大规模语言模型的混合和统一微调：在资源受限下的方法和基准测试

Haomin Qi, Zihan Dai, Chengbo Huang
Information Engineering, The Chinese University of Hong Kong, Hong Kong
h.chee@link.cuhk.edu.hk, daizihan@link.cuhk.edu.hk, ch4019@link.cuhk.edu.hk

***Abstract*—微调大型语言模型（LLMs）由于其规模和内存需求，仍然是一个计算瓶颈。本文对参数高效微调（PEFT）技术进行了全面评估，包括 LoRA、BOFT、LoRA-GA 和 uRNN，并介绍了一种新的混合策略，该策略动态整合了 BOFT 的正交稳定性和 LoRA-GA 的梯度对齐快速收敛。通过计算由梯度范数指导的每层自适应更新，混合方法在各种任务中实现了卓越的收敛效率和泛化能力。我们还首次探讨了将全体 RNN（uRNN）原则适用于基于变压器的 LLMs，通过结构化全体约束来增强梯度稳定性。在四个基准——GLUE、GSM8K、MT-Bench 和 HumanEval 上进行的实证评估中，使用从 7B 到 405B 参数的模型结果表明，我们的混合方法一贯优于单独的 PEFT 基线，接近完全微调的精度，同时在训练时间上减少高达 2.1 倍，并在内存使用方面减少 50% 的资源消耗。这些发现确立了混合方法作为一种实用且可扩展的微调解决方案，可在资源受限情况下实际部署 LLMs。**

Keywords*—*Large Language Models, Parameter-Efficient Fine-Tuning, Low-Rank Adaptation, Orthogonal Transformations, Unitary RNN

I. 引言

大型语言模型（LLMs）已成为自然语言处理（NLP）中的基础，推动了从机器翻译到代码生成的应用。然而，微调这些大型模型的成本仍然是一个重要的障碍，特别是在资源受限的环境中。参数高效微调（PEFT）策略 [1] 如低秩自适应（LoRA） [2] 、蝶形正交微调（BOFT） [3] 和梯度感知的 LoRA-GA 被提出以减少可训练参数的数量 [4] ，但它们通常在稳定性、收敛速度和表示能力之间存在权衡。

本论文介绍了一种新颖的混合微调框架，该框架通过基于梯度范数的动态加权，在每层级别上整合了 LoRA-GA 和 BOFT 更新。LoRA-GA 通过奇异值分解（SVD）计算与主要梯度方向对齐的低秩更新，从而实现快速的早期收敛。BOFT 利用反对称矩阵的 Cayley 变换来强制保持正交性，实现稳定的梯度传播。我们的方法融合了两种更新方案，通过自适应系数动态平衡训练过程中的每个组成部分的影响。这种设计确保了初始阶段的快速适应性和后期阶段的稳健稳定性。

此外，我们首次通过嵌入幺正演化 RNN（uRNN）结构 [5] 扩展了微调框架到基于 transformer 的大型语言模型中。具体而言，我们用由傅里叶变换、排列和 Householder 反射参数化的结构化幺正矩阵替换选定的注意力或前馈子层权重。这些幺正约束维持了梯度范数，并在深层架构中提高了训练的鲁棒性。

我们在跨越 7B 到 405B 参数的 LLMs 上，对四个标准基准（GLUE、GSM8K、MT-Bench 和 HumanEval）评估了所提出的混合和 uRNN 增强微调策略。我们的结果表明，混合方法始终优于单独的 PEFT 技术，实现了近乎完全微调的性能，同时将训练时间减少到不到一半，并将内存使用量减少了近 50 % 。uRNN 增强的 transformer 变体进一步有助于梯度稳定性，尤其是在需要长程依赖建模的任务中。

我们的主要贡献如下：

- 我们提出了一种新颖的混合微调算法，该算法动态地融合了 LoRA-GA 的梯度对齐低秩更新和 BOFT 的正交变换。通过基于实时梯度范数计算每层自适应混合系数，我们的方法在训练阶段实现了快速收敛和稳定优化。
- 我们首次将单位演化 RNN（uRNN）原理应用于基于 Transformer 的 LLM，通过将结构化的单位矩阵嵌入到注意力和前馈子层中，从而在微调过程中增强梯度稳定性。
- 我们在四个 PEFT 基准上进行了全面评估——LoRA、BOFT、LoRA-GA 和 uRNN——以及我们的方法，对四个标准的 NLP 和代码生成基准（GLUE、GSM8K、MT-Bench 和 HumanEval）使用四个主要的 LLM 进行评估。

II. 背景与动机

A. 参数高效的大型语言模型微调

大型语言模型（LLMs）的参数数量日益增加，使得完全模型微调在 GPU 内存、训练时间和能耗方面的代价高得令人望而却步。因此，参数高效微调（PEFT）策略作为实用的替代方案出现，旨在减少可训练参数的数量，同时保持下游性能 [6] 。这些方法通常冻结大部分预训练权重，并注入轻量、可学习的组件，以适应新任务。

早期的 PEFT 技术包括适配器模块 [7] ，它在变压器块之间插入瓶颈层，以及前缀微调 [8] ，学习任务特定的提示向量并在输入嵌入前加上前缀。最近，低秩适应（LoRA）引入了可训练的矩阵分解来近似权重更新，在内存使用和任务性能之间提供了良好的权衡。这些技术导致了一种新的模块化、可重用且资源感知的大规模部署微调范式 [9] 。

B. LoRA、BOFT 和 LoRA-GA 方法

LoRA 将权重更新分解为一对低秩矩阵，显著减少可调参数的数量和训练存储 [2] 。它的简洁性和有效性已被广泛采用。然而，LoRA 在整个训练过程中假定固定的低秩结构，这可能会在高度复杂或动态任务中限制适应性。

BOFT 通过对模型权重施加蝶形结构的正交更新来解决训练稳定性问题。其正交性约束有助于保持梯度范数，并减轻训练不稳定性 [10] 。尽管如此，蝶形变换的限制性结构可能会限制表达能力，尤其是在需要非线性适应的任务中。

978-1-6654-8045-1/22/ $ 31.00 ©2022 IEEE

LoRA-GA 基于 LoRA，通过使用梯度对齐奇异值分解（SVD） [4] 来初始化低秩矩阵。这种梯度感知的初始化改进了早期收敛性，但在噪声梯度下可能引入额外的计算成本和不稳定性。

这些方法各自提供了独特的优势，但没有一个能够在单一框架中充分解决鲁棒性、适应性和资源意识的综合需求。

C. 幺正 RNN 与梯度稳定性

幺正 RNN（uRNN）最初是为了解决递归网络中的梯度爆炸和消失问题而提出的 [5] 。通过将转换矩阵限制为幺正，这些模型在长序列中保持梯度幅度。随后的一些工作 [11, 12] 使用傅里叶变换、置换和 Householder 反射提出了高效的参数化方法。

虽然通常用于序列建模，但 uRNN 的稳定性保证在基于 transformer 的模型中具有理论吸引力，尤其是在深层微调过程中。据我们所知，我们是首个将结构化酉矩阵整合到 transformer 层中用于 LLM 微调的，提供了一种在参数高效适配期间稳定梯度的新视角。

