Governance-Aware Hybrid Fine-Tuning for Multilingual Large Language Models

Haomin Qi[1], Chengbo Huang[2], Zihan Dai[3], Yunkai Gao[4]
[1]University of California San Diego, La Jolla, CA, USA
[2]Columbia University, New York City, NY, USA
[3]University of Copenhagen, Copenhagen, Denmark
[4]Duke University, Durham, NC, USA
Email: h5qi@ucsd.edu, ch4019@columbia.edu, cjh841@alumni.ku.dk, yunkai.gao@duke.edu

Abstract—We present a governance-aware hybrid fine-tuning framework for multilingual, low-resource adaptation of large language models. The core algorithm mixes gradient-aligned low-rank updates with structured orthogonal transformations through layer-wise mixing and introduces unitary constraints in selected sub-layers to stabilize deep optimization. In tandem with lightweight, label-free data governance—language identification, near-duplicate removal, and quality filtering—the framework targets accuracy, calibration, and cross-language parity under tight compute budgets. Across XNLI and FLORES, the hybrid approach delivers consistent gains over strong PEFT baselines while maintaining directional balance and improving probability calibration, as shown in Tables II and III. It is more resilient to lightweight orthographic variants, as shown in Table IV, and benefits additively from simple governance steps, as shown in Table V. Training-footprint measurements indicate modest overhead and a favorable cost–quality frontier, as shown in Table VI and Figure 2. Together these results show that hybrid and unitary PEFT provide a stable and accessible path to resource-efficient multilingual adaptation when paired with practical data governance.

Index Terms—*parameter-efficient fine-tuning, multilingual nlp, low-resource learning, large language models, robustness*

I. INTRODUCTION

Large Language Models (LLMs) have become central to Natural Language Processing (NLP) applications ranging from machine translation to code generation, yet full-model adaptation is often prohibitive in memory and time. Parameter-Efficient Fine-Tuning (PEFT) mitigates this by updating a small subset of parameters via lightweight adapters or structured transforms, with notable instances including Low-Rank Adaptation (LoRA) [1], Orthogonal/OFT-style updates refined as Butterfly Orthogonal Fine-Tuning (BOFT) [2], and gradient-aligned variants that steer updates along dominant directions [3]. Despite their efficiency, existing PEFT approaches trade off rapid early adaptation, stability in deep stacks, and representational capacity, particularly when supervision is scarce or distributional shifts induce fragile optimization.

We propose a *hybrid* PEFT framework that fuses gradient-aligned low-rank updates with structured orthogonal adjustments through layer-wise, gradient-norm–based mixing. Concretely, the low-rank branch (LoRA-style with gradient alignment) accelerates early progress by emphasizing dominant update directions, while the orthogonal branch (BOFT-style Cayley/orthogonal updates) preserves gradient magnitudes and

encourages stable late-phase optimization [4]. To further steady very deep Transformer stacks, we impose *selective unitary constraints* in chosen sub-layers using structured unitary parameterizations inspired by Unitary Recurrent Neural Networks (uRNNs) [5]. The result is a single-step, per-layer fusion that retains PEFT simplicity while improving stability and invariance at modest overhead; the mechanism is compatible with standard training loops and does not require altering the backbone architecture.

Our evaluation spans general-purpose and multilingual, low-resource settings across models from 7B to 405B parameters. We report accuracy and *Expected Calibration Error* (ECE) [6] on standard benchmarks—GLUE [7], GSM8K [8], MT-Bench [9], and HumanEval [10]—and study cross-language accuracy and parity on XNLI [11] and FLORES [12] under a 32-shot protocol. Because deployability depends on *how* models are evaluated and curated, we complement accuracy with cross-language parity (Parity Gap, Δ) and calibration, probe robustness to routine orthographic variants, and quantify the effect of lightweight, label-free curation (language identification and near-duplicate/quality filtering) using widely adopted tools and heuristics [13, 14]. Empirically, the hybrid method tracks or narrows the gap to full fine-tuning with far fewer trainable parameters, improves calibration and parity under limited supervision, exhibits smaller drops under orthographic variation, and remains near a favorable cost–quality frontier, as detailed in Tables I–VI and Figures 2–4.

