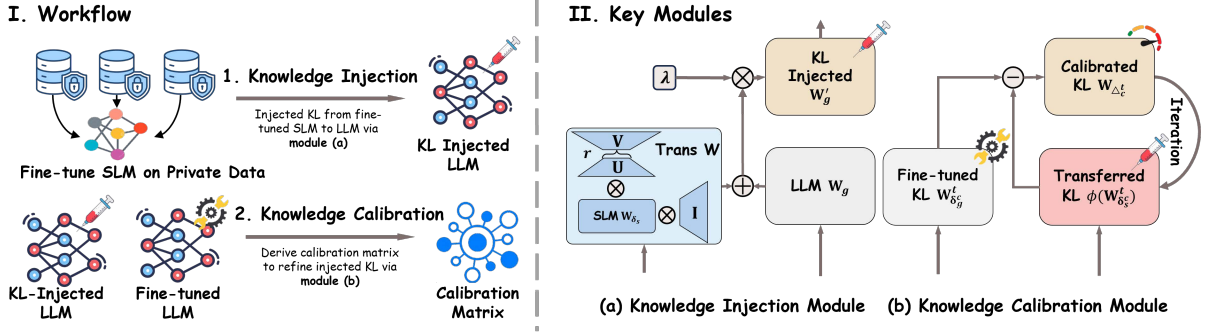


Figure 2: Overview of the workflow (left) of our method. It consists of two main phases: (1) **knowledge injection** and (2) **knowledge calibration**. The subfigure on the right illustrates the technical details of the key modules within our method. KL denotes knowledge.

Error Bound of Knowledge Transformation. Define $\mathbf{E}_{\text{proj}}$ as the knowledge transformation projection error, where $\mathbf{E}_{\text{proj}} = \mathbf{W}_g - \mathbf{W}_s \mathbf{W}_r$. Because the transfer matrix $\mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{W}_g$ acts as an orthogonal projector, the error is inherently bounded by the Frobenius norm of the target weights, $\|\mathbf{W}_g\|_F$. Applying SVD demonstrates that this squared projection error is strictly constrained by the spectral energy of the rank- r target model: $\|\mathbf{E}_{\text{proj}}\|_F^2 \leq \sum_{i=1}^r \sigma_i^2$, where σ_i represents the i -th singular value of $\mathbf{W}_g$ (ref. Appendix A.3).

Low-Rank Approximation. To improve computational efficiency and reduce parameter overhead, we further apply Singular Value Decomposition (SVD) to the transfer matrix ($\mathbf{W}_\ell$ or $\mathbf{W}_r$) for low-rank approximation (Golub and Kahan, 1965):

$$\mathbf{W}_\ell = \mathbf{U}_\ell \Sigma_\ell \mathbf{V}_\ell^\top \quad \text{or} \quad \mathbf{W}_r = \mathbf{U}_r \Sigma_r \mathbf{V}_r^\top, \quad (11)$$

where $\mathbf{U}_\ell \in \mathbb{R}^{m \times m}$, $\mathbf{V}_\ell \in \mathbb{R}^{p \times p}$, $\mathbf{U}_r \in \mathbb{R}^{n \times n}$ and $\mathbf{V}_r \in \mathbb{R}^{q \times q}$ are orthogonal matrices. $\Sigma_\ell \in \mathbb{R}^{m \times p}$ and $\Sigma_r \in \mathbb{R}^{n \times q}$ are diagonal matrices consisted by the singular values of $\mathbf{W}_\ell$ and $\mathbf{W}_r$, respectively. We retain the top- r singular values from Σ_ℓ and Σ_r for low-rank matrix factorization.

By substituting Eqs. (8), (9), and (11) into Eq. (3), we finally derive the updated parameter $\mathbf{W}'_g$ that fuses the domain-specific knowledge parameters $\mathbf{W}_{\delta_s} \in \mathbb{R}^{m \times n}$:

$$\begin{aligned} \mathbf{W}'_g &= \mathbf{W}_g + \lambda \cdot \phi(\mathbf{W}_{\delta_s}) \\ &= \begin{cases} \mathbf{W}_g + \lambda \cdot (\mathbf{I}_{m \times p}^\top \mathbf{W}_{\delta_s} (\mathbf{U}_r \Sigma_r \mathbf{V}_r^\top)) & \text{if } n \leq m, \\ \mathbf{W}_g + \lambda \cdot ((\mathbf{U}_\ell \Sigma_\ell \mathbf{V}_\ell^\top)^\top \mathbf{W}_{\delta_s} \mathbf{I}_{n \times q}) & \text{otherwise.} \end{cases} \end{aligned} \quad (12)$$

Through Eq. (12), domain-specific knowledge can be efficiently injected into the target LLMs.

Layer-wise Knowledge Injection. Given the scale discrepancy between the SLM and the LLM, the

layers of SLM are generally fewer than those of LLM. Therefore, we propose a sensitivity-aware strategy to select a matching number of layers from LLM for layer-wise knowledge injection. Specifically, parameter sensitivity is quantified by the gradient of the loss $\mathcal{L}$ with respect to the i -th parameter $\mathbf{w}_i$ (He et al., 2023; Pizarroso et al., 2022). This can be approximated using a first-order Taylor expansion by calculating the change in loss incurred when the parameter $\mathbf{w}_i$ is set to zero. A larger increase in loss indicates higher sensitivity, and thus greater importance and contributions of the parameter to the LLM. This process can be formally expressed as:

$$S(\mathbf{w}_i) = |\mathbf{w}_i \nabla_{\mathbf{w}_i} \mathcal{L}| \approx |\mathcal{L} - \mathcal{L}(\mathbf{w}_i = 0)|, \quad (13)$$

where $S(\mathbf{w}_i)$ measures the sensitivity of $\mathbf{w}_i$.

The layer-wise sensitivity S^j of the j -th layer is then computed by summing the sensitivities of all parameters within this layer, which reflects the overall contribution of layer j to the model’s performance. To ensure comparability across layers, we apply ℓ_2 -normalization to all layer sensitivities. The normalized sensitivity S^j for the j -th layer is computed as:

$$S^j = \frac{S^j}{\|\mathcal{S}\|_2} = \frac{S^j}{\sqrt{\sum_{j=1}^v (S^j)^2}}, \quad S^j = \sum_{\mathbf{w}_i \in \mathcal{P}(\mathcal{W}^j)} S(\mathbf{w}_i), \quad (14)$$

where $\mathcal{P}(\mathcal{W}^j) = \{\mathbf{w}_1^j, \mathbf{w}_2^j, \dots\}$ denotes the set of trainable parameters in layer j . $\mathcal{S} = \{S^1, S^2, \dots\}$ denotes the set of layer sensitivities.

We select the top- K layers in LLM with the highest layer-wise sensitivities as the target knowledge injection layers¹, where K equals the total number

¹If the SLM has more layers than the LLM, Top- K selection is instead performed on SLM.

of layers in the SLM. Formally, this process can be expressed as:

$$\mathcal{K} = \text{TopK}(\{S^1, S^2, \dots, S^v\}, K), \quad (15)$$

where $\mathcal{K}$ denotes the index set of top- K layers selected from LLM for knowledge injection².

3.4 Knowledge Calibration

Knowledge Calibration Matrix. Due to the architectural heterogeneity between SLM and LLM and the transformation error in Eq. (10), which is bounded by the spectral energy of the pre-trained weights of θ_g , (ref. Appendix A.3 for details), it is non-trivial to achieve lossless knowledge injection. To this end, we propose a knowledge calibration strategy to mitigate the discrepancy arising from the knowledge injection. The core idea is to calculate the difference between the fine-tuned parameters of the target LLM and the fused parameters delivered from the fine-tuned SLM via Eq. (12). This difference serves as a calibration matrix to refine knowledge injection. Specifically, users randomly sample a very small subset from the domain-specific dataset $\mathcal{D}_s$ as the calibration data $\tilde{\mathcal{D}}_c$ ³, which is then shared with developers and used to fine-tune the SLM and the top- K knowledge-injected layers of the LLM, respectively. Following Eq. (4), the knowledge vector δ_g^c and δ_s^c can be obtained. We, thereby, substitute Eq. (12) into Eq. (5), the calibration matrix is derived as follows:

$$\begin{aligned} \mathbf{W}_{\Delta^c} &= \mathbf{W}_{\delta_g^c} - \lambda \cdot \phi(\mathbf{W}_{\delta_s^c}) \\ &= \begin{cases} \mathbf{W}_{\delta_g^c} - \lambda \cdot (\mathbf{I}_{m \times p}^\top \mathbf{W}_{\delta_s^c} (\mathbf{U}_r \Sigma_r \mathbf{V}_r^\top)) & \text{if } n \leq m, \\ \mathbf{W}_{\delta_g^c} - \lambda \cdot ((\mathbf{U}_\ell \Sigma_\ell \mathbf{V}_\ell^\top)^\top \mathbf{W}_{\delta_s^c} \mathbf{I}_{n \times q}) & \text{otherwise,} \end{cases} \end{aligned} \quad (16)$$

where $\mathbf{W}_{\delta_g^c}$ and $\mathbf{W}_{\delta_s^c}$ are the parameters from the knowledge vector δ_g^c and δ_s^c , respectively, *i.e.*, $\mathbf{W}_{\delta_g^c} \in \delta_g^c$, $\mathbf{W}_{\delta_s^c} \in \delta_s^c$. $\mathbf{W}_{\Delta^c}$ is the corresponding calibration matrix associated with the calibration knowledge vector Δ_c , *i.e.*, $\mathbf{W}_{\Delta^c} \in \Delta_c$.

Knowledge Calibration via Iterative Refinement.

To mitigate the inherent gap between the ideal calibration matrix and that computed via Eq. (16), we

²To preserve the structural consistency between the source and target models, the injection adheres to a *relative layer alignment* principle. Specifically, the layers in $\mathcal{K}$ are ordered according to their relative depth in LLM for layer-wise knowledge injection.

³Figure 8 in Appendix F.4 illustrates the performance variations across different sizes of the calibration dataset. Performance steadily improves as the calibration data size increases, matching FPPT at 5% and surpassing it at 10%.

extend the calibration process beyond a single step by reformulating Eq. (16) into an iterative framework. The iteration refinement comprehensively considers both the current and historical information to achieve more precise and stable knowledge calibration. Concretely, we split the calibration dataset $\tilde{\mathcal{D}}_c$ into T groups for a total of T iterations. The iterative calibration matrix can be derived:

$$\mathbf{Z}^t = \mathbf{W}_{\delta_g^c}^t - \lambda \cdot \phi(\mathbf{W}_{\delta_s^c}^t), \quad (17)$$

$$\tilde{\mathbf{W}}_{\Delta^c}^t = \tilde{\mathbf{W}}_{\Delta^c}^{t-1} + \mathbf{F}^t (\mathbf{Z}^t - \tilde{\mathbf{W}}_{\Delta^c}^{t-1}), \quad (18)$$

$$\mathbf{F}^t = \mathbf{P}^{t-1} (\mathbf{P}^{t-1} + \boldsymbol{\zeta})^{-1}, \quad (19)$$

$$\mathbf{P}^t = (\mathbf{I} - \mathbf{F}^t) (\mathbf{P}^{t-1} + \boldsymbol{\xi}), \quad (20)$$

where $\mathbf{Z}^t$ denotes the calibration matrices at step t , obtained based on Eq. (16). $\tilde{\mathbf{W}}_{\Delta^c}^t$ is the matrix that integrates both the historical and current step calibrations. $\mathbf{F}^t$ denotes the gain derived from historical measurement. $\mathbf{P}^t$ denotes the covariance matrix of the calibration estimate at iteration t . $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ are hyperparameters controlling the covariances of injection and calibration updates, respectively, thereby enhancing the robustness and generalization of LLMs. Specifically, $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma_\xi^2)$ and $\boldsymbol{\zeta} \sim \mathcal{N}(0, \sigma_\zeta^2)$. $\mathbf{P}^0 = \alpha \mathbf{I}$.

Convergence and Error Analysis. Observing the Eq. (18), the posterior error of the calibration process evolves as (ref. Appendix A.4 for details): $\mathbf{e}_t = (\mathbf{I} - \mathbf{F}_t) \mathbf{e}_{t-1} + (\mathbf{F}_t - \mathbf{I}) \boldsymbol{\omega}_{t-1} + \mathbf{F}_t \boldsymbol{\nu}_t$, where $\boldsymbol{\omega}_{t-1}$ and $\boldsymbol{\nu}_t$ denote zero-mean Gaussian noise with covariances $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$, respectively. This yields a contraction factor $\mathbf{M}_t = \boldsymbol{\zeta} (\mathbf{P}_{t-1} + \boldsymbol{\xi})^{-1}$ whose eigenvalues strictly lie within $(0, 1)$ for $\boldsymbol{\zeta} \succ 0$, ensuring bounded input-bounded output (BIBO) stability across the entire iteration. As the multi-step refinement progresses, the internal covariance converges to the geometric mean $\mathbf{P}^* = \sqrt{\boldsymbol{\zeta} \boldsymbol{\xi}}$, and the true error covariance reaches a steady-state analytical bound of $C^* = \frac{2\boldsymbol{\zeta} \sqrt{\boldsymbol{\xi}}}{\sqrt{\boldsymbol{\xi}} + 2\sqrt{\boldsymbol{\zeta}}}$. Ultimately, these closed-form solutions theoretically guarantee that the proposed iterative knowledge calibration is provably stable, convergent, and mathematically bounded.

