arXiv:2301.01821v1 [cs.CL] 4 Jan 2023

参数高效微调设计空间

Jiaao Chen[†*]、**Aston Zhang**[‡]、**Xingjian Shi**[‡]、**Mu Li**[‡]、**Alex Smola**[‡]、**Diyi Yang**[◇]
[†]佐治亚理工学院，[‡]亚马逊云科技，[◇]斯坦福大学

摘要

参数高效微调旨在以更少的可训练参数，实现与微调相当的性能。已有多种策略（例如适配器、前缀微调、BitFit 和 LoRA）被提出。然而，这些策略的设计彼此独立，均由人工构思；参数高效微调是否存在某些设计模式，仍不清楚。因此，我们提出一种参数高效微调设计范式，并发现了适用于不同实验设置的设计模式。我们不再专注于设计另一种独立的调优策略，而是引入参数高效微调设计空间，用以参数化微调结构和调优策略。具体而言，任何设计空间都由四个组成部分刻画：层分组、可训练参数分配、可调组和策略分配。从一个初始设计空间出发，我们根据每种设计选择对模型质量的影响，逐步细化该空间，并在每个阶段围绕这四个组成部分进行贪心选择。我们发现了以下设计模式：（i）按纺锤形模式对层进行分组；（ii）将可训练参数均匀分配给各层；（iii）对所有组进行调优；（iv）为不同组分配合适的调优策略。这些设计模式催生了新的参数高效微调方法。实验结果表明，在自然语言处理[1]的不同任务和不同骨干模型上，这些方法始终显著优于所研究的参数高效微调策略。

1 引言

大型预训练模型通过在特定任务的标注数据上进行微调，在各种下游自然语言处理任务中取得了最先进的性能 [Devlin 等人，2019，Liu 等，2019，Yang 等，2019，Joshi 等人，2019，Sun 等人，2019，Clark 等人，2019，Lewis 等人，2020a，Bao 等人，2020，He 等人，2020，Raffel 等，2020，Ziems 等人， 2022]。然而，为不同任务微调全部参数并分别存储这些参数，会带来高昂的计算和存储开销（例如，RoBERTa 有 3.55 亿个参数 [Liu 等， 2019]，GPT-3 [则有 1750 亿个参数 2020]Brown 等人）。这使得将其部署到由多个任务组成的现实世界自然语言处理（NLP）系统中变得困难。

为了以更高的参数效率，将预训练模型中的通用知识适配到特定下游任务，研究者提出了多种策略：仅学习少量（额外）参数，同时冻结其余预训练参数 [Houlsby 等人，2019a，Pfeiffer 等人，2021，Li 和 Liang，2021，Brown 等人，2020，Lester 等人，2021a，Schick 和 Sch¨utze，2021，Ziems 等人， 2022]。适配器微调 [Houlsby 等人，2019a] 是最早一批通过有限数量的参数来引导预训练模型的策略之一。它会在预训练网络的每一层插入适配器（小型神经网络模块），并且在微调时只训练适配器。受提示方法成功经验的启发——这类方法通过文本提示来控制预训练语言模型 [Brown 等人，2020], 前缀微调 [Li 和 Liang， 2021] 以及提示微调 [Lester 等人，2021b] 会在输入或隐藏层前添加额外的可调令牌，并且在针对下游任务进行微调时只训练这些软提示。BitFit [Zaken 等人， 2021] 会更新预训练模型中的偏置项，同时冻结其余参数。LoRA [Hu 等人，2021] 将注意力权重梯度分解为低秩矩阵，以减少可训练参数的数量。鉴于此类研究取得了令人鼓舞的成果，He 等人 [2022]提出了对这些现有策略及其

[*]在亚马逊云科技实习期间完成的工作。联系 Jiaao Chen： <jiaaochen@gatech.edu>；Aston Zhang： <astonz@amazon.com>。[1]代码可在此处获取：https://github.com/amazon-science/peft-design-spaces。

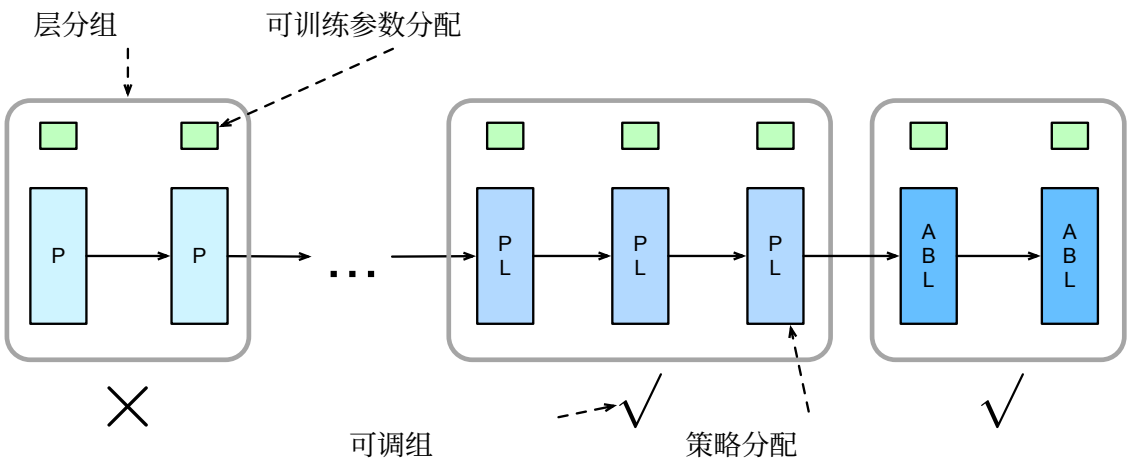

图1：参数高效微调设计空间。其特点包括：(i) 层分组（如何对连续的层进行分组）；(ii) 可训练参数分配（如何将可训练参数的数量分配给各层）；(iii) 可调组（哪些组将进行微调）；以及 (iv) 策略分配（如何为各组分配合适的策略，例如在**A**dapter、**P**refix、**B**itFit 和**L**oRA之间进行选择）。

所示差异　　相互联系的统一视角。与其前身一样，由此得到的方法仍被同等地 分配
到不同的预训练　　层。

尽管效果显著，大多数参数高效微调策略都是通过人工设计过程开发的，并未充分考虑这些不同策略之间是否存在设计模式，以及此类模式如何应用于不同的骨干模型和下游任务。此外，不同策略通常是分别应用的；因此，尚不清楚何时何处哪种策略效果最佳 [Mao et al., 2022], ，也不清楚这些不同策略如何相互强化或互为补充。鉴于此，我们的目标是以更全面的视角理解参数高效微调设计，并发现既具有可解释性、又能适用于不同实验设置的设计模式。

我们没有再设计一种同样应用于不同预训练层的单独策略，而是提出了**参数高效微调设计空间**，对微调结构和策略进行参数化。具体而言，如图1所示，这些设计空间均由四个主要组成部分构成：层分组、可训练参数分配、可调组和策略分配。

从一个相对不受约束的参数高效微调设计空间出发，我们通过比较从施加不同约束的设计空间中随机抽取的模型的整体质量，逐步细化该空间（例如，确保每个组包含相同数量的层）。在整个实验过程中，我们发现了若干参数高效微调的设计模式，例如以纺锤形模式分组各层、将可训练参数均匀分配到各层、调整所有组，以及为不同组分配恰当的调优策略。我们进一步提出了采用所有这些已发现设计模式的新型参数高效微调方法。大量实验表明，我们的方法始终优于所研究的参数高效微调策略。尽管我们以 T5 [Raffel 等人， 2020]和分类任务为示例，但我们发现，采用所有这些已发现设计模式的方法也适用于其他骨干网络（例如，RoBERTa [Liu 等人， 2019], BART [Lewis 等，2020b], 和 XLNet [Yang 等人， 2019]）以及不同的自然语言处理任务（例如，摘要、机器翻译和八个 SuperGLUE 数据集）。

我们的贡献可总结如下：(i) 我们提出了参数高效微调设计空间。(ii) 基于这些设计空间，我们通过全面的实验发现了参数高效微调中的若干设计模式。(iii) 我们发现的设计模式催生了参数高效微调方法；在不同骨干模型和不同自然语言处理任务上，这些方法始终优于所研究的参数高效微调策略。

2 相关工作

我们的工作与网络设计空间和参数高效微调相关，并以相关研究为基础。下面我们将讨论其中的联系与区别。

网络设计空间 许多研究通过临时探索新的设计选择来设计神经网络模型，以提升性能 [Radosavovic 等人, 2019], 例如采用更深的架构或残差连接。近期，一些研究 [Radosavovic et al., 2020, You et al., 2020, Radosavovic 等人， 2019] 从设计空间层面着手，探索卷积神经网络 [Radosavovic 等人，2020] 和图神经网络 [You et al.，2020]的新设计原则。受这类研究启发，我们从设计空间的视角重新思考参数高效微调，旨在发现适用于不同实验设置的设计模式。

