适应一次，更新中蓬勃发展：在不断演变的基础模型上进行可转移的参数高效微调

Naibin Gu[1,2], **Peng Fu**[1,2*], **Xiyu Liu**[1,2], **Ke Ma**[3], **Zheng Lin**[1,2], **Weiping Wang**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]School of Electronic, Electrical and Communication Engineering, UCAS, Beijing, China
{ gunaibin,fupeng } @iie.ac.cn

Abstract

参数高效微调（PEFT）已成为微调大型语言模型的常用方法，其中一个基础模型可以通过 PEFT 模块切换为多个用户服务。为了提升用户体验，基础模型需要定期更新。然而，一旦更新，针对以前版本微调的 PEFT 模块往往在新版本上表现出明显的性能下降。重新调整这些大量模块以恢复性能将带来巨大的计算开销。通过对基础模型更新期间发生变化的综合分析，我们发现了一个有趣的现象：持续训练主要影响存储在前馈网络（FFN）中的任务特定知识，而对注意机制中的任务特定模式影响较小。基于这些发现，我们引入了 Trans-PEFT，这是一种新方法，通过关注任务特定模式，同时减少对基础模型中某些知识的依赖性来增强 PEFT 模块。进一步的理论分析支持了我们的方法。针对 7 个基础模型和 12 个数据集的大量实验表明，训练在 Trans-PEFT 上的模块无需重新调整，就能在更新的基础模型上保持性能，从而显著减少实际应用中的维护开销 [1] 。

大规模语言模型在各个领域表现出了令人印象深刻的性能。虽然微调是为了特定应用适应这些模型的一种常用方法，但随着模型规模的扩大，其在计算资源和存储需求方面变得越来越具有挑战性。参数高效微调（PEFT）技术通过只更新一小部分参数而在保持相当性能的同时解决了这一挑战。这使得模型的高效定制化成为可能，其中一个基础模型可以通过在 PEFT 模块间的动态切换为众多用户服务。

为了提升用户体验，基础模型需要定期更新其知识和能力。这些更新通常通过持续预训练来实现，而无需改变模型结构，这可以通过从 Qwen2 (**?**) 更新到 Qwen2.5 (**?**) 来说明。然而，这就出现了一个关键挑战：当基础模型更新后，之前版本上微调的 PEFT 模块无法直接应用于新版本，因为这样的直接迁移会导致性能显著下降 (**?**) 。对于具有大量 PEFT 模块的大规模部署，重新调整不仅会带来巨大的计算开销，还会因为需要长期存储用户数据而引发重大的隐私问题。

为了应对迁移后性能下降的挑战，我们对基础模型更新过程中发生的变化进行了分析，并揭示了几个有趣的观察结果：（1）注意力激活分布在对应的层中保持相似，这表明微调后的基础模型在不同版本中维持了类似的任务特定模式。（2）在对应层间，FFN 激活分布在版本之间有所不同，这表明层内任务特定知识存储发生变化。（3）不同 FFN 子层对激活幅度的影响发生变化，表明层内和跨层知识存储都发生了转变。这些知识存储的变化导致迁移后任务性能下降，突显了减少对某些知识依赖性的必要性。

基于这些发现，我们引入了 Trans-PEFT，一种提高 PEFT 模块可转移性的新方法。在微调过程中，Trans-PEFT 采用了两种关键策略：层内知识屏蔽和跨层知识丢弃。这些策略系统地减少了 PEFT 模块对某些知识的依赖，从而鼓励 PEFT 在注意机制中捕获不变的任务特定模式。此外，我们通过限定基础模型版本间的损失差异，为迁移后的任务性能提供了理论保证。在实际应用中，Trans-PEFT 可以无缝集成到现有的 PEFT 方法中，而无需架构修改或引入额外的计算成本。这种实用的设计使得在旧基础模型上训练的 PEFT 模块能够高效地迁移到其新版本中。

我们使用来自 3 个不同来源的 7 个不同基础模型在 12 个数据集上验证了我们的方法，这些数据集涵盖了数学推理、代码生成和常识推理。实验结果表明，Trans-PEFT 在不重新调整的情况下保持了基础模型版本的性能，与直接传输相比，性能提升高达 30 %。此外，Trans-PEFT 不仅表现出维护性能的能力，还有可能利用基础模型更新带来的改进。

总之，我们的贡献如下：

- 我们揭示了基础模型更新的影响：持续训练显著改变了存储在 FFN 子层中的任务特定知识，而对注意力机制的任务特定模

* Corresponding author: Peng Fu.

[1]代码可以在 https://github.com/gccnlp/Trans-PEFT 获取。

式影响较小。

- 我们介绍了 Trans-PEFT，这是一种新颖的方法，可以将 PEFT 模块从旧的基础模型版本转移到其新版本，而无需重新调优。我们提供了理论分析以奠定其有效性的基础。
- 在 12 个数据集上的 7 个基模型中进行的大量实验表明，Trans-PEFT 训练的模块可以直接使用，无需重新调优，同时保持性能。

在我们的 Trans-PEFT 方法中，我们选择了两种广泛使用的 PEFT 技术，LoRA (**?**) 和 Adapter (**?**)，以验证可迁移性的有效性。

LoRA。对于基础模型中的线性权重 $\boldsymbol{W} \in \mathbb{R}^{d\times k}$，LoRA 引入一个可训练的模块，与线性权重并行。每个模块包含一个降维投影层 $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d\times r}$ 和一个升维投影层 $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{r\times k}$，其中 d 是基础模型的隐藏层大小以及 $r \ll d$。引入 LoRA 后，前向传播被修改如下：

$$\boldsymbol{h} \leftarrow \boldsymbol{X}\boldsymbol{W} + \boldsymbol{X}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}, \tag{1}$$

，其中 $\boldsymbol{X} \in \mathbb{R}^{d}$ 是 $\boldsymbol{W}$ 的输入。

适配器。对于基础模型中的每一层，适配器方法在注意力子层和 FFN 子层之后顺序添加两个可训练模块。添加适配器后，前向传递修改如下：

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + f(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}, \tag{2}$$

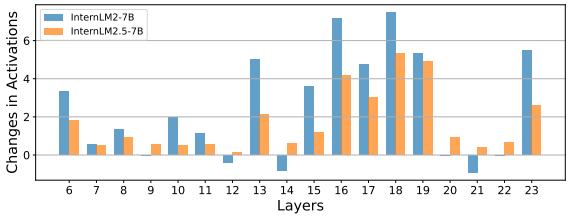
这里，$h \in \mathbb{R}^{d}$ 是子层的输出，f 是一个非线性激活函数。

