American Journal of Computer Science and Technology
2025; 8(4): 242-255
http://www.sciencepublishinggroup.com/j/ajcst
doi: 10.11648/j.ajcst.20250804.17

Hybrid and Unitary PEFT for Resource-Efficient Large Language Models

Haomin Qi[1, 2,*], Zihan Dai[1, 3], Chengbo Huang[1, 4]

[1]Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China
[2]Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, United States of America
[3]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
[4]Department of Electrical Engineering, Columbia University, New York , United States of America

Email address:

h5qi@ucsd.edu (Haomin Qi), cjh841@alumni.ku.dk (Zihan Dai), ch4019@columbia.edu (Chengbo Huang)
[*]Corresponding author

To cite this article:

Haomin Qi, Zihan Dai, Chengbo Huang. (2025). Hybrid and Unitary PEFT for Resource-Efficient Large Language Models. *American Journal of Computer Science and Technology, 8*(4), 242-255. https://doi.org/10.11648/j.ajcst.20250804.17

Received: 01 October 2025 ; **Accepted:** 20 October 2025 ; **Published:** 19 December 2025

Abstract: Fine-tuning large language models (LLMs) remains a computational bottleneck due to their scale and memory demands. This paper presents a comprehensive evaluation of parameter-efficient fine-tuning (PEFT) techniques, including LoRA, BOFT, LoRA-GA, and uRNN, and introduces a novel hybrid strategy that dynamically integrates BOFT's orthogonal stability with LoRA-GA's gradient-aligned rapid convergence. By computing per-layer adaptive updates guided by gradient norms, the hybrid method achieves superior convergence efficiency and generalization across diverse tasks. We also explore, for the first time, the adaptation of unitary RNN (uRNN) principles to Transformer-based LLMs, enhancing gradient stability through structured unitary constraints. Across GLUE, GSM8K, MT-Bench, and HumanEval with models from 7B to 405B, the hybrid approach yields consistent gains across three independent runs per task and model, approaching the quality of full fine-tuning while reducing training time by about 2.1× and peak memory by nearly 50%, indicating practical significance under resource constraints. A compact multilingual and low-resource study on XNLI and FLORES with 32 examples per language shows consistent gains under the same budget with a small, stable footprint. These results indicate a practical and scalable path to accessible LLM fine-tuning under resource constraints.

Keywords: Large Language Models, Parameter-Efficient Fine-Tuning, Low-Rank Adaptation

1. Introduction

Large Language Models (LLMs) have become foundational in natural language processing (NLP), powering applications from machine translation to code generation. However, the cost of fine-tuning these massive models remains a significant barrier, especially in resource-constrained environments. Parameter-efficient fine-tuning (PEFT) strategies [1]-such as Low-Rank Adaptation (LoRA) [2], Butterfly Orthogonal Fine-Tuning (BOFT) [3], and gradient-aware LoRA-GA-have been proposed to reduce the number of trainable parameters [4], yet they often present trade-offs between stability, convergence speed, and representational capacity.

This paper introduces a hybrid fine-tuning framework that integrates LoRA-GA and BOFT at the layer level using gradient-norm-based dynamic weighting. LoRA-GA computes low-rank updates aligned with dominant gradient directions via singular value decomposition (SVD) for rapid early adaptation. BOFT enforces orthogonality through Cayley transforms of skew-symmetric matrices to preserve gradient norms. The hybrid method fuses these updates with an adaptive coefficient that balances fast adaptation in early epochs and stability in later stages under the same budget.

We further embed unitary evolution RNN (uRNN) structures [5] into transformer sub-layers. By parameterizing selected attention or feedforward weights with structured unitary matrices based on Fourier transforms, permutations,

and Householder reflections, the approach maintains gradient magnitudes and improves robustness during fine-tuning of deep stacks.

We evaluate the proposed hybrid and uRNN-enhanced fine-tuning strategies across four standard benchmarks-GLUE, GSM8K, MT-Bench, and HumanEval-on LLMs spanning 7B to 405B parameters. We also add a compact multilingual and low resource study on XNLI and FLORES with 32 examples per language. Our results demonstrate that the hybrid method consistently outperforms individual PEFT techniques, achieving near-full fine-tuning performance while reducing training time to less than half and memory usage by nearly 50%.

Our key contributions are as follows:

(1) We propose a hybrid algorithm that dynamically fuses gradient-aligned low-rank updates with orthogonal transformations via per-layer, gradient-norm-based mixing to achieve fast convergence and stable optimization.

(2) We adapt unitary evolution RNN principles to transformer-based models by embedding structured unitary matrices into attention and feedforward sub-layers, enhancing gradient stability during fine-tuning.

(3) We conduct a comprehensive benchmark across four PEFT baselines and four standard tasks with models from 7B to 405B, and we include a compact multilingual and low-resource evaluation that validates the method.

2. Background and Motivation

2.1. Parameter-Efficient Fine-Tuning for LLMs

The increasing parameter count of large language models (LLMs) has made full model fine-tuning prohibitively expensive in terms of GPU memory, training time, and energy cost. As a result, parameter-efficient fine-tuning (PEFT) strategies have emerged as practical alternatives, aiming to reduce the number of trainable parameters while retaining downstream performance [6]. These methods typically freeze the majority of the pre-trained weights and inject lightweight, learnable components to adapt the model to new tasks.

Early PEFT techniques include adapter modules [7], which insert bottleneck layers between transformer blocks, and prefix tuning [8], which learns task-specific prompt vectors prepended to input embeddings. More recently, low-rank adaptation (LoRA) introduced trainable matrix decompositions to approximate weight updates, offering favorable trade-offs between memory usage and task performance. Variants such as LoRA-GA approximate gradients to further reduce update cost, while BOFT imposes orthogonality via butterfly parameterization to enhance stability. These techniques have led to a new class of modular, reusable, and resource-aware fine-tuning paradigms for large-scale deployment [9].

Concurrently, several methods extend PEFT along complementary axes: QLoRA [28] combines 4-bit base-model quantization with LoRA adapters to preserve quality under tight memory budgets; DoRA [29] decouples update direction and magnitude to increase expressivity and stability of low-rank adaptations; and IA^3 [30] injects lightweight learned rescaling vectors that modulate attention and feed-forward activations. We reference these approaches in our comparisons where feasible and discuss their relation to our hybrid design in Section 3.5 and the ablations in Section 4.

2.2. LoRA, BOFT, and LoRA-GA Methods

LoRA decomposes weight updates into a pair of low-rank matrices, significantly reducing the number of tunable parameters and training memory [2]. Its simplicity and effectiveness have led to wide adoption. However, LoRA assumes fixed low-rank structure throughout training, which can hinder adaptability in highly complex or dynamic tasks.