面向自然语言处理的参数高效微调 随着预训练模型规模不断增大，存储微调模型的成本变得极高，而对于缺乏大量计算资源的人而言，微调也变得难以实现。越来越多的研究致力于寻找参数高效的替代方案，以降低内存和存储成本的方式适配大规模预训练模型。Houlsby 等人 [2019b] 提出在每一层之间使用瓶颈层（带有跳跃连接）来适配大型模型。这一思路已扩展到许多领域 [Stick- land and Murray, 2019, Pfeiffer et al., 2020, Rebuffi et al., 2017, Lin 等人， 2020]。其他研究旨在通过识别并仅训练全部模型参数中的一个子集，避免引入额外参数 [Zhao et al., 2020, Guo et al., 2020, Mallya et al., 2018, Radiya-Dixit and Wang, 2020, Sung et al., 2021, Zaken 等人， 2021]。近期研究还探索了基于克罗内克积的参数化超复数乘法进行秩分解的思路 [Zhang et al., 2021a] ，并将可训练的秩分解矩阵注入每一层 [Hu et al., 2021, Karimi Mahabadi et al.，2021]。Li 和 Liang [2021] 提出了前缀微调，即在自回归语言模型前添加一组前缀，或在编码器和解码器前都添加前缀。更新前缀参数，同时固定预训练参数。Lester 等人 [2021a] 提出了类似的方法，但只在大规模模型的嵌入层添加虚拟词元，而非离散提示 [Deng et al., 2022, Zhong et al.，2022]。Bari 等人 [2022] 提出了更易收敛的半参数提示微调，其中记忆提示可根据输入自适应，无需调优。近期，He 等人 [2022]和 Ding 等人 [2022] 提出了对现有参数高效微调策略的统一视角，并阐明了它们之间的差异与联系。Mao 等人 [2022] 还提出了一个统一框架，通过混合专家机制结合不同方法。

与上述将各自方法同等应用于不同预训练层的工作不同，我们聚焦于更通用的参数高效微调设计空间。这有助于从微调结构和调优策略两个方面，更全面地理解参数高效微调。通过逐步细化设计空间的实验，我们发现了参数高效微调的设计模式。

3 设计空间的组成部分

在定义参数高效微调的设计空间时，我们旨在涵盖关键设计组成部分，并为每个组成部分提供一组具有代表性的选项。需要注意的是，我们的目标并非枚举所有可能的设计空间，而是展示如何利用设计空间为参数高效微调研究提供参考。

具体而言，在我们的工作中，参数高效微调设计空间由参数高效微调中的一组代表性选项构成，包括以下四个组成部分：(i) 层分组，(ii) 可训练参数分配，(iii) 可调组，以及 (iv) 策略分配。参照图1所示的设计空间示例，我们将在下文详细介绍这四个设计组成部分，并在第4节探讨它们的设计选项。

层分组预训练模型中的不同层会捕获不同的信息，表现也各不相同。例如，Jawahar等人 [2019] 发现，BERT 中第 $\{3, 4, 5, 6, 7, 9, 12\}$ 层的表征能力最强，而且每一层捕获的信息类型各不相同，涵盖文本表层、句法层面乃至语义层面的表征。例如，第 9 层对语义任务具有预测能力，例如检查并列分句合取项是否随机交换；而第 3 层在预测句子长度等表层任务上的表现最佳。因此，在将这些预训练模型适配到下游任务时，如何将行为相似的层分组至关重要，这关系到合适的参数高效微调策略的设计与应用。对于这一设计要素，我们研究在微调过程中如何对预训练模型（例如 T5 中的 Transformer 层）的连续层进行分组。

可训练参数分配在参数高效微调中，可训练参数的总数通常是预先设定的，例如设为预训练模型参数总数的一小部分。我们将研究如何将预先设定数量的可训练参数分配给各层的不同设计选择。

可调组 Zaken 等人 [2021] 发现，在下游任务上进行微调时，并非所有参数都需要调整。例如，BitFit [Zaken 等人， 2021] 只更新预训练模型中的偏置参数，同时冻结其余参数。因此，我们研究在参数高效微调过程中需要学习哪些组，才能获得更好的性能。

策略分配 为了提高参数效率，研究者提出了多种策略组合 [Li 和 Liang，2021，Lester 等人，2021a，Houlsby 等人，2019a，Hu 等人， 2021] 这些方法只调整少量（额外）参数，并冻结这些预训练模型中的其余参数，从而将其通用知识适配到特定的下游任务。受提供架构灵活性的有效性启发 [Zhang et al., 2021a,b], 我们假设，不同组可能适合采用不同的策略（或策略组合），以捕获不同类型的信息。更正式地说，给定一组用于分配的独立策略 $\mathcal{A}$ ，对于任意组 G_i，为 G_i中的每一层分配一个子集 $\mathcal{U}_i \subset \mathcal{A}$。

4 发现设计模式

基于 PEFT 设计空间的这四种不同设计组件，我们将从一个相对不受约束的设计空间着手，逐步发现设计模式。

4.1 设计空间实验设置

我们首先介绍用于发现设计模式的实验设置。请注意，我们的方法也适用于其他任务和未来的预训练微调后的骨干模型。 r

数据集 我们根据广泛使用的 GLUE 基准 [Wang 等人， 2018]上的平均表现，来发现参数高效微调的设计模式。该基准涵盖了广泛的自然语言理解任务。首先，单句任务 包括 (i) 斯坦福情感树库 (SST-2) 和 (ii) 语言可接受性语料库 (CoLA)。其次，相似度和释义任务 包括 (i) Quora 问题对 (QQP)、(ii) 语义文本相似度基准 (STS-B) 和 (iii) 微软研究院释义语料库 (MRPC)。第三，推理任务 包括 (i) 多体裁自然语言推理 (MNLI)、(ii) 问句自然语言推理 (QNLI) 和 (iii) 文本蕴含识别 (RTE)。为比较表现，CoLA 使用马修斯相关系数衡量；STS-B 使用斯皮尔曼相关系数，其余 GLUE 任务则使用准确率衡量。

预训练骨干模型与模型设置 我们使用 T5-base/3b [Raffel 等人， 2020] 作为骨干模型，通过 PEFT 设计空间发现设计模式。我们使用 Hugging Face [2] 进行实现，并遵循默认设置。在探索过程中，我们参照 He 等人 [2022]的做法，将可训练参数总量设为骨干模型参数量的 0.5%。

4.2 使用 T5-base 发现设计模式

在本小节中，我们以 T5-base（预训练骨干模型）为例，介绍发现设计模式的实证过程。每个 PEFT 设计空间（记为 $\mathcal{S}_i$）都包含一组模型（$\mathcal{S}_i$-模型），这些模型满足刻画该空间的约束，涉及层分组、可训练参数分配、可调组和策略分配。为了发现设计模式，我们从一个约束相对宽松的 PEFT 设计空间（$\mathcal{S}_0$）入手。随后，我们逐步细化设计空间（从 $\mathcal{S}_0$ 到 $\mathcal{S}_{1:4}$），比较施加不同约束的设计空间中模型的整体质量（例如，每组包含相同数量的层）。为了在低计算量、低训练轮数设置 [下量化任意设计空间 $\mathcal{S}_i$ 中模型的整体质量，我们按照 Radosavovic 等人 2020], 的方法，从 $\mathcal{S}_i$中随机抽取 100 个模型，进行 3 个训练轮次的微调[3], ，并计算 GLUE 平均表现的平均值。

[2]https://huggingface.co/docs/transformers/index[3]我们通过观察模型能否在较少的训练轮数内获得稳定的性能，从而得出一致的结论，来确定训练轮数（见附录中的表 7）。

我们强调，我们的目标是展示设计空间这一视角如何帮助指导 PEFT 研究，而不是找出“最佳”设计空间或方法。出于计算效率方面的考虑，枚举设计空间各组成部分（第3节）所有可能的约束不在本文的研究范围之内。

4.2.1 初始 $\mathcal{S}_0$ 设计空间

初始的、相对不受约束的设计空间 $\mathcal{S}_0$ 包含所有在设计空间组成部分（第3节）上不受约束的模型。各个参数高效微调策略包括适配器、前缀、BitFit 和 LoRA。可以将这个 $\mathcal{S}_0$ 设计空间视为由随机设计模式组成的一组随机模型（$\mathcal{S}_0$-模型）。具体而言，在没有分组约束的情况下，预训练层中的每一层都有一半的概率会被微调：如果被选中，则会为该层分配随机策略（或策略组合），以及随机数量的可训练参数。
在比较更细致的设计模式（例如如何在适配器、前缀、BitFit 和 LoRA 之间恰当地分配可训练策略）之前，我们先探究如何对层进行分组，以及如何将可训练参数总量分配给各层。

4.2.2 具有额外分组约束的 $\mathcal{S}_1$ 设计空间

受 Radosavovic 等人 [2020], 的启发，我们在实验中也考虑了*4* 组（$G_1, \ldots, G_4$，按前向传播顺序排列）[4]。记 N_i 为 G_i中的层数。如图 2 所示，我们比较以下层分组模式：**(i) 递增 ($N_{i+1} > N_i$)**：各组中的层数逐渐增加；**(ii) 均匀($N_{i+1} = N_i$)**：各组中的层数相同；**(iii)递减 ($N_{i+1} < N_i$)**：各组中的层数逐渐减少；**(iv) 纺锤形 ($N_1 < N_2 = N_3 > N_4$)**：两端组中的层数较少；**(v)瓶颈 ($N_1 > N_2 = N_3 < N_4$)**：两端组中的层数较多。

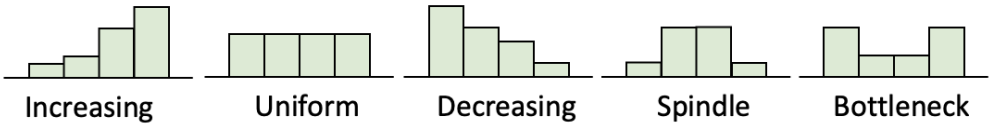

图 2：层分组模式，横轴和纵轴分别表示组（G_1、.....、 G_4）和各组中的层数。

这些层分组模式形成了 5 种不同的设计空间。每种设计空间都由 $\mathcal{S}_0$ 设计空间中满足某一种分组模式约束的所有模型构成。为了比较不同设计空间中模型的整体质量，我们：**(i)** 从满足每种分组模式约束的 $\mathcal{S}_0$ 设计空间中随机抽取 100 个模型（图 2）；**(ii)** 进行 3 个训练轮次的微调；**(iii)** 计算每种设计空间的平均性能。之后逐步添加新的约束时，我们将沿用这一流程。
平均性能如表 1[5] 所示。我们发现，在全部 8 个 GLUE 任务上，具有纺锤形分组模式的设计空间（图 2）中的模型始终优于其他设计空间中的模型。这可能是因为大型预训练模型不同层所捕获的信息复杂度各异，而这种复杂性有利于在发现的层分组模式中进行信息适配。