1 基础模型更新分析

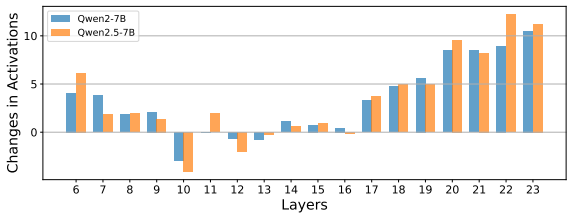
现代基础模型通常采用 Transformer (**?**) 架构，它由多个 transformer 层组成，每个层包含一个注意力子层和一个 FFN 子层。为研究更新影响，我们在 Qwen2-7B 和其新版本 Qwen2.5-7B，InternLM2-7B (**?**) 和其新版本 InternLM2.5-7B 上使用 MetaMathQA 数据集 (**?**) 微调 LoRA，并分析这些模型子层的激活。更多实现细节见附录 6 。

1.1 层内发生的变化

我们比较两个微调后的模型版本在每个层中的注意力和 FFN 子层的激活分布，如图 **??** 和 **??** 所示。蓝色矩形突出显示了相对高激活的区域。我们发现微调后两个版本的注意力子层的激活分布非常相似。然而，我们观察到 FFN 子层在中间激活分布上有明显差异。之前的可解释性研究表明，注意力子层的主要作用是提取 token 之间的依赖关系 (**?**)，而 FFN 子层主要



(a) InternLM2-7B vs InternLM2.5-7B



(b) Qwen2-7B vs Qwen2.5-7B

Figure 1: 对比不同微调基础模型版本中每个 FFN 子层对激活的影响。该影响通过相邻层之间激活幅度的差异进行衡量。

作为键值存储知识 (**?**)。这些结果表明，在微调后，两个版本的注意力机制对于捕获 token 关系的任务特定模式变化较小。通常，对基础模型的更新在不改变其架构的情况下，旨在通过持续训练改变模型知识和增强能力。FFN 激活的差异证实了两个版本在任务特定知识区域上的区别。

1.2 不同层次的变化

此外，我们还研究了不同 FFN 子层之间的知识存储变化。如图 1 所示，我们分析了每个 FFN 子层对激活幅度的影响，针对旧版本和新版本进行了比较，这反映了任务特定知识存储的变化。据观察，许多层在两个版本之间甚至表现出相反的效果，这表明在不同的 FFN 子层之间知识存储也发生了变化。总而言之，PEFT 模块未能直接转移到更新的基础模型可以归因于两个原因：基础模型的层内和跨层知识存储发生了变化。

尽管我们发现传输性能受 FFN 子层中重新分配的知识存储影响，但我们无法精确追踪基础模型如何为不同任务修改其存储。这使得无法确定 PEFT 模块在其微调任务中所需的确切修改。

为了解决这个挑战，我们提出了 Trans-PEFT 方法，它通过修改旧版本模型的微调过程来提高 PEFT 模块的可迁移性。如图 2 所示，我们的方法主要结合了两个关键策略：层内知识掩蔽和跨层知识丢弃。

层内知识掩模。由于更新导致的知识存储变化存在不确定性，我们有动机引入随机性以

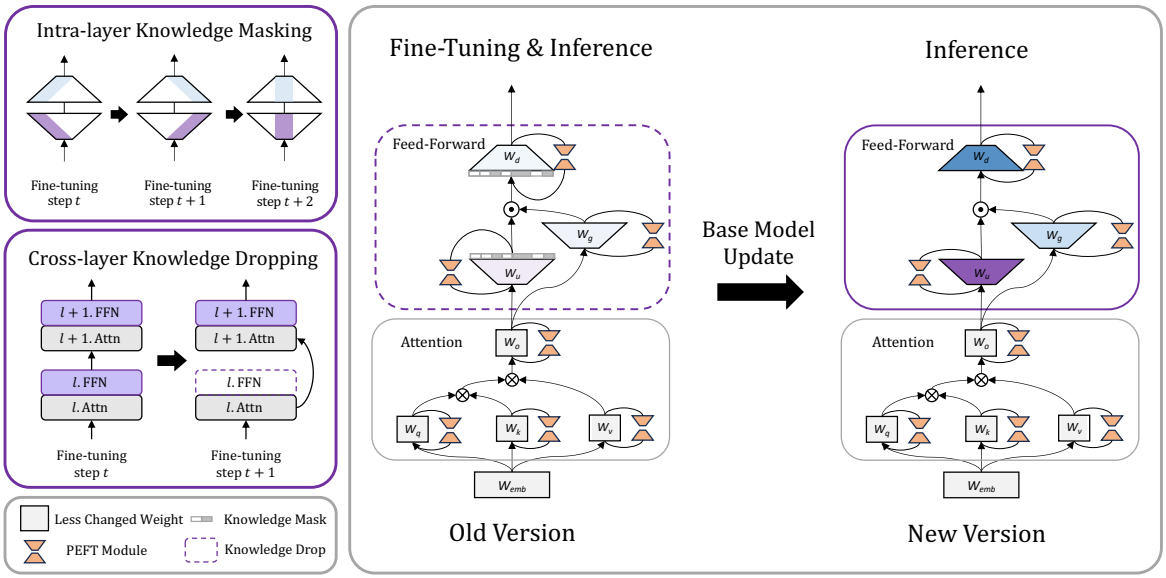


Figure 2: Trans-PEFT 的示例。Trans-PEFT 方法主要包括两种策略：层内知识屏蔽和跨层知识丢弃。在每次前向传递中，它随机选择丢弃整个 FFN 层的输出或屏蔽某些中间维度。

减少对某些知识的依赖。具体而言，给定一个 FFN 子层，它主要由一个上投影矩阵 $\boldsymbol{W}_{\text{fc1}} \in \mathbb{R}^{d \times d_{ff}}$ 、一个下投影矩阵 $\boldsymbol{W}_{\text{fc2}} \in \mathbb{R}^{d_{ff} \times d}$ 和一个激活函数 σ 组成，其中 d_{ff} 是中间尺寸 [2] 。当加入 PEFT 模块时，以 LoRA 为例，给定输入 $\boldsymbol{X}$ 的输出为：

$$\begin{aligned}\text{FFN}(\boldsymbol{X}) = &\sigma(\boldsymbol{X}(\boldsymbol{W}_{\text{fc1}} + \Delta\boldsymbol{W}_{\text{fc1}})) \\ &\cdot (\boldsymbol{W}_{\text{fc2}} + \Delta\boldsymbol{W}_{\text{fc2}}),\end{aligned} \tag{3}$$