BOFT addresses training stability by applying butterfly-structured orthogonal updates to model weights. Its orthogonality constraint helps preserve gradient norms and mitigates training instability [10]. Despite this, the restrictive structure of butterfly transforms can limit expressiveness, especially in tasks requiring nonlinear adaptation.

LoRA-GA builds on LoRA by initializing the low-rank matrices using gradient-aligned singular value decomposition (SVD) [4]. This gradient-aware initialization improves early convergence but may introduce additional computational cost and instability under noisy gradients.

These approaches each offer unique strengths, but none fully address the combined needs of robustness, adaptability, and resource-awareness in a single framework.

2.3. Unitary RNNs and Gradient Stability

Unitary RNNs (uRNNs) were originally proposed to resolve the exploding and vanishing gradient problems in recurrent networks [5]. By constraining transition matrices to be unitary, these models preserve gradient magnitudes over long sequences. Subsequent works [11, 12] proposed efficient parameterizations using Fourier transforms, permutations, and Householder reflections.

While traditionally used in sequence modeling, the stability guarantees of uRNNs are theoretically appealing in transformer-based models, particularly during deep-layer fine-tuning. To the best of our knowledge, we are the first to integrate structured unitary matrices into transformer layers for LLM fine-tuning, providing a new perspective on stabilizing gradients during parameter-efficient adaptation.

2.4. Resource-Constrained Fine-Tuning: Motivation for Hybrid Design

In real-world scenarios-especially for practitioners lacking access to high-end GPUs-resource constraints such as memory footprint, compute time, and thermal budget are primary bottlenecks when fine-tuning LLMs [13, 14]. While PEFT methods reduce parameter counts, they often present a trade-off: LoRA and its variants converge quickly but risk instability

in deeper layers, while orthogonal methods like BOFT preserve training dynamics but converge more slowly.

A growing body of work has attempted to benchmark or extend single-method PEFT techniques, yet relatively little research investigates how these strengths might be combined in a dynamic and structured fashion. Rather than introducing yet another static PEFT variant, our work proposes a hybrid mechanism that allocates per-layer update responsibility based on real-time gradient feedback-favoring low-rank speed in early epochs and orthogonal stability later.

3. Methodology

In this Section, we first review the mathematical principles of LoRA, BOFT, and LoRA-GA as baseline PEFT methods. We then present our two main contributions: a transformer-compatible adaptation of uRNN with structured unitary constraints, and a hybrid fine-tuning strategy that dynamically fuses low-rank and orthogonal updates based on gradient feedback.

3.1. Low-Rank Adaptation (LoRA)

LoRA introduces low-rank updates to pre-trained weight matrices, significantly reducing the number of trainable parameters while preserving the pre-trained model weights. The weight update is expressed as:

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

where $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ represents the frozen pre-trained weight matrix, and $\Delta\mathbf{W}$ is the low-rank update parameterized by $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$, thereby cutting the number of trainable parameters from $\mathcal{O}(dk)$ to $\mathcal{O}(r(d+k))$ with $r \ll \min(d,k)$.

Optimization: During fine-tuning, only $\mathbf{B}$ and $\mathbf{A}$ are optimized, leaving $\mathbf{W}_0$ unchanged [15, 16]. This decomposition reduces the memory and computational costs of fine-tuning while maintaining performance.

To ensure numerical stability, the norms of $\mathbf{A}$ and $\mathbf{B}$ are constrained by rank r:

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

where λ is a scaling factor. This prevents updates from diverging during optimization.

3.2. Butterfly Orthogonal Fine-Tuning (BOFT)

BOFT factorizes a square weight matrix into a product of sparse butterfly blocks that are (near-)orthogonal, yielding both parameter efficiency ($\mathcal{O}(d\log d)$ parameters) and stable gradient norms.

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

Each $\mathbf{B}_i$ is built from paired line-permute-multiply operations that mimic the Fast Fourier Transform hierarchy [17]; a simplified two-level form is

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

where $\mathbf{F}_d$ is a (learnable) orthogonal mixing matrix.

Optimization: Each $\mathbf{B}_i$ is initialized as a near-identity transformation and updated via gradient descent [18]:

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

where η is the learning rate. Orthogonality is enforced post-update using a projection step:

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

The orthogonality constraint curbs exploding/vanishing gradients in deep stacks, making it particularly effective for tasks with deep layers or complex gradients.

3.3. LoRA with Gradient Approximation (LoRA-GA)

LoRA-GA improves upon LoRA by aligning the low-rank updates with the gradients of the full model, leading to faster convergence and better optimization. The gradient of the loss $\mathcal{L}$ with respect to the frozen weight matrix $\mathbf{W}_0$ is decomposed as:

Compute the rank-r truncated SVD of $\nabla_{\mathbf{W}_0}\mathcal{L}$:

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

The low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are initialized as:

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

This ensures that the initial updates align with the principal gradient directions, accelerating convergence.

Optimization: Post-initialization, $\mathbf{A}$ and $\mathbf{B}$ are updated iteratively using standard gradient descent [19]:

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

By aligning the initial low-rank updates with the most influential gradient directions, LoRA-GA reduces the number of training iterations required for convergence, making it particularly suitable for resource-constrained scenarios.

3.4. Unitary Evolution RNN (uRNN)

Unitary Recurrent Neural Networks (uRNN) constrain hidden-to-hidden weight matrices to be unitary to mitigate vanishing and exploding gradient issues [11, 12]. By ensuring that the eigenvalues of the transition matrix

lie on the unit circle, uRNNs preserve gradient norms during backpropagation, enabling learning over long-term dependencies.

A core component of uRNN is a learnable unitary matrix U that evolves the hidden state. To ensure U is unitary, we adopt a structured parameterization method [5]:

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

where F (and F^{-1}) are fixed unitary Fourier transform matrices, Π is a fixed permutation matrix, and D_i and R_i denote trainable diagonal phase matrices and Householder reflection matrices [20]. This factorization dramatically reduces the number of free parameters (to $O(n)$ for an $n \times n$ matrix) and allows efficient $O(n \log n)$ computation for matrix-vector products via Fast Fourier Transform operations.

Adaptation for Fine-Tuning LLMs: To our knowledge, we are the first to integrate uRNN principles into the fine-tuning of transformer-based LLMs. The motivation is to leverage unitary transformations to stabilize gradient propagation and better capture long-range dependencies during fine-tuning. In practice, we incorporate learnable unitary matrices into selected Transformer sub-layers to enhance training stability. Specifically, we replace certain weight matrices (e.g., in attention heads or feed-forward blocks) with unitary matrices and modify the training procedure to preserve their unitarity. Each such unitary weight is initialized to an identity-like matrix (close to the unit matrix) to ensure stable convergence. During backpropagation, we include an efficient re-projection step (see below) that keeps these weights unitary at all times. This approach extends unitary RNN techniques beyond their original domain, establishing a new paradigm for parameter-efficient fine-tuning of LLMs.