Algorithm 1 in Appendix B describes the completed process of our knowledge injection method.

4 Experiments

4.1 Experimental Setup

Knowledge injection aims to enhance the LLM’s capability in specialized domains, we therefore first conduct experiments on two domain-specific

Table 1: The overall performance of our method and baselines fine-tuning on Qwen2.5-3B. We report pass@1 as the evaluation metric for the HumanEval (HE) and MBPP benchmarks. B-4, R-1, and R-L correspond to BLEU-4, Rouge-1 and Rouge-L scores, respectively. $\uparrow$ indicates that higher values are better. The best results for each metric are in **bold**, and the second-best results are in underlined, respectively. PEFT refers to parameter-efficient fine-tuning. Our method consistently surpasses all baseline methods by a large margin and achieves comparable performance against FPFT, while our method only fine-tunes a 0.5B model.

Method	MedCaseReasoning				ChEBI20				Python	
	B-4 $\uparrow$	R-1 $\uparrow$	R-2 $\uparrow$	R-L $\uparrow$	B-4 $\uparrow$	R-1 $\uparrow$	R-2 $\uparrow$	R-L $\uparrow$	HE $\uparrow$	MBPP $\uparrow$
FPFT	0.316	0.377	0.202	0.274	0.563	0.646	0.479	0.559	0.695	0.780
P-Tuning v2	0.183	0.258	0.120	0.150	0.111	0.235	0.075	0.096	0.670	0.725
OFT	<u>0.207</u>	0.317	0.156	0.178	0.075	0.319	0.185	0.101	0.641	0.730
QOFTv2	0.179	<u>0.329</u>	0.165	0.171	0.081	0.364	0.205	0.101	0.665	0.754
LoRA	0.163	0.310	0.156	0.155	0.083	0.357	0.216	0.107	0.665	0.753
QLoRA	0.176	0.320	0.159	0.164	0.075	0.291	0.155	0.087	0.488	0.574
DoRA	0.175	0.311	0.150	0.152	0.079	0.416	0.273	0.112	0.659	0.754
DeLoRA	0.187	0.261	0.122	0.155	0.045	0.255	0.102	0.045	0.652	0.738
HiRA	0.196	<u>0.329</u>	0.159	0.173	0.110	0.338	0.187	0.128	0.646	0.761
Offsite-Tuning	0.199	0.324	<u>0.167</u>	<u>0.195</u>	<u>0.394</u>	<u>0.483</u>	<u>0.285</u>	<u>0.385</u>	0.490	0.659
DP-SGD	0.146	0.220	0.090	0.102	0.043	0.136	0.074	0.053	0.250	0.556
Ours	0.297	0.358	0.185	0.256	0.431	0.507	0.303	0.407	0.672	<u>0.759</u>

datasets: *MedCaseReasoning* (Wu et al., 2025) from the medical domain dataset and *ChEBI20* (Edwards et al., 2021) from the chemistry domain dataset. *MedCaseReasoning* evaluates an LLM’s capability to perform clinical reasoning from medical case reports, while *ChEBI20* examines chemical knowledge by requiring the LLM to yield molecule captions, *i.e.*, textual descriptions that accurately describe the key structural and functional features of molecules, based on the SMILES notation. Dataset details are provided in Appendix C. To further assess the capability of LLMs in coding, we fine-tune the baseline methods and SLM on *CodeFeedback* (Zheng et al., 2024). The resulting models are evaluated on the *HumanEval* (Chen et al., 2021) and *MBPP* (Austin et al., 2021) datasets for code generation. We select a series of LLMs from the Qwen2.5 (Team, 2024) and LLaMA3.2 (Grattafiori et al., 2024) families. The smaller-scale models are identified as SLMs, while the larger models serve as target LLMs for knowledge injection.

Baselines. We select PEFT and privacy-preserving fine-tuning methods as baselines for performance comparison, as they are the most closely related research directions to our work. These including P-Tuning v2 (Liu et al., 2022), OFT (Qiu et al., 2023), QOFTv2 (Qiu et al., 2025), LoRA (Hu et al., 2021), and its variants, *e.g.*, QLoRA (Dettmers et al.,

2023), DoRA (Liu et al., 2024), HiRA (Huang et al., 2025), DeLoRA (Bini et al., 2025), Offsite-Tuning (Xiao et al., 2023) and DP-SGD (Abadi et al., 2016). A detailed introduction of these baselines and implementation details are provided in Appendix D and E.

4.2 Main Results

Overall Performance. Table 1 shows the performance of our method compared with baselines on the medical reasoning, molecular caption, and coding generation, three highly specialized tasks. Overall, our method consistently outperforms the baselines by a large margin across nearly all metrics, *i.e.*, up to 46.07% and 9.39% relative improvements over the second-best method on the *MedCaseReasoning* and *ChEBI20* datasets, respectively. LLM-as-a-Judge evaluation demonstrates that our method achieves a higher win rate than LoRA (ref. Appendix F.1 for details). On code generation tasks, our method attains the best performance on *HumanEval* and competitive result on *MBPP*. Furthermore, it is noteworthy that our solution fine-tunes only the Qwen2.5-0.5B for knowledge injection, yet achieves comparable results to full-parameter fine-tuning of the Qwen2.5-3B.

Memory Usage. Figure 3 reports the peak GPU memory usage of our method and the baselines when fine-tuning Qwen2.5-7B on the *ChEBI20*

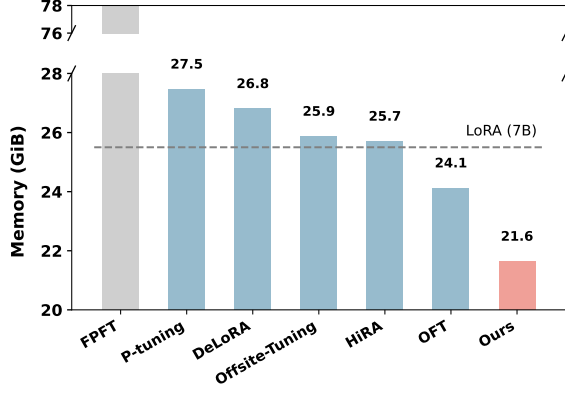


Figure 3: Peak memory usage on the *ChEBI20* dataset. All the experiments are conducted with Qwen2.5-7B. Our method demonstrates superior memory efficiency compared to PEFT methods, as it only fine-tunes a small language model during the knowledge injection.

dataset. Our approach achieves the lowest memory usage across other PEFT methods, as it requires users to fine-tune only a small language model (*i.e.*, Qwen2.5-0.5B) during knowledge injection. Importantly, the computational overhead of existing PEFT approaches scales with the size of the LLM (*e.g.*, the memory usage of LoRA increases by 78% when performing from 3B to 7B). In contrast, our method maintains a consistently low overhead, as it depends solely on the size of SLM, demonstrating superior scalability on larger model injections.

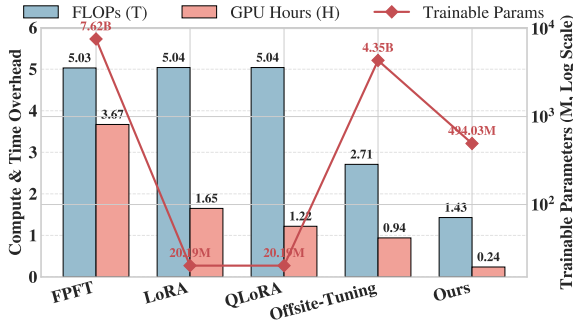


Figure 4: Comparison of our method against FPFT, LoRA, QLoRA, and Offsite-Tuning on the *ChEBI20* dataset using Qwen2.5-7B. Our approach substantially minimizes computational overhead (1.43T FLOPs / 0.24 GPU hours) and strictly limits raw data sharing to a 3% calibration subset, effectively decoupling client-side training from the target LLM backbone.

FLOPs, Trainable Parameters, and GPU Hours. We further profile our method against FPFT,

Table 2: Evaluation of layer selection strategies (Random selection and Top- K selection) and FPFT on the calibration dataset $\tilde{\mathcal{D}}_c$. Our approach yields the best results across all compared variations.

Method	MedCaseReasoning			
	B-4	R-1	R-2	R-L
w. Top-K	0.265	0.346	0.179	0.208
w. Random	0.243	0.340	0.174	0.196
w. $\tilde{\mathcal{D}}_c$ FPFT	0.211	0.331	0.162	0.185
Ours	0.297	0.358	0.185	0.256

LoRA, QLoRA, and Prefix-Tuning on the *ChEBI20* dataset, using Qwen2.5-7B as the target model, as shown in Figure 4. Existing PEFT and Offsite-Tuning baselines require access to the full raw dataset, our method limits data sharing to only a 3% calibration subset while achieving competitive performance. Moreover, our framework requires only 1.43T FLOPs and 0.24 GPU hours in total, substantially reducing the computational overhead compared with existing approaches. These results highlight the benefit of decoupling client-side training from the target LLM backbone, which substantially alleviates the memory and computational burden associated with adapting LLMs.

4.3 Ablation Studies

Sensitivity-aware Layer Selection and Calibration. To evaluate the effectiveness of our sensitivity-aware layer selection, we replace it with two typical strategies: random selection (randomly sampling K layers) and top- K selection (choosing the uppermost K layers of the LLM). We then measure the resulting performance differences in layer-wise knowledge injection. In addition, we also conduct full-parameter fine-tuning using only the calibration set to verify the significance of the knowledge injection stage. The results are reported in Table 2. Compared with random and Top- K selections, our sensitivity-aware layer selection strategy consistently achieves superior performance across all evaluation metrics. Moreover, fine-tuning solely on the calibration set yields the weakest performance, highlighting the necessity of our proposed knowledge injection stage for domain adaptation.

Scaling Evaluation. We investigate performance variations across different scales of SLMs and LLMs on the *MedCaseReasoning* dataset. Results are presented in Table 3 (ref. Appendix F.2 for details). Overall, Qwen2.5-7B with FPFT exhibits su-

Table 3: Performance of our method across different model scales, in comparison with FPFT and LoRA fine-tuned on Qwen2.5-3B and Qwen2.5-7B. Our solution achieves comparable results to FPFT across all scales while significantly outperforming LoRA.

Method		MedCaseReasoning			
		B-4	R-1	R-2	R-L
FPFT	3B	0.316	0.377	0.202	0.274
	7B	0.330	0.379	0.204	0.280
LoRA	3B	0.163	0.310	0.156	0.155
	7B	0.223	0.324	0.163	0.190
Ours	0.5B to 3B	<u>0.297</u>	0.358	0.185	0.256
	0.5B to 7B	0.288	0.351	0.179	0.249
	3B to 7B	<u>0.297</u>	<u>0.367</u>	<u>0.197</u>	<u>0.260</u>

prior performance compared to 3B, demonstrating the benefits of scaling up. Our solution achieves performance marginally below FPFT while substantially outperforming LoRA (*e.g.*, 82.21% improvement on BLEU-4 for Qwen2.5-3B), highlighting the robustness and scalability of our method.

Impact of Different LLM Families We further evaluate the performance of our proposed method across different LLM families. Specifically, we select LLaMA3.2-1B and Phi1.5-1.3B as the SLMs, the LLaMA3.2-3B and Phi2-2.7B are the corresponding target LLMs, for knowledge injection. Results are shown in Table 4. Overall, the observations are consistent with the experimental results using the Qwen2.5 family, *i.e.*, our method is consistently outperforms LoRA while slightly lower than FPFT, demonstrating the robustness of our approach and highlighting the practical applicability.

4.4 In-Depth Analyses

Iterative Knowledge Calibration. We investigate the performance with different iteration steps (*i.e.*, T) in Eq. (18), as well as the effect of removing knowledge calibration (*i.e.*, w/o Δ^c), to better understand the contributions of this component in our proposed method. All experiments are conducted using Qwen2.5-3B on the *MedCaseReasoning* dataset. As illustrated in Figure 5 (a), our method exhibits degraded performance when the calibration stage is omitted. Incorporating knowledge calibration leads to substantial performance improvements, underscoring the necessity of calibration in refining the injected knowledge to better align with the target model. Furthermore, we observe that only a moderate number of iterations

Table 4: Performance on LLaMA3.2-3B and Phi2-2.7B. Our method consistently matches FPFT while substantially surpassing LoRA.