其中 $\Delta\boldsymbol{W}$ 代表在相应权重上添加的 LoRA 模块。

为实现这一目标，我们在对旧的基本模型版本进行微调时，为每个 FFN 子层引入一个不可训练的二进制掩码向量 m 。这个掩码向量对激活函数的输出进行逐元素应用，减少其对特定维度的依赖。该掩码向量的值根据一个率 p_i 确定，如下所示：

$$m \sim \text{Bernoulli}(1 - p_i), \tag{4}$$

$$\begin{aligned}\text{FFN}(\boldsymbol{X}) = &\sigma(\boldsymbol{X}(\boldsymbol{W}_{\text{fc1}} + \Delta\boldsymbol{W}_{\text{fc1}})) \odot m \\ &\cdot (\boldsymbol{W}_{\text{fc2}} + \Delta\boldsymbol{W}_{\text{fc2}}),\end{aligned} \tag{5}$$

其中 m 在每次前向传播时变化，随机掩盖中间维度的输出。

跨层知识删除。除了层内视角策略之外，基于我们在第 **??** 节中的发现，我们从跨层视角引入随机性，以缓解 PEFT 模块对特定层知识的依赖性。具体来说，我们在每个 FFN 层之前引入一个二进制元素 z ，以决定是否删除该层的输出。对于每个层，z 的值根据速率 p_c 确定，如下所示：

$$z \sim \text{Bernoulli}(1 - p_c), \tag{6}$$

$$\widetilde{\text{FFN}}(\boldsymbol{X}) = z \cdot \text{FFN}(\boldsymbol{X}), \tag{7}$$

其中 $\widetilde{\text{FFN}}(\boldsymbol{X})$ 表示 FFN 层的最终输出。此策略在哪些层对最终输出的贡献上引入了可变性，减少了 PEFT 模块对任何特定层的依赖。

通过整合这两种策略，层内屏蔽引入了细粒度的随机性，以确保 PEFT 模块不会过度依赖 FFN 层中的某些知识。同时，跨层知识删除防止 PEFT 模块过度依赖某些层的知识，从而鼓励 PEFT 模块在注意机制中捕捉持久的任务特定模式。这些策略共同使得 PEFT 模块能够在不同版本的基础模型中保持有效性。

1.3 理论分析

为了提供我们方法有效性的理论见解，我们分析了 Trans-PEFT 如何影响 PEFT 模块在不同模型版本之间的可迁移性。

令 $\mathcal{M}_0$ 和 $\mathcal{M}_1$ 表示一个基础 Transformer 模型的两个版本，其中 $\mathcal{M}_1$ 是从 $\mathcal{M}_0$ 更新而来的。我们定义 $\boldsymbol{W}_{\text{ffn}} = \{\boldsymbol{W}_{\text{fc1}}, \boldsymbol{W}_{\text{fc2}}\}$ 为 FFN 子层中的权重，$\theta_{\text{ffn}} = \{\Delta\boldsymbol{W}_{\text{fc1}}, \Delta\boldsymbol{W}_{\text{fc2}}\}$ 为 FFN 子层的 PEFT 参数，$\boldsymbol{W}_{\text{att}}$ 为注意力子层的权重，$\Delta\boldsymbol{W}_{\text{att}}$ 为相应的 PEFT 参数。对于输入 $\boldsymbol{X} \in \mathbb{R}^d$ ，Transformer 层的输出可以表示为：

$$y = \text{Attn}(\boldsymbol{X}) + \text{FFN}(\text{Attn}(\boldsymbol{X})), \tag{8}$$

，其中 $\text{Attn}(\cdot)$ 和 $\text{FFN}(\cdot)$ 分别表示注意力和 FFN 操作。当集成 PEFT 模块（例如 LoRA）

[2] 为了简化，省略了 LLaMA 风格模型中 gate_proj 的详细描述。

时，输出变为：

$$y = \text{Attn}(\boldsymbol{X}; \theta_{\text{att}}) + \text{FFN}(\text{Attn}(\boldsymbol{X}; \theta_{\text{att}}); \theta_{\text{ffn}}), \quad (9)$$

，具体任务的损失为 $\mathcal{L}(\theta_{\text{att}}, \theta_{\text{ffn}}; \mathcal{M})$ 。

如图 **??** 和 **??** 所示，当基础模型从 $\mathcal{M}_0$ 更新到 $\mathcal{M}_1$ 时，与任务特定微调相关的注意力子层在两个版本中基本保持不变：$\boldsymbol{W}_{\text{att}}^{(1)} \approx \boldsymbol{W}_{\text{att}}^{(0)}$ 。相比之下，FFN 子层的参数发生了显著变化。因此，我们假设：

1. 注意力稳定性: $\|\boldsymbol{W}_{\text{att}}^{(1)} - \boldsymbol{W}_{\text{att}}^{(0)}\|_2 \leq \epsilon_{\text{att}} \ll 1$
2. FFN 扰动有界性：$\|\boldsymbol{W}_{\text{ffn}}^{(1)} - \boldsymbol{W}_{\text{ffn}}^{(0)}\|_2 \leq \rho$
3. 损失的光滑性：$\mathcal{L}$ 在 $\boldsymbol{W}_{\text{att}}, \boldsymbol{W}_{\text{ffn}}$ 中是 L -Lipschitz ，在 θ_{ffn} 中是 β -Lipschitz
4. 损失可分解性：总损失可以分解为来自注意力和 FFN 子层的加性组件 $\mathcal{L}(\theta; \mathcal{M}) = \mathcal{L}_{\text{att}}(\theta_{\text{att}}; \mathcal{M}) + \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}; \mathcal{M})$

基于这些假设，Trans-PEFT 被设计用来在从 $\mathcal{M}_0$ 迁移到 $\mathcal{M}_1$ 时保持 PEFT 模块的有效性，通过保留对稳定注意力子层的依赖性，同时通过控制随机性适应 FFN 的变化。下面，我们提供一个使用 Trans-PEFT 将 $\mathcal{M}_0$ 上微调的 PEFT 参数应用于更新的模型 $\mathcal{M}_1$ 时损失差距的上界。