D. 资源受限的微调：混合设计的动机

在现实场景中，尤其是对于缺乏高端 GPU 的从业者来说，资源约束如内存占用、计算时间和热预算是微调 LLMs [13, 14] 时的主要瓶颈。虽然 PEFT 方法减少了参数数量，但它们往往会带来一个权衡：LoRA 及其变体收敛速度快，但在较深层可能出现不稳定性，而像 BOFT 这样的正交方法则能够保持训练动态但收敛速度较慢。

越来越多的研究尝试对单一方法的 PEFT 技术进行基准测试或扩展，但相对较少的研究探讨如何以动态和结构化的方式结合这些优势。我们的工作并没有引入另一种静态的 PEFT 变体，而是提出了一种混合机制，该机制基于实时的梯度反馈分配每层更新责任——在早期阶段倾向于低秩的速度，后来则倾向于正交的稳定性。

在本节中，我们首先回顾作为基线 PEFT 方法的 LoRA、BOFT 和 LoRA-GA 的数学原理。然后，我们介绍两个主要贡献：一种兼容 Transformer 的 uRNN 结构性单位约束适应方法，以及一种基于梯度反馈动态融合低秩和正交更新的混合微调策略。

E. 低秩适应 (LoRA)

LoRA 引入了对预训练权重矩阵的低秩更新，大大减少了可训练参数的数量，同时保留了预训练模型的权重。权重更新表达为：

$$\boldsymbol{W}' = \boldsymbol{W}_0 + \Delta\boldsymbol{W}, \quad \Delta\boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}, \tag{1}$$

，其中 $\boldsymbol{W}_0 \in \mathbb{R}^{d\times k}$ 表示被冻结的预训练权重矩阵，$\Delta\boldsymbol{W}$ 是由 $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ 和 $\boldsymbol{A} \in \mathbb{R}^{r\times k}$ 参数化的低秩更新，从而将可训练参数的数量从 $\mathcal{O}(dk)$ 减少到 $\mathcal{O}(r(d+k))$ ，加上 $r \ll \min(d,k)$ 。

优化：在微调期间，只优化 $\boldsymbol{B}$ 和 $\boldsymbol{A}$ ，而保持 $\boldsymbol{W}_0$ 不变 [15, 16] 。这种分解在保持性能的同时减少了微调的内存和计算成本。

为了确保数值稳定性，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的范数受到 r 阶的约束：

$$\|\Delta\boldsymbol{W}\|_F \le \lambda \cdot \|\boldsymbol{W}_0\|_F, \tag{2}$$

其中 λ 是一个缩放因子。这防止在优化过程中更新发散。

F. 蝶形正交微调 (BOFT)

BOFT 将一个方阵权重矩阵分解为（近似）正交的稀疏蝴蝶块的乘积，从而实现参数效率（$\mathcal{O}(d\log d)$ 个参数）和稳定的梯度范数。

$$\boldsymbol{W} = \prod_{i=1}^{m} \boldsymbol{B}_i, \qquad \boldsymbol{B}_i \in \mathbb{R}^{d\times d}. \tag{3}$$

每个 $\boldsymbol{B}_i$ 是由成对的行置换-乘法操作构建的，这些操作模仿快速傅里叶变换层次结构 [17] ；一个简化的两级形式是

$$\boldsymbol{B}(d,2) = \begin{bmatrix} \boldsymbol{I}_{d/2} & \boldsymbol{0} \\ [2pt]\boldsymbol{0} & \boldsymbol{I}_{d/2} \end{bmatrix} \boldsymbol{F}_d \begin{bmatrix} \boldsymbol{I}_{d/2} & \boldsymbol{0} \\ [2pt]\boldsymbol{0} & \boldsymbol{I}_{d/2} \end{bmatrix}, \tag{4}$$

，其中 $\boldsymbol{F}_d$ 是一个（可学习的）正交混合矩阵。

优化：每个 $\boldsymbol{B}_i$ 被初始化为近似单位变换，并通过梯度下降更新 [18] ：

$$\boldsymbol{B}_i^{t+1} = \boldsymbol{B}_i^t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{B}_i^t}, \tag{5}$$

，其中 η 是学习率。正交性通过投影步骤在更新后进行强制：

$$\boldsymbol{B}_i \leftarrow \mathrm{Proj}_{\mathrm{orthogonal}}(\boldsymbol{B}_i). \tag{6}$$

正交约束可以抑制深层堆栈中的梯度爆炸或消失现象，使其在具有深层或复杂梯度的任务中特别有效。

G. 具有梯度近似的低秩自适应（LoRA-GA）

LoRA-GA 通过将低秩更新与完整模型的梯度对齐来改进 LoRA，从而实现更快的收敛和更好的优化。损失 $\mathcal{L}$ 对固定权重矩阵 $\boldsymbol{W}_0$ 的梯度被分解为：

计算 r 截断的 $\nabla_{\boldsymbol{W}_0}\mathcal{L}$ 的秩-截断奇异值分解：

$$\nabla_{\boldsymbol{W}_0}\mathcal{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \qquad \boldsymbol{U} \in \mathbb{R}^{d\times r}, \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{k\times r}. \tag{7}$$

低秩矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 被初始化为：

$$\boldsymbol{A}_0 = \boldsymbol{U}\boldsymbol{\Sigma}^{1/2}, \quad \boldsymbol{B}_0 = \boldsymbol{V}\boldsymbol{\Sigma}^{1/2}. \tag{8}$$

。这确保了初始更新与主要梯度方向一致，加速了收敛。

优化：初始化后，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 使用标准梯度下降法在 [19] 中迭代更新：

$$\boldsymbol{A}^{t+1} = \boldsymbol{A}^t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{A}^t}, \quad \boldsymbol{B}^{t+1} = \boldsymbol{B}^t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{B}^t}. \tag{9}$$

通过将最初的低秩更新与最具影响力的梯度方向对齐，LoRA-GA 减少了收敛所需的训练迭代次数，使其特别适合资源受限的情境。

H. 酉演化循环神经网络 (uRNN)

单位旋转递归神经网络（uRNN）通过将隐藏到隐藏的权重矩阵限制为单位矩阵，以缓解梯度消失和爆炸问题 [11, 12] 。通过确保转移矩阵的特征值位于单位圆上，uRNN 在反向传播中保持梯度范数，从而实现对长期依赖性的学习。

uRNN 的核心组件是一个可学习的酉矩阵 U ，用于演化隐藏状态。为了确保 U 是酉矩阵，我们采用了一种结构化参数化方法 [5] :

$$\boldsymbol{U} = \boldsymbol{D}_3\boldsymbol{R}_2\boldsymbol{F}^{-1}\boldsymbol{D}_2\Pi\boldsymbol{R}_1\boldsymbol{F}\boldsymbol{D}_1, \tag{10}$$

，其中 F （和 F^{-1} ）是固定的酉傅里叶变换矩阵，Π 是固定的置换矩阵，D_i 和 R_i 表示可训练的对角相位矩阵和 Householder 反射矩阵 [20] 。这种分解显著减少了自由参数的数量（对于一个 $n \times n$ 矩阵减少到 $O(n)$ 个），并且允许通过快速傅里叶变换操作有效地进行矩阵-向量乘积的 $O(n \log n)$ 计算。