从现在起，我们将按纺锤形模式对层进行分组。 我们将这种采用额外设计模式的 $\mathcal{S}_0$ 称为新的$\mathcal{S}_1$ 设计空间。

4.2.3 具有额外参数约束的 $\mathcal{S}_2$ 设计空间

我们继续探索可训练参数分配中的设计模式，以细化 $\mathcal{S}_1$ 设计空间。令 n_i表示预训练骨干模型第 i 层的可训练参数数量，我们比较以下设计模式：**(i) 递增 ($n_{i+1} \geq n_i$)**：每一层中的可训练参数数量逐渐增加（或保持不变）；**(ii) 均匀 ($n_{i+1} = n_i$)**：每一层中的可训练参数数量相同；以及 **(iii)递减 ($n_{i+1} \leq n_i$)**：每一层中的可训练参数数量逐渐减少（或保持不变）。按照第4.2.2节所述的流程，我们针对这 3 个新的设计空间分别得到 100 个模型。表 2 报告了这 3 个设计空间的平均性能。均匀分配这一设计模式取得了最高的 GLUE 平均性能，因此，这种相对简单且易于解释的设计模式更具优势。

[4]8 组的实验结果见附录中的表16。[5]该步骤的训练时间见附录中的表18。

表1：使用T5-base预训练骨干模型在GLUE数据集上的平均性能（低计算量、低训练轮数设置：100个随机模型，3个微调轮数）。我们比较了向$\mathcal{S}_0$ 设计空间添加不同层分组约束的效果。

层分组	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
$\mathcal{S}_0$-模型	76.9	70.1	72.5	73.3	63.6	71.7	73.8	24.3	65.7
递增	85.3	74.9	77.2	77.5	66.8	76.2	76.0	33.0	70.8
均匀	84.8	73.7	78.1	78.6	68.5	77.8	79.2	36.1	72.1
递减	81.9	72.1	78.3	76.7	67.3	75.9	78.6	28.7	70.0
纺锤形	**86.9**	**75.5**	**79.8**	**79.4**	**69.8**	**78.3**	**80.1**	**37.3**	**73.3**
瓶颈	84.5	74.6	76.9	78.1	69.2	76.2	78.6	32.1	71.3

表 2：使用 T5-base 预训练骨干模型在 GLUE 数据集上取得的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了向 $\mathcal{S}_1$ 设计空间添加不同参数分配约束的效果。

参数分配	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
递增	87.2	**77.9**	79.4	78.7	71.6	77.6	**81.4**	32.0	73.2
均匀	**87.8**	77.4	**80.1**	**80.5**	**73.9**	**78.1**	80.4	34.3	**74.0**
递减	86.4	75.8	78.4	77.0	70.4	77.1	78.7	**35.8**	72.4

我们将均匀地向各层分配可训练参数数量。我们将 $\mathcal{S}_1$ 采用这一额外设计模式的方案称为新的 $\mathcal{S}_2$ 设计空间。

4.2.4 具有额外可调组约束的设计空间

在深入研究策略分配设计模式之前，有必要先考察哪些组需要进行微调。毕竟，只有确定哪些组需要微调之后，研究如何为不同组分配策略才有意义。如表3所示，我们探索了可调组中的各种设计模式，以进一步约束 $\mathcal{S}_2$设计空间。根据 GLUE 平均性能，我们发现，要获得最佳性能，所有组都需要进行微调。这表明，预训练层的所有组都捕获了有用的信息，应针对下游任务进行适配。

我们将微调 所有组。 我们将这种包含额外设计模式的 $\mathcal{S}_2$ 设计空间称为新的 $\mathcal{S}_3$ 设计空间 .

4.2.5 增加策略约束的 $\mathcal{S}_4$ 设计空间

最后，我们通过进一步约束所推导出的 $\mathcal{S}_3$ 设计空间，研究如何分配合适策略这一细致的设计模式。具体而言，每个设计空间都包含这样的模型：为任意组 $G_i (i = 1, ..., 4)$中的所有层分配 {Adapter（A）、前缀（P）、BitFit（B）和 LoRA（L）} 中的一个子集。我们首先向该 $\mathcal{S}_3$ 空间添加不同的 G_1策略分配约束。按照相同的模式发现流程（第4.2.2节），我们发现了 G_1的策略分配模式。然后，我们逐步将 $G_i (i > 1)$策略分配约束，以及为所有 $G_j (j = 1, ..., i - 1)$发现的策略分配模式，添加到 $\mathcal{S}_3$ 空间中。由于篇幅有限，我们在附录中展示这一过程的结果（G_1 见表8， G_2 表9， G_3 见表10， G_4 见表11），结果表明，对于 T5-base 预训练骨干模型，策略分配为 G_1-(A, L) – G_2-(A, P) – G_3-(A, P, B) – G_4-(P, B, L)。

我们将把发现的合适调优策略分配给各组。 我们将加入这一额外设计模式的 $\mathcal{S}_3$ 称为新的 $\mathcal{S}_4$ 设计空间，其中包含最终的 $\mathcal{S}_4$-模型。

4.3 使用 T5-3b 发现设计模式

随后，我们在 T5-3b 上重复上述过程，以考察使用较小模型（T5-base）发现的设计模式是否同样适用于更大的模型。结果见附录中的表 12（层分组）、表 13（可训练参数分配）、表 14（可调组）和表 15（策略分配）。我们发现，使用 T5-3b 这类更大的模型时，这些设计模式仍然适用：(i) 按纺锤形模式对层进行分组（表 12）； (ii) 将可训练参数的数量均匀分配给各层（表 13）； (iii) 调优所有组

表3：使用T5-base预训练骨干模型在GLUE数据集上的平均性能（低计算量、低训练轮数设置：100个随机模型，3个微调轮数）。我们比较了向$\mathcal{S}_2$ 设计空间中添加不同可调组约束的效果。

可调组	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_1	82.6	72.1	77.6	70.6	65.3	71.9	77.6	27.6	68.2
G_2	83.3	72.8	77.5	72.8	63.6	72.8	77.5	27.5	68.4
G_3	83.6	73.3	78.2	73.3	66.4	71.3	77.9	22.9	68.4
G_4	83.2	73.0	77.9	73.7	63.9	72.0	77.9	27.9	68.7
G_1, G_2	83.5	73.2	78.0	75.4	67.7	73.2	78.0	28.0	69.6
G_3, G_4	87.8	74.6	78.3	76.9	68.6	74.3	78.3	28.3	70.7
G_1, G_2, G_3	86.0	75.8	79.0	77.8	71.8	78.8	79.0	33.0	72.6
G_2, G_3, G_4	85.2	76.6	79.1	78.6	70.1	77.6	79.1	31.9	72.2
$\boldsymbol{G_1, G_2, G_3, G_4}$	**88.3**	**77.4**	**82.1**	**81.5**	**74.9**	**79.4**	**81.4**	**34.3**	**74.9**

表4：分别使用 T5-base（上半部分）和 T5-3b（下半部分）预训练骨干模型时，不同微调方法在 GLUE数据集上的性能。结果为 20 次随机运行的平均值（下标表示标准差）。在显著性水平 $p < 0.05(*)$甚至 $p < 0.01(**)$下， $\mathcal{S}_4$-模型和 $\mathcal{S}_4$-3b-模型在全部八个数据集上的表现均显著优于排名第二的参数高效微调方法。

方法	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	平均
full	95.2	87.1	93.7	89.4	80.1	89.4	90.7	51.1	84.5
适配器	94.6	85.5	89.8	86.7	75.3	86.7	89.1	59.2	83.3
前缀	94.0	81.6	87.8	83.4	64.3	83.1	84.8	34.0	76.6
BitFit	94.4	84.5	90.6	88.3	74.3	86.6	90.1	57.7	83.3
LoRA	94.8	84.7	91.6	88.5	75.8	86.3	88.7	51.5	82.7
$\mathcal{S}_4$-模型	$\mathbf{95.5}^{**}_{1.7}$	$\mathbf{87.6}^{**}_{1.0}$	$\mathbf{92.7}^{**}_{1.1}$	$\mathbf{88.8}^{**}_{1.0}$	$\mathbf{80.4}^{*}_{2.3}$	$\mathbf{87.4}^{*}_{2.0}$	$\mathbf{91.2}^{**}_{2.4}$	$\mathbf{62.2}^{*}_{3.2}$	**85.7**
full	97.4	91.4	96.3	89.7	91.1	90.6	92.5	67.1	89.5
适配器	96.3	89.9	94.7	87.8	83.4	90	89.7	65.2	87.1
前缀	96.3	82.8	88.9	85.5	78.3	83.5	85.4	42.7	80.4
BitFit	95.8	89.5	93.5	88.5	86.2	90.7	88.6	64.2	87.1
LoRA	96.2	90.6	94.9	89.1	91.2	91.1	91.1	67.4	88.9
$\mathcal{S}_4$-3b-模型	$\mathbf{97.2}^{**}_{1.8}$	$\mathbf{91.6}^{**}_{1.2}$	$\mathbf{96.6}^{**}_{1.0}$	$\mathbf{89.5}^{**}_{1.5}$	$\mathbf{91.5}^{*}_{2.8}$	$\mathbf{91.5}^{*}_{2.5}$	$\mathbf{91.9}^{*}_{2.0}$	$\mathbf{69.7}^{*}_{3.4}$	**89.9**