定理 1 设 $\theta_{att}^{*(0)}$ 和 $\theta_{ffn}^{*(0)}$ 是通过 *Trans-PEFT* 在 $\mathcal{M}_0$ 上微调的 *PEFT* 参数。对于 $\mathcal{M}_1$ ，损失差异满足：

$$|\mathcal{L}(\theta^{*(0)}; \mathcal{M}_1) - \mathcal{L}(\theta^{*(0)}; \mathcal{M}_0)| \leq \underbrace{L\rho}_{\textit{FFN Shift}} + \underbrace{2\beta\|\theta_{ffn}^{*(0)} - \theta_{ffn}^{*(1)}\|}_{\textit{Parameter Deviation}} + \underbrace{C \cdot (p_i + p_c)}_{\textit{Regularization}}, \quad (10)$$

，其中 $\theta_{ffn}^{*(1)}$ 是针对 $\mathcal{M}_1$ 的假设的最佳 *PEFT* 参数。而损失函数满足光滑性和可分解性。

这个界限揭示了影响迁移性能的三个关键组成部分：首先，FFN 偏移项（$L\rho$ ）反映了基础模型更新的影响；其次，参数偏差项衡量了我们迁移的 PEFT 参数与 $\mathcal{M}_1$ 的最优 PEFT 参数之间的接近度，随着 Trans-PEFT 鼓励 PEFT 模块捕捉注意力中的持续模式，这一项可以减少；第三，正则化项，通过 Trans-PEFT 控制，量化了由其随机扰动（例如，掩码和丢弃）造成的性能损失。具体而言，层内知识掩码通过 p_i 减少模型对层内维度扰动的敏感性，而跨层知识丢弃通过 p_c 增强参数针对跨层变化的鲁棒性。通过调节 p_i 和 p_c ，我们在 $\mathcal{M}_0$ 的学习和对 $\mathcal{M}_1$ 的适应性之间取得平衡。完整证明见附录 **??** 。

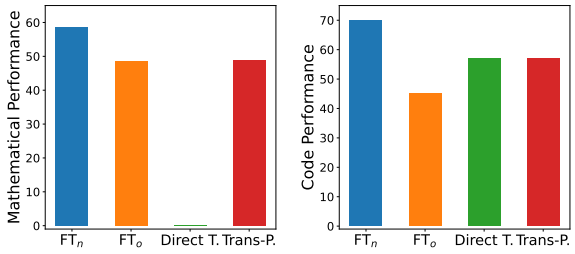
基模型。为了彻底评估我们方法的可迁移性，我们进行了一些实验，这些实验涉及来自三个不同来源的七个基模型，所有这些模型都通过持续预训练进行了更新。这些模型包括从 Qwen 家族中的 Qwen2-7B 到 Qwen2.5-7B 的一般更新，从 InternLM 家族中的 InternLM2-7B 到 InternLM2.5-7B 的一般更新，以及从 DeepSeek-7B 到 DeepSeek-Coder-7B-v1.5 和 DeepSeek-Math-7B 的领域特定更新。

数据集。我们实验中使用的数据集主要涵盖三个领域：常识推理、数学推理和代码生成。在常识推理任务中，我们按照 **?** 对模型进行微调，并在 CommonsenseQA 数据集上进行评估，包括 BoolQ (**?**) 、PIQA (**?**) 、SIQA (**?**) 、HellaSwag (**?**) 、WinoGrande (**?**) 、ARC-e、ARC-c (**?**) 和 OBQA (**?**) 数据集。在数学推理任务中，我们在 MetaMathQA 数据集的 10 万子集上微调模型，并评估在 GSM8K (**?**) 和 MATH (**?**) 数据集上的迁移性能。在代码生成任务中，我们在 CodeFeedback105K 数据集 (**??**) 上微调模型，并在 HumanEval (**?**) 和 MBPP (**?**) 数据集上评估性能。

基线与设置。我们使用三个有代表性的 PEFT 方法来评估我们的方案：LoRA (**?**) 、Adapter (**?**) 、以及 DoRA (**?**) ，这是 LoRA 的一种流行变体。我们与以下基线方法进行比较：Fine-tune $_o$ ，表示对旧版本基础模型进行微调并在旧版本上评估 PEFT；Fine-tune $_n$ ，表示在新版本上进行微调并在新版本上评估 PEFT，代表理想性能；以及 Direct Transfer $_{o \to n}$ ，即在旧版本上微调 PEFT 并直接转移到新版本而无需重新微调。我们提出的方法，Trans-PEFT $_{o \to n}$ ，应用了两种新的策略来在旧版本上微调 PEFT，然后直接应用到新版本，无需重新微调。更多细节见附录 6 。

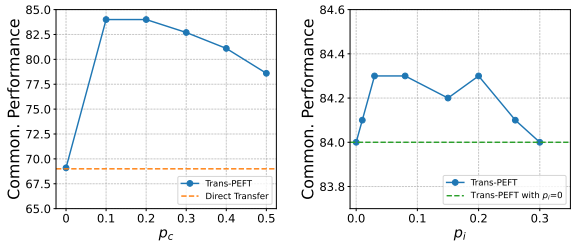
1.4 不同 PEFT 类型的实验

表格 **??** 显示了在 Qwen 基础模型上关于常识推理任务的三种不同类型 PEFT 的实验结果。如图所示，Trans-PEFT 方法在所有三种 PEFT 类型上均持续超越 Direct Transfer 方法，分别在 LoRA、Adapter 和 DoRA 上实现了 15.2 % , 9.0 % , 和 30.7 % 的显著改进。值得注意的是，Direct Transfer 方法在使用 DoRA 时平均表现仅为 55.2 % ，并且在多个任务上表现不佳，表明其在实际应用中的不实用性。相比之下，Trans-PEFT 在所有 PEFT 类型上实现了与在新版本上微调相媲美的表现，无需重新微调。这证明了其能够有效减少对某些知识的依赖，并鼓励 PEFT 模块关注注意力机制中的不变量。



(a) Mathematical Reasoning (b) Code Generation

Figure 3: 在具有领域更新的基础模型上的结果。我们评估了在 DeepSeek 基础模型上使用 Adapter 的迁移性能。对于代码任务，我们从基础模型迁移到代码更新版本，而对于数学任务，我们迁移到数学更新版本。