Method		MedCaseReasoning			
		B-4	R-1	R-2	R-L
LLaMA	FPFT	0.326	0.382	0.206	0.279
	LoRA	0.285	0.362	0.193	0.251
	Ours	0.319	0.371	0.199	0.276
Phi	FPFT	0.308	0.372	0.202	0.269
	LoRA	0.126	0.289	0.137	0.106
	Ours	0.285	0.362	0.193	0.251

(*e.g.*, $T = 2$ or 3) is sufficient to achieve stable and competitive performance, which demonstrates that the iterative calibration procedure is effective in improving model performance and efficient in terms of computational cost.

Layer Sensitivity. Following Eq. (14), we compute and compare the layer sensitivity of Qwen2.5-3B on both the *MedCaseReasoning* calibration dataset and the *ChEBI20* calibration datasets, respectively. As shown in Figure 5 (b), the most significant layers of LLMs are primarily concentrated in the deeper portion of the network (*e.g.*, layers 28 to 32 in Qwen2.5-3B), while several middle layers and the initial layer (*i.e.*, the first layer) also play a non-negligible role when model training. This observation is consistent with findings in (Fan et al., 2020; Michel et al., 2019) that while deeper layers dominate the model’s output, early and middle layers still encode complementary information essential for overall performance.

Training Convergence of $\phi(\mathbf{W}_{\delta_s})$ -SVD Initialization. Considering that the vanilla initialization in LoRA results in slow convergence and suboptimal fine-tuning (Zhang et al., 2025), we, thereby, investigate the potential benefits of incorporating our knowledge-injection strategy into LoRA-based methods. Specifically, instead of initializing the low-rank trainable matrices A and B in LoRA with the Gaussian distribution and zeros, we initialize them using the SVD of the injected knowledgeable parameters (*i.e.*, $\phi(\mathbf{W}_{\delta_s})$). Afterwards, we observe the training loss trajectories of FPFT, LoRA, and the LoRA initializing with $\phi(\mathbf{W}_{\delta_s})$ -SVD (denoted as KJ Loss in Figure 5 (c)). Following the experimental settings and implementation in (Pizarroso et al., 2022), we adopt a two-layer neural network pretrained on 10,000 odd-labeled MNIST samples (LeCun, 1998) and then fine-tuned on 1,000

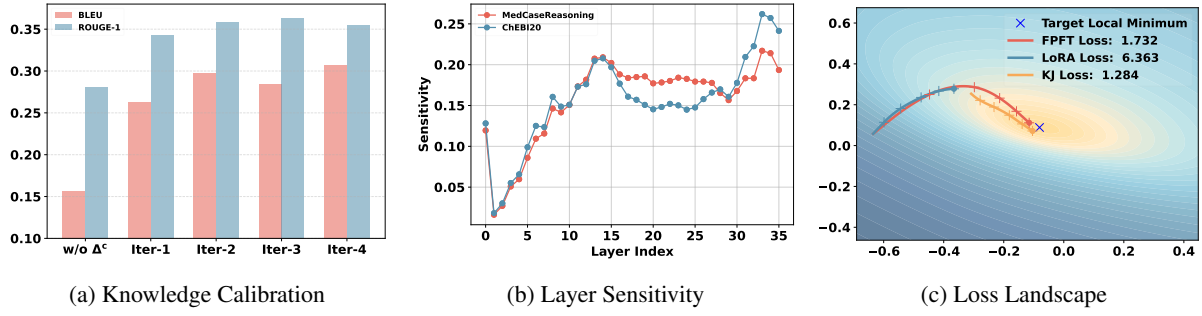


Figure 5: (a) Performance of our method with and without knowledge calibration (w/o Δ^c), evaluated on *MedCaseReasoning* datasets using Qwen2.5-3B. After incorporating knowledge calibration, our method exhibits substantial gains. Besides, performance variation across different calibration iterations is minor, suggesting that a moderate number of iterations (e.g., $T = 2$ or 3) is sufficient. (b) Layer sensitivity scores of Qwen2.5-3B on *MedCaseReasoning* and *ChEBI20* datasets. In general, the most influential layers are concentrated toward the deeper part of the network, while a few mid-level layers also display notable contributions. (c) Instead of initializing low-rank matrices B and A with zero and a Gaussian distribution in LoRA (Hu et al., 2021), we initialize them via the SVD of the injected knowledgeable parameters (denoted as KJ). We observe the loss trajectories among FPFT, LoRA, and KJ under gradient descent. KJ presents a better initialization point and achieves a lower loss with the same training steps, demonstrating the great potential of knowledge injection initialization in LoRA-based methods.

even-labeled samples. All methods share the same learning rate of 5×10^{-4} . As illustrated in Figure 5 (c), compared to FPFT and LoRA, initializing with the parameters obtained via knowledge injection achieves a more favorable starting point, exhibits faster convergence, and reaches a lower loss under the same optimization steps. This result highlights the strong potential of our method in LoRA-based methods enhancement, as the parameters yielded by knowledge injection provide richer prior information for more effective initialization.

5 Conclusion

We propose a knowledge injection method, enhancing LLMs’ performance in specific domains under constraints of computational resources and raw data sharing. Specifically, our method allows users to fine-tune small language models on domain-specific data, thereby mitigating substantial computational overhead. The fine-tuned parameters are subsequently shared with developers. We derive a closed-form bilinear matrix transformation, thereby enabling the domain knowledge from the fine-tuned small language model to be efficiently injected into an LLM of any scale. An iterative knowledge calibration strategy is further proposed to alleviate information loss during injection. We theoretically prove the error bound for the knowledge injection, as well as the stability and convergence of the calibration. Extensive experiments demonstrate that our method achieves performance comparable to FPFT while using only 28% GPU memory, and

consistently and substantially outperforms LoRA-based approaches on medical, chemical, and coding benchmarks. In-depth analysis further illustrates the potential benefits of knowledge injection initialization in LoRA-based methods.

Limitations

While our knowledge injection framework reduces computational overhead and raw-data exposure, several limitations remain. First, it relies on a small calibration set, which may be infeasible under strict no-data-usage constraints. Besides, although raw data are not shared, the transferred weights lack formal privacy guarantees, leaving potential risks such as membership inference, model inversion, and gradient leakage unexplored. Finally, our evaluation primarily uses surface-level metrics (BLEU and ROUGE), which may not fully capture factual consistency and higher-order reasoning. Finally, computational constraints limit experiments on very large datasets and target LLMs, while the effectiveness of the closed-form bilinear transformation across highly heterogeneous architectures remains unvalidated.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62272349); Natural Science Foundation of Hubei Province under Grant Numbers 2023BAB160; Chenliang Li is the corresponding author.

References

- Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.
- Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, and 1 others. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. 2025. Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3737–3746.
- Massimo Bini, Leander Gierbach, and Zeynep Akata. 2025. Delora: Decoupling angles and strength in low-rank adaptation. *arXiv preprint arXiv:2503.18225*.
- Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *International conference on machine learning*, pages 1059–1071. PMLR.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in neural information processing systems*, volume 36, pages 10088–10115.
- Kennedy Edemacu and Xintao Wu. 2025. Privacy preserving prompt engineering: A survey. *ACM Computing Surveys*, 57(10):1–36.
- Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 375–413.
- Carl Edwards, ChengXiang Zhai, and Heng Ji. 2021. Text2mol: Cross-modal molecule retrieval with natural language queries. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 595–607.
- Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.
- Gene Golub and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, and 1 others. 2023. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. 2023. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11825–11835.
- Roger A Horn and Charles R Johnson. 2012. *Matrix analysis*. Cambridge university press.
- 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 *International conference on machine learning*, pages 2790–2799. PMLR.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. 2025. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*.
- Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674.
- Peter Kairouz and H Brendan McMahan. 2021. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210.

- Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.
- Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.
- Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5260–5271.
- Yann LeCun. 1998. [The mnist database of handwritten digits](#). Online.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.
- Vladislav Lialin, Sherin Muckatira, Namrata Shiva-gunde, and Anna Rumshisky. 2024. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, volume 2024, pages 49405–49421.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32100–32121.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.
- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr.
- Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.
- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503.
- Jaime Pizarroso, José Portela, and Antonio Muñoz. 2022. Neursens: sensitivity analysis of neural networks. *Journal of Statistical Software*, 102:1–36.
- Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. 2023. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362.
- Zeju Qiu, Weiyang Liu, Adrian Weller, and Bernhard Schölkopf. 2025. Orthogonal finetuning made scalable. *arXiv preprint arXiv:2506.19847*.
- A Saxe, J McClelland, and S Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations 2014*. International Conference on Learning Representations 2014.
- Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning, ICML’25*. PMLR.
- Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Xun Wang, Jing Xu, Franziska Boenisch, Michael Backes, Christopher A Choquette-Choo, and Adam Dziedzic. 2025. Efficient and privacy-preserving soft prompt transfer for llms. *arXiv preprint arXiv:2506.16196*.
- Kevin Wu, Eric Wu, Rahul Thapa, Kevin Wei, Angela Zhang, Arvind Suresh, Jacqueline J Tao, Min Woo Sun, Alejandro Lozano, and James Zou. 2025. Medcasereasoning: Evaluating and learning diagnostic reasoning from clinical case reports. *arXiv preprint arXiv:2505.11733*.
- Wenhan Xia, Chengwei Qin, and Elad Hazan. 2024. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*.
- Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

- Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, and 1 others. 2021. Opacus: User-friendly differential privacy library in pytorch. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.
- Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. 2023. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pages 40770–40803. PMLR.
- Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*.
- Yuanhe Zhang, Fanghui Liu, and Yudong Chen. 2025. Lora-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. *arXiv preprint arXiv:2502.01235*.
- Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Yi Xia, Hongshen Xu, Zichen Zhu, Su Zhu, Shuai Fan, and 1 others. 2024. Chemdfm: A large language foundation model for chemistry. In *Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges*.
- Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhui Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12834–12859.

A Theoretical Analysis

A.1 On the Invertibility of the Knowledge Transfer Matrix

Lemma A.1. *Let $\mathbf{W} \in \mathbb{R}^{m \times n}$ denote the weight matrix of a linear transformation in an LLM, where $m \leq n$. Under the near-orthogonality assumption of the weight vectors, we demonstrate that the Gram matrix $\mathbf{W}\mathbf{W}^\top \in \mathbb{R}^{m \times m}$ is full rank and therefore invertible.*

Proof. Let $\mathbf{W}^\top = [w_1, w_2, \dots, w_m]$, where $w_i \in \mathbb{R}^n$ are the row vectors. The matrix $\mathbf{G}$ is formalized as:

$$\mathbf{G} = \mathbf{W}\mathbf{W}^\top = \begin{pmatrix} w_1^\top w_1 & w_1^\top w_2 & \dots & w_1^\top w_m \\ w_2^\top w_1 & w_2^\top w_2 & \dots & w_2^\top w_m \\ \vdots & \vdots & \ddots & \vdots \\ w_m^\top w_1 & w_m^\top w_2 & \dots & w_m^\top w_m \end{pmatrix}. \quad (21)$$

Due to large-scale isotropic initialization (Saxe et al., 2014) and self-normalizing optimization of LLMs, both empirical observations and theoretical analyses have revealed that weight matrices in LLMs exhibit near-orthogonal structures (Brock et al., 2021; Zhai et al., 2023). Accordingly, the **near-orthogonality** assumption is formalized as bounded off-diagonal terms relative to the diagonal terms:

$$|w_i^\top w_j| \leq \epsilon, \quad \forall i \neq j, \quad (22)$$

where $\epsilon \ll 1$ is a small coupling constant, and the norms are bounded from below:

$$\|w_i\|_2^2 = w_i^\top w_i \geq \sigma_0^2 > 0. \quad (23)$$

We decompose the Gram matrix $\mathbf{G}$ into a diagonal component $\mathbf{D}$ and a perturbation component $\mathbf{P}$ as follows:

$$\mathbf{G} = \mathbf{D} + \mathbf{P} \quad (24)$$

where $\mathbf{D} = \text{diag}(\|w_1\|_2^2, \|w_2\|_2^2, \dots, \|w_m\|_2^2)$, $\mathbf{P} = \mathbf{G} - \mathbf{D}$, containing the off-diagonal terms.