（表14），以及（iv）使用适当的策略调整不同的组（表15）。对于 T5-3b，发现的适当策略分配为 G_1-(P, L) – G_2-(A, L) – G_3-(P, B, L) – G_4-(A, P, B)。我们将最终设计空间称为 $\mathcal{S}_4$-3b，并将该设计空间中的最终模型称为 $\mathcal{S}_4$-3b-模型。

5 评估

$\mathcal{S}_4$-模型（第4.2.5节）和 $\mathcal{S}_4$-3b-模型（第4.3节）分别采用了使用 T5-base 和 T5-3b 发现的所有设计模式。因此，它们都是新的参数高效微调方法。我们将评估它们应用于不同预训练骨干模型和不同自然语言处理任务时的效果。

5.1 实验设置

数据集 除了 GLUE 数据集 [Wang 等人，2018](第4节 4.1)之外，我们还在 He 等人 [2022]使用的两项生成任务上评估我们的方法：(i) 使用 XSUM [Narayan 等人，2018], 进行抽象式摘要，以及 (ii) 使用 WMT 2016 英罗数据集 [Bojar 等人, 2016]进行机器翻译。我们报告 XSUM 测试集上的 ROUGE 分数 [Lin，2004]，以及英语-罗马尼亚语测试集上的 BLEU 分数 [Papineni 等人, 2002]。

模型与模型设置 我们主要将所提出的方法与以下基线进行比较：(i)**全量微调**(full)：对预训练模型中的所有模型参数进行微调；(ii) **适配器** [Houlsby 等人, 2019a]：在每个 Transformer 层中添加适配器模块；(iii) **前缀** [Li 和 Liang，2021]：优化一组添加到 Transformer 层前面的较小连续向量；(iv) **BitFit** [Zaken 等人，2021]：仅更新预训练模型中的偏置项；(v) **LoRA** [Hu 等人，2021]：将注意力权重分解为低秩矩阵，以减少可训练参数的数量。除了 T5 [Raffel 等人，2020], 的方法外，我们还将所提出的方法应用于其他骨干模型

表5：使用 RoBERTa-base（上半部分）和 RoBERTa-large（下半部分）预训练骨干模型时，不同微调方法在 GLUE数据集上的表现。结果取 20 次随机运行的平均值（下标为标准差）。这里还纳入了两个基线：(i) $\mathcal{S}_0$-*model*，其中 *RoBERTa* 的所有设计均按 $\mathcal{S}_0$ 设计空间中的方式随机选取；*(ii)* $\mathcal{S}_3$-*model*，其中策略按照 $\mathcal{S}_3$ 设计空间中的方式随机分配给不同的 *RoBERTa* 层组。 $\mathcal{S}_4$-*model* 和 $\mathcal{S}_4$-*3b-model* 在全部八个数据集上的表现均显著优于排名第二的参数高效微调方法，显著性水平为 $p < 0.05(*)$，甚至达到 $p < 0.01(**)$。

方法	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	平均
full	94.8	87.6	92.8	91.9	80.8	90.3	90.2	63.6	86.5
适配器	94.2	87.1	93.1	90.2	71.5	89.7	88.5	60.8	84.4
前缀	94.0	86.8	91.3	90.5	74.5	90.3	88.2	61.5	84.6
BitFit	93.7	84.8	91.3	84.5	77.8	**90.8**	90.0	61.8	84.3
LoRA	**94.9**	87.5	93.1	90.8	83.1	90.0	89.6	62.6	86.4
$\mathcal{S}_0$-模型	94.2	95.3	90.4	90.6	75.6	89.6	88.0	60.9	85.6
$\mathcal{S}_3$-模型	94.3	87.2	92.8	91.0	81.8	90.3	89.2	63.2	86.2
$\mathcal{S}_4$-模型	$94.8_{1.6}$	$\mathbf{87.8}^{**}_{0.8}$	$\mathbf{93.4}^{**}_{1.3}$	$\mathbf{91.6}^{*}_{1.2}$	$\mathbf{85.8}^{**}_{1.8}$	$90.4^{*}_{2.0}$	$\mathbf{90.0}^{**}_{1.8}$	$\mathbf{63.2}^{*}_{3.5}$	**87.1**
full	96.4	90.2	94.7	92.2	86.6	92.4	90.9	68.0	88.9
适配器	96.6	90.5	94.8	91.7	80.1	92.1	90.9	67.8	88.1
前缀	95.7	87.6	92.1	88.7	82.3	89.6	87.4	62.8	85.7
BitFit	96.1	88.0	93.4	90.2	86.2	90.9	**92.7**	64.2	87.7
LoRA	96.2	90.6	94.7	91.6	**87.4**	92.0	89.7	68.2	88.8
$\mathcal{S}_0$-模型	95.5	86.5	92.3	89.8	84.6	89.2	86.3	61.2	85.6
$\mathcal{S}_3$-模型	96.3	89.4	93.8	90.2	85.9	90.8	90.9	63.4	87.6
$\mathcal{S}_4$-3b-模型	$\mathbf{96.6}^{**}_{1.3}$	$\mathbf{90.8}^{*}_{1.1}$	$\mathbf{95.1}^{**}_{0.8}$	$\mathbf{92.0}^{**}_{1.2}$	$87.2_{2.8}$	$\mathbf{92.3}^{*}_{2.2}$	$91.8^{**}_{1.8}$	$\mathbf{68.4}^{*}_{3.2}$	**89.3**

包括 RoBERTa-base/large [Liu 等人， 2019]和 BART-base/large [Lewis 等人，2020a]。我们采用默认设置。我们按照 He 等人 [2022]的方法，设置可训练参数总量（占骨干模型参数总量的百分比）。具体而言，适配器、前缀、LoRA 和我们的方法均设为 0.5%，BitFit 则设为 0.1%。

在所有实验中，我们遵循 Liu 等人 [2019]的做法，使用预热比例为 0.06 的线性衰减调度器进行训练。基础模型的批大小为 128，大型模型的批大小为 64。最大学习率为 $5e-5$，训练轮数的上限设为 5 或 10。所有实验均使用 8 张 A100 GPU 完成。

5.2 使用 T5 骨干网络在 GLUE 上的有效性

借助我们发现的设计模式，我们在 GLUE 上对 T5-base（$\mathcal{S}_4$-模型）和 T5-3b（$\mathcal{S}_4$-3b-模型）进行微调，并将其与所有基线方法进行比较。结果如表4所示，其中关键指标是 GLUE 平均性能（最后一列）。我们发现，在这一关键指标上，我们的 $\mathcal{S}_4$-模型和 $\mathcal{S}_4$- 3b-模型始终优于所研究的方法。仅调整 0.5% 的参数，我们的方法甚至优于调整所有参数的全量微调基线，这表明我们发现的参数高效微调设计模式十分有效。

表 6：使用 BART-base（上半部分）和 BART-large（下半部分）预训练骨干模型时，不同调优方法在生成任务（XSUM 和英语-罗马尼亚语）上的表现。

方法	XSUM（R-1/2/L）	英语-罗马尼亚语（BLEU）
full	40.5/19.2/34.8	34.5
适配器	37.7/17.9/33.1	33.3
前缀	38.2/18.4/32.4	33.8
BitFit	37.2/17.5/31.4	33.2
LoRA	38.9/18.6/33.5	33.6
PA	39.3/18.7/33.8	33.8
$\mathcal{S}_4$-模型	**40.2/19.3/34.2**	**34.1**
full	45.1/22.3/37.2	37.9
适配器	43.8/20.8/35.7	35.3
前缀	43.4/20.4/35.5	35.6
BitFit	42.8/18.7/33.2	35.2
LoRA	42.9/19.4/34.8	35.8
PA	43.9/20.6/35.6	36.4
$\mathcal{S}_4$-3b-模型	**44.3/21.7/36.8**	**37.2**

5.3 RoBERTa 主干网络上 GLUE 的普遍有效性

我们直接将 $\mathcal{S}_4$-model 和 $\mathcal{S}_4$-3b-model（采用使用 T5-base 和 T5-3b 发现的设计模式）分别用于微调 RoBERTa-base 和 RoBERTa-large 预训练骨干模型（无需额外的发现过程）。我们保持其他所有设置不变，并在 GLUE 数据集上对其进行评估。我们还与从两个设计空间中随机采样的变体方法进行比较：(

（i） $\mathcal{S}_0$-*model*，其中为 RoBERTa 随机选择所有设计，如 $\mathcal{S}_0$ 中所示；（ii） $\mathcal{S}_3$-*model*，其中将策略随机分配给不同的 RoBERTa 层组，如 $\mathcal{S}_3$ 中所示。表5显示：（i）使用 T5 模型发现的设计模式（由 $\mathcal{S}_4$-model 和 $\mathcal{S}_4$-3b-model 采用）适用于 RoBERTa 骨干模型，且在 GLUE 平均性能上优于所研究的方法，无需额外的发现过程；（ii）从 $\mathcal{S}_0$-models、 $\mathcal{S}_3$-models 到 $\mathcal{S}_4$-(3b)-models，性能逐步提升，这支持在模式发现过程中加入更多约束（第4节）。