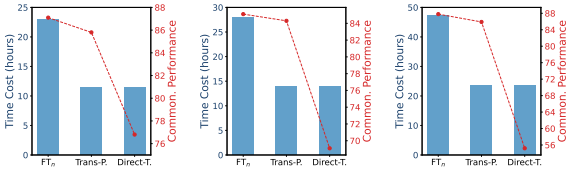
(a) Effect of p_c (b) Effect of p_i

Figure 4: 在我们提出的策略中，p_c 和 p_i 的效果。结果是在使用 LoRA 的常识推理任务上获得的。在 (a) 中，我们固定 $p_i = 0$ 并改变 p_c 。在 (b) 中，我们使用最佳设置 $p_c = 0.2$ 并改变 p_i 。

1.5 不同基础模型的实验

表 **??** 展示了在数学推理和代码生成任务中不同基础模型的结果，这些任务通常被认为更具挑战性。在常识推理实验中，两个基础模型版本表现出相当的微调性能，表明基础模型的更新并未显著增强常识推理能力。然而，对于数学和编码任务，Qwen 和 InternLM 模型家族在新版本（即，微调 $_n$ ）上的微调得分明显更高，表明基础模型更新特别增强了这些领域的能力。如图所示，Trans-PEFT 不仅在整体性能上优于微调 $_o$ ，而且接近重新调优方法微调 $_n$ 的效果，展示了它能够利用基础模型持续预训练带来的性能提升。这一成功经验验证了我们的理论分析，表明 Trans-PEFT 能够将微调过程中的参数偏差的第二项最小化。相比之下，直接迁移方法不仅未能利用模型更新带来的性能提升，甚至表现不如微调 $_o$ 。这表明微调的模块在迁移后变得无效。

此外，我们在特定领域更新的基础模型上进行了实验，如图 3 所示。在数学和编码任务中，Trans-PEFT 不仅保持了原始模型的性能，还在编码任务中取得了改进，并在数学任务中取得了轻微的提升。相比之下，直接迁移虽能匹配我们在编码任务中的表现，但在转移到更复杂的数学任务后几乎无法使用。这表明我们的方法能够有效地捕捉注意机制中的任务特定模式，从而实现基础模型版本间的 PEFT 迁移。



(a) Adapter (b) LoRA (c) DoRA

Figure 5: 在 Qwen 模型上，不同方法在不同类型的 PEFT 上的微调时间成本和性能。在 A800 GPU 上测试微调时间，这包括在旧版和新版基础模型上进行微调的时间开销。

1.6 分析

p_c 和 p_i 的效果。我们研究了 Trans-PEFT 中两个关键参数的效果：层丢弃概率 p_c 和维度掩码概率 p_i 。这些参数控制正则化效果，以减少对某些知识的依赖，从而捕捉不变的任务特定模式。参数 p_c 确定正向传播过程中丢弃每个 FFN 层的概率。如图 4 (a) 所示，当 p_c 为零时，我们提出的策略均未使用，导致次优的迁移性能。随着 p_c 的增加，迁移性能提高，并在 $p_c = 0.2$ 处达到峰值。进一步增加 p_c 开始影响 PEFT 的学习，导致迁移性能下降。

参数 p_i 控制前向传播过程中 FFN 维度被掩盖的比例。如图 4 (b) 所示，当 $p_i = 0$ 时，仅应用层知识丢弃。与单独使用层知识丢弃相比，适度增加 p_i 进一步提高了传输性能，验证了我们的双重策略方法。然而，过高的 p_i 值会对 PEFT 学习产生负面影响，这与高 p_c 值的影响类似。

微调效率。我们进行了一项分析，以评估 Trans-PEFT 在提高微调效率方面的影响。如图 5 所示，在常识推理数据集上对不同的 PEFT 方法进行微调通常需要在 A800 GPU 上耗费 14 到 24 个 GPU 小时。如果进行重新微调，包括旧版本的初始微调时间，总时间会翻倍至 28 到 48 小时（平均额外增加 20 个 GPU 小时）。在实际应用中，这种效率差距变得更加明显。例如，当处理数千个定制的 PEFT 模块时，累计的重新微调时间可能会增加到大约 20,000 个 GPU 小时 [3] 。我们的 Trans-PEFT 方法通过有效消除重新微调的需求，同时保持相当的性能

[3]需要注意的是，与预训练不同，并行地批量处理和更新成千上万用户定制的 PEFT 模块存在重大挑战。这主要是由于不同模块的目标不一致，导致需要多次前向传递来单独更新每个模块。

水平，为这一挑战提供了解决方案。它只需要在旧的基础模型版本上进行一次微调，这使得其在实际应用中非常高效和实用。这种在计算资源和时间上显著的减少使得 Trans-PEFT 对于大规模应用和频繁的模型更新特别有价值。

2 相关工作

2.1 参数高效微调

参数高效微调 (PEFT) 方法在添加可训练模块方面有几种主流方法：基于提示的方法 (**???**) 微调在模型层的隐藏状态前添加的软提示，但由于表现相对不稳定和输入长度减少而较少使用 (**?**) 。Adapter Tuning 及其变体 (**??**) 在模型层之间插入可训练模块，提供高效的参数更新。重新参数化方法，比如 LoRA 及其变体 (**????**) ，采用不同的方法，通过在模型的线性层旁边插入低秩模块来近似原始权重的更新。这两种方法通常可以实现与完整参数微调相当的性能。在这些已有的方法基础上，我们提出了 Trans-PEFT 方法，该方法允许跨不同版本的基础模型高效转移微调的 PEFT 模块，同时保持性能。

2.2 更新基础模型的迁移方法

首先由 **?** 提出将 PEFT 模块转移到基础模型的挑战。后续研究 (**??**) 发现，尽管同源模型在某种程度上可以重用 PEFT 模块，但与重新调优相比，性能仍然存在显著差距，从而限制了其实际效用。为了弥合这一差距，Trans-PEFT 专注于在通过持续预训练更新的模型之间转移 PEFT 模块，并引入了一种新的转移机制，该机制在降低计算成本的同时保持性能。从数据隐私的角度来看，**?** 提出通过使用从旧模型版本生成的合成数据来取消存储用户数据以进行重新调优的需求。虽然这种方法保护了隐私，但由于需要重新调优 PEFT 模块，仍需花费大量计算成本。Trans-PEFT 通过增强旧版本的微调过程提供了更高效的解决方案，这既减少了计算需求，又消除了存储敏感用户数据的需求。相关工作的进一步讨论见附录 5 。