Given $\mathbf{G}$ is Symmetric Positive Semi-Definite (SPSD), its eigenvalues are real and non-negative. Based on the **Weyl's inequality** (Horn and Johnson, 2012) for eigenvalues, we can obtain a bound on the smallest eigenvalue:

$$\lambda_{\min}(\mathbf{G}) \geq \lambda_{\min}(\mathbf{D}) + \lambda_{\min}(\mathbf{P}). \quad (25)$$

As $\mathbf{D}$ is diagonal, its smallest eigenvalue is:

$$\lambda_{\min}(\mathbf{D}) = \min_i \|w_i\|_2^2 \geq \sigma_0^2. \quad (26)$$

Following the **Gershgorin Circle Theorem**, we further consider the matrix norm bound related to the perturbation $\mathbf{P}$:

$$\lambda_{\min}(\mathbf{G}) \geq \lambda_{\min}(\mathbf{D}) - \|\mathbf{P}\|_2, \quad (27)$$

where $\|\mathbf{P}\|_2$ is the spectral norm of the perturbation matrix $\mathbf{P}$. By definition, the ∞ -norm of $\mathbf{P}$ is bounded:

$$\|\mathbf{P}\|_\infty = \max_i \sum_{j=1}^m |\mathbf{P}_{ij}| = \max_i \sum_{j \neq i} |w_i^\top w_j|. \quad (28)$$

Considering the near-orthogonality condition, $|w_i^\top w_j| \leq \epsilon$, we have:

$$\|\mathbf{P}\|_\infty \leq (m-1)\epsilon. \quad (29)$$

Since $\mathbf{P}$ is a symmetric matrix, its spectral norm $\|\mathbf{P}\|_2$ is equal to its largest absolute eigenvalue $|\lambda_{\max}|$ and we have $\|\mathbf{P}\|_1 = \|\mathbf{P}\|_\infty$. Considering a fundamental relationship, i.e., $\|\mathbf{P}\|_2 \leq \sqrt{\|\mathbf{P}\|_1 \|\mathbf{P}\|_\infty}$, we have:

$$\|\mathbf{P}\|_2 \leq \|\mathbf{P}\|_\infty \leq (m-1)\epsilon. \quad (30)$$

Substituting the bounds (Eq. (30)) back into the perturbation inequality (Eq. (27)), we have:

$$\lambda_{\min}(\mathbf{G}) \geq \lambda_{\min}(\mathbf{D}) - (m-1)\epsilon \geq \sigma_0^2 - (m-1)\epsilon. \quad (31)$$

As weights in LLMs are typically initialized isotropically ($\sigma_0^2 > 0$ and non-zero norms) and optimization maintains near-orthogonality (implying a sufficiently small ϵ), in practice, the near-orthogonality constant ϵ satisfies:

$$\epsilon < \frac{\sigma_0^2}{m-1}. \quad (32)$$

Therefore, $\lambda_{\min}(\mathbf{G}) > 0$, which proves that the Gram matrix $\mathbf{G} = \mathbf{W}\mathbf{W}^\top$ is Positive Definite (PD), hence **full rank** (rank m) and **invertible**. We could alternatively derive the closed-form formulation of the transfer matrix by fixing one from the other. $\square$

A.2 Closed-Form of the Knowledge Transfer Matrix.

Lemma A.2. Denote $\mathbf{W}_s \in \mathbb{R}^{m \times n}$ as any weight from the source SLM, $\mathbf{W}_g \in \mathbb{R}^{p \times q}$ is the corresponding weight from the target LLM for knowledge injection, where $m < p$ and $n < q$. We define $\mathbf{W}_\ell \in \mathbb{R}^{m \times p}$ and $\mathbf{W}_r \in \mathbb{R}^{n \times q}$ as the left and right knowledge transfer matrices, respectively, such that

$$\mathbf{W}_g = \mathbf{W}_\ell^\top \mathbf{W}_s \mathbf{W}_r. \quad (33)$$

We demonstrate that the transfer matrices $\mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{I}_{n \times q}^\top \mathbf{W}_s^\top (\mathbf{W}_s \mathbf{W}_s^\top)^\dagger$ and $\mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{I}_{m \times p} \mathbf{W}_g$.

Proof. We consider the bilinear equation for knowledge transfer:

$$\mathbf{W}_g = \mathbf{W}_\ell^\top \mathbf{W}_s \mathbf{W}_r, \quad (34)$$

where $\mathbf{W}_s \in \mathbb{R}^{m \times n}$ and $\mathbf{W}_g \in \mathbb{R}^{p \times q}$ are known source and target weight matrices, respectively. The transfer matrices $\mathbf{W}_\ell \in \mathbb{R}^{m \times p}$ and $\mathbf{W}_r \in \mathbb{R}^{n \times q}$ are the unknown variables to be solved.

Given that Eq. (34) represents a non-convex, underdetermined system, a direct analytical solution for both $\mathbf{W}_\ell$ and $\mathbf{W}_r$ simultaneously is intractable. Accordingly, rather than deriving a closed-form joint solution, we formulate two distinct least-squares problems, each derived under a specific constraint.

Specifically, we re-frame this problem by recasting the relation as a Frobenius-norm minimization:

$$\min_{\mathbf{W}_\ell, \mathbf{W}_r} \mathcal{L}(\mathbf{W}_\ell, \mathbf{W}_r) = \left\| \mathbf{W}_g - \mathbf{W}_\ell^\top \mathbf{W}_s \mathbf{W}_r \right\|_F^2. \quad (35)$$

We then apply an **Alternating Least Squares (ALS)** approach, *i.e.*, optimize one variable while holding the other fixed. Each sub-problem is convex, yielding a unique least-squares solution. The two closed-form expressions in Lemma A.2 can thus be interpreted as the first iteration of ALS under different initialization constraints.

1. Derivation of $\mathbf{W}_r$ (Right Transfer Matrix)

First, we solve for $\mathbf{W}_r$ by fixing $\mathbf{W}_\ell$. The optimization sub-problem becomes:

$$\min_{\mathbf{W}_r} \left\| \mathbf{W}_g - (\mathbf{W}_\ell^\top \mathbf{W}_s) \mathbf{W}_r \right\|_F^2. \quad (36)$$

We thereby define the constant matrix $\mathbf{C} = \mathbf{W}_\ell^\top \mathbf{W}_s$. The dimensions are $\mathbf{C} \in \mathbb{R}^{p \times n}$. Thus, the problem reduces to a standard linear least-squares formulation:

$$\min_{\mathbf{W}_r} \left\| \mathbf{W}_g - \mathbf{C} \mathbf{W}_r \right\|_F^2. \quad (37)$$

To find the minimum, we compute the gradient of the loss $\mathcal{L}(\mathbf{W}_r) = \text{Tr}((\mathbf{W}_g - \mathbf{C}\mathbf{W}_r)^\top(\mathbf{W}_g - \mathbf{C}\mathbf{W}_r))$ with respect to $\mathbf{W}_r$ and set it to zero, we have:

$$\mathcal{L}(\mathbf{W}_r) = \text{Tr}((\mathbf{W}_g - \mathbf{C}\mathbf{W}_r)^\top(\mathbf{W}_g - \mathbf{C}\mathbf{W}_r)) \quad (38)$$

$$= \text{Tr}(\mathbf{W}_g^\top \mathbf{W}_g - \mathbf{W}_g^\top \mathbf{C}\mathbf{W}_r - \mathbf{W}_r^\top \mathbf{C}^\top \mathbf{W}_g + \mathbf{W}_r^\top \mathbf{C}^\top \mathbf{C}\mathbf{W}_r). \quad (39)$$

Using the cyclic property of the matrix trace ($\text{Tr}(AB) = \text{Tr}(BA)$), we have $\text{Tr}(\mathbf{W}_g^\top \mathbf{C}\mathbf{W}_r) = \text{Tr}(\mathbf{W}_r^\top \mathbf{C}^\top \mathbf{W}_g)$. Therefore,

$$\mathcal{L}(\mathbf{W}_r) = \text{Tr}(\mathbf{W}_g^\top \mathbf{W}_g) - 2\text{Tr}(\mathbf{W}_g^\top \mathbf{C}\mathbf{W}_r) + \text{Tr}(\mathbf{W}_r^\top \mathbf{C}^\top \mathbf{C}\mathbf{W}_r). \quad (40)$$

Given the matrix differentiation rules $\frac{\partial \text{Tr}(\mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = \mathbf{A}^\top$ and $\frac{\partial \text{Tr}(\mathbf{X}^\top \mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{X}$. Take the partial derivative of $\mathcal{L}(\mathbf{W}_r)$ with respect to $\mathbf{W}_r$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_r} = 0 - 2\frac{\partial \text{Tr}(\mathbf{W}_g^\top \mathbf{C}\mathbf{W}_r)}{\partial \mathbf{W}_r} + \frac{\partial \text{Tr}(\mathbf{W}_r^\top (\mathbf{C}^\top \mathbf{C})\mathbf{W}_r)}{\partial \mathbf{W}_r} \quad (41)$$

$$= -2(\mathbf{W}_g^\top \mathbf{C})^\top + ((\mathbf{C}^\top \mathbf{C}) + (\mathbf{C}^\top \mathbf{C})^\top) \mathbf{W}_r. \quad (42)$$

Since $\mathbf{C}^\top \mathbf{C}$ is itself a symmetric matrix, $(\mathbf{C}^\top \mathbf{C})^\top = \mathbf{C}^\top \mathbf{C}$, we have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_r} = -2\mathbf{C}^\top \mathbf{W}_g + 2(\mathbf{C}^\top \mathbf{C})\mathbf{W}_r. \quad (43)$$

Set the partial derivative to the zero matrix $\mathbf{0}$ to find the extremum:

$$2(\mathbf{C}^\top \mathbf{C})\mathbf{W}_r = 2\mathbf{C}^\top \mathbf{W}_g, \mathbf{W}_r = (\mathbf{C}^\top \mathbf{C})^\dagger \mathbf{C}^\top \mathbf{W}_g. \quad (44)$$

This is precisely $\mathbf{W}_r = \mathbf{C}^\dagger \mathbf{W}_g$, where $\mathbf{C}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{C}$.

We further impose $\mathbf{W}_\ell = \mathbf{I}_{m \times p}$. Note $\mathbf{W}_\ell \in \mathbb{R}^{m \times p}$, thus $\mathbf{W}_\ell^\top = \mathbf{I}_{m \times p}^\top = \mathbf{I}_{p \times m}$. We substitute this constraint into $\mathbf{C} = \mathbf{W}_\ell^\top \mathbf{W}_s$:

$$\mathbf{C} = \mathbf{I}_{p \times m} \mathbf{W}_s. \quad (45)$$

Therefore, $\mathbf{C}^\top \mathbf{C}$ and $\mathbf{C}^\top$ can be obtained:

$$\mathbf{C}^\top \mathbf{C} = (\mathbf{I}_{p \times m} \mathbf{W}_s)^\top (\mathbf{I}_{p \times m} \mathbf{W}_s) = \mathbf{W}_s^\top \mathbf{I}_{p \times m}^\top \mathbf{I}_{p \times m} \mathbf{W}_s = \mathbf{W}_s^\top \mathbf{I}_{m \times p} \mathbf{I}_{p \times m} \mathbf{W}_s. \quad (46)$$

Supposing $m < p$, $\mathbf{I}_{m \times p} \mathbf{I}_{p \times m} = \mathbf{I}_m$ (an $m \times m$ identity matrix), we have:

$$\mathbf{C}^\top \mathbf{C} = \mathbf{W}_s^\top \mathbf{I}_m \mathbf{W}_s = \mathbf{W}_s^\top \mathbf{W}_s, \quad (47)$$

$$\mathbf{C}^\top = (\mathbf{I}_{p \times m} \mathbf{W}_s)^\top = \mathbf{W}_s^\top \mathbf{I}_{p \times m}^\top = \mathbf{W}_s^\top \mathbf{I}_{m \times p}. \quad (48)$$

Substituting these results into $\mathbf{W}_r = (\mathbf{C}^\top \mathbf{C})^\dagger \mathbf{C}^\top \mathbf{W}_g$:

$$\boxed{\mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{I}_{m \times p} \mathbf{W}_g}. \quad (49)$$

2. Derivation of $\mathbf{W}_\ell$ (Left Transfer Matrix)

Now, we fix $\mathbf{W}_r$ and solve $\min_{\mathbf{W}_\ell} \mathcal{L}(\mathbf{W}_\ell, \mathbf{W}_r)$. To simplify notation, we define $\mathbf{B} = \mathbf{W}_s \mathbf{W}_r \in \mathbb{R}^{m \times q}$. The subproblem becomes:

$$\min_{\mathbf{W}_\ell} \left\| \mathbf{W}_g - \mathbf{W}_\ell^\top \mathbf{B} \right\|_F^2. \quad (50)$$

Considering this is a *right-multiply* form of the least-squares problem, we again expand the loss function:

$$\mathcal{L}(\mathbf{W}_\ell^\top) = \text{Tr}\left((\mathbf{W}_g - \mathbf{W}_\ell^\top \mathbf{B})^\top (\mathbf{W}_g - \mathbf{W}_\ell^\top \mathbf{B})\right) \quad (51)$$

$$= \text{Tr}\left(\mathbf{W}_g^\top \mathbf{W}_g - \mathbf{W}_g^\top \mathbf{W}_\ell^\top \mathbf{B} - \mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top \mathbf{W}_g + \mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top \mathbf{W}_\ell^\top \mathbf{B}\right). \quad (52)$$