We summarize the contributions of our paper as follows:

1) **Hybrid PEFT.** A layer-wise fusion of gradient-aligned low-rank and structured orthogonal updates via gradient-norm mixing, yielding fast early adaptation and stable late optimization under a fixed parameter budget.
2) **Unitary stabilization.** Selective unitary parameterizations for key Transformer sub-layers that control gradient growth in deep stacks with negligible overhead.
3) **Validated effectiveness.** Consistent gains over strong PEFT baselines from 7B to 405B on general-purpose and 32-shot multilingual tasks, with improved calibration, cross-language parity, and robustness to orthographic variants.

II. RELATED WORK

A. *Parameter-Efficient Fine-Tuning (PEFT)*

Parameter-Efficient Fine-Tuning (PEFT) reduces the number of trainable parameters for Large Language Models

(LLMs) in Natural Language Processing (NLP) by freezing most pre-trained weights and learning small task-specific modules. Early approaches include *adapters*, which insert bottleneck layers in Transformer blocks [15], and *prefix/prompt tuning*, which optimizes continuous prompts while keeping backbone parameters fixed [16, 17]. Subsequent methods improved parameter sharing and compression, e.g., *Compacter* [18], and even showed that updating only biases (*BitFit*) can be competitive for many tasks [19]. These techniques collectively emphasize modularity, small memory footprints, and ease of deployment in constrained environments.

Low-Rank Adaptation (LoRA) decomposes weight updates into low-rank matrices, offering a favorable accuracy–memory trade-off and becoming a widely used PEFT baseline. *QLoRA* combines PEFT with 4-bit quantization to further reduce memory while preserving headroom for quality [20]. In parallel, *gradient-aligned* schemes project or initialize updates along dominant gradient subspaces to improve early progress and memory use. Our work follows this line by combining a gradient-aligned low-rank branch with a structured orthogonal branch under a fixed PEFT budget, aiming to retain rapid adaptation while strengthening stability in deeper layers.

Whereas prior PEFT methods typically commit to a single update family (low-rank, orthogonal, prompts/adapters) or a single stabilization strategy, we combine *gradient-aligned low-rank* and *structured orthogonal* updates through a layer-wise mixing rule, and we introduce *selective unitary* constraints to steady deep stacks.

B. Orthogonal and Unitary Parameterizations

Orthogonal and unitary parameterizations preserve norms and have long been used to stabilize deep optimization. In recurrent networks, unitary/orthogonal transitions alleviate vanishing and exploding gradients [5, 21]. In the PEFT setting, *Butterfly Orthogonal Fine-Tuning* (BOFT) constrains updates via butterfly factorization, linking to fast linear transforms and orthogonal structure [2, 22]. Rather than replacing low-rank updates, we *fuse* orthogonal and low-rank branches per layer and add *selective unitary* constraints to key Transformer sublayers, seeking a complementary balance: the low-rank path emphasizes data-aligned efficiency, while the orthogonal path encourages stable long-horizon optimization in deep stacks.

C. Multilingual Adaptation, Calibration, and Robustness

Multilingual evaluation emphasizes cross-language generalization and parity across families and scripts. XNLI benchmarks cross-lingual natural language inference [11]; FLORES-101/200 measures translation quality across many languages [12, 23]. Beyond accuracy, *Expected Calibration Error* (ECE) captures the reliability of predicted probabilities in modern neural networks. Robustness to lightweight orthographic variation (e.g., punctuation normalization, diacritics removal, whitespace changes) is relevant for user-facing systems; small character-level perturbations can degrade performance in classification and translation [24, 25]. Our experimental protocol aligns with this perspective by reporting accuracy with Parity Gap (Δ), ECE, and sensitivity to orthographic variants under a matched few-shot budget.