3 结论

本文介绍了 Trans-PEFT，这是一种新颖的方法，使 PEFT 模块能够在更新的基础模型上工作。我们的分析表明，基础模型的更新主要影响存储在 FFN 子层中的任务特定知识，而对注意机制中的任务特定模式影响最小。利用这些见解，我们开发了层内和层间策略，以减少 PEFT 模块对 FFN 存储知识的依赖，而鼓励它们捕捉注意机制内的持久模式。大量实验表明，Trans-PEFT 成功地在不需要重新调校的情况下保持了基础模型更新后的性能，从而显著降低了 PEFT 的维护成本。

4

局限性 Trans-PEFT 显示了在连续训练的基础模型之间转移 PEFT 模块的有效性。然而，当应用于重新预训练场景时，我们的方法面临制约。这类场景通常包括架构变化或大规模数据集扩展，例如从 LLaMA2 (**?**) 更新到 LLaMA3 (**?**) 。这种限制源于具有不同随机初始化 (**?**) 的基础模型之间预训练参数空间的根本差异。由于 PEFT 在特定的参数空间 (**?**) 内运行，在完全不同的空间之间转移参数仍然不可行。探索适应我们的方法以应用于这些条件的途径代表了未来研究的方向。

我们感谢匿名审稿人的宝贵意见。同时，我们也感谢来自信息工程研究所的刘正笑的帮助。本研究得到了中国国家自然科学基金（编号：62472419, 62472420, 62376257）的支持。

设 $\mathcal{M}_0$ 和 $\mathcal{M}_1$ 表示一个基变换模型的两个版本，其中 $\mathcal{M}_1$ 是从 $\mathcal{M}_0$ 更新而来的。我们定义 $\boldsymbol{W}^{(0)} = \{\boldsymbol{W}^{(0)}_{\text{ffn}}, \boldsymbol{W}^{(0)}_{\text{att}}\}$ 为 $\mathcal{M}_0$ 的权重，$\theta^{(0)} = \{\theta^{(0)}_{\text{ffn}}, \theta^{(0)}_{\text{att}}\}$ 为在 $\mathcal{M}_0$ 上微调的 PEFT 权重；$\boldsymbol{W}^{(1)} = \{\boldsymbol{W}^{(1)}_{\text{ffn}}, \boldsymbol{W}^{(1)}_{\text{att}}\}$ 为 $\mathcal{M}_1$ 的权重，$\theta^{(1)} = \{\theta^{(1)}_{\text{ffn}}, \theta^{(1)}_{\text{att}}\}$ 为在 $\mathcal{M}_1$ 上微调的 PEFT 权重。

假设。

1. 注意稳定性：$\|\boldsymbol{W}^{(1)}_{\text{att}} - \boldsymbol{W}^{(0)}_{\text{att}}\|_2 \leq \epsilon_{\text{att}} \ll 1$
2. FFN 扰动有界性：$\|\boldsymbol{W}^{(1)}_{\text{ffn}} - \boldsymbol{W}^{(0)}_{\text{ffn}}\|_2 \leq \rho$
3. 损失平滑性：$\mathcal{L}$ 在 $\boldsymbol{W}_{\text{att}}, \boldsymbol{W}_{\text{ffn}}$ 中是 L -Lipschitz，在 θ_{ffn} 中是 β -Lipschitz
4. 损失可分解性：总损失可以分解为来自注意力和 FFN 子层的可加成分 $\mathcal{L}(\theta; \mathcal{M}) = \mathcal{L}_{\text{att}}(\theta_{\text{att}}; \mathcal{M}) + \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}; \mathcal{M})$

定理 1。设 $\theta^{*(0)} = \{\theta^{*(0)}_{\text{att}}, \theta^{*(0)}_{\text{ffn}}\}$ 是使用 Trans-PEFT 在 $\mathcal{M}_0$ 上微调的最优 PEFT 参数。在注意力稳定性和 FFN 扰动有界性的假设下，$\mathcal{M}_1$ 上的损失差异满足：

$$|\mathcal{L}(\theta^{*(0)}; \mathcal{M}_1) - \mathcal{L}(\theta^{*(0)}; \mathcal{M}_0)| \leq L\rho + 2\beta\|\theta^{*(0)}_{\text{ffn}} - \theta^{*(1)}_{\text{ffn}}\| + C \cdot (p_i + p_c), \tag{11}$$

其中 $\theta^{*(1)}_{\text{ffn}}$ 是 $\mathcal{M}_1$ 的假定最优 PEFT 参数。损失函数满足光滑性和可分解性。

4.1 证明

由于 Trans-PEFT 在 FFN 模块 $\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)} = \boldsymbol{W}_{\text{ffn}}^{(0)} + \delta(m,z)$ 中引入了掩码，因此我们可以推导出：

$$\begin{aligned}&|\mathcal{L}(\theta^{*(0)};\mathcal{M}_1) - \mathcal{L}(\theta^{*(0)};\mathcal{M}_0)| \\ &\quad = |\mathcal{L}(\theta^{*(0)};[\boldsymbol{W}_{\text{att}}^{(1)},\boldsymbol{W}_{\text{ffn}}^{(1)}]) - \mathcal{L}(\theta^{*(0)};[\boldsymbol{W}_{\text{att}}^{(0)},\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)}])|.\end{aligned} \tag{12}$$

总损失可以分为注意力和 FFN 组件（假设 4）：

$$\begin{aligned}&|\mathcal{L}(\theta^{*(0)};[\boldsymbol{W}_{\text{att}}^{(1)},\boldsymbol{W}_{\text{ffn}}^{(1)}]) - \mathcal{L}(\theta^{*(0)};[\boldsymbol{W}_{\text{att}}^{(0)},\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)}])| \\ &=|\mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(1)}) + \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(0)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})| \\ &=|\mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(1)}) - \mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(0)}) + \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})| \\ &\le|\mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(1)}) - \mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(0)})| + |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})|.\end{aligned} \tag{13}$$

因为 $\boldsymbol{W}_{\text{att}}^{(1)} \approx \boldsymbol{W}_{\text{att}}^{(0)}$（假设 1）和 $\mathcal{L}_{\text{att}}$ 的 L_{att} -Lipschitz 连续性（假设 3），注意力损失差异可以忽略：

$$|\mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(1)}) - \mathcal{L}_{\text{att}}(\theta_{\text{att}}^{*(0)};\boldsymbol{W}_{\text{att}}^{(0)})| \le L_{\text{att}}\epsilon_{\text{att}} \approx 0. \tag{14}$$