用于微调 LLMs 的自适应：据我们所知，我们是首个将 uRNN 原理整合到基于 Transformer 的 LLMs 微调中的研究。其动机是利用酉变换来稳定梯度传播，并在微调过程中更好地捕捉长距离依赖关系。在实践中，我们将可学习的酉矩阵纳入到一些选定的 Transformer 子层中以增强训练的稳定性。具体来说，我们用酉矩阵替换某些权重矩阵（例如，在注意力头或前馈块中），并修改训练过程以保持它们的酉性。每个这样的酉权重初始化为类似于单位矩阵（接近于单位矩阵）的形式，以确保稳定收敛。在反向传播期间，我们包括一个高效的再投影步骤（详见下文），以在任何时候都保持这些权重的酉性。这种方法将酉 RNN 技术扩展到其原始领域之外，建立了一个参数高效地微调 LLMs 的新范式。

通过重新投影的梯度更新：我们在酉矩阵的流形上通过梯度下降训练单位权重 U 。设 $\nabla_U L$ 为损失 L 对 U 的梯度（由反向传播计算得到）。我们首先从梯度构造一个斜厄密矩阵 B （即 $B^H = -B$ ）:

$$\boldsymbol{B} = \nabla_{\boldsymbol{UL}}\,\boldsymbol{U}^{\boldsymbol{H}} - \boldsymbol{U}\,(\nabla_{\boldsymbol{UL}})^{\boldsymbol{H}}, \tag{11}$$

，其中 U^H 表示 U 的共轭转置。按构造，B 位于酉群的李代数 $\mathfrak{u}(n)$ 中。换句话说，B 是斜厄密矩阵，因此对于任意实数步长，$\exp(\eta B)$ 是酉矩阵。然后我们通过酉旋转更新 U :

$$\boldsymbol{U}_{t+1} = \exp(\eta\,\boldsymbol{B})\,\boldsymbol{U}_{\boldsymbol{t}}, \tag{12}$$

，其中 η 是学习率。这个指数映射更新保证 U_{t+1} 保持酉性。在实现中，$\exp(\eta B)$ 可以通过截断的泰勒级数或缩放-平方算法高效近似，我们根据需要重新归一化 U ，以纠正由酉性引起的任何数值漂移。

微调算法：算法 1 概述了使用 uRNN 原则的整体微调过程。我们在选定的 Transformer 层的前向传递中应用 U ，然后在每个训练步骤中使用上述规则更新 U ，而其他模型权重则按照通常的方式进行更新。值得注意的是，尽管 uRNNs 最初是为递归序列模型设计的，我们的策略是将这些单位约束应用于 Transformer 结构中的前馈或注意力层。从概念上讲，通过 transformer 子层的每一次前向传递类似于 RNN 的单步，其中子层的输入相当于“隐藏状态”。保持 U 为单位因此有助于通过网络深度保持梯度范数，即使没有显式的递归 [21] 。[1]

[1]在实践中，我们将每个酉约束子层的输入视为 RNN 隐状态的代理，这确保了跨多个 transformer 层的稳定反向传播。

Algorithm 1 基于 uRNN 的微调程序

Initialize: Unitary matrix $\boldsymbol{U}$ using structured parameterization:
  Set $\boldsymbol{F}$, $\boldsymbol{F}^{-1}$, Π as fixed matrices; initialize diagonal phase matrices $\boldsymbol{D}_i$ and Householder reflection matrices $\boldsymbol{R}_i$ near identity.
Choose learning rate η and total epochs E .
for *epoch* $= 1$ **to** E **do**
  for each minibatch in the dataset **do**
    Forward Pass:
    for each transformer layer with unitary-constrained weight **do**
      Replace original weight matrix with current unitary matrix $\boldsymbol{U}$.
      Compute the forward pass using the updated unitary matrix $\boldsymbol{U}$.
    end for
    Compute the task-specific loss $\mathcal{L}$ based on current minibatch predictions.
    Backward Pass:
    Compute gradient $\nabla_{\boldsymbol{U}}\mathcal{L}$ via backpropagation.
    Construct skew-Hermitian matrix $\boldsymbol{B}$:
      $\boldsymbol{B} = \nabla_{\boldsymbol{U}}\mathcal{L}\,\boldsymbol{U}^H - \boldsymbol{U}\,(\nabla_{\boldsymbol{U}}\mathcal{L})^H$,
    where $\boldsymbol{U}^H$ denotes conjugate transpose of $\boldsymbol{U}$.
    Update the unitary matrix via matrix exponential:
      $\boldsymbol{U} \leftarrow \exp(\eta\boldsymbol{B})\,\boldsymbol{U}$
      (Use truncated Taylor series or scaling-and-squaring approximation for efficiency)
    If numerical drift occurs, re-normalize $\boldsymbol{U}$ to strictly enforce unitarity.
    Update all other non-unitary parameters of the model as usual via standard gradient descent.
  end for
end for

将 uRNN 的原则整合到 Transformer 的微调中提供了几个显著的好处。首先，强制单位变换确保了梯度的稳定性：即使在非常深的网络或具有长程依赖的任务中，它也能防止梯度的幅度消失或爆炸。

其次，该方法倾向于在微调过程中改善收敛性，因为稳定的梯度范数有助于加快和更可靠的训练（减少达到给定性能水平所需的迭代次数）。

第三，这种方法可以适应不同的模型组件；酉约束可以在不改变架构的情况下应用于 LLM 的各种层或子层（例如，注意力投影或前馈模块），从而将正交变换的使用扩展到其传统的递归设置之外。通过将酉变换扩展到基于变压器的 LLM，我们建立了一种新的微调范式，将参数效率与训练稳定性结合起来。

I. 混合微调方法

我们提出了一种逐层融合 LoRA-GA 和 BOFT 更新的方法，以捕捉低秩和正交规范的适应模式。具体来说，这种混合策略在每一层的相同权重矩阵中，计算来自 LoRA-GA 的梯度对齐的低秩更新和来自 BOFT 的结构化正交更新，然后使用一个动态系数将它们混合。直观上，这允许通过低秩组件进行快速初始适应，同时随着训练的进行，逐渐将重点转向 BOFT 组件以稳定学习。

数学公式：考虑一个具有预训练权重矩阵 $\boldsymbol{W}^\ell \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ 的层 ℓ 。我们引入低秩因子 $\boldsymbol{A}^\ell \in \mathbb{R}^{d_{\text{out}} \times r}$ 和 $\boldsymbol{B}^\ell \in \mathbb{R}^{r \times d_{\text{in}}}$ （秩为 r ），类似于 LoRA [4] ，并引入一