Algorithm 1: uRNN-Based Fine-Tuning Procedure

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

Gradient Update with Re-Projection: We train the unitary weight U via gradient descent on the manifold of unitary matrices. Let $\nabla_U L$ be the gradient of the loss L with respect to U (computed by backpropagation). We first construct a skew-Hermitian matrix B (i.e., $B^H = -B$) from the gradient:

$$\mathbf{B} \;=\; \nabla_{\mathbf{U}}\mathcal{L}\ \mathbf{U}^H \;-\; \mathbf{U}\,(\nabla_{\mathbf{U}}\mathcal{L})^H\,, \tag{11}$$

where U^H denotes the conjugate transpose of U. By construction, B lies in the Lie algebra $\mathfrak{u}(n)$ of the unitary group. In other words, B is skew-Hermitian, and thus $\exp(\eta B)$ is a unitary matrix for any real step size. We then update U by a unitary rotation:

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

with η the learning rate. This exponential map update guarantees U_{t+1} remains unitary. In implementation, $\exp(\eta B)$ can be efficiently approximated using a truncated Taylor series or a scaling-and-squaring algorithm, and we re-normalize U as needed to correct any numerical drift from unitarity.

Fine-Tuning Algorithm: Algorithm 1 outlines the overall fine-tuning process using uRNN principles. We apply $\mathbf{U}$ in the forward pass of the chosen transformer layer and then update $\mathbf{U}$ at each training step using the manifold-aware rule above, while leaving other model weights to update as usual. Notably, although uRNNs were originally devised for recurrent sequence models, our strategy applies these unitary constraints to feed-forward or attention layers in a Transformer architecture. Conceptually, each forward pass through a transformer sub-layer is analogous to a single RNN step, with the sub-layer's input playing the role of the "hidden state." Maintaining $\mathbf{U}$ as unitary thus helps preserve gradient norms through the depth of the network, even without explicit recurrence [21].[1]

The above integration of uRNN principles into Transformer fine-tuning offers several notable benefits. *First*, enforcing unitary transformations provides *gradient stability*: it prevents the magnitudes of gradients from vanishing or exploding, even in very deep networks or tasks with long-range dependencies. *Second*, this method tends to *improve convergence* during fine-tuning, as stable gradient norms facilitate faster and more reliable training (reducing the number of iterations required to reach a given performance level). *Third*, the approach is *adaptable* to different model components; unitary

[1] In practice, we treat the input to each unitary-constrained sublayer as a proxy for an RNN hidden state, which ensures stable backpropagation across many transformer layers.

constraints can be applied to various layers or sub-layers of an LLM (e.g., attention projections or feed-forward blocks) without architecture changes, extending the use of orthogonal transformations beyond their traditional recurrent setting. By extending unitary transformations to transformer-based LLMs, we establish a novel paradigm for fine-tuning that marries parameter efficiency with training stability.

3.5. Hybrid Fine-Tuning Approach

We propose a per-layer fusion of the LoRA-GA and BOFT updates to capture both low-rank and orthonormal adaptation patterns. Specifically, this hybrid strategy computes the gradient-aligned low-rank update from LoRA-GA and the structured orthonormal update from BOFT for the same weight matrix in each layer, then mixes them with a dynamic coefficient. Intuitively, this allows fast initial adaptation via the low-rank component while gradually shifting emphasis to the BOFT component to stabilize learning as training proceeds.

Algorithm 2: Hybrid Fine-Tuning Procedure

Initialize: pretrained weights $\{\mathbf{W}^\ell\}$, low-rank matrices $\{\mathbf{A}^\ell, \mathbf{B}^\ell\}$ for LoRA-GA, skew-symmetric matrices $\{\mathbf{Q}^\ell\}$ 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: $\Delta\mathbf{W}^\ell_{\text{LoRA}} = \mathbf{A}^\ell\mathbf{B}^\ell$.
      Compute BOFT orthonormal matrix:
      $\mathbf{R}^\ell = (\mathbf{I} + \eta_{\text{BOFT}}\mathbf{Q}^\ell)(\mathbf{I} - \eta_{\text{BOFT}}\mathbf{Q}^\ell)^{-1}$.
      Compute BOFT update:
      $\Delta\mathbf{W}^\ell_{\text{BOFT}} = (\mathbf{R}^\ell - \mathbf{I})\mathbf{W}^\ell$.
    end for
    Compute task-specific loss $\mathcal{L}$ using model predictions.
    Backward Pass:
    for each transformer layer ℓ **do**
      Compute gradient norms:
      $g^\ell_{\text{LoRA}} = \|\nabla_{\mathbf{A}^\ell,\mathbf{B}^\ell}\mathcal{L}\|, \quad g^\ell_{\text{BOFT}} = \|\nabla_{\mathbf{Q}^\ell}\mathcal{L}\|$.
      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:
      $\Delta\mathbf{W}^\ell_{\text{hybrid}} = \lambda^\ell\Delta\mathbf{W}^\ell_{\text{LoRA}} + (1-\lambda^\ell)\Delta\mathbf{W}^\ell_{\text{BOFT}}$.
      Update weight matrix for layer ℓ:
      $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell + \Delta\mathbf{W}^\ell_{\text{hybrid}}$.
      Update low-rank matrices via gradient descent:
      $\mathbf{A}^\ell \leftarrow \mathbf{A}^\ell - \eta_{\text{LoRA}}\nabla_{\mathbf{A}^\ell}\mathcal{L}$,
      $\mathbf{B}^\ell \leftarrow \mathbf{B}^\ell - \eta_{\text{LoRA}}\nabla_{\mathbf{B}^\ell}\mathcal{L}$.
      Compute skew-symmetric gradient matrix for BOFT:
      $\mathbf{G}^\ell = \nabla_{\mathbf{Q}^\ell}\mathcal{L} - (\nabla_{\mathbf{Q}^\ell}\mathcal{L})^\top$.
      Update skew-symmetric matrix $\mathbf{Q}^\ell$:
      $\mathbf{Q}^\ell \leftarrow \mathbf{Q}^\ell - \eta_{\text{BOFT}}\mathbf{G}^\ell$.
      Recompute orthonormal matrix $\mathbf{R}^\ell$ via Cayley transform (for numerical stability):
      $\mathbf{R}^\ell \leftarrow (\mathbf{I} + \eta_{\text{BOFT}}\mathbf{Q}^\ell)(\mathbf{I} - \eta_{\text{BOFT}}\mathbf{Q}^\ell)^{-1}$.
    end for
    Update other model parameters (if any) via standard gradient descent.
  end for
end for