5.4 在 BART 骨干网络上生成任务的总体效果

与第 5.3 节类似，我们进一步将 $\mathcal{S}_4$-model 和 $\mathcal{S}_4$-3b-model（采用使用 T5-base 和 T5-3b 发现的设计模式）直接应用于 BART-base 和 BART-large 预训练骨干模型的微调（未进行额外的发现过程）。我们按照 He 等人 [2022] 的方法，在两个生成任务上评估这些模型：摘要（XSUM）和机器翻译（英语-罗马尼亚语）。我们还与可训练参数数量相同的 PA（并行适配器）进行比较 [He 等人， 2022]。表 6 显示，尽管我们的方法采用了使用 T5 在分类任务上发现的设计模式，但在采用不同 BART 骨干网络的生成任务上，仍优于所考察的参数高效微调策略。

6 结论

参数高效微调以更高的参数效率，将预训练模型中的知识迁移到下游任务中。我们没有一开始就专注于设计另一种策略，而是引入了参数高效微调设计空间。我们通过实证发现了参数高效微调中的若干设计模式。这些设计模式催生了新的参数高效微调方法。实验表明，在自然语言处理领域的不同骨干模型和不同任务上，这些方法始终优于所考察的参数高效微调策略。

参考文献

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。Bert：用于语言理解的深度双向 Transformer 预训练。发表于*NAACL-HLT*，2019年。Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。Roberta：一种稳健优化的 BERT 预训练方法。*arXiv* 预印本 *arXiv:1907.11692*，2019年。Zhilin Yang、Zihang Dai、Yiming Yang、Jaime Carbonell、Russ R Salakhutdinov 和 Quoc V Le。Xlnet：用于语言理解的广义自回归预训练。发表于神经信息处理系统进展，第5754–5764页，2019年。Mandar Joshi、Danqi Chen、Yinhan Liu、Daniel S. Weld、Luke Zettlemoyer 和 Omer Levy。Spanbert：通过表示和预测跨度改进预训练。计算语言学协会会刊，8：64–77，2019年。Yu Sun、Shuohuan Wang、Yukun Li、Shikun Feng、Xuyi Chen、Han Zhang、Xin Tian、Danxiang Zhu、Hao Tian 和 Hua Wu。Ernie：通过知识整合增强表征。*arXiv* 预印本 *arXiv:1904.09223*，2019年。Kevin Clark、Minh-Thang Luong、Quoc V Le 和 Christopher D Manning。Electra：将文本编码器作为判别器而非生成器进行预训练。发表于国际学习表征会议，2019年。Mike Lewis、Yinhan Liu、Naman Goyal、Marjan Ghazvininejad、Abdelrahman Mohamed、Omer Levy、Ves Stoyanov 和 Luke Zettlemoyer。Bart：用于自然语言生成、翻译和理解的去噪序列到序列预训练。*SCL*，2020a年。Hangbo Bao、Li Dong、Furu Wei、Wenhui Wang、Nan Yang、Xiaodong Liu、Yu Wang、Songhao Piao、Jianfeng Gao、Ming Zhou 等。Unilmv2：用于统一语言模型预训练的伪掩码语言模型。*arXiv* 预印本 *arXiv:2002.12804*，2020年。Pengcheng He、Xiaodong Liu、Jianfeng Gao 和 Weizhu Chen。Deberta：采用解耦注意力的解码增强型 BERT。*arXiv* 预印本 *arXiv:2006.03654*，2020年。Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。探索使用统一文本到文本 Transformer 进行迁移学习的极限，2020年。Caleb Ziems、Jiaao Chen、Camille Harris、Jessica Anderson 和 Diyi Yang。VALUE：理解自然语言理解中的方言差异。发表于计算语言学协会第*60*届年会论文集（第

1：长篇论文），第3701–3720页，爱尔兰都柏林，2022年5月。计算语言学协会。doi：10.18653/v1/2022.acl-long.258。URL https://aclanthology.org/2022.acl-long.258。Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeffrey Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。语言模型是少样本学习者，2020年。Neil Houlsby、Andrei Giurgiu、Stanislaw Jastrzebski、Bruna Morrone、Quentin De Laroussilhe、Andrea Gesmundo、Mona Attariyan 和 Sylvain Gelly。自然语言处理的参数高效迁移学习。发表于Kamalika Chaudhuri和Ruslan Salakhutdinov编，第*36*届国际机器学习会议论文集，机器学习研究论文集第97卷，第2790–2799页。PMLR，2019年6月9–15日。URL http://proceedings.mlr.press/v97/houlsby19a.html。Jonas Pfeiffer、Aishwarya Kamath、Andreas R¨uckl´e、Kyunghyun Cho 和 Iryna Gurevych。AdapterFusion：用于迁移学习的非破坏性任务组合。发表于计算语言学协会欧洲分会第*16*届会议论文集：主会场，第487–503页，线上，2021年4月。计算语言学协会。URL https://www.aclweb.org/anthology/2021.eacl-main.39。Xiang Lisa Li 和 Percy Liang。前缀微调：优化用于生成的连续提示，2021年。Brian Lester、Rami Al-Rfou 和 Noah Constant。规模在参数高效提示微调中的力量，2021a。Timo Schick 和 Hinrich Sch¨utze。利用完形填空问题进行少样本文本分类和自然语言推理。发表于计算语言学协会欧洲分会第*16*届会议论文集：主会场，第255–269页，线上，2021年4月。计算语言学协会。doi：10.18653/v1/2021.eacl-main.20。URL https://aclanthology.org/2021.eacl-main.20。Brian Lester、Rami Al-Rfou 和 Noah Constant。规模在参数高效提示微调中的力量。发表于*2021*年自然语言处理实证方法会议论文集，第3045–3059页，线上及多米尼加共和国蓬塔卡纳，2021年11月。计算语言学协会。doi：10.18653/v1/2021.emnlp-main.243。URL https://aclanthology.org/2021.emnlp-main.243。Elad Ben Zaken、Shauli Ravfogel 和 Yoav Goldberg。Bitfit：面向 Transformer 掩码语言模型的简单参数高效微调，2021年。URL https://arxiv.org/abs/2106.10199。Edward J. Hu、Yelong Shen、Phillip Wallis、Zeyuan Allen-Zhu、Yuanzhi Li、Shean Wang、Lu Wang 和 Weizhu Chen。LoRA：大语言模型的低秩适配，2021年。URL https://arxiv.org/abs/2106.09685。Junxian He、Chunting Zhou、Xuezhe Ma、Taylor Berg-Kirkpatrick 和 Graham Neubig。迈向参数高效迁移学习的统一视角。发表于国际学习表征会议，2022 年。Yuning Mao 、Lambert Mathias、Rui Hou、Amjad Almahairi、Hao Ma、Jiawei Han、Scott Yih 和 Madian Khabsa。UniPELT：参数高效语言模型调优的统一框架。发表于计算语言学协会第*60*届年会论文集（第*1*卷：长篇论文），第6253–6264页，爱尔兰都柏林，2022年5月。计算语言学协会。doi：10.18653/v1/2022.acl-long.433。URL https: //aclanthology.org/2022.acl-long.433。Mike Lewis、Yinhan Liu、Naman Goyal、Marjan Ghazvininejad、Abdelrahman Mohamed、Omer Levy、Veselin Stoy-anov 和 Luke Zettlemoyer。BART：用于自然语言生成、翻译和理解的去噪序列到序列预训练。发表于计算语言学协会第*58*届年会论文集，第7871–7880页，线上，2020年7月。计算语言学协会。doi：10.18653/v1/2020.acl-main.703。URL https://www.aclweb.org/anthology/2020.acl-main.703。Ilija Radosavovic、Justin Johnson、Saining Xie、Wan-Yen Lo 和 Piotr Doll´ar。关于视觉识别的网络设计空间，2019年。URL https://arxiv.org/abs/1905.13214。Ilija Radosavovic、Raj Prateek Kosaraju、Ross Girshick、Kaiming He 和 Piotr Doll´ar。设计网络设计空间，2020年。URL https://arxiv.org/abs/2003.13678。Jiaxuan You、Rex Ying 和 Jure Leskovec。图神经网络的设计空间，2020年。URL https://arxiv. org/abs/2011.08843。Neil Houlsby、Andrei Giurgiu、Stanislaw Jastrzebski、Bruna Morrone、Quentin De Laroussilhe、Andrea Gesmundo、Mona Attariyan 和 Sylvain Gelly。自然语言处理的参数高效迁移学习。发表于国际机器学习会议，第2790–2799页。PMLR，2019b。Asa Cooper Stickland 和 Iain Murray。Bert 与 Pals：用于多任务学习中高效适配的投影注意力层