Algorithm 2 混合微调程序

Initialize: pretrained weights $\{\boldsymbol{W}^\ell\}$, low-rank matrices $\{\boldsymbol{A}^\ell, \boldsymbol{B}^\ell\}$ for LoRA-GA, skew-symmetric matrices $\{\boldsymbol{Q}^\ell\}$ for BOFT.
Set learning rates $\eta_{\text{LoRA}}, \eta_{\text{BOFT}}$; choose rank r , total epochs E .
for *epoch* $= 1$ **to** E **do**
  for each minibatch in the dataset **do**
    Forward Pass:
    for each transformer layer ℓ **do**
      Compute LoRA-GA update: $\Delta \boldsymbol{W}^\ell_{\text{LoRA}} = \boldsymbol{A}^\ell \boldsymbol{B}^\ell$.
      Compute BOFT orthonormal matrix:
      $\boldsymbol{R}^\ell = (\boldsymbol{I} + \eta_{\text{BOFT}} \boldsymbol{Q}^\ell)(\boldsymbol{I} - \eta_{\text{BOFT}} \boldsymbol{Q}^\ell)^{-1}$.
      Compute BOFT update:
      $\Delta \boldsymbol{W}^\ell_{\text{BOFT}} = (\boldsymbol{R}^\ell - \boldsymbol{I}) \boldsymbol{W}^\ell$.
    end for
    Compute task-specific loss $\mathcal{L}$ using model predictions.
    Backward Pass:
    for each transformer layer ℓ **do**
      Compute gradient norms:
      $g^\ell_{\text{LoRA}} = \|\nabla_{\boldsymbol{A}^\ell, \boldsymbol{B}^\ell} \mathcal{L}\|$, $g^\ell_{\text{BOFT}} = \|\nabla_{\boldsymbol{Q}^\ell} \mathcal{L}\|$.
      Compute dynamic weighting coefficient:
      $\lambda^\ell = \frac{g^\ell_{\text{LoRA}}}{g^\ell_{\text{LoRA}} + g^\ell_{\text{BOFT}}}$.
      Form hybrid update for layer ℓ :
      $\Delta \boldsymbol{W}^\ell_{\text{hybrid}} = \lambda^\ell \Delta \boldsymbol{W}^\ell_{\text{LoRA}} + (1 - \lambda^\ell) \Delta \boldsymbol{W}^\ell_{\text{BOFT}}$.
      Update weight matrix for layer ℓ :
      $\boldsymbol{W}^\ell \leftarrow \boldsymbol{W}^\ell + \Delta \boldsymbol{W}^\ell_{\text{hybrid}}$.
      Update low-rank matrices via gradient descent:
      $\boldsymbol{A}^\ell \leftarrow \boldsymbol{A}^\ell - \eta_{\text{LoRA}} \nabla_{\boldsymbol{A}^\ell} \mathcal{L}$,
      $\boldsymbol{B}^\ell \leftarrow \boldsymbol{B}^\ell - \eta_{\text{LoRA}} \nabla_{\boldsymbol{B}^\ell} \mathcal{L}$.
      Compute skew-symmetric gradient matrix for BOFT:
      $\boldsymbol{G}^\ell = \nabla_{\boldsymbol{Q}^\ell} \mathcal{L} - (\nabla_{\boldsymbol{Q}^\ell} \mathcal{L})^\top$.
      Update skew-symmetric matrix $\boldsymbol{Q}^\ell$:
      $\boldsymbol{Q}^\ell \leftarrow \boldsymbol{Q}^\ell - \eta_{\text{BOFT}} \boldsymbol{G}^\ell$.
      Recompute orthonormal matrix $\boldsymbol{R}^\ell$ via Cayley transform (for numerical stability):
      $\boldsymbol{R}^\ell \leftarrow (\boldsymbol{I} + \eta_{\text{BOFT}} \boldsymbol{Q}^\ell)(\boldsymbol{I} - \eta_{\text{BOFT}} \boldsymbol{Q}^\ell)^{-1}$.
    end for
    Update other model parameters (if any) via standard gradient descent.
  end for
end for

个斜对称矩阵 $\boldsymbol{Q}^\ell \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$ （$\boldsymbol{Q}^\ell = -\boldsymbol{Q}^{\ell\top}$ ），类似于 BOFT [3] 。低秩 LoRA-GA 更新为

$$\Delta \boldsymbol{W}^\ell_{\text{LoRA}} = \boldsymbol{A}^\ell \boldsymbol{B}^\ell.$$

正交 BOFT 更新通过 Cayley 变换 [10] 获得：

$$\boldsymbol{R}^\ell = (\boldsymbol{I} + \eta \boldsymbol{Q}^\ell)(\boldsymbol{I} - \eta \boldsymbol{Q}^\ell)^{-1}, \qquad \boldsymbol{Q}^\ell = -\boldsymbol{Q}^{\ell\top},$$

这确保了 $\boldsymbol{R}^\ell$ 是正交的（对于小步长 η ）。对 $\boldsymbol{W}^\ell$ 的 BOFT 更新为

$$\Delta \boldsymbol{W}^\ell_{\text{BOFT}} = (\boldsymbol{R}^\ell - \boldsymbol{I}_{d_{\text{out}}}) \boldsymbol{W}^\ell.$$

我们结合一个随着训练步骤 t 而自适应的分层混合系数 $\lambda^\ell_t \in [0, 1]$ 。具体地，我们设置

$$\lambda^\ell_t = \frac{\|\nabla_{\boldsymbol{A}^\ell, \boldsymbol{B}^\ell} L(\theta_t)\|}{\|\nabla_{\boldsymbol{A}^\ell, \boldsymbol{B}^\ell} L(\theta_t)\| + \|\nabla_{\boldsymbol{Q}^\ell} L(\theta_t)\|},$$

这样梯度范数较大的分量将获得更高的权重。混合更新为

$$\Delta \boldsymbol{W}^\ell_{\text{hybrid}} = \lambda^\ell_t \Delta \boldsymbol{W}^\ell_{\text{LoRA}} + (1 - \lambda^\ell_t) \Delta \boldsymbol{W}^\ell_{\text{BOFT}},$$

并且权重更新为 $\boldsymbol{W}^\ell \leftarrow \boldsymbol{W}^\ell + \Delta \boldsymbol{W}^\ell_{\text{hybrid}}$ 。这里 $\nabla_{\boldsymbol{A}^\ell, \boldsymbol{B}^\ell} L$ 表示相对于 LoRA 参数 [22] 的训练损失 $L(\theta_t)$ 的梯度，$(\boldsymbol{A}^\ell, \boldsymbol{B}^\ell)$ 以及相对于 $\boldsymbol{Q}^\ell$ 的梯度 $\nabla_{\boldsymbol{Q}^\ell} L$ 。以上所有符号均按层定义 ℓ 。

伪代码算法：算法 2 总结了混合微调更新。在每次迭代中，我们为每一层计算 LoRA-GA 和 BOFT 更新，计算混合系数 λ^ℓ_t ，然后应用加权组合来更新权重。

每层的混合融合仅在个别的 LoRA-GA 和 BOFT 更新之外增加了适度的开销。在每一层中，低秩更新的成本是 $\mathcal{O}(d_{\text{out}} r + r\, d_{\text{in}})$ ，而 BOFT 转换的成本是 $\mathcal{O}(d_{\text{out}} \log d_{\text{out}})$ 。计算 λ^ℓ_t 仅需要已经计算出的梯度的范数，这是可以忽略不计的。

可调参数的总数是 LoRA 因子和任何 BOFT 参数（例如，蝴蝶因子）的总和，这类似于独立使用这两种方法。通过构造，Cayley 参数化强制 $\boldsymbol{R}^\ell$ 为正交规范，这有助于在优化过程中保持梯度范数。因此，混合更新在一个统一的步骤中集成了快速低秩适应和结构化正交调整。

III. 实验

本部分评估所提出的微调策略在不同任务和模型中的效果，以调查其有效性、可扩展性和计算效率。实验设计主要集中在系统地比较 LoRA、BOFT、LoRA-GA、uRNN 以及混合方法与完整微调（Full FT）基线的表现。此外，我们探索任务特定收益和资源消耗之间的权衡，以提供对所提出方法实用性的全面理解。