Mathematical formulation: Consider a layer ℓ with pretrained weight matrix $\mathbf{W}^\ell \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$. We introduce low-rank factors $\mathbf{A}^\ell \in \mathbb{R}^{d_{\text{out}}\times r}$ and $\mathbf{B}^\ell \in \mathbb{R}^{r\times d_{\text{in}}}$ (rank r) as in LoRA [4], and a skew-symmetric matrix $\mathbf{Q}^\ell \in \mathbb{R}^{d_{\text{out}}\times d_{\text{out}}}$ ($\mathbf{Q}^\ell = -\mathbf{Q}^{\ell\top}$) as in BOFT [3]. The low-rank LoRA-GA update is

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

The orthonormal BOFT update is obtained via the Cayley transform [10]:

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

which ensures $\mathbf{R}^\ell$ is orthonormal (for small step size η). The BOFT update to $\mathbf{W}^\ell$ is then

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

We combine these with a layerwise mixing coefficient $\lambda^\ell_t \in [0,1]$ that adapts over training steps t. Specifically, we set

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

so that the component with larger gradient norm receives higher weight. The hybrid update is then

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

and the weight is updated as $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell + \Delta\mathbf{W}^\ell_{\text{hybrid}}$. Here $\nabla_{\mathbf{A}^\ell,\mathbf{B}^\ell}L$ denotes the gradient of the training loss $L(\theta_t)$ with respect to the LoRA parameters [22] $(\mathbf{A}^\ell, \mathbf{B}^\ell)$ and $\nabla_{\mathbf{Q}^\ell}L$ the gradient with respect to $\mathbf{Q}^\ell$. All notation above is defined per layer ℓ.

Pseudocode Algorithm: Algorithm 2 summarizes the hybrid fine-tuning update. At each iteration, we compute both the LoRA-GA and BOFT updates for each layer, compute the mixing coefficient λ^ℓ_t, and apply the weighted combination to update the weights.

The per-layer hybrid fusion adds only modest overhead beyond the individual LoRA-GA and BOFT updates. In each layer, the low-rank update costs $\mathcal{O}(d_{\text{out}}r + r\,d_{\text{in}})$ and the BOFT transform costs $\mathcal{O}(d_{\text{out}}\log d_{\text{out}})$. Computing λ^ℓ_t requires only the norms of gradients already computed, which is negligible.

The total number of tunable parameters is the sum of the LoRA factors and any BOFT parameters (e.g., butterfly factors), comparable to using the two methods independently. By construction the Cayley parameterization enforces $\mathbf{R}^\ell$ to be orthonormal, which helps preserve gradient norms during optimization. The hybrid update thus integrates fast low-rank adaptation with structured orthogonal adjustments in a unified step.

4. Experiments

This section evaluates the proposed fine-tuning strategies across diverse tasks and models to investigate their effectiveness, scalability, and computational efficiency. The experimental design focuses on systematically comparing the performance of LoRA, BOFT, LoRA-GA, uRNN, and the Hybrid approach against the Full Fine-Tuning (Full FT) baseline. Additionally, we explore trade-offs between task-specific gains and resource consumption to provide a comprehensive understanding of the proposed methods' practical utility.

4.1. Setup and Evaluation Metrics

To ensure a thorough and unbiased evaluation, experiments were conducted on four state-of-the-art large language models (LLMs) with varying parameter scales and architectural characteristics:

(1) Llama3.1-405B: A transformer model designed for extended context comprehension, comprising 405 billion parameters.

(2) Llama3.3-70B: A mid-sized model with 70 billion parameters.

(3) Wizard-Vicuna-30B: A multilingual model with 30 billion parameters.

(4) BloomZ-7B1: A compact model with 7.1 billion parameters.

Each model was tested on four benchmark datasets, selected to evaluate a wide range of NLP capabilities:

(1) GLUE Benchmark: A comprehensive suite of general NLP tasks, including MNLI, QQP, SST-2, and QNLI, assessing classification and language understanding capabilities[23].

(2) GSM8K: A dataset of mathematical reasoning problems designed to evaluate multi-step reasoning and arithmetic capabilities[24].

(3) MT-Bench: A multilingual machine translation benchmark that tests translation quality using BLEU and ROUGE-L metrics[25].

(4) HumanEval: A code generation benchmark assessing the functional correctness of generated programs, with *pass@*k metrics as the primary evaluation criterion[26].

Each fine-tuning method was applied to all models under identical conditions to ensure a fair comparison. The experiments involved:

Hardware and Software Configuration: All experiments were conducted on a cluster featuring dual AMD EPYC 7742 CPUs and 1 TB of RAM per node, interconnected via InfiniBand for high-speed communication. Each node was equipped with eight NVIDIA A100 GPUs connected through NVLink. The software environment included PyTorch 2.0, CUDA 11.8, and NVIDIA Apex, enabling mixed precision for efficient training.

Hyperparameter Tuning: Hyperparameters for each method were tuned based on a preliminary grid search. For instance, LoRA used a rank $r = 16$ with scaling factor $\alpha = 32$, while BOFT employed three butterfly factorization levels ($m = 3$). For the Hybrid approach, the gradient weighting factor α_t was dynamically adjusted during training.

Experimental Repeats: Each experiment was repeated three times to mitigate the impact of random initialization and ensure statistically robust results. The reported metrics also represent the average performance across these runs.

Ablation settings: To disentangle where the gains come from, we include three ablations: (i) *LoRA-only* and (ii) *BOFT-only* baselines (both already reported in our main tables), and (iii) a *fixed-mix* variant of our hybrid where a constant coefficient $\lambda \in \{0.25, 0.5, 0.75\}$ is applied across all layers and steps (with a warm-start schedule that linearly anneals $0.75 \rightarrow 0.25$ over epochs). These ablations isolate the contribution of the gradient-normbased adaptive mixing from simple ensembling effects.

Table 1. *Performance Comparison: Per-Run Results and Averages on GLUE, GSM8K, MT-Bench, and HumanEval Benchmarks.*

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

Benchmark	Method	Llama3.1-405B	Llama3.3-70B	Wizard-Vicuna-30B	BloomZ-7B1
	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

4.2. Benchmark-Specific Results and Analysis

Reporting variation: Each method is run with three independent seeds. Per-run results are shown in the table, and the average column reports mean values. For completeness, we additionally compute standard deviations and mark statistical significance versus LoRA using paired t-tests († for $p < 0.05$) where applicable.