g。在国际会议 *f*机器学习 *g*, pag第5986–5995页。PMLR，2019年。

Jonas Pfeiffer、Aishwarya Kamath、Andreas R¨uckl´e、Kyunghyun Cho 和 Iryna Gurevych。AdapterFusion：用于迁移学习的非破坏性任务组合。*arXiv* 预印本 *arXiv:2005.00247*，2020年。Sylvestre-Alvise Rebuffi、Hakan Bilen 和 Andrea Vedaldi。通过残差适配器学习多个视觉域。*arXiv* 预印本 *arXiv:1705.08045*，2017年。Zhaojiang Lin、Andrea Madotto 和 Pascale Fung。通过参数高效迁移学习探索多功能生成式语言模型。*arXiv* 预印本 *arXiv:2004.03829*，2020年。Mengjie Zhao、Tao Lin、Fei Mi、Martin Jaggi 和 Hinrich Sch¨utze。将掩码作为预训练语言模型微调的高效替代方案。*arXiv* 预印本 *arXiv:2004.12406*，2020年。Demi Guo、Alexander M Rush 和 Yoon Kim。通过差分剪枝实现参数高效迁移学习。*arXiv* 预印本 *arXiv:2012.07463*，2020年。Arun Mallya、Dillon Davis 和 Svetlana Lazebnik。Piggyback：通过学习掩码权重，让单个网络适应多项任务。载于欧洲计算机视觉会议（*ECCV*）论文集，第67–82页，2018年。Evani Radiya-Dixit 和 Xin Wang。微调可以有多精细？学习高效语言模型。载于国际人工智能与统计学会议，第2435–2443页。PMLR，2020年。Yi-Lin Sung、Varun Nair 和 Colin A Raffel。使用固定稀疏掩码训练神经网络。载于 M. Ranzato、A. Beygelzimer、Y. Dauphin、P.S. Liang 和 J. Wortman Vaughan（编），神经信息处理系统进展，第34卷，第24193–24205页。Curran Associates, Inc.，2021年。网址：https://proceedings.neurips.cc/paper/2021/file/cb2653f548f8709598e8b5156738cc51-Paper.pdf。Aston Zhang、Yi Tay、Shuai Zhang、Alvin Chan、Anh Tuan Luu、Siu Hui 和 Jie Fu。超越全连接层的四元数：用 $1/n$ 参数参数化超复数乘法。载于国际学习表征会议，2021a。Rabeeh Karimi Mahabadi、James Henderson 和 Sebastian Ruder。Compacter：高效的低秩超复数适配器层。神经信息处理系统进展，34：1022–1035，2021年。Mingkai Deng、Jianyu Wang、Cheng-Ping Hsieh、Yihan Wang、Han Guo、Tianmin Shu、Meng Song、Eric P. Xing 和 Zhiting Hu。RLPrompt：使用强化学习优化离散文本提示，2022年。网址：https: //arxiv.org/abs/2205.12548。Wanjun Zhong、Yifan Gao、Ning Ding、Zhiyuan Liu、Ming Zhou、Jiahai Wang、Jian Yin 和 Nan Duan。通过统一模式提示提升任务泛化能力，2022年。网址：https://arxiv.org/abs/2208.03229。M Saiful Bari、Aston Zhang、Shuai Zheng、Xingjian Shi、Yi Zhu、Shafiq Joty 和 Mu Li。SPT：面向多任务提示学习的半参数提示微调。*arXiv* 预印本 *arXiv:2212.10929*，2022年。Ning Ding、Yujia Qin、Guang Yang、Fuchao Wei、Zonghan Yang、Yusheng Su、Shengding Hu、Yulin Chen、Chi-Min Chan、Weize Chen、Jing Yi、Weilin Zhao、Xiaozhi Wang、Zhiyuan Liu、Hai-Tao Zheng、Jianfei Chen、Yang Liu、Jie Tang、Juanzi Li 和 Maosong Sun。Delta调优：预训练语言模型参数高效方法的全面研究，2022年。网址：https://arxiv.org/abs/2203.06904。Ganesh Jawahar、Benoˆıt Sagot 和 Djam´e Seddah。BERT 对语言结构有何认识？载于计算语言学协会第*57*届年会论文集，第3651–3657页，意大利佛罗伦萨，2019年7月。计算语言学协会。Aston Zhang、Yi Tay、Yikang Shen、Alvin Chan 和 Shuai Zhang。具有动态软递归的自实例化循环单元。神经信息处理系统进展，34：6503–6514，2021b。Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。GLUE：自然语言理解的多任务基准与分析平台。载于*BlackboxNLP@EMNLP*，2018年。Shashi Narayan、Shay B. Cohen 和 Mirella Lapata。别给我细节，只要摘要！面向极限摘要的主题感知卷积神经网络。载于*2018*年自然语言处理实证方法会议论文集，第1797–1807页，比利时布鲁塞尔，2018年10月至11月。计算语言学协会。doi：10.18653/v1/D18-1206。网址：https://aclanthology.org/D18-1206。Ondˇrej Bojar、Rajen Chatterjee、Christian Federmann、Yvette Graham、Barry Haddow、Matthias Huck、Antonio Ji-meno Yepes、Philipp Koehn、Varvara Logacheva、Christof Monz、Matteo Negri、Aur´elie N´ev´eol、Mariana Neves、Martin Popel、Matt Post、Raphael Rubino、Carolina Scarton、Lucia Specia、Marco Turchi、Karin Verspoor 和 Mar-cos Zampieri。2016年机器翻译会议成果。载于第一届机器翻译会议论文集：第*2*卷，共享任务论文，第131–198页，德国柏林，2016年8月。计算语言学协会。doi：10.18653/v1/W16-2301。网址：https://aclanthology.org/W16-2301。

Chin-Yew Lin. ROUGE：一种用于自动评估摘要的工具包。载于*Text Summarization BranchesOut*，第 74–81 页，西班牙巴塞罗那，2004年7月。计算语言学协会。网址：https: //aclanthology.org/W04-1013。
Kishore Papineni、Salim Roukos、Todd Ward 和 Wei-Jing Zhu。Bleu：一种自动评估机器翻译的方法。载于*ACL*，2002年。

更多实验结果

表 7：使用 T5-base 预训练骨干模型在 GLUE数据集上进行评估的平均表现（低计算量、低训练轮数设置：100 个随机模型，五种不同模块的微调轮数为 $= 1$，2，3，4，20）。我们比较了向 $\mathcal{S}_0$ 设计空间添加不同分组约束的效果。

分组模式	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
1 个训练轮次									
递增	73.2	63.3	**67.8**	68.8	**63.8**	67.2	64.1	11.0	**59.9**
均匀	**72.8**	64.1	63.4	63.4	62.5	**69.8**	65.8	12.1	59.2
递减	72.4	63.2	65.1	69.8	59.3	62.7	63.6	**18.7**	59.4
纺锤形	72.6	**64.8**	66.8	**71.1**	62.1	62.3	64.8	12.3	59.6
瓶颈	72.2	63.7	65.3	68.3	61.2	63.2	**66.6**	12.1	59.0
2 训练轮次									
递增	76.2	69.3	73.2	76.5	65.8	72.2	**74.0**	21.0	66.0
均匀	74.8	70.9	**74.1**	75.6	66.5	73.4	71.2	22.1	66.1
递减	71.4	70.1	72.1	**76.8**	64.3	71.7	73.6	18.7	64.8
纺锤形	**76.6**	**71.9**	71.8	74.4	**67.5**	**73.5**	71.8	22.3	**66.2**
瓶颈	74.2	71.1	69.6	73.3	65.2	73.3	73.6	**24.1**	65.5
3 个训练轮次									
递增	85.3	74.9	77.2	77.5	66.8	76.2	76.0	33.0	70.8
均匀	84.8	73.7	78.1	78.6	68.5	77.8	79.2	36.1	72.1
递减	81.9	72.1	78.3	76.7	67.3	75.9	78.6	28.7	69.9
纺锤形	**86.9**	**75.5**	**79.8**	**79.4**	**69.8**	**78.3**	**80.1**	**47.3**	**74.6**
瓶颈	84.5	74.6	76.9	78.1	69.2	76.2	78.6	32.1	71.3
4个训练周期									
递增	88.3	78.5	80.2	80.5	70.8	80.2	80.0	37.0	74.4
均匀	88.8	78.9	81.9	81.5	71.5	80.8	81.4	39.1	75.4
递减	87.6	74.1	80.8	81.7	79.3	78.9	79.6	38.7	75.1
纺锤形	**89.6**	**79.8**	**83.6**	**82.8**	**71.8**	**81.3**	**82.1**	**39.3**	**76.3**
瓶颈	86.5	77.6	82.7	81.1	70.2	70.9	81.6	36.1	73.3
20个训练周期									
递增	92.3	83.3	86.2	82.5	71.8	82.2	84.0	51.0	79.1
均匀	92.8	83.9	86.1	83.6	72.5	83.8	84.2	52.1	79.9
递减	91.4	82.1	85.1	83.1	69.3	81.7	83.6	48.7	78.1
纺锤形	**93.6**	**84.8**	**87.8**	**84.4**	**73.5**	**84.3**	**85.8**	**52.3**	**80.8**
瓶颈	92.1	82.6	85.6	83.3	71.2	83.2	84.6	52.1	79.3

B XLNet骨干网络在SuperGLUE上的总体效果

我们还直接使用 $\mathcal{S}_4$-model 和 $\mathcal{S}_4$-3b-model（采用通过 T5-base 和 T5-3b 发现的设计模式），在 XLNet-base 和 XLNet-large 预训练骨干模型上进行微调，且不经过任何额外的发现过程。我们保持其他所有设置不变，并在 SuperGLUE 数据集上对它们进行评估。表 17 再次说明，我们从 T5 模型中发现的参数高效微调设计模式能够泛化至 XLNet 骨干模型，并且在其他任务（SuperGLUE）上优于所研究的方法，无需额外的发现过程。

C 发现序列

在这项工作中，我们遵循“分组模式 – 可训练参数分配 – 可调组 – 策略分配”这一发现序列：

1. 为了大规模探索和理解大型预训练模型所有层中的设计模式，有必要也更高效地以组为单位研究这些层。因此，我们从分组模式入手。

表8：使用 T5-base 预训练骨干模型在 GLUE数据集上取得的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了在 $\mathcal{S}_3$ 设计空间中添加不同 G_1 策略分配约束的效果。