A. 设置和评估指标

为了确保全面和公正的评估，对 4 种不同参数规模和架构特性的最新大语言模型（LLMs）进行了实验：

- Llama3.1-405B：一个前沿的变换器模型，旨在扩展上下文理解，包含 4050 亿个参数。
- Llama3.3-70B：一个中等规模的模型，具有 700 亿个参数，平衡了计算效率和高准确性。
- Wizard-Vicuna-30B：一个拥有 300 亿参数的多语言模型，经过指令跟随任务的微调。
- BloomZ-7B1：一个拥有 71 亿参数的紧凑型模型，专为资源受限环境中的部署而优化。

每个模型都在四个基准数据集上进行了测试，选择这些数据集是为了评估广泛的自然语言处理能力：

- GLUE 基准：一个全面的通用 NLP 任务集，包括 MNLI、QQP、SST-2 和 QNLI，用于评估分类和语言理解能力 [23] 。
- GSM8K：一个数学推理问题的数据集，旨在评估多步推理和算术能力 [24] 。
- MT-Bench：一个多语种机器翻译基准，使用 BLEU 和 ROUGE-L 指标测试翻译质量 [25] 。
- HumanEval：一个用于评估生成程序功能正确性的代码生成基准测试，采用 pass@ k 度量作为主要评估标准 [26] 。

每种微调方法在相同条件下应用于所有模型，以确保公平比较。实验包括：

a) 硬件和软件配置： 所有实验都在一个集群上进行，该集群配备了双 AMD EPYC 7742 CPU 和每节点 1 TB 的内存，节点之间通过 InfiniBand 进行高速通信。每个节点配备了八个通过 NVLink 连接的 NVIDIA A100 GPU。软件环境包括 PyTorch 2.0、CUDA 11.8 和 NVIDIA Apex，支持混合精度以实现高效训练。

b) 超参数调优： 每种方法的超参数都是基于初步的网格搜索进行调优的。例如，LoRA 使用了秩 $r = 16$ 和缩放因子 $\alpha = 32$ ，而 BOFT 采用了三个蝶形分解层次 ($m = 3$)。对于混合方法，梯度加权因子 α_t 在训练过程中动态调整。

c) 实验重复： 每个实验重复进行三次，以减轻随机初始化的影响并确保统计上稳健的结果。报告的指标也代表了这些运行的平均表现。

B. 基准特定结果和分析

GLUE Benchmark: GLUE 基准测试结果总结在表格 I 中。混合方法在所有模型规模上表现出一致的优越性，有效地结合了 LoRA-GA 的快速适应性和 BOFT 的结构化稳定性。在最大的模型 Llama3.1-405B 上，混合方法取得了平均得分 92.3 % ，接近于完整微调（Full FT）基准的 92.5 % 。这表明混合方法可以捕捉到细微的语言模式而不需要完整微调相关的高计算成本。

在中型 Llama3.3-70B 模型上，Hybrid 方法以 0.8 % 超过 BOFT，以 0.6 % 超过 LoRA-GA，突显了其快速收敛和强大泛化的平衡能力。较小的模型如 BloomZ-7B1，从 Hybrid 策略中显著获益，比 LoRA 提高了 1.1 %，进一步缩小了与 Full FT 的性能差距，同时保持参数效率。

BOFT 在 MNLI 和 QNLI 等语言复杂任务上实现了强大的泛化能力，其中结构化正交更新增强了梯度稳定性。同时，LoRA-GA 在需要快速特征提取的任务如 SST-2 和 QQP 中表现出色。然而，两种方法在实现适应性和稳定性方面同时存在局限性，而混合方法有效地解决了这些问题。

GSM8K 基准测试：GSM8K 数据集评估多步骤算术和推理能力，结果如表 I 所示。在所有模型规模中，混合方法 (GSM8K Benchmark) 实现了最高的准确率，其中在 Llama3.1-405B 上达到了显著的 55.9 %，比 Full FT 高出 0.2 %，比 LoRA 高出 1.7 %。这些结果反映了混合方法在复杂推理任务中适应能力，同时保持了稳定的梯度传播。

对于中型模型，例如 Llama3.3-70B，Hybrid 方法比 LoRA-GA 提高了 0.9 %，比 BOFT 提高了 1.0 %，表明其在收敛速度和结构稳健性之间的平衡整合。在较小的模型如 Wizard-Vicuna-30B 和 BloomZ-7B1 上，Hybrid 方法持续优于其他方法，缩小了与 Full FT 的性能差距，同时实现了资源高效的微调。

混合策略在较小模型上的显著优势突出了其动态平衡特征适应和稳定优化的能力，使其特别适合复杂的推理任务。

MT-Bench 基准：MT-Bench 数据集评估模型执行多语言翻译任务的能力，强调语言的忠实度和语义一致性。表格 I 展示了不同微调方法在四种模型规模下的 BLEU 分数。混合方法表现优异，始终优于完整微调和独立的高效参数方法。在 Llama3.1-405B 上，混合方法的 BLEU 分数为 29.4，略微超过完整微调 0.1。对于像 BloomZ-7B1 这样的小型模型，混合方法比 LoRA-GA 提高了 0.7，反映了其在资源受限环境下的适应性。

Hybrid 方法与 uRNN 之间的性能差距突显了在跨语言任务中应用单位矩阵变换的挑战。尽管 uRNN 在长序列中保持梯度稳定性方面表现出色，但它缺乏多语言翻译所需的适应性。

研究结果还表明，混合方法成功地结合了 BOFT 和 LoRA-GA 的优势。BOFT 的结构化正交更新在处理多样的语言模式时保证了稳定性和一致性，而 LoRA-GA 的梯度对齐加速了早期收敛，尤其是在多语言数据集上。例如，混合方法在 Wizard-Vicuna-30B 上比独立的 LoRA 提高了 0.9，强调了其在平衡稳定性和收敛速度方面的能力。

在较小的模型中，例如 BloomZ-7B1，Hybrid 方法通过实现 25.4 的 BLEU 分数展示了其效率。鉴于较小模型固有的参数限制，这一表现值得注意，而传统的全面微调在充分利用其计算强度方面举步维艰。在 Llama3.3-70B 模型上，Hybrid 方法实现了 28.1 的 BLEU 分数，分别比 LoRA-GA 和 BOFT 高出 0.8 和 0.6。这表明了 Hybrid 策略在不牺牲效率的情况下，能够跨不同参数规模进行泛化的能力。

HumanEval 代码生成：[2] HumanEval 数据集评估模型生成功能正确代码的能力，指标主要关注通过率 @1 和通过率 @10 的准确性 [3] 。表格 I 展示了所有方法和模型规模的结果，强调了混合方法作为表现最佳的方法。在 Llama3.1-405B 上，混合方法实现了 62.5 % 的通过率 @1 准确性，优于完整微调 0.5 %。值得注意的是，对于较小的模型，如 BloomZ-7B1，混合方法缩小了与完整微调整个性能差距，实现了 40.5 % 通过率 @1，比 BOFT 提高 1.0 %，比 LoRA 提高 1.5 %。

混合方法通过将 LoRA-GA 的快速收敛特性与 BOFT 的稳定性相结合而表现出色。这种组合特别适合生成功能正确的代码，因为它解决了快速适应任务特定细微差别和在较长训练期间保持稳健学习动态的双重挑战。例如，混合方法在 Wizard-Vicuna-30B 上展示了比 BOFT 在 pass@10 准确率上提高 0.7 %，反映出其在不同代码模式之间的泛化能力。