Using the trace properties $\text{Tr}(\mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top \mathbf{W}_g) = \text{Tr}(\mathbf{W}_g^\top \mathbf{W}_\ell^\top \mathbf{B})$ and $\text{Tr}(\mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top \mathbf{W}_\ell^\top \mathbf{B}) = \text{Tr}(\mathbf{W}_\ell^\top \mathbf{B} \mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top)$. We have:

$$\mathcal{L}(\mathbf{W}_\ell^\top) = \text{Tr}(\mathbf{W}_g^\top \mathbf{W}_g) - 2\text{Tr}(\mathbf{W}_g^\top \mathbf{W}_\ell^\top \mathbf{B}) + \text{Tr}(\mathbf{W}_\ell^\top \mathbf{B} \mathbf{B}^\top (\mathbf{W}_\ell^\top)^\top). \quad (53)$$

We take the partial derivative with respect to $\mathbf{W}_\ell^\top$. Using $\frac{\partial \text{Tr}(\mathbf{C}\mathbf{X}\mathbf{D})}{\partial \mathbf{X}} = \mathbf{C}^\top \mathbf{D}^\top$ and $\frac{\partial \text{Tr}(\mathbf{X}\mathbf{C}\mathbf{X}^\top)}{\partial \mathbf{X}} = \mathbf{X}(\mathbf{C} + \mathbf{C}^\top)$, we have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\ell^\top} = 0 - 2 \frac{\partial \text{Tr}(\mathbf{W}_g^\top \mathbf{W}_\ell^\top \mathbf{B})}{\partial \mathbf{W}_\ell^\top} + \frac{\partial \text{Tr}(\mathbf{W}_\ell^\top (\mathbf{B} \mathbf{B}^\top) (\mathbf{W}_\ell^\top)^\top)}{\partial \mathbf{W}_\ell^\top} \quad (54)$$

$$= -2(\mathbf{W}_g^\top)^\top \mathbf{B}^\top + \mathbf{W}_\ell^\top \left((\mathbf{B} \mathbf{B}^\top) + (\mathbf{B} \mathbf{B}^\top)^\top \right). \quad (55)$$

Given $\mathbf{B} \mathbf{B}^\top$ is a symmetric matrix, we have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\ell^\top} = -2\mathbf{W}_g \mathbf{B}^\top + 2\mathbf{W}_\ell^\top (\mathbf{B} \mathbf{B}^\top). \quad (56)$$

Set the partial derivative to the zero matrix $\mathbf{0}$:

$$2\mathbf{W}_\ell^\top (\mathbf{B} \mathbf{B}^\top) = 2\mathbf{W}_g \mathbf{B}^\top, \quad (57)$$

$$\mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{B}^\top (\mathbf{B} \mathbf{B}^\top)^\dagger. \quad (58)$$

This is precisely the right pseudoinverse $\mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{B}^\dagger$. Again, let $\mathbf{W}_r = \mathbf{I}_{n \times q}$. Note $\mathbf{W}_r \in \mathbb{R}^{n \times q}$. We substitute this set into $\mathbf{B} = \mathbf{W}_s \mathbf{W}_r$:

$$\mathbf{B} = \mathbf{W}_s \mathbf{I}_{n \times q}. \quad (59)$$

Therefore, $\mathbf{B}^\top$ and $\mathbf{B} \mathbf{B}^\top$ can be delivered:

$$\mathbf{B}^\top = (\mathbf{W}_s \mathbf{I}_{n \times q})^\top = \mathbf{I}_{n \times q}^\top \mathbf{W}_s^\top = \mathbf{I}_{q \times n} \mathbf{W}_s^\top, \quad (60)$$

$$\mathbf{B} \mathbf{B}^\top = (\mathbf{W}_s \mathbf{I}_{n \times q})(\mathbf{W}_s \mathbf{I}_{n \times q})^\top = \mathbf{W}_s \mathbf{I}_{n \times q} \mathbf{I}_{q \times n} \mathbf{W}_s^\top. \quad (61)$$

Assuming $n < q$, $\mathbf{I}_{n \times q} \mathbf{I}_{q \times n} = \mathbf{I}_n$ (an $n \times n$ identity matrix), we have:

$$\mathbf{B} \mathbf{B}^\top = \mathbf{W}_s \mathbf{I}_n \mathbf{W}_s^\top = \mathbf{W}_s \mathbf{W}_s^\top. \quad (62)$$

Substituting these results into $\mathbf{W}_\ell^\top = \mathbf{W}_g \mathbf{B}^\top (\mathbf{B} \mathbf{B}^\top)^\dagger$:

$$\boxed{\mathbf{W}_\ell^\top = \mathbf{W}_g (\mathbf{I}_{q \times n} \mathbf{W}_s^\top) (\mathbf{W}_s \mathbf{W}_s^\top)^\dagger = \mathbf{W}_g \mathbf{I}_{n \times q}^\top \mathbf{W}_s^\top (\mathbf{W}_s \mathbf{W}_s^\top)^\dagger}. \quad (63)$$

□

A.3 Error Bound of Knowledge Transformation

Lemma A.3. Let $\mathbf{W}_s$ and $\mathbf{W}_g$ denote the base weight matrices of the source SLM and target LLM, respectively. Under the simplifying assumption that the left transfer matrix $\mathbf{W}_\ell = \mathbf{I}$, the transfer matrix $\mathbf{W}_r$ in Eq. (10) is derived via the pseudo-inverse such that $\mathbf{W}_r = (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{W}_g$. The

Frobenius norm of the projection error, defined as $\mathbf{E}_{proj} = \mathbf{W}_g - \mathbf{W}_s \mathbf{W}_r$, is strictly bounded by the spectral energy of the target weights $\mathbf{W}_g$, satisfying:

$$\|\mathbf{E}_{proj}\|_F^2 \leq \sum_{i=1}^r \sigma_i^2, \quad (64)$$

where σ_i represents the i -th singular value of $\mathbf{W}_g$, and r is the rank.

Proof. Given the derivation of the transfer matrix $\mathbf{W}_r$, we can define the approximated target weights as $\tilde{\mathbf{W}}_g^{\text{approx}} = \mathbf{W}_s \mathbf{W}_r$. Substituting $\mathbf{W}_r$ into this equation yields:

$$\tilde{\mathbf{W}}_g^{\text{approx}} = \mathbf{W}_s (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{W}_g \quad (65)$$

The projection error $\mathbf{E}_{proj}$ evaluates the discrepancy between the true target weights and the approximated target weights. This can be formulated as:

$$\mathbf{E}_{proj} = \mathbf{W}_g - \tilde{\mathbf{W}}_g^{\text{approx}} = \mathbf{W}_g - \mathbf{W}_s (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top \mathbf{W}_g. \quad (66)$$

By factoring out $\mathbf{W}_g$, we obtain:

$$\mathbf{E}_{proj} = (\mathbf{I} - \mathbf{W}_s (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top) \mathbf{W}_g. \quad (67)$$

The term $(\mathbf{I} - \mathbf{W}_s (\mathbf{W}_s^\top \mathbf{W}_s)^\dagger \mathbf{W}_s^\top)$ acts as an orthogonal projector onto the left null space of $\mathbf{W}_s$. By definition, the spectral norm of an orthogonal projector is exactly 1. Applying this property to the Frobenius norm of the error yields our first bound:

$$\|\mathbf{E}_{proj}\|_F \leq \|\mathbf{W}_g\|_F. \quad (68)$$

To explicitly express this error constraint through the spectral energy of the pre-trained weights, we apply Singular Value Decomposition (SVD) to the target weights, such that $\mathbf{W}_g = \mathbf{U}_g \mathbf{\Sigma}_g \mathbf{V}_g^\top$. Leveraging the orthogonal invariance of the Frobenius norm, we establish:

$$\|\mathbf{W}_g\|_F^2 = \|\mathbf{U}_g \mathbf{\Sigma}_g\|_F^2. \quad (69)$$

Expanding the Frobenius norm using the trace operator $\text{Tr}(\cdot)$, we derive the final bound:

$$\|\mathbf{E}_{proj}\|_F^2 \leq \text{Tr}((\mathbf{U}_g \mathbf{\Sigma}_g)^\top (\mathbf{U}_g \mathbf{\Sigma}_g)) = \text{Tr}(\mathbf{\Sigma}_g^\top \mathbf{\Sigma}_g) = \sum_{i=1}^r \sigma_i^2. \quad (70)$$

□

A.4 Knowledge Calibration Stability Convergence and Error Analysis

Let $\hat{\mathbf{w}}_t$ and $\mathbf{w}_t^*$ as the estimated calibration (*i.e.*, $\hat{\mathbf{w}}_t = \tilde{\mathbf{W}}_{\Delta^c}^{t-1}$) and the expected ground-truth. we define the **a posteriori estimation error** as:

$$\mathbf{e}_t = \hat{\mathbf{w}}_t - \mathbf{w}_t^* \quad (71)$$

$$= [\hat{\mathbf{w}}_{t-1} + \mathbf{F}_t(\mathbf{z}_t - \hat{\mathbf{w}}_{t-1})] - [\mathbf{w}_{t-1}^* + \boldsymbol{\omega}_{t-1}] \quad (72)$$

$$= \hat{\mathbf{w}}_{t-1} - \mathbf{w}_{t-1}^* - \boldsymbol{\omega}_{t-1} + \mathbf{F}_t(\mathbf{w}_{t-1}^* + \boldsymbol{\nu}_t - \hat{\mathbf{w}}_{t-1}) \quad (73)$$

$$= \mathbf{e}_{t-1} - \boldsymbol{\omega}_{t-1} + \mathbf{F}_t([\mathbf{w}_{t-1}^* + \boldsymbol{\omega}_{t-1}] + \boldsymbol{\nu}_t - [\mathbf{e}_{t-1} + \mathbf{w}_{t-1}^*]) \quad (74)$$

$$= \mathbf{e}_{t-1} - \boldsymbol{\omega}_{t-1} + \mathbf{F}_t(\boldsymbol{\omega}_{t-1} + \boldsymbol{\nu}_t - \mathbf{e}_{t-1}) \quad (75)$$

$$= (\mathbf{I} - \mathbf{F}_t)\mathbf{e}_{t-1} + (\mathbf{F}_t - \mathbf{I})\boldsymbol{\omega}_{t-1} + \mathbf{F}_t\boldsymbol{\nu}_t, \quad (76)$$

where $\mathbf{w}_t^* = \mathbf{w}_{t-1}^* + \boldsymbol{\omega}_{t-1}$ and $\mathbf{z}_t = \mathbf{w}_t^* + \boldsymbol{\nu}_t$. $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$ represent the noise from knowledge injection and calibration update, respectively. $\boldsymbol{\omega}_{t-1} \sim \mathcal{N}(0, \boldsymbol{\xi})$, $\boldsymbol{\xi} \succeq 0$ and $\boldsymbol{\nu}_t \sim \mathcal{N}(0, \boldsymbol{\zeta})$, $\boldsymbol{\zeta} \succeq 0$.