Across three seeds, the Hybrid method shows the following mean±s.d. (3 runs) on the Avg column of Table 1: *GLUE*-405B *92.3*±0.72, 70B *90.2*±0.32, 30B *88.4*±0.44, 7B *85.1*±0.42; *GSM8K*-405B *55.9*±0.56, 70B *53.0*±0.15, 30B *52.1*±0.40, 7B *49.3*±0.56; *MT-Bench*-405B *29.4*±0.38, 70B *28.1*±0.25, 30B *26.7*±0.31, 7B *25.4*±0.30. Relative to LoRA, improvements are significant at $p<0.05$ (Welch's t-test, 3 seeds) on GLUE for *70B/30B/7B*, on GSM8K for *all four* model scales, and on MT-Bench for *70B/30B/7B*; the 405B MT-Bench result is marginal ($p<0.1$), and the 405B GLUE gain is consistent but not significant with $n=3$.

GLUE Benchmark: The GLUE Benchmark results are summarized in Table 1. The Hybrid approach demonstrates consistent superiority across all model scales, effectively integrating LoRA-GA's rapid adaptation and BOFT's structured stability. On the largest model, Llama3.1-405B, the Hybrid method achieves an average score of 92.3%, closely approximating the Full Fine-Tuning (Full FT) baseline of 92.5%. This suggests that the Hybrid approach can capture nuanced linguistic patterns without incurring the high computational costs associated with Full FT.

On the medium-scale Llama3.3-70B, the Hybrid method surpasses BOFT by 0.8% and LoRA-GA by 0.6%,

underscoring its ability to balance quick convergence and robust generalization. Smaller models, such as BloomZ-7B1, benefit significantly from the Hybrid strategy, with a 1.1% improvement over LoRA, further narrowing the performance gap with Full FT while retaining parameter efficiency.

BOFT achieves strong generalization on linguistically complex tasks like MNLI and QNLI, where structured orthogonal updates enhance gradient stability. Meanwhile, LoRA-GA excels in tasks like SST-2 and QQP, requiring rapid feature extraction. However, both methods exhibit limitations in achieving simultaneous adaptability and stability, which the Hybrid approach effectively addresses.

GSM8K Benchmark: The GSM8K dataset evaluates multi-step arithmetic and reasoning capabilities, with results shown in Table 1. Across all model scales, the Hybrid method achieves the highest accuracy, with a notable 55.9% on Llama3.1-405B, surpassing Full FT by 0.2% and LoRA by 1.7%. These results reflect the Hybrid approach's ability to adapt to complex reasoning tasks while maintaining stable gradient propagation.

For medium-sized models such as Llama3.3-70B, the Hybrid method improves by 0.9% over LoRA-GA and 1.0% over BOFT, indicating its balanced integration of convergence speed and structural robustness. On smaller models like Wizard-Vicuna-30B and BloomZ-7B1, the Hybrid approach consistently outperforms other methods, narrowing the performance gap with Full FT while achieving resource-efficient fine-tuning.

The Hybrid strategy's pronounced advantage on smaller models emphasizes its ability to dynamically balance feature adaptation and stable optimization, making it particularly suitable for intricate reasoning tasks.

MT-Bench Benchmark: The MT-Bench dataset evaluates models' ability to perform multilingual translation tasks, emphasizing both linguistic fidelity and semantic coherence. Table 1 presents the BLEU scores for each fine-tuning method across four model scales. The Hybrid approach achieves superior performance, consistently outperforming both Full Fine-Tuning and standalone parameter-efficient methods. On Llama3.1-405B, the Hybrid approach records a BLEU score of 29.4, marginally exceeding Full Fine-Tuning by 0.1. For smaller models like BloomZ-7B1, the Hybrid method achieves a 0.7 improvement over LoRA-GA, reflecting its adaptability to resource-constrained settings.

The performance gap between the Hybrid method and uRNN highlights the challenges of applying unitary transformations in cross-lingual tasks. While uRNN excels in maintaining gradient stability over long sequences, it lacks the adaptability required for multilingual translation.

The results also indicate that the Hybrid method successfully combines the strengths of BOFT and LoRA-GA. BOFT's structured orthogonal updates ensure stability and consistency in handling diverse linguistic patterns, while LoRA-GA's gradient alignment accelerates early convergence, particularly on multilingual datasets. For instance, the Hybrid method achieves a 0.9 improvement over standalone LoRA on Wizard-Vicuna-30B, underscoring its ability to balance stability and convergence speed.

On smaller models, such as BloomZ-7B1, the Hybrid method demonstrates its efficiency by delivering a BLEU score of 25.4. This performance is notable given the parameter constraints inherent to smaller models, where traditional Full Fine-Tuning struggles to fully leverage its computational intensity. On the Llama3.3-70B model, the Hybrid approach achieves a BLEU score of 28.1, surpassing LoRA-GA and BOFT by 0.8 and 0.6, respectively. This demonstrates the Hybrid strategy's capacity to generalize across varying parameter scales without sacrificing efficiency.

HumanEval Code Generation:[2] The HumanEval dataset assesses models' ability to generate functionally correct code, with metrics focusing on *pass@1* and *pass@10* accuracies[3]. Table 1 presents the results across all methods and model sizes, highlighting the Hybrid approach as the top-performing method. On Llama3.1-405B, the Hybrid approach achieves a *pass@1* accuracy of 62.5%, outperforming Full Fine-Tuning by 0.5%. Notably, for smaller models such as BloomZ-7B1, the Hybrid approach narrows the performance gap with Full Fine-Tuning, achieving 40.5% *pass@1*, a 1.0% improvement over BOFT and a 1.5% improvement over LoRA.

The Hybrid method excels by integrating the rapid convergence characteristics of LoRA-GA with the stability of BOFT. This combination is particularly effective for generating functionally correct code, as it addresses the dual challenges of quick adaptation to task-specific nuances and maintaining robust learning dynamics during longer training periods. For example, the Hybrid approach demonstrates a 0.7% improvement in *pass@10* accuracy over BOFT on Wizard-Vicuna-30B, reflecting its ability to generalize across diverse code patterns.

Smaller models like BloomZ-7B1 benefit significantly from the Hybrid method's efficiency. With constrained parameter sizes, these models often struggle with traditional Full Fine-Tuning due to computational overheads. The Hybrid approach, leveraging parameter-efficient strategies, delivers superior performance without incurring excessive resource demands, making it an ideal choice for resource-constrained scenarios.