像 BloomZ-7B1 这样的小型模型显著受益于混合方法的高效性。由于计算开销，这些模型在传统的全参数微调中经常遇到困难。混合方法利用参数高效策略，在不增加过多资源需求的情况下提供卓越的性能，使其成为资源受限场景中的理想选择。

C. 资源和性能分析

本小节详细评估了四种模型规模（Llama3.1-405B、Llama3.3-70B、Wizard-Vicuna-30B 和 BloomZ-7B1）上六种微调方法（Full FT、LoRA、BOFT、LoRA-GA、uRNN、Hybrid）的训练时间、GPU 内存使用、梯度范数和验证损失。图 1 和图 2 所示的结果突出了计算效率与任务特定泛化之间的权衡关系。

图 1 比较了每个 epoch 的训练时间和 GPU 内存使用情况。与 Full FT 相比，Hybrid 方法在训练时间上实现了 2.1 倍的加速，并且在所有模型大小上都有一致的减

[2]由于计算资源的限制以及在 HumanEval 基准测试中所选大型语言模型的相似性评估性能，同样属于 Llama3 系列的 Llama3.3-70B 模型未在 HumanEval 代码生成实验中重复使用。

[3]HumanEval 以百分比形式报告性能为一个 (pass@ 1，pass@ 10) 的组。这里，pass@1 表示模型的 top-1 生成的解决方案功能正确的问题的百分比，而 pass@10 表示前十个生成的解决方案中有任何一个正确的问题的百分比。在表格 I 中，对每个模型-方法组合，左侧数字表示 pass@1，右侧数字表示 pass@10。

TABLE I. 性能比较：GLUE、GSM8K、MT-BENCH 和 HUMANEVAL 基准测试的每次运行结果及平均值

Benchmark	Method	Llama3.1-405B	Llama3.3-70B	Wizard-Vicuna-30B	BloomZ-7B1
GLUE (%)	Full FT	91.0 / 94.0 / 92.5	90.0 / 91.0 / 91.1	87.8 / 90.0 / 89.0	84.9 / 86.2 / 86.3
	Avg	92.5	90.7	88.9	85.8
	LoRA	90.2 / 91.5 / 92.2	88.9 / 89.2 / 89.3	87.3 / 87.4 / 87.8	83.7 / 84.2 / 84.1
	Avg	91.3	89.1	87.5	84.0
	BOFT	91.5 / 92.0 / 91.6	89.1 / 89.8 / 89.3	87.7 / 87.9 / 87.8	84.0 / 84.5 / 84.4
	Avg	91.7	89.4	87.8	84.3
	LoRA-GA	91.4 / 92.3 / 92.0	89.4 / 89.8 / 89.6	87.6 / 87.9 / 88.2	84.1 / 84.3 / 84.8
	Avg	91.9	89.6	87.9	84.4
	uRNN	90.1 / 91.5 / 91.1	88.0 / 88.8 / 88.7	86.0 / 86.7 / 87.5	82.6 / 83.9 / 84.0
	Avg	90.9	88.5	86.7	83.5
	Hybrid	91.7 / 93.1 / 92.1	89.8 / 90.3 / 90.4	87.9 / 88.6 / 88.7	84.6 / 85.2 / 85.4
	Avg	92.3	90.2	88.4	85.1
GSM8K (%)	Full FT	55.6 / 56.0 / 55.5	53.0 / 53.5 / 52.8	51.3 / 51.9 / 51.2	48.7 / 49.0 / 49.0
	Avg	55.7	53.1	51.5	48.9
	LoRA	53.8 / 54.4 / 54.3	51.1 / 51.9 / 51.5	50.6 / 51.0 / 50.8	47.8 / 48.2 / 48.0
	Avg	54.2	51.5	50.8	48.0
	BOFT	54.7 / 55.0 / 54.7	51.9 / 52.1 / 52.0	51.0 / 51.2 / 51.4	48.4 / 48.5 / 48.6
	Avg	54.8	52.0	51.2	48.5
	LoRA-GA	54.2 / 54.8 / 54.8	52.1 / 52.4 / 52.0	51.3 / 51.6 / 51.2	48.5 / 48.8 / 48.7
	Avg	54.6	52.2	51.4	48.7
	uRNN	54.2 / 54.7 / 54.6	51.7 / 51.9 / 51.8	50.7 / 51.1 / 51.2	48.0 / 48.4 / 48.2
	Avg	54.5	51.8	51.0	48.2
	Hybrid	55.3 / 56.0 / 56.4	52.8 / 53.1 / 53.0	51.7 / 52.1 / 52.5	48.7 / 49.4 / 49.8
	Avg	55.9	53.0	52.1	49.3
MT-Bench (BLEU)	Full FT	28.7 / 29.8 / 29.4	27.5 / 28.2 / 27.9	26.0 / 26.8 / 26.4	24.5 / 25.0 / 24.8
	Avg	29.3	27.9	26.4	24.8
	LoRA	28.4 / 29.1 / 28.5	27.0 / 27.6 / 27.3	25.4 / 26.0 / 25.9	23.8 / 24.4 / 24.7
	Avg	28.7	27.3	25.8	24.3
	BOFT	28.6 / 29.2 / 29.0	27.1 / 27.8 / 27.5	25.6 / 26.2 / 26.0	24.0 / 24.6 / 24.9
	Avg	28.9	27.5	26.0	24.5
	LoRA-GA	28.7 / 29.3 / 29.1	27.2 / 27.8 / 27.7	25.9 / 26.4 / 26.2	24.2 / 24.7 / 25.2
	Avg	29.0	27.7	26.2	24.7
	uRNN	28.2 / 29.1 / 28.5	26.7 / 27.5 / 27.1	25.2 / 26.1 / 25.7	23.7 / 24.5 / 24.4
	Avg	28.6	27.1	25.7	24.2
	Hybrid	29.0 / 29.6 / 29.7	27.8 / 28.3 / 28.1	26.4 / 26.8 / 27.0	25.1 / 25.7 / 25.4
	Avg	29.4	28.1	26.7	25.4
HumanEval	Full FT	61.8 / 62.2 / 62.0 / 77.3 / 77.8 / 77.5	—	54.0 / 54.3 / 54.1 / 70.2 / 70.6 / 70.4	41.1 / 41.5 / 41.3 / 59.0 / 59.3 / 59.1
	Avg	62.0 / 77.5	—	54.1 / 70.4	41.3 / 59.1
	LoRA	60.8 / 61.2 / 61.0 / 75.9 / 76.3 / 76.1	—	51.9 / 52.2 / 52.0 / 68.7 / 69.3 / 69.0	39.0 / 39.7 / 39.5 / 56.8 / 57.4 / 57.3
	Avg	61.0 / 76.1	—	52.0 / 69.0	39.5 / 57.3
	BOFT	61.3 / 61.6 / 61.4 / 76.5 / 76.8 / 76.6	—	52.3 / 52.7 / 52.5 / 69.3 / 69.6 / 69.4	39.6 / 40.1 / 39.9 / 57.4 / 57.9 / 57.7
	Avg	61.4 / 76.6	—	52.5 / 69.4	39.9 / 57.7
	LoRA-GA	61.4 / 61.7 / 61.5 / 76.7 / 77.0 / 76.8	—	52.6 / 52.8 / 52.7 / 69.4 / 69.7 / 69.5	39.5 / 39.9 / 39.7 / 57.5 / 58.0 / 57.6
	Avg	61.5 / 76.8	—	52.7 / 69.5	39.7 / 57.6
	uRNN	60.5 / 61.1 / 60.8 / 75.7 / 76.3 / 76.0	—	51.6 / 52.2 / 51.9 / 68.3 / 69.1 / 68.8	38.6 / 39.4 / 39.0 / 56.4 / 57.2 / 56.8
	Avg	60.8 / 76.0	—	51.9 / 68.8	39.0 / 56.8
	Hybrid	62.1 / 62.9 / 62.5 / 77.4 / 78.1 / 77.8	—	53.2 / 53.8 / 53.5 / 70.0 / 70.6 / 70.1	40.2 / 40.8 / 40.5 / 57.9 / 58.5 / 58.3
	Avg	62.5 / 77.8	—	53.5 / 70.1	40.5 / 58.3