1. Stability Analysis To analyze the stability, we observe the boundary of $\mathbf{e}_t$, which depends on the contraction properties of the homogeneous term $\mathbf{I} - \mathbf{F}_t$. We define:

$$\mathbf{M}_t = \mathbf{I} - \mathbf{F}_t = \mathbf{I} - \mathbf{P}_{t-1}(\mathbf{P}_{t-1} + \boldsymbol{\zeta})^{-1}. \quad (77)$$

Given the matrix identity

$$\mathbf{I} - \mathbf{A}(\mathbf{A} + \mathbf{B})^{-1} = \mathbf{B}(\mathbf{A} + \mathbf{B})^{-1}, \quad (78)$$

we obtain

$$\mathbf{M}_t = \boldsymbol{\zeta}(\mathbf{P}_{t-1} + \boldsymbol{\zeta})^{-1}. \quad (79)$$

Lemma A.4. Stability: *Given the initial covariance of calibration matrix $\mathbf{P}_0 \succeq 0$ and the process noises in knowledge injection and calibration $\boldsymbol{\xi} \succeq 0$, $\boldsymbol{\zeta} \succ 0$. Then, the posterior estimation error $\mathbf{e}_t$ is stable and remains statistically bounded.*

Proof. The covariance update is

$$\mathbf{P}_t = \mathbf{M}_t(\mathbf{P}_{t-1} + \boldsymbol{\xi}). \quad (80)$$

Considering $\boldsymbol{\zeta} \succ 0$ and $\mathbf{P}_{t-1} \succeq 0$ (as $\mathbf{P}_0 \succeq 0$ and the update preserves positive semi-definiteness), it follows that $(\mathbf{P}_{t-1} + \boldsymbol{\zeta}) \succ 0$ and is invertible. For any eigenvalue λ_M of $\mathbf{M}_t$:

$$\lambda_M = \lambda_i(\boldsymbol{\zeta}(\mathbf{P}_{t-1} + \boldsymbol{\zeta})^{-1}). \quad (81)$$

As both $\boldsymbol{\zeta}$ and $\mathbf{P}_{t-1}$ are symmetric positive semi-definite (SPSD), we have $\lambda_i(\mathbf{P}_{t-1}) \geq 0$, and

$$\lambda_i(\mathbf{P}_{t-1} + \boldsymbol{\zeta}) = \lambda_i(\mathbf{P}_{t-1}) + \lambda_i(\boldsymbol{\zeta}). \quad (82)$$

Thus,

$$\lambda_M = \frac{\lambda_i(\boldsymbol{\zeta})}{\lambda_i(\mathbf{P}_{t-1}) + \lambda_i(\boldsymbol{\zeta})}. \quad (83)$$

Given $\boldsymbol{\zeta} \succ 0$, we have $\lambda_i(\boldsymbol{\zeta}) > 0$, thus, $\lambda_i(\mathbf{P}_{t-1}) \geq 0$ and

$$0 < \lambda_M \leq 1. \quad (84)$$

The spectral radius $\rho(\mathbf{M}_t) \leq 1$. For finite iterations, $\rho(\mathbf{M}_t) < 1$ almost always holds, ensuring bounded error. Therefore, the system is **BIBO-stable** (bounded input–bounded output). $\square$

2. Convergence of Internal Covariance $\mathbf{P}_t$

The iterative equation is

$$\mathbf{P}_t = \boldsymbol{\zeta}(\mathbf{P}_{t-1} + \boldsymbol{\zeta})^{-1}(\mathbf{P}_{t-1} + \boldsymbol{\xi}). \quad (85)$$

Assume a steady-state solution $\mathbf{P}^* = \lim_{t \rightarrow \infty} \mathbf{P}_t$. It satisfies the **non-standard algebraic Riccati equation**:

$$\mathbf{P}^* = \boldsymbol{\zeta}(\mathbf{P}^* + \boldsymbol{\zeta})^{-1}(\mathbf{P}^* + \boldsymbol{\xi}), \quad (86)$$

$$\mathbf{P}^*(\mathbf{P}^* + \boldsymbol{\zeta}) = \boldsymbol{\zeta}(\mathbf{P}^* + \boldsymbol{\xi}), \quad (87)$$

$$(\mathbf{P}^*)^2 + \mathbf{P}^*\boldsymbol{\zeta} = \boldsymbol{\zeta}\mathbf{P}^* + \boldsymbol{\zeta}\boldsymbol{\xi}. \quad (88)$$

Let $\boldsymbol{\xi} = \xi\mathbf{I}$, $\boldsymbol{\zeta} = \zeta\mathbf{I}$, and $\mathbf{P}_t = p_t\mathbf{I}$. Then

$$(p^*)^2 = \zeta\xi \implies \boxed{p^* = \sqrt{\zeta\xi}}. \quad (89)$$

The internal covariance estimate $\mathbf{P}_t$ converges to $\mathbf{P}^*$, the geometric mean of the knowledge injection and calibration noise covariances.

3. Stability True Error Covariance

Let $\mathbf{F}^* = \lim_{t \rightarrow \infty} \mathbf{F}_t = \mathbf{P}^*(\mathbf{P}^* + \zeta)^{-1}$, considering scalar case, we have:

$$f^* = \frac{p^*}{p^* + \zeta} = \frac{\sqrt{\xi}}{\sqrt{\xi} + \sqrt{\zeta}}. \quad (90)$$

At steady state, the true error covariance $\mathbf{C}^* = \lim_{t \rightarrow \infty} \mathbf{C}_t$ satisfies a discrete-time **Lyapunov equation**:

$$\mathbf{C}^* = (\mathbf{I} - \mathbf{F}^*)\mathbf{C}^*(\mathbf{I} - \mathbf{F}^*)^\top + (\mathbf{F}^* - \mathbf{I})\xi(\mathbf{F}^* - \mathbf{I})^\top + \mathbf{F}^*\zeta(\mathbf{F}^*)^\top, \quad (91)$$

assuming ω and ν are uncorrelated, we have:

$$M^* = 1 - f^* = 1 - \frac{\sqrt{\xi}}{\sqrt{\xi} + \sqrt{\zeta}} = \frac{\sqrt{\zeta}}{\sqrt{\xi} + \sqrt{\zeta}}, \quad (92)$$

$$C^* = (M^*)^2 C^* + (M^*)^2 \xi + (f^*)^2 \zeta, \quad (93)$$

$$C^*(1 - (M^*)^2) = (M^*)^2 \xi + (f^*)^2 \zeta, \quad (94)$$

$$C^*(1 - M^*)(1 + M^*) = (M^*)^2 \xi + (f^*)^2 \zeta, \quad (95)$$

$$C^* f^*(1 + M^*) = (M^*)^2 \xi + (f^*)^2 \zeta, \quad (96)$$

$$C^* = \frac{(M^*)^2 \xi + (f^*)^2 \zeta}{f^*(1 + M^*)}, \quad (97)$$

$$C^* = \frac{\frac{\zeta}{(\sqrt{\xi} + \sqrt{\zeta})^2} \xi + \frac{\xi}{(\sqrt{\xi} + \sqrt{\zeta})^2} \zeta}{\frac{\sqrt{\xi}}{\sqrt{\xi} + \sqrt{\zeta}} \left(1 + \frac{\sqrt{\zeta}}{\sqrt{\xi} + \sqrt{\zeta}}\right)}, \quad (98)$$

$$C^* = \frac{\frac{2\xi\zeta}{(\sqrt{\xi} + \sqrt{\zeta})^2}}{\frac{\sqrt{\xi}}{\sqrt{\xi} + \sqrt{\zeta}} \cdot \frac{\sqrt{\xi} + 2\sqrt{\zeta}}{\sqrt{\xi} + \sqrt{\zeta}}} = \frac{2\xi\zeta}{\sqrt{\xi}(\sqrt{\xi} + 2\sqrt{\zeta})}, \quad (99)$$

$$\boxed{C^* = \frac{2\zeta\sqrt{\xi}}{\sqrt{\xi} + 2\sqrt{\zeta}}}. \quad (100)$$

Eq. (100) gives the stability true error covariance in closed form.

B Algorithm of Knowledge Injection

Algorithm 1 summarizes the proposed knowledge injection framework, which consists of two stages: knowledge injection and iterative calibration. Given $\mathcal{D}_s$, a calibration subset $\tilde{\mathcal{D}}_c$, a source SLM θ_s , and a target LLM θ_g , we first perform sensitive-aware layer selection and construct layer-wise transfer matrices ($\mathbf{W}_\ell, \mathbf{W}_r$). The knowledge vector δ_s is obtained by fine-tuning θ_s on $\mathcal{D}_s$ and injected into θ_g via a bilinear transformation. We then iteratively refine the injected knowledge using $\tilde{\mathcal{D}}_c$, updating calibration matrices, uncertainty, and covariance statistics. The final model θ'_g is obtained by integrating the calibrated updates.

C Datasets

We select two datasets *i.e.*, *MedCaseReasoning* and *ChEBI20*, from the medical and chemical domains, respectively, to evaluate the effectiveness of our method for domain-specific knowledge injection. Table 7 and Table 8 in Appendix H present examples from these two datasets, respectively.

MedCaseReasoning. *MedCaseReasoning* (Wu et al., 2025) is an open-access medical dataset designed to evaluate LLMs' capability on clinical diagnostic generation. It comprises a total of 14,489 case reports collected from over 800 medical journals. In this dataset, LLMs are tasked with generating clinical analysis based on the textual descriptions of patient symptoms. We split the dataset into 80/10/10% as the training/validation/test set for PEFT methods fine-tuning and evaluation. BLEU and ROUGE are calculated to assess the model's performance.

ChEBI20. *ChEBI20* (Edwards et al., 2021, 2022) is a curated benchmark derived from the Chemical Entities of Biological Interest (ChEBI) database, aiming to evaluate the specific ability of LLMs in

Algorithm 1: Knowledge Injection.

- 1: **INPUT:**
 - 2: *Data:* Specific domain dataset $\mathcal{D}_s$; Calibration dataset $\tilde{\mathcal{D}}_c = \{\tilde{\mathcal{D}}_c^1, \dots, \tilde{\mathcal{D}}_c^T\}$ sampled from $\mathcal{D}_s$, where T is number of iteration.
 - 3: *Model:* Small-scale source language model θ_s ; Large-scale target language model θ_g .
 - 4: **PHASE I. KNOWLEDGE INJECTION:**
 - 5: *Sensitive-aware Layer Selection:* Obtaining $\mathcal{K} = \text{TopK}(\{S^1, S^2, \dots, S^v\}, K)$; // Eqs. (13), (14), and (15)
 - 6: *Knowledge Transfer Matrix:* Obtaining $\mathbf{W}_\ell$ and $\mathbf{W}_r$ regarding each layer in $\mathcal{K}$; // Eqs. (9), (10), and (11)
 - 7: *Knowledge Vector:* Fine-tuning θ_s with $\mathcal{D}_s$ and obtain knowledge vector δ_s ; // Eq. (1)
 - 8: *Knowledge Injection:* Injecting knowledge encoded in δ_s , based on $\mathbf{W}_\ell$ and $\mathbf{W}_r$; // Eqs. (3) and (12)
 - 9: **PHASE II. KNOWLEDGE CALIBRATION:**
 - 10: **for** step t in Iterations T **do**
 - 11: *Injected Knowledgeable Parameters at t :* Obtaining $\mathbf{W}_{\delta_g^t}^t$ and $\mathbf{W}_{\delta_s^t}^t$ based on $\tilde{\mathcal{D}}_c^t$; // Eq. (8)
 - 12: *Knowledge Calibration Matrix at t :* Calculating $\mathbf{Z}^t$; // Eq. (17)
 - 13: *Integrated Knowledge Calibration Matrix:* Incorporating historical calibration information and updating $\tilde{\mathbf{W}}_{\Delta c}^t$; // Eq. (18)
 - 14: *Uncertainty and Covariance Matrix:* Updating $\mathbf{F}^t$ and $\mathbf{P}^t$; // Eqs. (19) and (20)
 - 15: **end for**
 - 16: **OUTPUT:**
 - 17: *Knowledge Injected Model:* $\theta_g' \leftarrow \theta_g + \lambda \cdot \phi(\delta_s) + \delta_g^{c(T)} - \lambda \cdot \phi^{(T)}(\delta_s^{c(T)})$; // Eq. (6)
-

molecular representation learning and text-to-molecule understanding. Specifically, it comprises 20 biochemical entity classes and 33,010 molecule-description pairs spanning diverse functional categories for molecular classification and structure–property modeling. Following (Guo et al., 2023; Zhao et al., 2024), we evaluate the LLMs’ molecule captioning ability with *ChEBI20* dataset *i.e.*, given a molecule SMILES notation, LLMs are asked to generate a textual brief description with natural language that accurately describes the key features, properties, and functional groups of the molecule. We adopt an 80/10/10% train/validation/test dataset split for PEFT methods fine-tuning, evaluation, and assess performance using standard captioning metrics, including BLEU and ROUGE.

D Baselines

We select six representative PEFT methods, including LoRA, adapter-based solutions, and prefix-tuning, alongside two privacy-preserving fine-tuning approaches (Off-site Tuning and DP-SGD), as baseline models for performance comparison. It is noteworthy that, despite their computational efficiency, the computational cost of these baselines grows with model scale, which introduces non-trivial computational overhead in source-limited scenarios.

- **P-Tuning v2.** Instead of fine-tuning full parameters, P-Tuning v2 (Liu et al., 2022) learns a small number of task-specific vectors inserted into the input embedding space, effectively optimizing the model’s internal representations toward expected behaviors while drastically reducing computational and storage overhead. As a result, P-Tuning v2 serves as a competitive and lightweight baseline for domain adaptation and instruction specialization.
- **OFT.** Rather than straightforwardly updating the raw parameters, OFT (Orthogonal Fine-Tuning) (Qiu et al., 2023) fine-tunes LLM weights within a set of orthogonal subspaces. This orthogonal constraint preserves the pre-trained representation capacity while facilitating effective task-specific adaptation.