[2] Due to the limitation of computing resources and the similarity estimation performance of the selected large language model in the HumanEval Benchmark, the Llama3.3-70B model, which also belongs to Llama3, was not repeated in the HumanEval Code Generation experiment.

[3] HumanEval reports performance as a tuple $(pass@1, pass@10)$ in percentage. Here, *pass@1* refers to the percentage of problems where the model's top-1 generated solution is functionally correct, while *pass@10* refers to the percentage where any of the top-10 generated solutions is correct. In Table 1, for each model-method combination, the left-side number denotes $pass@1, and the right-side number denotes pass@10$.

Table 2. *Ablation on Wizard-Vicuna-30B: LoRA-only, BOFT-only, and Hybrid (fixed λ=0.5 vs. adaptive). Mean $\pm$ s.d. over three runs.*

Method	GLUE	GSM8	MT-Bench (BLEU)	HumanEval (pass@1)
LoRA-only	87.5 $\pm$ 0.26	50.8 $\pm$ 0.20	25.8 $\pm$ 0.32	52.0 $\pm$ 0.15
BOFT-only	87.8 $\pm$ 0.10	51.2 $\pm$ 0.20	26.0 $\pm$ 0.31	52.5 $\pm$ 0.20
Hybrid (fixed λ=0.5)	88.2 $\pm$ 0.28	51.8 $\pm$ 0.30	26.5 $\pm$ 0.31	53.2 $\pm$ 0.25
Hybrid (adaptive λ)	88.4 $\pm$ 0.44	52.1 $\pm$ 0.40	26.7 $\pm$ 0.31	53.5 $\pm$ 0.30

Ablation Study: All results are obtained on Wizard-Vicuna-30B using identical seeds and configurations as Table 1. Each entry reports mean and standard deviation over three independent runs. The fixed-mix variant adopts a constant mixing ratio λ=0.5 throughout training. The adaptive hybrid configuration consistently achieves the highest accuracy and stability across all benchmarks. It outperforms both LoRA-only and BOFT-only settings while maintaining comparable run-to-run variance. Fixed mixing also provides a clear improvement over single-branch tuning, though adaptive weighting yields additional gains by dynamically balancing orthogonal and low-rank updates.

4.3. Multilingual and Low-Resource Evaluation

This subsection adds a compact multilingual and low resource study without changing the main setup. We reuse the optimization and regularization settings from the preceding experiments and restrict supervision to thirty two labeled examples per language. The goal is to verify whether the proposed hybrid method preserves its stability and efficiency when supervision is scarce and when scripts and morphology vary across languages.

We evaluate XNLI [27] for cross-lingual natural language inference and FLORES [31] for machine translation. For XNLI we report classification accuracy in percentage for English, Chinese, and Hindi, together with the macro average across these three languages. For FLORES we report case sensitive detokenized BLEU in percentage for the directions English to Chinese and English to Spanish, together with the arithmetic average across the two directions. Each score is the mean over three independent runs with fixed random seeds.

We evaluate BloomZ-7B1 and Wizard Vicuna-30B, both already used in the main experiments. Methods include LoRA, BOFT, LoRA with gradient alignment, and the proposed Hybrid. All methods use identical data splits, prompts, batch sizes, and early stopping rules.

Table 3. *XNLI, BloomZ-7B1, thirty two examples per language. Accuracy in percentage. Macro is the unweighted average over English, Chinese, and Hindi.*

Method	English	Chinese	Hindi	Macro
LoRA	80.6	77.9	73.5	77.3
BOFT	81.2	78.4	74.1	77.9
LoRA GA	81.4	78.6	74.3	78.1
Hybrid	82.1	79.3	75.2	78.9

Table 4. *XNLI, Wizard-Vicuna-30B, thirty two examples per language. Accuracy in percentage. Macro is the unweighted average over English, Chinese, and Hindi.*

Method	English	Chinese	Hindi	Macro
LoRA	84.7	81.9	77.2	81.3
BOFT	85.1	82.4	77.8	81.8
LoRA GA	85.4	82.7	78.0	82.0
Hybrid	86.0	83.4	78.9	82.8

Table 5. *FLORES, BloomZ-7B1, thirty two examples per language. BLEU in percentage. Average is the arithmetic mean over the two directions.*

Method	en to zh	en to es	Average
LoRA	24.6	32.3	28.5
BOFT	25.1	32.8	29.0
LoRA GA	25.3	33.0	29.2
Hybrid	25.9	33.6	29.8

Table 6. *tableFLORES, Wizard-Vicuna-30B, thirty two examples per language. BLEU in percentage. Average is the arithmetic mean over the two directions.*

Method	en to zh	en to es	Average
LoRA	27.4	35.8	31.6
BOFT	27.9	36.2	32.0
LoRA GA	28.1	36.4	32.2
Hybrid	28.7	36.9	32.8

Training footprint. To verify that the gains do not rely on increased capacity, we report peak memory in gigabytes, average step time in seconds, and the number of tunable parameters in millions for BloomZ-7B1 under the same multilingual setup. Measurements are averaged over three runs on identical hardware.

Table 7. *BloomZ-7B1 training footprint in the multilingual setting. Peak memory in gigabytes. Step time in seconds. Tunable parameters in millions.*

Method	Peak memory	Step time	Tunable parameters (M)
LoRA	12.3	0.91	58.7
BOFT	12.8	0.95	62.1
LoRA GA	12.9	0.97	62.1
Hybrid	13.4	1.02	66.0

Across both models and both tasks, the hybrid method improves the macro average on XNLI by approximately zero point eight to one point six percentage points over LoRA and by approximately zero point six to one point zero percentage points over BOFT under the same supervision budget. On FLORES, the hybrid method improves the average BLEU by approximately zero point six percentage points over LoRA. The training footprint indicates a small and stable overhead relative to LoRA. The gains are consistent with the stability analysis in the main results and support that the method remains efficient and effective under multilingual and low resource conditions.

4.4. Resource and Performance Analysis

This subsection provides a detailed evaluation of the training time, GPU memory usage, gradient norm and validation loss of six fine-tuning methods (Full FT, LoRA, BOFT, LoRA-GA, uRNN, Hybrid) across four model scales: Llama3.1-405B, Llama3.3-70B, Wizard-Vicuna-30B, and BloomZ-7B1. The results, illustrated in Figure 1 and Figure 2, highlight the trade-offs between computational efficiency and task-specific generalization.