D. Data Governance and Budget-Aware Adaptation

Data governance practices such as language identification, near-duplicate removal, and light quality filtering are standard in web-scale pipelines and benefit multilingual robustness [26]. These steps are complementary to PEFT and quantization when compute is constrained. Our study integrates such lightweight curation into the evaluation protocol and quantifies its effect in a 32-shot setting. Related retrieval-augmented pipelines likewise emphasize budget-aware adaptation and robustness in applied domains [27, 28], reinforcing the value of pairing efficient algorithms with pragmatic governance to meet quality–cost targets.

III. 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.

A. 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 [18, 29]. 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:

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

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

B. 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 [30]; 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 [31]:

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

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

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

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

C. 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 $\nabla_{\mathbf{W}_0} \mathcal{L}$:

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

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

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

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 [32]:

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

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.

D. 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 [21, 33]. 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 [34]. 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

1: **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.
2: Choose learning rate η and total epochs E.
3: **for** $epoch = 1$ **to** E **do**
4:     **for** each minibatch in the dataset **do**
5:         **Forward Pass:**
6:         **for** each transformer layer with unitary-constrained weight **do**
7:             Replace original weight matrix with current unitary matrix $\mathbf{U}$.
8:             Compute the forward pass using the updated unitary matrix $\mathbf{U}$.
9:         **end for**
10:         Compute the task-specific loss $\mathcal{L}$ based on current minibatch predictions.
11:         **Backward Pass:**
12:         Compute gradient $\nabla_{\mathbf{U}} \mathcal{L}$ via backpropagation.
13:         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 $\mathbf{U}$.
14:         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)
15:         If numerical drift occurs, re-normalize $\mathbf{U}$ to strictly enforce unitarity.
16:         Update all other non-unitary parameters of the model as usual via standard gradient descent.
17:     **end for**
18: **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:

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

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:

$$\mathbf{U}_{t+1} = \exp(\eta\,\mathbf{B})\,\mathbf{U_t}, \qquad (12)$$

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 U in the forward pass of the chosen transformer layer and then update U using the above rule at each training step, 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 U as unitary thus helps preserve gradient norms through the depth of the network, even without explicit recurrence [35].[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 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.