- **QOFTv2.** Considering the expensive cubic time complexity of matrix–matrix in OFT, OFTv2 (Qiu et al., 2025) reconceptualizes the update process in an input-centric manner, replacing matrix–matrix with matrix–vector multiplications to enable matrix-free computation, thus reducing the complexity to quadratic. In addition, OFTv2 further adopts a quantization strategy to improve memory efficiency.
- **LoRA.** Low-Rank Adaptation (LoRA) (Hu et al., 2021) is a representative work in PEFT that decomposes LLM weight into a pair of low-rank learnable matrices. By incorporating these matrices into pretrained weight layers, LoRA enables efficient task adaptation with minimal additional parameters.
- **QLoRA.** QLoRA (Dettmers et al., 2023) extends the LoRA framework by introducing quantization to further reduce memory consumption. Specifically, it quantizes the pre-trained model weights into a low-bit representation while preserving a set of trainable low-rank adapters in full precision. As such, QLoRA represents a strong baseline for parameter-efficient adaptation of LLMs.
- **DoRA.** DoRA (Weight-Decomposed Low-Rank Adaptation) (Liu et al., 2024) introduces an orthogonal improvement over LoRA by reparameterizing each weight as the product of a learnable magnitude scalar and a direction vector. This decomposition enables more stable optimization and finer-grained control over adaptation dynamics, effectively reducing parameter interference while preserving representational consistency.
- **HiRA.** HiRA (Huang et al., 2025) proposes a Hadamard product high-rank adaptation. The element-wise product effectively amplifies the representational rank of the low-rank factors, allowing richer perturbations while maintaining a small parameter footprint.
- **DeLoRA.** To alleviate the trade-off between robustness of hyperparameter configurations and representation capability, DeLoRA (Bini et al., 2025) jointly normalizes and scales learnable low-rank updates. By explicitly constraining transformation distance, DeLoRA disentangles directional learning dynamics from adaptation magnitude, thereby achieving enhanced robustness without sacrificing representational flexibility.
- **Offsite-Tuning.** Offsite-Tuning (Xiao et al., 2023) first fine-tunes a lightweight surrogate model (e.g., a compressed or distilled proxy) on the client using private data, and then transferred to the server to guide the adaptation of the full-scale LLM. This design reduces client-side memory and computation requirements while preserving privacy by avoiding direct exposure of the backbone model to local data.
- **DP-SGD.** Differentially Private Stochastic Gradient Descent (DP-SGD) (Abadi et al., 2016) is a standard privacy-preserving optimization method that enables model training with formal differential privacy guarantees. It enforces bounded sensitivity by clipping per-sample gradients and injecting Gaussian noise into the aggregated gradients during optimization. As a result, DP-SGD prevents any single training instance from having excessive influence on the learned model, thereby reducing the risk of sensitive data leakage.

E Implementation Details

To ensure a fair comparison, we conduct thorough hyperparameter tuning for all methods compared in our experiments. Specifically, in the knowledge injection stage, we conduct six epochs of fine-tuning for the SLM θ_s to ensure that SLM memorizes the domain-specific knowledge, while in the knowledge calibration stage, we fine-tune the LLM θ_g for three epochs using only 3% of the training data as the calibration set (Figure 8 in Appendix F.4 investigates that performance steadily improves with calibration data size, matching FPFT at 5% and surpassing it at 10%). We preliminarily set the learning rate to 1×10^{-5} , utilizing a cosine learning rate schedule with a warm-up phase accounting for 10% of the steps over one epoch. We set the batch size as 1 and the maximum tokens to 5120 for the *MedCaseReasoning* dataset, 2048 for the *ChEBI20* dataset, and 512 for the *CodeFeedback* dataset, across all baselines. The

Table 5: Performance of our method across varying scales of SLMs and LLMs for knowledge injection. Experiments are conducted using the Qwen2.5 family. FPFT denotes full-parameter fine-tuning. B-4, R-1, and R-L correspond to BLEU-4, Rouge-1 and Rouge-L scores, respectively. We report pass@1 as the coding evaluation metric for the HumanEval (HE) and MBPP benchmarks.

Method		MedCaseReasoning				ChEBI20				Python	
		B-4	R-1	R-2	R-L	B-4	R-1	R-2	R-L	HE	MBPP
FPFT	3B	0.316	0.377	0.202	0.274	0.563	0.646	0.479	0.559	0.695	0.780
	7B	0.330	0.379	0.204	0.280	0.576	0.654	0.493	0.570	0.844	0.837
LoRA	3B	0.163	0.310	0.156	0.155	0.083	0.357	0.216	0.107	0.665	0.753
	7B	0.223	0.324	0.163	0.190	0.372	0.444	0.306	0.331	0.825	0.802
Ours	0.5B to 3B	<u>0.297</u>	0.358	0.185	0.256	0.431	0.507	0.303	0.407	0.672	0.759
	0.5B to 7B	0.288	0.351	0.179	0.249	<u>0.503</u>	0.576	<u>0.390</u>	<u>0.486</u>	<u>0.831</u>	<u>0.827</u>
	3B to 7B	<u>0.297</u>	<u>0.367</u>	<u>0.197</u>	<u>0.260</u>	0.498	<u>0.579</u>	0.389	<u>0.486</u>	0.829	0.816

hyperparameter λ , σ_ξ and σ_ζ are set to 1, 0.0001, and 0.0001, respectively. We set the rank- r of SVD in Eq. (11) as 8 and investigate the performance variations across different values. Across all baselines, we fix the number of training epochs to three. For LoRA, QLoRA, DoRA, HiRA, and DeLoRA, we apply SVD to all linear transformations in each layer (*i.e.*, q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj) and set the rank- $r = 8$, scaling factor $\alpha = 16$. The initial learning rate is set as 1×10^{-4} , while tuning the learning rate within the range $[1 \times 10^{-5}, 5 \times 10^{-3}]$ to identify the optimal configuration. We select a cosine learning-rate schedule with a warm-up ratio of 10% of the steps over one epoch. For QLoRA, we initialize the boundary parameter λ to 15, which constrains the Frobenius norm of the weight updates, preventing the fine-tuned model from drifting excessively from the original pretrained weights. Besides, we follow the original QLoRA setup and quantize all the fine-tuning weights using the *bitsandbytes* (bnb) 4-bit NF4 quantization. For P-Tuning v2, we set the number of virtual tokens to 20 with a hidden size of 128. For OFT and OFT v2, we set the block size to 32 across all layers. For Offsite-Tuning, we adopt the top and bottom four Transformer layers plus the lm_head as adapters and set the layer dropout rate to 0.5. For DP-SGD, we employ the Opacus PyTorch library (Yousefpour et al., 2021) to perform full-parameter differentially private fine-tuning. The experiments are conducted on $2 \times$ NVIDIA Tesla H100 94G GPUs.

F Experiments

F.1 LLM-as-Judge for MedCaseReasoning Dataset

Beyond BLEU and ROUGE evaluation on the *MedCaseReasoning* dataset, we further assess generation quality using the Gemini 3 Flash API as an LLM-as-a-Judge evaluator, with the evaluation prompt provided in Appendix G. The results show that our method outperforms than LoRA with a win/tie/loss ratio of 33.4%/41.3%/25.3%, indicating superior response quality and reasoning capability.

F.2 Scaling of LLMs

To assess the scalability of the proposed method, we evaluate its performance across multiple model sizes. Specifically, Qwen2.5-0.5B and Qwen2.5-3B are selected as source SLMs, Qwen2.5-3B and Qwen2.5-7B are their corresponding target LLMs, for knowledge injection. The experimental results are shown in Table 5. Across all scales, our solution consistently outperforms LoRA by a large margin, while remaining only slightly behind FPFT, demonstrating the effectiveness and scalability of our proposed solution.

F.3 Performance across Different Base Models

We further evaluate the performance of our method in comparison with FPFT and LoRA across other base models, including LLaMA3.2-3B and Phi-2-2.7B. The results are shown in Table 6. Experiments demonstrate that our method consistently performs well across different LLMs on both the *MedCaseReasoning*

Table 6: Performance of LLaMA 3.2-3B and Phi2-2.7B families. FPFT denotes full-parameter fine-tuning. B-4, R-1, and R-L correspond to BLEU-4, Rouge-1 and Rouge-L scores, respectively. We report pass@1 as the coding evaluation metric for the HumanEval (HE) and MBPP benchmarks. Our method is comparable to FPFT and outperforms LoRA consistently.

Method		MedCaseReasoning				ChEBI20				Python	
		B-4	R-1	R-2	R-L	B-4	R-1	R-2	R-L	HE	MBPP
LLaMA	FPFT	0.326	0.382	0.206	0.279	0.571	0.657	0.491	0.561	0.616	0.665
	LoRA	0.285	0.362	0.193	0.251	0.457	0.544	0.351	0.452	0.390	0.640
	Ours	<u>0.319</u>	<u>0.371</u>	<u>0.199</u>	<u>0.276</u>	<u>0.499</u>	<u>0.582</u>	<u>0.395</u>	<u>0.487</u>	<u>0.573</u>	<u>0.614</u>
Phi	FPFT	0.308	0.372	0.202	0.269	0.448	0.533	0.332	0.435	0.451	0.762
	LoRA	0.126	0.289	0.137	0.106	0.104	0.401	0.218	0.119	0.250	0.608
	Ours	<u>0.285</u>	<u>0.362</u>	<u>0.193</u>	<u>0.251</u>	<u>0.345</u>	<u>0.437</u>	<u>0.229</u>	<u>0.340</u>	<u>0.375</u>	<u>0.730</u>

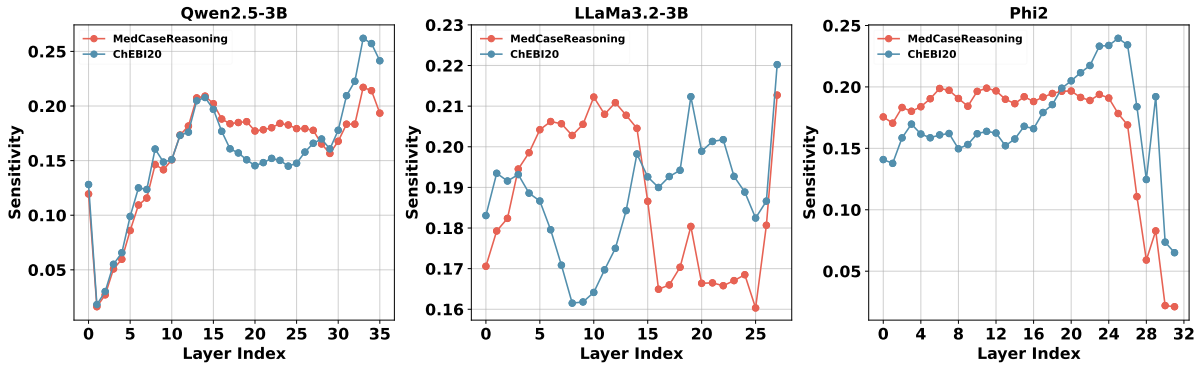


Figure 6: Layer sensitivity across different base models.

dataset, *ChEBI20* dataset and the coding dataset. In particular, our method achieves substantially larger performance gains over LoRA, indicating strong robustness and cross-model effectiveness.

F.4 Hyperparameter Analysis

Layer Sensitivity. Figure 6 illustrates the layer-wise importance across different base models, including Qwen2.5-3B, LLaMA3.2-3B, and Phi-2.7B. In general, despite case-specific variations, deeper layers tend to exhibit higher sensitivity (*i.e.*, reflected by larger magnitudes of parameter updates) than shallower layers in most scenarios, with the exception of LLaMA on the *ChEBI20* dataset. In certain cases, the sensitivity generally diminishes toward the topmost layers, a phenomenon that can be attributed to the *compression valley* (Skean et al., 2025), where intermediate layers achieve the optimal balance between information preservation and noise reduction, whereas final layers become overly specialized to the next-token prediction objective, thereby contributing less effectively to overall performance. Given the observed variability across models and datasets, we recommend tailoring layer selection strategies to each specific application.

Low Rank. We further investigate the effect of the low-rank dimensionality (*i.e.*, r in Eq. (11)) on the final performance. Figure 7 reports the results of our method, where Qwen2.5-0.5B and Qwen2.5-3B are employed as source SLM and target LLM, respectively, when knowledge injection. As shown, the performance remains slightly improved as the rank increases from 4 to 64 on *MedCaseReasoning* dataset, whereas on the *ChEBI20* dataset, performance fluctuates with increasing rank. Considering the trade-off between performance and efficiency, a moderate rank (*e.g.*, 8) can be a practical choice.