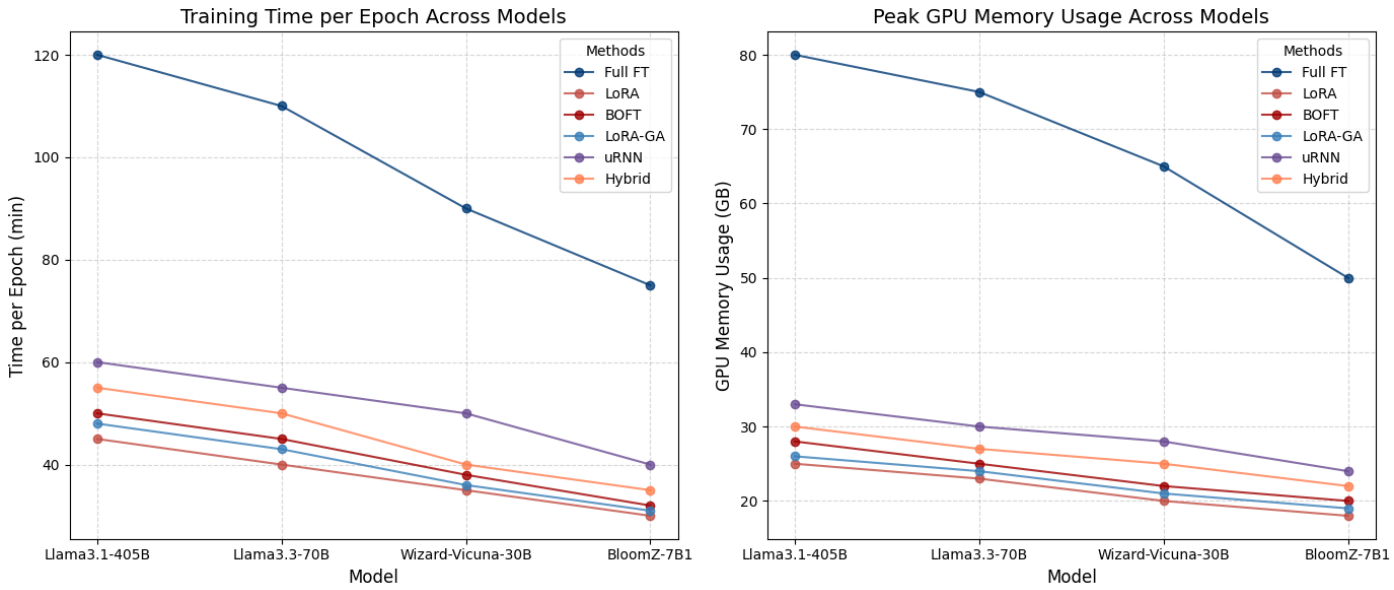

Fig. 1. 不同模型规模下每种方法的训练时间和 GPU 内存使用情况。

少。例如，在 Llama3.1-405B 上，Hybrid 方法每个 epoch 需要 55 分钟，而 Full FT 需要 120 分钟。较小的模型如 BloomZ-7B1 也有类似的改进，Hybrid 方法将训练时间减少到 35 分钟。内存使用方面呈现出类似的趋势，Hybrid 方法在 Llama3.1-405B 上消耗 30 GB 内存，而 Full FT 则为 80 GB。尽管 LoRA 在内存使用方面最低（在最大模型上为 25 GB），但其在泛化方面的限制使其适用性仅限于较小的模型。Hybrid 方法通过结合 LoRA-GA 的梯度对齐与 BOFT 的结构正交性，平衡了计算效率和泛化能力，确保了在不需要过多资源的情况下实现稳健的性能。

为了进一步研究训练的稳定性，我们以 Wizard-Vicuna-30B 作为中等规模模型的代表，跟踪两个关键指标——梯度范数和验证损失——在十个训练轮次中的变化。图 2 展示了对于相同的六种微调方法的两个并排图。在左侧，我们可以看到梯度范数在每个训练轮次中的演变；LoRA 和 LoRA-GA 起初具有相对较大的梯度（高于 5.0），这表明参数初始更新较快，但在第 5 轮后逐渐趋于平稳。相比之下，BOFT 和 uRNN 从一开始就保持较小的梯度幅度，反映了它们的正交或单位约束。值得注意的是，Hybrid 起始于 5.4，然后在第 10 轮时稳步下降到 3.6，平衡了快速适应和稳定更新。

在右侧，验证损失曲线证实了这些观察结果。虽然完整微调的验证损失到训练结束时下降到约 0.83，混合方法的表现非常接近，达到 0.88，尽管其需要的可调参数显著减少。LoRA-GA 的损失也明显下降，但在第 7 个 epoch 左右短暂趋于平稳，然后继续下降到 0.93。同时，BOFT 和 uRNN 强调稳定性，取得平滑的收敛，但最终的损失值稍高（0.94–1.00）。因此，这些趋势证实了正交或酉约束有助于限制梯度尖峰，而低秩梯度对齐促进了更快的早期学习阶段。通过融合两者的优势，混合方法实现了平衡的优化，从而验证了结合 LoRA 型更新与结构变换的有效性。

基于性能和资源分析，混合方法作为一种高效的解决方案，将低秩梯度对齐与结构稳定的更新结合起来。通过利用 LoRA-GA 和 BOFT 之间的协同作用，它能够在显著降低训练时间和内存需求的同时，持续提供接近 Full FT 的准确性。在 Wizard-Vicuna-30B 上的历元趋势进一步表明，混合方法在保持控制的梯度范数的同时，到最后一个历元实现了具有竞争力的验证损失。因此，对于需要快速适应和稳定性的的大规模或资源受限的部署而言，混合方法可以作为 Full FT 的一个可靠替代方案，在准确性和效率之间取得实用的平衡。

IV. 结论与展望

我们的研究对在计算预算受限情况下大语言模型的参数高效微调（PEFT）方法进行了原则性的探索。现有的策略如 LoRA、BOFT 和 LoRA-GA 在适应性、稳定性或收敛性方面各具优势，但在共同优化这些维度方面有所不足。为弥补这一差距，我们引入了两个关键创新：一种兼容 transformer 的单位 RNN（uRNN）适配方法，用于改进梯度保留，以及一个基于每层梯度反馈动态结合低秩与正交更新的混合微调框架。