策略分配 SST-2		MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_1-适配器 (A)	89.8	83.5	84.9	80.8	72.5	80.8	**78.5**	**37.7**	76.1
G_1-前缀 (P)	89.3	83.1	84.4	80.1	70.1	80.0	77.6	33.0	74.7
G_1-BitFit (B)	89.0	82.9	84.1	81.4	72.0	81.1	77.0	30.8	74.8
G_1-LoRA (L)	89.9	83.6	85.0	81.1	71.8	81.0	78.8	35.3	75.8
G_1-(P, L)	89.1	82.8	85.1	81.2	71.9	81.5	79.1	35.0	75.7
G_1-(A, P)	89.8	82.8	84.8	81.1	72.2	81.3	79.2	36.4	75.9
G_1-(A, L)	89.6	**83.8**	**85.6**	81.3	**72.9**	**81.7**	**79.5**	**36.8**	**76.4**
G_1-(A, P, L)	89.6	83.5	85.2	81.5	72.2	81.4	79.2	35.2	75.9
G_1-(P, B, L)	89.3	83.6	85.5	81.6	72.3	81.0	78.8	35.7	76.0
G_1-(A, P, B)	89.2	83.3	84.8	**81.8**	72.5	81.1	78.6	35.6	75.8
G_1-(A、B、L)	89.8	83.4	84.8	81.1	72.6	81.6	79.4	34.8	75.9
G_1-(A, P, B, L)	**90.0**	83.1	85.3	81.6	72.6	81.4	79.2	36.5	76.1

表9：使用 T5-base 预训练骨干模型在 GLUE数据集上的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较在 $\mathcal{S}_3$ 设计空间中，将不同的 G_2 策略分配约束与 G_1-(L, A)结合的效果。

策略分配 SST-2		MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_2-适配器 (A)	91.6	84.3	85.5	**82.3**	73.5	82.8	81.3	38.8	77.5
G_2-前缀 (P)	89.6	84.0	86.5	81.5	73.3	82.5	80.5	36.2	76.7
G_2-BitFit (B)	91.2	83.6	85.7	82.9	72.6	82.6	80.8	33.1	76.5
G_2-LoRA (L)	91.4	84.4	86.1	82.0	72.8	81.8	81.6	39.8	77.4
G_2-(P, L)	91.6	84.6	86.8	81.8	73.8	82.8	82.0	38.5	77.7
G_2-(A, P)	**92.2**	**84.2**	**87.1**	82.2	**74.4**	83.0	**82.5**	40.8	**78.3**
G_2-(A, L)	92.0	84.4	86.5	81.8	73.6	82.6	82.2	40.1	77.9
G_2-(A, P, L)	91.8	84.8	86.8	81.8	74.1	83.0	82.1	37.9	77.7
G_2-(P, B, L)	91.6	84.1	87.1	82.0	74.0	82.9	82.4	35.8	77.4
G_2-(A, P, B)	91.8	84.2	86.8	82.1	73.7	**83.3**	82.2	41.2	78.1
G_2-(A、B、L)	**92.2**	84.3	86.1	82.0	74.1	83.2	82.0	37.6	77.6
G_2-(A, P, B, L)	92.0	84.1	87.0	81.9	74.2	83.1	81.3	**42.4**	78.1

2. 找出最佳分组模式后，接下来便要探索如何将可训练参数分配给这些不同的组，以便在公平比较的基础上研究更细致的设计（例如，这样便可以比较不同的策略分配模式，而不受可训练参数差异的影响）。

3. 接下来，在深入探讨策略分配模式之前，先考察微调过程中需要学习哪些组，这一点至关重要。因为只有弄清楚需要学习哪些组之后，研究如何为不同组分配策略才有意义。

4. 最后，我们研究调优策略分配，这是最微妙的设计。

表10：使用 T5-base 预训练骨干模型在 GLUE 数据集上的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较在 $\mathcal{S}_3$ 设计空间中添加不同的 G_3 策略分配约束与 G_1-(L, A) – G_2-(P, A) 的效果。

策略分配 SST-2		MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_3-适配器（A）	92.5	85.3	87.5	**83.3**	73.9	84.0	83.8	**44.9**	79.4
G_3-前缀（P）	91.5	84.7	86.7	82.6	74.2	83.8	82.9	40.5	78.4
G_3-BitFit（B）	91.9	84.3	87.0	82.0	73.6	84.1	83.3	36.1	77.8
G_3-LoRA（L）	92.8	85.4	87.8	83.5	74.7	82.4	84.0	44.0	79.3
G_3-(P, L)	93.0	85.2	88.3	83.8	75.2	84.4	84.2	37.9	79.0
G_3-(A, P)	92.4	85.6	88.1	83.6	75.0	84.2	84.0	41.8	79.3
G_3-(A, L)	92.0	85.9	88.2	83.1	75.3	84.3	83.9	42.2	79.4
G_3-(A, P, L)	92.6	86.0	87.5	83.4	75.6	84.6	83.5	43.9	79.6
G_3-(P, B, L)	92.7	85.8	87.2	83.7	75.2	84.5	83.8	40.8	79.2
G_3-**(A, P, B)**	93.3	**85.8**	**88.6**	**84.0**	75.5	**84.9**	84.1	42.1	**79.8**
G_3-(A, B, L)	**93.7**	86.5	88.0	83.2	**75.8**	84.2	84.2	39.7	79.4
G_3-(A, P, B, L)	93.3	85.6	87.7	83.8	75.2	84.3	**84.4**	41.6	79.4

表11：使用 T5-base 预训练骨干模型在 GLUE 数据集上取得的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了向 $\mathcal{S}_3$ 设计空间添加不同 G_4 策略分配约束的效果： G_1-(A, L)、 G_2-(A, P)、 G_3-(A, P, B)。

策略分配 SST-2		MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_4-适配器 (A)	93.8	85.8	88.6	84.8	76.3	85.8	86.0	**48.5**	81.2
G_4-前缀 (P)	93.5	85.2	88.3	83.6	76.8	85.3	85.6	44.8	80.3
G_4-BitFit (B)	94.1	85.3	88.9	84.4	77.1	85.4	86.2	46.1	80.9
G_4-LoRA (L)	94.0	86.0	89.2	85.0	77.2	85.5	85.8	47.7	81.3
G_4-(P, L)	94.3	86.2	89.3	85.8	78.0	86.0	88.2	47.2	81.8
G_4-(A, P)	94.1	86.2	89.6	85.4	77.9	86.2	86.9	45.3	81.4
G_4-(A, L)	94.2	85.9	89.2	85.5	77.8	86.2	88.0	46.8	81.7
G_4-(A, P, L)	94.1	85.8	88.8	85.7	77.4	86.5	87.9	44.8	81.3
G_4-**(P, B, L)**	**94.6**	**86.4**	**90.4**	**86.1**	78.2	**86.8**	**88.5**	47.2	**82.3**
G_4-(A、P、B)	94.5	86.0	89.6	86.0	78.0	86.2	88.1	44.8	81.6
G_4-(A、B、L)	94.3	**86.4**	89.2	85.6	78.2	86.4	88.3	46.6	81.9
G_4-(A, P, B, L)	94.2	86.2	89.2	85.9	**78.5**	86.1	88.0	45.3	81.6

表 12：使用 T5-3b 预训练骨干模型在 GLUE数据集上取得的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了向 $\mathcal{S}_0$ 设计空间添加不同层分组约束的效果。

分组模式	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
$\mathcal{S}_0$-模型	80.3	72.1	74.7	72.8	76.9	75.2	71.0	32.2	69.4
递增	84.4	75.7	83.0	78.3	82.7	80.3	76.3	42.1	75.3
均匀	86.8	77.1	82.6	76.2	83.8	**81.6**	77.3	**48.9**	76.8
递减	83.2	74.3	81.8	77.3	82.8	79.9	76.5	40.8	74.5
纺锤形	**88.6**	**78.8**	**83.7**	77.7	**84.2**	80.9	**78.3**	44.6	**77.1**
瓶颈	86.3	77.0	82.2	75.6	83.3	80.2	77.1	41.5	75.4

表13：使用 T5-3b 预训练骨干模型在 GLUE数据集 上的平均表现（低计算量、低训练轮数设置：100 个随机模型，微调 3 轮）。我们比较了向 $\mathcal{S}_1$ 设计空间添加不同层参数约束的效果。

参数分配	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
递增	90.3	79.3	**84.9**	79.3	85.2	**82.8**	**79.2**	50.1	78.9
均匀	**90.6**	**80.8**	84.6	**79.7**	**85.5**	82.4	78.9	**50.8**	**79.1**
递减	88.6	78.2	83.5	78.1	84.4	81.5	78.1	49.6	77.7

表14：使用 T5-3b 预训练骨干模型在 GLUE 数据集上的平均性能（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了向 $\mathcal{S}_2$ 设计空间添加不同微调组约束的效果。

可调组	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_1	88.3	78.3	82.2	77.4	82.1	80.7	76.1	49.4	76.8
G_2	89.1	78.8	82.1	77.2	82.3	81.2	76.4	49.6	77.1
G_3	89.6	78.5	82.6	78.1	83.8	81.9	77.4	48.7	77.5
G_4	89.8	79.3	82.7	77.9	83.5	81.9	77.9	48.5	77.1
G_1, G_2	90.1	80.2	83.4	78.5	84.3	82.4	78.5	51.1	78.5
G_3, G_4	90.5	80.6	83.8	78.7	84.2	83	78.2	50.3	78.6
G_1, G_2, G_3	90.6	80.3	84.9	79.3	84.7	82.9	79.3	50.2	79.0
G_2, G_3, G_4	90.8	80.9	84.6	79.1	85.1	83.1	79.1	49.2	78.9
$\boldsymbol{G_1, G_2, G_3, G_4}$	**91.1**	**81.4**	**85.2**	**80.4**	**85.9**	**83.5**	**80.0**	**51.6**	**79.9**

表15：使用 T5-3b 预训练骨干模型在 GLUE数据集上取得的平均性能（低计算量、低训练轮数设置：100 个随机模型，微调 3 个训练轮次）。我们按照第4.2.5节中的流程，比较添加不同策略分配约束的效果。