因此，总损失差异由 FFN 项主导：

$$\begin{aligned}&|\mathcal{L}(\theta^{*(0)};\mathcal{M}_1) - \mathcal{L}(\theta^{*(0)};\mathcal{M}_0)| \\ &\quad \le |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})|.\end{aligned} \tag{15}$$

因此，我们可以推导出损失的子项：

$$\begin{aligned}&|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})| \\ &= |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(0)}) + \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(0)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})| \\ &\le |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(0)})| + \underbrace{|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(0)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\tilde{\boldsymbol{W}}_{\text{ffn}}^{(0)})|}_{\text{sub-term } \mathcal{R}} \\ &\le \underbrace{|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(1)})|}_{\text{sub-term } \mathcal{A}_1} + |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)})| + \mathcal{R} \\ &\le \mathcal{A}_1 + \underbrace{|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(0)})|}_{\text{sub-term } \mathcal{B}} + \underbrace{|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(0)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(0)})|}_{\text{sub-term } \mathcal{A}_2} + \mathcal{R} \\ &= \mathcal{A}_1 + \mathcal{B} + \mathcal{A}_2 + \mathcal{R}.\end{aligned} \tag{16}$$

使用 $\mathcal{L}_{\text{ffn}}$ 的 L -Lipschitz 连续性（假设 3）和 FFN 权重的边界（假设 2），我们界定子项 $\mathcal{B}$：

$$\begin{aligned}\mathcal{B} &= |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(0)})| \\ &\le L \cdot \|\boldsymbol{W}_{\text{ffn}}^{(1)} - \boldsymbol{W}_{\text{ffn}}^{(0)}\|_2 \le L\rho.\end{aligned} \tag{17}$$

在 θ_{ffn} 中使用 β -Lipschitz（假设 3）界定子项 $\mathcal{A}_1 + \mathcal{A}_2$：

$$\begin{aligned}\mathcal{A}_1 + \mathcal{A}_2 =&|\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(0)};\boldsymbol{W}_{\text{ffn}}^{(1)}) - \mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\boldsymbol{W}_{\text{ffn}}^{(1)})| + |\mathcal{L}_{\text{ffn}}(\theta_{\text{ffn}}^{*(1)};\\ &\le 2\beta\|\theta_{\text{ffn}}^{*(0)} - \theta_{\text{ffn}}^{*(1)}\|.\end{aligned} \tag{18}$$

在模型 $\mathcal{M}_0$ 上，Trans-PEFT 的微调目标是最小化期望损失：

$$\theta_{\text{ffn}}^{*(0)} = \arg\min_{\theta} \mathbb{E}_{m,z}\left[\mathcal{L}_{\text{ffn}}(\theta;\boldsymbol{W}_{\text{ffn}}^{(0)} + \delta(m,z))\right], \tag{19}$$

其中 $\delta(m,z)$ 表示通过使用概率 p_i 掩盖 m 和使用概率 p_c 丢弃 z 而引起的隐式扰动。在权重 $\boldsymbol{W}_{\text{ffn}}^{(0)}$ 处对损失函数进行三阶 Taylor 展开：

$$\mathcal{R} = |\mathcal{L}_{\text{ffn}}(\theta;\boldsymbol{W}_{\text{ffn}}^{(0)}+\delta) - \mathcal{L}_{\text{ffn}}(\theta;\boldsymbol{W}_{\text{ffn}}^{(0)})| \approx \nabla_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta + \frac{1}{2}\delta^\top\nabla^2_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta + \frac{1}{6}\nabla^3_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta^3. \tag{20}$$

取期望后，由于 $\delta(m,z)$ 是零均值随机变量，即 $\mathbb{E}_{m,z}[\delta(m,z)] = 0$，故一阶项消失。并且二阶项主导扰动的影响：

$$\mathbb{E}[\mathcal{R}] \approx \frac{1}{2}\mathbb{E}\left[\delta^\top\nabla^2_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta\right]. \tag{21}$$

假设 Hessian 矩阵 $\nabla^2_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}$ 的最大特征值为 $\lambda_{\max}$，则：

$$\mathbb{E}\left[\delta^\top\nabla^2_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta\right] \le \lambda_{\max}\cdot\mathbb{E}\left[\|\delta\|_2^2\right], \tag{22}$$

其中，$\mathbb{E}\left[\|\delta\|_2^2\right]$ 是通过正交设计策略引入的，包括概率为 p_i 的层内掩盖和概率为 p_c 的跨层丢弃：

$$\mathbb{E}[\|\delta\|_2^2] = \mathbb{E}[\|\delta_i\|_2^2] + \mathbb{E}[\|\delta_c\|_2^2] \approx C_1 p_i + C_2 p_c, \tag{23}$$

其中，C_1 和 C_2 表示 Trans-PEFT 掩盖权重的平方范数之和。因此，我们将 $\mathcal{R}$ 的界限：

$$\mathbb{E}\left[\delta^\top\nabla^2_{\boldsymbol{W}}\mathcal{L}_{\text{ffn}}\cdot\delta\right] \le \lambda_{\max}(C_1 p_i + C_2 p_c), \tag{24}$$

$$\mathbb{E}[\mathcal{R}] \le \frac{1}{2}\lambda_{\max}(C_1 p_i + C_2 p_c) \le C\cdot(p_i + p_c), \tag{25}$$

其中 $C = \frac{1}{2}\lambda_{\max}\cdot\max(C_1, C_2)$。基于上述推导，我们得到了定理中的三个项。

5 扩展的相关工作

参数高效模块的跨模型可转移性挑战首次由 **?** 引入，该方法观察到将提示模块参数从原始模型转移到新模型往往会导致性能下降甚至完全失败。解决这个问题最直接的方法是通过在新模型上重新微调来恢复性能。MUSCLE (**?**) 试图通过利用知识蒸馏来缓解这一问题，确保在新模型中保持性能，同时像 Trans-LoRA (**?**) 和 Plug-n-Play (**?**) 等工作进一步降低或消除了通过蒸馏策略进行微调所需的额外下游数据。然而，这些方法最终仍依赖于在新模型上重新微调，因此无法真正实现参数的可转移性。