在 GLUE、GSM8K、MT-Bench 和 HumanEval 四个基准上的大量实验表明，我们的方法是有效的。该混合方法在使用 LLama3.1-405B 的 HumanEval 上达到了 62.5 % 的 pass@1 准确率，超过 LoRA 1.6 %，并且在训练时间减少 2.1 倍和内存使用减少近 50 % 的情况下，效果接近完整微调。在 MT-Bench 上，它的平均 BLEU 得分比 LoRA 高出 2.4 分。此外，单位约束的整合显著提高了深层配置中的训练稳定性，特别是在 GSM8K 和代码生成任务上。这些结果证实了我们的方法在资源受限条件下进行真实世界的 LLM 微调的实际可行性。

在未来的工作中，我们计划通过探索更细粒度的、特定层的控制来优化混合策略，可能会结合学习控制器或元梯度。此外，将单位参数化进一步扩展到多头注意力块并更深入地集成到模型预训练管道中，可能会解锁额外的提升。最后，将这些技术应用于真实世界的部署场景——包括领域特定的大语言模型、边缘设备和低带宽的分布式训练——将有助于验证它们在这些基准测试之外的稳健性和广泛性。

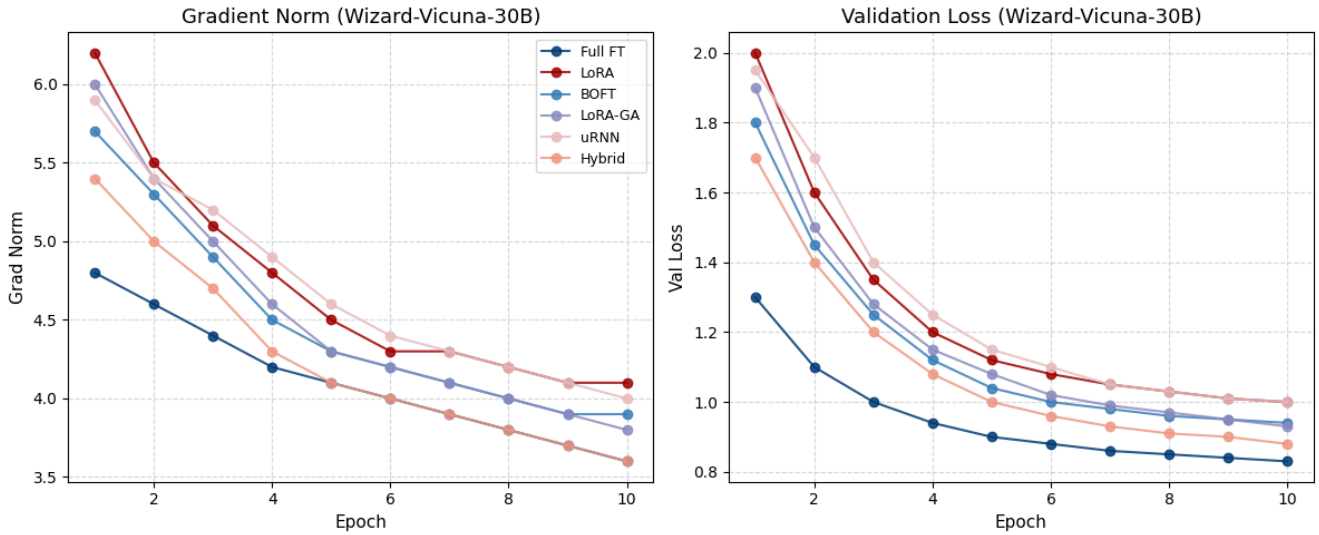

Fig. 2. 在整个十个周期中，Wizard-Vicuna-30B 的训练稳定性。（左）梯度范数演变；（右）验证损失。

References

[1] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," arXiv preprint arXiv:2403.14608 , 2024.

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in International Conference on Learning Representations , 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[3] W. Liu, Z. Qiu, Y. Feng, Y. Xiu, Y. Xue, L. Yu, H. Feng, Z. Liu, J. Heo, S. Peng, Y. Wen, M. J. Black, A. Weller, and B. Schölkopf, "Parameter-efficient orthogonal finetuning via butterfly factorization," in ICLR , 2024.

[4] S. Wang, L. Yu, and J. Li, "LoRA-GA: Low-rank adaptation with gradient approximation," 2024. [Online]. Available: https://arxiv.org/abs/2407.05000

[5] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in International Conference on Machine Learning . PMLR, 2016, pp. 1120–1128.

[6] P. He, Y. Chen, Y. Wang, and Y. Zhang, "Protum: A new method for prompt tuning based on "[mask]"," arXiv preprint arXiv:2201.12109 , 2022.

[7] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in International Conference on Machine Learning . PMLR, 2019, pp. 2790–2799.

[8] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," arXiv preprint arXiv:2110.07602 , 2021.

[9] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," arXiv preprint arXiv:2012.13255 , 2020.

[10] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, "Learning fast algorithms for linear transforms using butterfly factorizations," in International Conference on Machine Learning . PMLR, 2019, pp. 1517–1527.

[11] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," Advances in Neural Information Processing Systems , vol. 29, 2016.

[12] M. Emami, M. Sahraee Ardakan, S. Rangan, and A. K. Fletcher, "Input-output equivalence of unitary and contractive rnns," Advances in Neural Information Processing Systems , vol. 32, 2019.

[13] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment," arXiv preprint arXiv:2312.12148 , 2023.

[14] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen et al. , "Parameter-efficient fine-tuning of large-scale pre-trained language models," Nature Machine Intelligence , vol. 5, no. 3, pp. 220–235, 2023.

[15] R. K. Mahabadi, S. Ruder, M. Dehghani, J. Henderson et al. , "Compacter: Efficient low-rank hypercomplex adapter layers," in Advances in Neural Information Processing Systems (NeurIPS) , 2021.

[16] E. Ben Zaken, Y. Goldberg, and S. Ravfogel, "Bitfit: Simple parameter-efficient fine-tuning for transformers," arXiv preprint arXiv:2106.16399, 2021.

[17] S. Li, K. Jia, Y. Wen, T. Liu, and D. Tao, "Orthogonal deep neural networks," IEEE Transactions on Pattern Analysis and Machine Intelligence , vol. 43, no. 4, pp. 1352–1368, 2019.

[18] A. Prabhu, A. Farhadi, M. Rastegari et al. , "Butterfly transform: An efficient fft based neural architecture design," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition , 2020, pp. 12 024–12 033.

[19] Z. Wang, J. Liang, R. He, Z. Wang, and T. Tan, "LoRA-Pro: Are low-rank adapters properly optimized?" arXiv preprint arXiv:2407.18242 , 2024.

[20] I. Shafran, T. Bagby, and R. Skerry-Ryan, "Complex evolution recurrent neural networks (cernns)," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) . IEEE, 2018, pp. 5854–5858.

[21] J.-P. Bernardy and S. Lappin, "Assessing the unitary rnn as an end-to-end compositional model of syntax," arXiv preprint arXiv:2208.05719 , 2022.

[22] J. Pfeiffer, A. Rücklé, and etc., "Adapterfusion: Non-destructive task composition for transfer learning," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) , 2020.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," arXiv preprint arXiv:1804.07461 , 2018.

[24] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano et al. , "Training verifiers to solve math word problems," arXiv preprint arXiv:2110.14168 , 2021.

[25] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing et al. , "Judging llm-as-a-judge with mt-bench and chatbot arena," Advances in Neural Information Processing Systems , vol. 36, pp. 46 595–46 623, 2023.

[26] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman et al. , "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374 , 2021.