策略分配 SST-2		MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
G_1-适配器 (A)	91.1	81.4	86.1	80.5	86.7	83.3	80.1	50.8	80.0
G_1-前缀 (P)	90.8	81.1	85.5	80.2	86.2	83.1	79.8	50.2	79.6
G_1-BitFit (B)	90.2	81.3	85.1	79.6	85.8	82.8	79.6	49.5	79.2
G_1-LoRA（L）	91.4	81.9	86.2	80.8	86.4	83.9	80.8	49.6	80.0
G_1-（**P, L**）	**91.8**	**82.9**	**86.8**	81.3	**87.1**	84.2	**81.6**	52.3	**81.0**
G_1-（A, P）	91.3	81.9	86.4	81.1	85.6	83.7	80.7	**52.8**	80.1
G_1-（A, L）	91.6	82.3	86.1	**81.5**	85.8	**84.9**	81.5	51.8	80.6
G_1-（A, P, L）	91.1	81.7	85.8	81.2	86.4	84.2	80.9	52.3	80.4
G_1-（P, B, L）	91.5	82.8	86.3	81.4	86.1	83.6	81.2	51.5	80.5
G_1-(A, P, B)	91.3	82.3	86.7	80.8	86.8	84.3	80.7	51.8	80.5
G_1-(A、B、L)	91.7	82.5	86.2	81.3	86.3	84.6	81.3	51.7	80.7
G_1-(A, P, B, L)	91.6	82.3	86.2	81.1	86.6	84.2	81.1	51.1	80.5
G_2-适配器 (A)	92.1	82.5	86.4	81.8	87.2	84.8	81.8	53.8	81.3
G_2-前缀 (P)	91.8	83.1	87.2	81.6	86.2	84.4	81.1	52.8	81.0
G_2-BitFit (B)	91.2	82.1	86.4	81.1	86.3	84.6	80.3	53.1	80.6
G_2-LoRA（L）	92.6	82.9	87.5	81.3	87.4	85.1	81.9	52.2	81.4
G_2-(P, L)	91.6	82.7	87.6	81.6	**87.8**	85.3	82.1	52.8	81.4
G_2-(A, P)	92.1	83.3	87.5	81.9	87.4	85.5	81.8	53.1	81.5
G_2-**(A, L)**	92.5	**83.7**	**88.1**	**82.2**	87.4	**85.7**	**82.9**	53.6	**82.1**
G_2-(A, P, L)	92.3	83.4	87.4	81.6	87.1	85.3	81.4	53.2	81.4
G_2-(P, B, L)	91.8	83.1	87.4	81.5	87.2	85.1	82.7	53.8	81.5
G_2-(A、P、B)	91.5	82.6	87.8	81.3	86.5	85.2	82.1	**54.2**	81.4
G_2-(A、B、L)	92.6	83.5	87.2	82	87.3	86.5	82.5	52.8	81.8
G_2-(A, P, B, L)	**92.8**	83.2	87.6	81.6	87.5	85.5	82.4	51.2	81.5
G_3-适配器 (A)	92.6	84.1	88.3	81.8	87.8	85.4	82.8	55.2	82.2
G_3-前缀 (P)	92.1	83.3	87.6	81.4	87.1	85.4	82.6	53.5	81.6
G_3-BitFit (B)	92.4	83.9	88.4	82.1	87.2	85.8	82.4	53.3	81.9
G_3-LoRA（L）	93.1	84.3	87.7	82.4	87.8	86.2	83.1	54.3	82.3
G_3-(P, L)	92.8	84.1	88.7	82.6	88.2	86.2	83.3	54.7	82.6
G_3-(A, P)	93.1	83.8	89.1	82.3	88.1	85.8	82.6	55.1	82.5
G_3-(A, L)	92.7	84.5	88.4	82.8	88.2	86.1	83.5	54.6	82.6
G_3-(A, P, L)	92.8	84.6	88.1	82.5	87.7	85.5	83.2	53.8	82.3
G_3-**(P, B, L)**	**93.6**	**84.9**	**89.3**	**83.1**	88.2	**86.5**	83.9	**55.8**	**83.2**
G_3-(A, P, B)	93.3	83.9	88.5	82.2	88.4	86.2	83.5	55.3	82.6
G_3-(A、B、L)	93.4	84.2	88.9	82.6	87.8	85.8	**84.2**	54.9	82.7
G_3-(A, P, B, L)	92.2	84.4	88.7	82.3	**88.5**	86.2	**84.2**	54.2	82.5
G_4-适配器 (A)	92.8	85.2	89.1	83.5	87.8	86.5	84.2	56.3	83.2
G_4-前缀 (P)	92.8	84.6	89.5	82.6	87.4	86.5	83.8	55.8	82.8
G_4-BitFit (B)	93.8	84.9	89.5	83.3	88.7	86.8	84.4	55.2	83.3
G_4-LoRA（L）	93.3	84.7	89.3	82.7	88.3	86.2	82.7	54.7	82.7
G_4-（P, L）	93.8	85.3	89.6	83.6	88.6	86.8	84.6	56.3	83.5
G_4-（A, P）	93.8	84.9	89.8	84.3	88.5	86.6	84.8	56.7	83.6
G_4-（A, L）	93.7	85.6	89.5	84.1	88.2	86.6	85.2	55.4	83.5
G_4-（A, P, L）	94.2	85.2	89.6	83.9	88.2	86.4	84.9	55.9	83.5
G_4-（P, B, L）	93.8	**85.9**	89.8	83.6	88.6	86.9	85.2	56.3	83.7
G_4-**(A, P, B)**	**94.4**	85.7	**90.1**	**84.8**	**88.9**	**87.2**	85.3	**57.3**	**84.2**
G_4-(A, B, L)	93.8	85.3	89.5	84.1	88.8	86.7	**85.5**	56.6	83.7
G_4-(A, P, B, L)	94.1	85.4	89.7	84.4	88.5	86.5	85.2	56.8	83.8

表 16：使用 T5-base 预训练骨干模型在 GLUE数据集上的平均表现（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数）。我们比较了向 $\mathcal{S}_0$ 设计空间添加不同层分组约束的效果。层分组基于 8 个组。

层分组	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
$\mathcal{S}_0$-模型	76.9	70.1	72.5	73.3	63.6	71.7	73.8	24.3	65.7
递增	83.2	74.1	76.6	77.1	67.7	76.8	74.7	30.0	70.0
均匀	83.6	73.4	78.0	77.9	68.2	76.4	78.6	34.2	71.3
递减	80.3	71.6	77.4	75.5	67.0	75.3	77.2	26.4	68.9
纺锤形	**86.2**	**74.3**	**79.1**	**78.6**	**68.5**	**77.4**	**79.5**	**35.1**	**72.3**
瓶颈	83.2	73.1	75.8	77.6	67.9	75.3	78.2	31.4	70.3

表 17：分别使用 XLNet-base（上半部分）和 XLNet-large（下半部分）预训练骨干模型，在 SuperGLUE 数据集上采用不同微调方法的性能。结果取 10 次随机运行的平均值。在显著性水平 $p < 0.05$ **(*)**，甚至 $p < 0.01$ **(**)**下，$\mathcal{S}_4$-model 和 $\mathcal{S}_4$-3b-model 在全部八个数据集上的表现均显著优于次优的参数高效微调方法。

方法	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	平均
适配器	72.8	71.3/78.0	64.0	67.0/24.5	71.0/71.8	76.2	65.0	60.8	66.2
前缀	72.0	70.5/77.0	63.3	66.4/23.8	69.9/71.0	75.5	64.4	60.8	65.9
BitFit	71.8	70.0/76.2	62.8	65.8/22.6	69.4/70.6	74.5	64.8	60.6	65.2
LoRA	72.2	71.1/77.8	64.7	67.4/24.8	70.8/71.3	76.8	65.1	61.1	66.4
$\mathcal{S}_4$-模型	$\mathbf{73.8}^{**}$	$\mathbf{71.7/78.4}^{*}$	$\mathbf{65.9}^{**}$	$\mathbf{68.2/25.5}^{**}$	$\mathbf{71.1/72.0}^{*}$	$\mathbf{78.4}^{**}$	$\mathbf{65.8}^{*}$	$\mathbf{62.6}^{*}$	**67.5**
适配器	74.4	71.4/81.1	67.4	68.8/26.4	71.7/72.4	80.8	68.0	64.6	68.8
前缀	72.4	70.0/78.3	66.9	68.8/25.8	70.9/71.2	78.8	66.9	64.0	67.7
BitFit	71.1	70.7/79.8	68.0	68.6/25.4	71.1/71.6	80.4	67.2	64.3	68.1
LoRA	74.1	72.1/80.9	67.9	69.1/26.8	72.0/72.8	81.0	67.8	64.4	69.0
$\mathcal{S}_4$-3b-模型	$\mathbf{76.8}^{**}$	$\mathbf{74.6/81.9}^{**}$	$\mathbf{68.6}^{**}$	$\mathbf{69.5/27.1}^{*}$	$\mathbf{72.4/73.3}^{*}$	$\mathbf{81.2}^{*}$	$\mathbf{68.2}^{**}$	$\mathbf{64.8}^{*}$	**69.7**

表18：使用 8 个 A100 GPU 对 GLUE数据集进行训练的总时间（低计算量、低训练轮数设置：100 个随机模型，3 个微调轮数），采用 T5-base 预训练骨干模型，从 $\mathcal{S}_0$ 到 $\mathcal{S}_1$。

SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA
18 分钟	22 分钟	20 分钟	40 分钟	8 分钟	12 分钟	8 分钟	6 分钟