现有研究旨在实现模块参数在不同模型间的可转移性，可以分为三个方向：（1）同一模型家族内的跨尺寸转移，（2）完全不同模型架构间的转移，以及（3）通过持续训练从一个模型更新到另一个模型之间的转移。第一个方向，以 Offsite-Tuning (**?**) 、CRaSh (**?**) 、Plug-in (**?**) 和 LoRAM (**?**) 为代表，解决微调效率问题。在这里，为了提高效率，模块首先在基础模型的更小、压缩版本上进行微调，然后再转移回原始的大模型。这些工作主要解决维度不匹配的问题，而不是基础模型的实质性参数变化，使得他们的方法与我们的方法是正交的。

第二个方向尝试在两个完全不同的模型之间转移基于提示的模块 (**???**) ，利用提示模块的小参数规模和模型间重叠词汇表的对齐。然而，由于基础模型是独立预训练的，因此由于不同的初始化和训练动态 (**?**) ，它们的权重参数空间存在根本差异。由于 PEFT 方法在这些特定于模型的参数空间内运行 (**?**) ，因此这种方法在传输性方面受到严重限制，无法扩展到主流 PEFT 方法如 LoRA 或更复杂的任务。

第三个方向解决了一个常见的实际场景，即模型通过持续预训练进行增量更新，而不是从头开始重新预训练。在这种情况下，直接迁移成为最简单的解决方案。例如，Chat Vector (**?**) 和 RE-Adapt (**?**) 都专注于跨微调基础模型版本的直接迁移，这涉及到有限的持续训练。对于更大幅度的持续预训练更新，Recycle Tuning (**?**) 和 PortLLM (**?**) 表明，版本之间的直接迁移可以在通过持续预训练更新的模型上优于零样本学习。然而，Recycle Tuning 还发现，直接迁移在更新版本上的性能通常远低于通过重新调优所达到的性能，这与我们的发现一致。我们提出的方法 Trans-PEFT 更进一步，通过同时考虑模型更新过程中 FFN 子层中知识的变化和注意力子层中任务特定模式的不变性，使得迁移的参数模块在模型更新后能够保持微调性能，甚至可以达到与重调优相当的性能。

从方法论的角度来看，Trans-PEFT 中的跨层知识丢弃策略与 LayerDrop (**?**) 有相似之处，该方法侧重于推理效率；相比之下，我们的目标是减少 PEFT 模块对特定子层的依赖，并增强跨模型的可迁移性，这标志着动机和结果的根本差异。

6 实现细节

观察实验。在观察实验中，我们使用 LoRA 对模型的新旧版本进行微调，等级为 64。为了获得激活，我们为微调后的模型使用一个固定子集，并通过推理获得激活。

常识推理实验。在常识推理实验中，我们将所有 PEFT 类型的等级设置为 32，并在 Commonsense170K 数据集上进行微调。微调遵循 **?** 的设置，批量大小为 16，共进行三个周期。在所有实验中，我们将随机种子设置为 42。Finetune $_n$ 和 Finetune $_o$ 表示典型的微调并在相应的版本上使用。对于 Direct Transfer 和 Trans-PEFT，PEFT 模块仅在旧版本基础模型上进行微调，然后直接应用于新版本。对于 Trans-PEFT 参数 p_i 和 p_c ，鉴于大数据集，我们从 { 0.1, 0.2, 0.3 } 中选择。

数学推理和代码生成实验。在数学推理和代码生成实验中，我们将所有 PEFT 类型的秩设为 64。根据 **?** 的实验设置，我们在 MetaMathQA (**?**) 的 10 万子集上微调模型，并使用 MetaMathQA 代码库评估它们在 GSM8K (**?**) 和 MATH (**?**) 上的 0-shot 性能。对于代码生成任务，我们在 CodeFeedback105K 数据集上微调模型 (**??**) ，并使用 evalplus (**?**) 评估它们在人类评估 (**?**) 和 MBPP (**?**) 上的性能。对于所有实验，我们将随机种子设为 42。与常识实验一样，我们使用 16 的批大小并在三个周期上进行微调。对于 Trans-PEFT 的参数 p_c ，我们考虑从 { 0.1, 0.2, 0.3 } 和 p_i 从 { 0.01, 0.05, 0.1 } 中选择的值。所有实验均在两个 A800 GPU 上进行。

7 扩展观察

在图 **??** 和图 **??** 中，我们展示了代码生成任务中注意力子层和 FFN 子层激活分布的变化。如图所示，这些发现与第 1 节中讨论的一致：在注意力子层中，任务特定模式的变化不太明显，而在 FFN 子层中，知识储存的变化则更为显著。

为了进一步验证我们在第 1 节中的发现，即持续的预训练主要影响存储在 FFN 中的任务特定知识，而对注意力机制中的任务特定模式影响较小，我们进行了额外的实验，使用 Trans-PEFT 应用于不同的子层。具体来说，我们将 Trans-PEFT 应用于注意力子层，以检验

减少 PEFT 对这些相对稳定的任务特定模式的学习的影响，同时保持其对 FFN 子层中某些知识的依赖性。如图 **??** 所示，仅将 Trans-PEFT 应用于注意力子层时，产生的转移结果与直接转移相似，无法适应更新后的基础模型。此外，当 Trans-PEFT 同时应用于注意力和 FFN 子层时，转移性能相比仅应用于 FFN 子层时有所下降。这是因为在注意力子层中引入的随机性影响了任务特定模式的捕捉。这些定量结果为我们的发现提供了额外的实证支持。

8 扩展实验

8.1 统计显著性

表 **??** 显示了表 **??** 中实验的统计显著性结果，我们将随机种子固定为 42，并进行了三次不同数据顺序的实验。如所示，比较 Trans-PEFT 和直接转移的所有任务的 p 值都小于 0.05，表明 Trans-PEFT 显著优于直接转移。此外，平均性能差异在统计上也具有显著性。这些结果表明 Trans-PEFT 可以有效提高不同任务上的转移性能。

8.2 方法的稳定性分析

在我们所有的实验中，我们使用固定的种子（种子 =42）进行微调。在表 **??** 中，我们进一步评估了不同种子下 Trans-PEFT 的性能稳定性。总体性能趋势与表 **??** 中的结果一致，并且 Trans-PEFT 取得了显著优于直接迁移的结果。这表明 Trans-PEFT 在减少对 FFN 子层任务特定知识的依赖的能力几乎不受种子的影响。