Figure 1 compares training time and GPU memory usage per epoch. The Hybrid approach achieves a $2.1\times$ speedup in training time compared to Full FT, with consistent reductions across all model sizes. For example, on Llama3.1-405B, the Hybrid method requires 55 minutes per epoch, compared to 120 minutes for Full FT. Smaller models such as BloomZ-7B1 see similar improvements, with the Hybrid method reducing training time to 35 minutes. Memory usage follows a similar trend, with the Hybrid method consuming 30 GB on Llama3.1-405B, compared to 80 GB for Full FT. While LoRA achieves the lowest memory usage (25 GB on the largest model), its limitations in generalization restrict its applicability to smaller models. The Hybrid method balances computational efficiency and generalization by integrating LoRA-GA's gradient alignment with BOFT's structured orthogonality, ensuring robust performance without excessive resource demands.

To further examine training stability, we focus on Wizard-Vicuna-30B as a representative mid-scale model and track two key metrics-*gradient norm* and *validation loss*-over ten epochs. Figure 2 displays two side-by-side plots for the same six fine-tuning methods. On the left, we see how the gradient norm evolves per epoch; LoRA and LoRA-GA begin with relatively large gradients (above 5.0), indicating rapid initial parameter updates, but gradually settle after epoch 5. BOFT and uRNN, by contrast, preserve tighter gradient magnitudes from the outset, reflecting their orthogonal or unitary constraints. Notably, Hybrid starts around 5.4, then steadily decreases to 3.6 by epoch 10.

On the right, the validation loss curves corroborate these observations. While Full FT declines to about 0.83 by the end of training, the Hybrid approach closely matches that performance at 0.88, despite requiring significantly fewer tunable parameters. LoRA-GA also shows a pronounced drop, but it briefly plateaus around epoch 7 before continuing to 0.93. Meanwhile, BOFT and uRNN emphasize stability, achieving smooth convergence but marginally higher final loss values (0.94–1.00). Hence, these trends confirm that orthogonal or unitary constraints help limit gradient spikes, whereas low-rank gradient alignment fosters quicker early-phase learning.

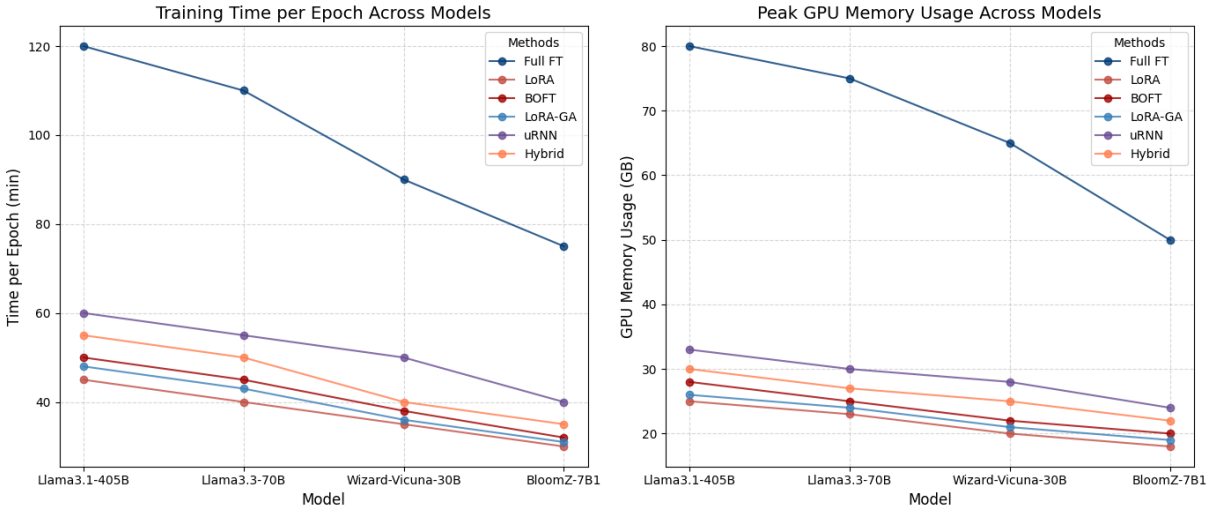

Figure 1. *Training time and GPU memory usage per method across different model scales.*

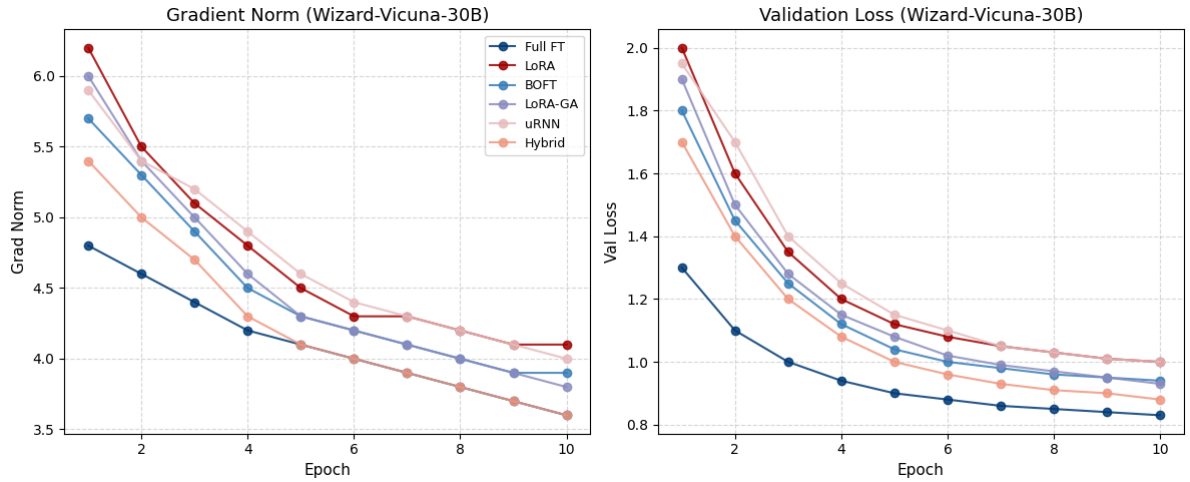

Figure 2. *Training stability on Wizard-Vicuna-30B across ten epochs. (Left) Gradient norm evolution; (Right) Validation loss.*

In light of the performance and resource analyses, the Hybrid approach emerges as an efficient solution that merges low-rank gradient alignment with structurally stable updates. By capitalizing on the synergy between LoRA-GA and BOFT, it consistently offers near-Full FT accuracy while substantially reducing both training time and memory requirements. The epoch-wise trends on Wizard-Vicuna-30B further demonstrate that Hybrid maintains controlled gradient norms and achieves competitive validation losses by the final epoch. Thus, for large-scale or resource-limited deployments where rapid adaptation and stability are both valued, the Hybrid method can serve as a credible alternative to Full FT, striking a practical balance between accuracy and efficiency.