Calibration Data Ratio. Due to the critical importance of knowledge calibration to final performance, we evaluate our method under varying calibration data ratios to analyze its sensitivity. Specifically, we randomly sample 1%, 3%, 5%, and 10% of the training data for knowledge calibration. The source and target LLMs, θ_s and θ_g , are Qwen2.5-0.5B and Qwen2.5-3B, respectively. As shown in Figure 8,

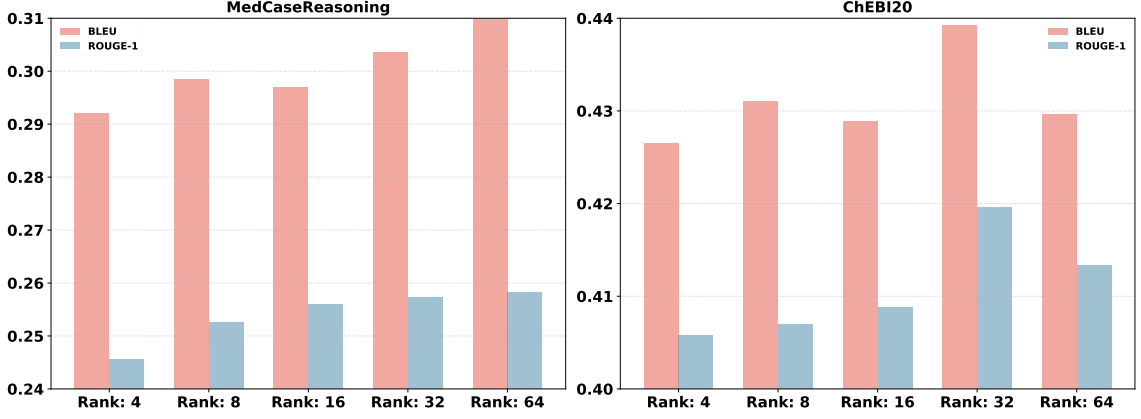


Figure 7: Performance variations across different low-rank settings. Experiments are conducted with Qwen2.5-0.5B as the source SLM and Qwen2.5-3B as the target LLM for knowledge injection.

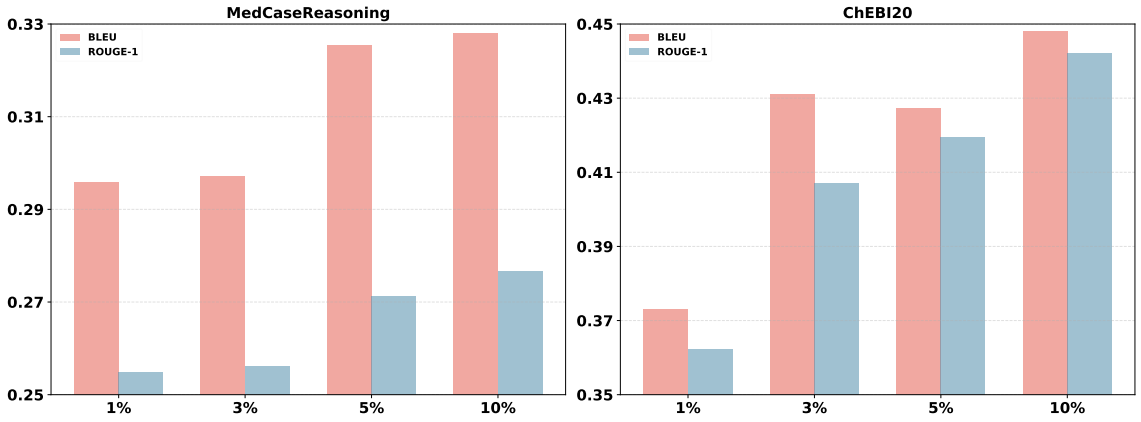


Figure 8: Performance variations under different calibration data ratios. Experiments are conducted on Qwen2.5-0.5B (source SLM) and Qwen2.5-3B (target LLM) for knowledge injection. A calibration data ratio of only 3% is sufficient to achieve satisfactory performance in practice, while adhering to privacy preservation requirements.

performance exhibits a steady upward trend with calibration data size increase. Moreover, it is encouraging to observe that with only 5% of the data, our method achieves performance approaching that of full-parameter fine-tuning (BLEU score of 0.325 vs 0.326) on the *MedCaseReasoning* dataset. When the ratio increases to 10%, our approach further surpasses FPFT (0.328 vs. 0.326), validating the effectiveness of knowledge injection. Nevertheless, considering practical constraints of privacy preservation, we recommend using a calibration ratio of 1% or 3% in real-world deployments.

F.5 Centered Kernel Alignment (CKA)

To empirically validate the semantic preservation of our knowledge injection method, we use Qwen2.5-3B to analyze the layer-wise representational similarity between our approach and the raw model. This similarity is measured using Centered Kernel Alignment (CKA) (Kornblith et al., 2019) on the *MedCaseReasoning* and *ChEBI-20* datasets. Specifically, let $\mathbf{H}_{kj}^{(\ell)} \in \mathbb{R}^{n \times d}$ and $\mathbf{H}_g^{(\ell)} \in \mathbb{R}^{n \times d}$ denote the hidden representations of the ℓ -th knowledge-injected layer and the corresponding layer from the raw model, respectively, where n is the number of samples and d is the hidden dimension. We compute the linear CKA score between the two representations as:

$$\text{CKA} \left(\mathbf{H}_g^{(\ell)}, \mathbf{H}_{kj}^{(\ell)} \right) = \frac{\left\| \left(\mathbf{H}_g^{(\ell)} \right)^\top \mathbf{H}_{kj}^{(\ell)} \right\|_F^2}{\left\| \left(\mathbf{H}_g^{(\ell)} \right)^\top \mathbf{H}_g^{(\ell)} \right\|_F \left\| \left(\mathbf{H}_{kj}^{(\ell)} \right)^\top \mathbf{H}_{kj}^{(\ell)} \right\|_F}, \quad (101)$$

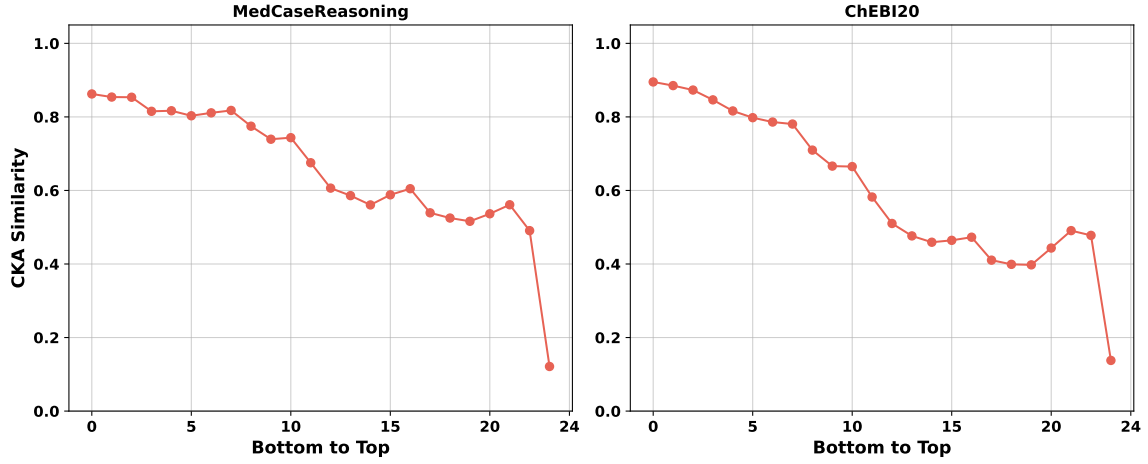


Figure 9: Layer-wise CKA similarity between the knowledge-injected layers and the correspondent layers of Qwen2.5-3B on *MedCaseReasoning* and *ChEBI20* datasets. The x-axis denotes the *relative order* of the injected layers sorted from bottom to top within the selected injection set. Higher CKA scores indicate stronger alignment of internal semantic representations between the injected and raw models. The results show that the proposed knowledge injection strategy largely preserves the hierarchical semantic structure of the raw model, while deeper injected layers exhibit relatively larger representational shifts.

where $\|\cdot\|_F$ denotes the Frobenius norm. Given CKA measures the alignment of internal feature representations while remaining invariant to isotropic scaling and orthogonal transformations, it is highly for comparing neural representations across models. High CKA scores indicate that the injected parameters preserve the representational geometry and hierarchical structure of the raw model, thereby demonstrating that our proposed strategy effectively transfers domain knowledge while maintaining the intrinsic semantic organization learned during pre-training. As visualized in the Figure 9, the results on both the *MedCaseReasoning* and *ChEBI20* datasets reveal a consistent trend, *i.e.*, representational similarity remains relatively high in the lower layers but exhibits a marked decline as the hierarchy ascends. This downward trajectory suggests that while features maintain structural consistency, the higher-level semantic abstractions diverge significantly under coarsening, indicating a layer-dependent sensitivity to input granularity.

G LLM-as-Judge Prompt

LLM-as-Judge for MedcaseReasoning

You are an expert medical educator and an AI evaluator. Your task is to evaluate two different AI-generated answers based on a medical case presentation. You must determine which answer is better by analyzing them across multiple specific perspectives.

CASE PRESENTATION:

ANSWER 1:

ANSWER 2:

Evaluation Criteria

Please evaluate both answers across the following four perspectives. Provide a brief explanation of how each answer performs for each criterion.

- 1. Information Synthesis vs. Extraction:** Does the answer naturally synthesize the medical reasoning into its own words, or does it awkwardly rely on direct, disjointed quotes from a source text?
- 2. Readability and Flow:** How smooth are the transitions between the different differential diagnoses? Is the text cohesive and easy to digest for a human reader?
- 3. Tone and Professionalism:** Which answer better reflects the authoritative, formal tone expected in a published clinical case report or a physician's clinical notes?
- 4. Formatting and Structure:** Assess the visual and structural presentation of the text (e.g., a clustered numbered list vs. a continuous narrative paragraph) and determine which is more appropriate for a differential diagnosis summary.

Output Format

Provide your evaluation in the following format:

- Analysis by Criterion: (Address each of the four criteria with specific references to the text).
- Critique of Answer 1: (Strengths and weaknesses).
- Critique of Answer 2: (Strengths and weaknesses).
- Final Verdict: (State clearly whether Answer 1, Answer 2, or a Tie is preferred, and briefly summarize why).

H Dataset Examples

Table 7: Example of *MedcaseReasoning* Dataset.

MedCaseReasoning

Instruction: Read the following case presentation, then output the reasoning for the diagnosis.

Input:

CASE PRESENTATION

A 53-year-old Afro-Caribbean man presented with a two-day history of vomiting, hiccups, and abdominal pain. He was initially treated for presumed gastroenteritis and discharged, but four days later he reported continued epigastric and lower abdominal pain, ongoing vomiting, and no bowel movements or urine output for four days. His history was notable only for hypercholesterolemia; he smoked ten cigarettes per day and denied alcohol or recent travel. On examination he was afebrile and clinically dehydrated, with a pulse of 108 beats per minute and blood pressure of 145/82 mm Hg. The abdomen was tender but soft, without peritoneal signs. A urethral catheter was inserted; urine dipstick was strongly positive for blood and protein, and positive for glucose and leukocytes.

Laboratory studies showed:

- Sodium 131 mmol/L, potassium 6.7 mmol/L
- Urea 45.2 mmol/L, creatinine 1885 μ mol/L
- C-reactive protein 421 mg/L, neutrophils 25.910⁹/L, ESR 100 mm/h
- Arterial blood gas on 28% O₂: pH 7.19, HCO₃⁻ 14.5 mmol/L, base deficit -12.9, anion gap 28 mmol/L

Renal ultrasound demonstrated normal-sized kidneys without hydronephrosis and an empty bladder. He was started on continuous veno-venous hemodiafiltration and empirical ciprofloxacin. A CT scan of the abdomen showed bilateral perinephric stranding suggestive of pyelonephritis but was otherwise unremarkable. Investigations for underlying causes of acute renal failure, including autoantibodies, complement levels, and protein electrophoresis, were all within normal limits.

YOUR ANSWER

Output:

1. Intra-abdominal sepsis leading to acute renal failure was initially favored — “our patient’s presenting symptoms of abdominal pain, constipation and anuria, along with the investigations revealing raised inflammatory markers, acute renal failure and an abdominal CT which suggested bilateral pyelonephritis, pointed towards a diagnosis of intra-abdominal sepsis leading to acute renal failure.”
2. Uremic polyneuropathy was considered for his later neuropathic signs — “Differential diagnoses considered included a polyneuropathy due to uremia, however this was less likely as the neuropathy developed several days after dialysis was commenced.”
3. Autoimmune and paraprotein disorders were evaluated but excluded - “Other differentials included diabetes mellitus, vasculitis, rheumatoid arthritis, systemic lupus erythematosus, amyloidosis, sarcoidosis, multiple myeloma or other light-chain deposition diseases, and HIV infection, which could also cause renal failure and neuropathies. These differentials were effectively ruled out with the negative autoantibody screen, normal serum electrophoresis and bone marrow biopsy, negative HIV test and normal blood glucose level.”
4. Guillain-Barré syndrome was a significant consideration for his polyradiculoneuropathy, but its likelihood diminished once the ingestion history emerged — Guillain-Barre syndrome was a significant differential; however, once the history of toxin ingestion was elucidated, this became less likely.

[illegible]