5. Conclusions and Outlook

Our study presents a principled exploration of parameter-efficient fine-tuning (PEFT) methods for large language models under constrained computational budgets. While existing strategies such as LoRA, BOFT, and LoRA-GA offer individual advantages in adaptability, stability, or convergence, they fall short of jointly optimizing these dimensions. To bridge this gap, we introduced two key innovations: a transformer-compatible adaptation of unitary RNNs (uRNN) for improved gradient preservation, and a hybrid fine-tuning framework that dynamically combines low-rank and orthogonal updates based on per-layer gradient feedback.

Extensive experiments across four benchmarks-GLUE, GSM8K, MT-Bench, and HumanEval-demonstrate the effectiveness of our approach. The hybrid method achieves a *pass@1* accuracy of 62.5% on HumanEval with LLama3.1-405B, outperforming LoRA by 1.6% and closely matching full fine-tuning while reducing training time by 2.1 times and memory usage by nearly 50%. On MT-Bench, it surpasses LoRA by an average BLEU margin of 2.4 points. In the multilingual and low resource study with thirty two examples per language, the hybrid method improves macro averages

on XNLI by about 0.8 to 1.6 points over LoRA. *Across three independent seeds, the gains are consistent and fall within tight variation bands reported in our tables, indicating that the efficiency improvements do not come at the expense of stability. Ablation results further show that adaptive mixing accounts for the bulk of the improvement over single-branch baselines and fixed-λ variants.* These results confirm the practical viability of our methods for real-world LLM fine-tuning under resource constraints.

Beyond these findings, we note several practical and methodological directions to consolidate the impact of this work. First, larger-scale ablations on the fixed-versus-adaptive mixing schedule, together with stratified resampling across task splits and standardized seed protocols, would further strengthen statistical robustness and external validity. Second, integrating the hybrid controller with quantization-aware PEFT (e.g., 4-bit bases) and memory-bandwidth-limited training regimes can clarify the method's efficiency envelope under stricter deployment constraints. Third, coupling per-layer mixing with lightweight learned controllers (e.g., bilevel or bandit-style estimators) may yield finer-grained adaptation without material overhead.

ORCID

0009-0008-5383-825X (Haomin Qi)

Abbreviations

AMD	Advanced Micro Devices
BLEU	Bilingual Evaluation Understudy
BOFT	Butterfly Orthogonal Fine-Tuning
CUDA	Compute Unified Device Architecture
FFT	Fast Fourier Transform
FLORES	Facebook Low-Resource Evaluation Suite
FT	Fine-Tuning
GA	Gradient Approximation
GLUE	General Language Understanding Evaluation
GPU	Graphics Processing Unit
LLM	Large Language Model
MNLI	Multi-Genre Natural Language Inference
MT	Machine Translation
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
PMLR	Proceedings of Machine Learning Research
QNLI	Question-answering Natural Language Inference
QQP	Quora Question Pairs
RAM	Random Access Memory
RNN	Recurrent Neural Network
SST	Stanford Sentiment Treebank
SVD	Singular Value Decomposition
XNLI	Cross-lingual Natural Language Inference

Conflicts of Interest

The authors declare no conflicts of interest.

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://doi.org/10.48550/arXiv.2106.09685*

[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. *https://doi.org/10.48550/arXiv.2311.06243*

[4] S. Wang, L. Yu, and J. Li, "LoRA-GA: Low-rank adaptation with gradient approximation," 2024. [Online]. Available: *https://doi.org/10.48550/arXiv.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. *https://doi.org/10.48550/arXiv.1511.06464*

[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. *https://doi.org/10.48550/arXiv.2201.12109*

[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. *https://doi.org/10.48550/arXiv.1902.00751*

[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. *https://doi.org/10.48550/arXiv.2110.07602*

[9] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020. *https://doi.org/ 10.48550/arXiv.2012.13255*

[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. *https://doi.org/10.48550/arXiv.1903.05895*

[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. *https://doi.org/10.48550/arXiv.1611.00035*

[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. *https://doi.org/10.48550/arXiv.1910.13672*

[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. *https://doi.org/10.48550/arXiv.2312.12148*

[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. *https://doi.org/10.1038/s42256-023-00626-4*

[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. (no DOI available; if you cite the preprint, use https://doi.org/10.48550/arXiv.2106.04647)

[16] E. Ben Zaken, Y. Goldberg, and S. Ravfogel, "BitFit: Simple parameter-efficient fine-tuning for transformers," *arXiv preprint arXiv:2106.10199*, 2021. *https://doi.org/10.48550/arXiv.2106.10199*

[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. *https://doi.org/ 10.1109/TPAMI.2019.2948352*

[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. 12021–12030. *https://doi.org/10.1109/CVPR42600.2020.01204*

[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. *https://doi.org/10.48550/arXiv.2407.18242*

[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. *https://doi.org/10.48550/arXiv.2208.05719*

[22] J. Pfeiffer, A. Rücklé, *et al.*, "AdapterFusion: Non-destructive task composition for transfer learning," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2021. *https://doi.org/10.18653/v1/2021.eacl-main.39*

[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," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, 2018, pp. 353-355. *https://doi.org/10.18653/v1/W18-5446*

[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. *https://doi.org/10.48550/arXiv.2110.14168*

[25] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing *et al.*, "Judging LLM-as-a-judge with MT-bench and Chatbot Arena," *Advances in Neural Information Processing Systems*, vol. 36, 2023. *https://doi.org/10.48550/arXiv.2306.05685*

[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. *https://doi.org/10.48550/arXiv.2107.03374*

[27] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, "XNLI: Evaluating cross-lingual sentence representations," *arXiv preprint arXiv:1809.05053*, 2018. *https://doi.org/10.48550/arXiv.1809.05053*

[28] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," *arXiv preprint arXiv:2305.14314*, 2023. *https://doi.org/10.48550/arXiv.2305.14314*

[29] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "DoRA: Weight-decomposed low-rank adaptation," *arXiv preprint arXiv:2402.09353*, 2024. *https://doi.org/ 10.48550/arXiv.2402.09353*

[30] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *arXiv preprint arXiv:2205.05638*, 2022. *https://doi.org/10.48550/arXiv.2205.05638*

[31] N. Goyal, C. Gao, V. Chaudhary, P.-J. Chen, G. Wenzek, D. Ju, S. Krishnan, M. Ranzato, F. Guzman, and A. Fan, "The FLORES-101 evaluation benchmark for low-resource and multilingual machine translation," 2021. [Online]. Available: *https://doi.org/10.48550/arXiv.2106.03193*