E. 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.

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

---

Algorithm 2 Hybrid Fine-Tuning Procedure

1: **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.
2: Set learning rates $\eta_{\text{LoRA}}, \eta_{\text{BOFT}}$; choose rank r, total epochs E.
3: **for** $epoch = 1$ **to** E **do**
4:   **for** each minibatch in the dataset **do**
5:     **Forward Pass:**
6:     **for** each transformer layer ℓ **do**
7:       Compute LoRA-GA update: $\Delta\mathbf{W}^\ell_{\text{LoRA}} = \mathbf{A}^\ell\mathbf{B}^\ell$.
8:       Compute BOFT orthonormal matrix:
      $\mathbf{R}^\ell = (\mathbf{I} + \eta_{\text{BOFT}}\mathbf{Q}^\ell)(\mathbf{I} - \eta_{\text{BOFT}}\mathbf{Q}^\ell)^{-1}$.
9:       Compute BOFT update:
      $\Delta\mathbf{W}^\ell_{\text{BOFT}} = (\mathbf{R}^\ell - \mathbf{I})\mathbf{W}^\ell$.
10:     **end for**
11:     Compute task-specific loss $\mathcal{L}$ using model predictions.
12:     **Backward Pass:**
13:     **for** each transformer layer ℓ **do**
14:       Compute gradient norms:
      $g^\ell_{\text{LoRA}} = \|\nabla_{\mathbf{A}^\ell,\mathbf{B}^\ell}\mathcal{L}\|, \quad g^\ell_{\text{BOFT}} = \|\nabla_{\mathbf{Q}^\ell}\mathcal{L}\|$.
15:       Compute dynamic weighting coefficient:
      $\lambda^\ell = \frac{g^\ell_{\text{LoRA}}}{g^\ell_{\text{LoRA}}+g^\ell_{\text{BOFT}}}$.
16:       Form hybrid update for layer ℓ:
      $\Delta\mathbf{W}^\ell_{\text{hybrid}} = \lambda^\ell\Delta\mathbf{W}^\ell_{\text{LoRA}} + (1 - \lambda^\ell)\Delta\mathbf{W}^\ell_{\text{BOFT}}$.
17:       Update weight matrix for layer ℓ:
      $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell + \Delta\mathbf{W}^\ell_{\text{hybrid}}$.
18:       Update low-rank matrices via gradient descent:
      $\mathbf{A}^\ell \leftarrow \mathbf{A}^\ell - \eta_{\text{LoRA}}\nabla_{\mathbf{A}^\ell}\mathcal{L}$,
      $\mathbf{B}^\ell \leftarrow \mathbf{B}^\ell - \eta_{\text{LoRA}}\nabla_{\mathbf{B}^\ell}\mathcal{L}$.
19:       Compute skew-symmetric gradient matrix for BOFT:
      $\mathbf{G}^\ell = \nabla_{\mathbf{Q}^\ell}\mathcal{L} - (\nabla_{\mathbf{Q}^\ell}\mathcal{L})^\top$.
20:       Update skew-symmetric matrix $\mathbf{Q}^\ell$:
      $\mathbf{Q}^\ell \leftarrow \mathbf{Q}^\ell - \eta_{\text{BOFT}}\mathbf{G}^\ell$.
21:       Recompute orthonormal matrix $\mathbf{R}^\ell$ via Cayley transform (for numerical stability):
      $\mathbf{R}^\ell \leftarrow (\mathbf{I} + \eta_{\text{BOFT}}\mathbf{Q}^\ell)(\mathbf{I} - \eta_{\text{BOFT}}\mathbf{Q}^\ell)^{-1}$.
22:     **end for**
23:     Update other model parameters (if any) via standard gradient descent.
24:   **end for**
25: **end for**

LoRA [36], 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 [2]. 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 [22]:

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

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

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

---

[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.

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

$$\Delta \mathbf{W}_{\mathrm{hybrid}}^{\ell} = \lambda_t^{\ell} \Delta \mathbf{W}_{\mathrm{LoRA}}^{\ell} + (1 - \lambda_t^{\ell}) \Delta \mathbf{W}_{\mathrm{BOFT}}^{\ell},$$

and the weight is updated as $\mathbf{W}^{\ell} \leftarrow \mathbf{W}^{\ell} + \Delta \mathbf{W}_{\mathrm{hybrid}}^{\ell}$. 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 [37] $(\mathbf{A}^{\ell}, \mathbf{B}^{\ell})$ 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_t^{ℓ}, 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 $\mathcal{O}(d_{\mathrm{out}} r + r\, d_{\mathrm{in}})$ and the BOFT transform costs $\mathcal{O}(d_{\mathrm{out}} \log d_{\mathrm{out}})$. Computing λ_t^{ℓ} 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.

IV. EXPERIMENTS

This section evaluates parameter-efficient fine-tuning strategies in both general-purpose and multilingual settings. The analysis focuses on two aspects: effectiveness under limited supervision and deployability under realistic compute budgets. We report accuracy, calibration, and cross-language parity together with training footprint, training dynamics, and the relationship between cost and quality. Unless stated otherwise, each number is the mean of three runs with fixed seeds and identical optimization settings across methods.

A. Setup and Evaluation Protocol

Models. We consider four open models that span size and multilingual coverage: Llama3.1-405B (405B parameters, long-context transformer) [38], Llama3.3-70B (70B) [38], Wizard-Vicuna-30B (30B, multilingual) [39, 40], and BloomZ-7B1 (7.1B, multilingual) [41].

Methods. The comparison includes Full Fine-Tuning (Full FT), LoRA, BOFT, LoRA with gradient alignment (LoRA-GA), unitary constraints inspired by uRNN (uRNN), and the proposed Hybrid method that combines low-rank gradient alignment with structured orthogonal updates. Hyperparameters follow a single protocol across tasks: LoRA rank $r=16$ with scaling $\alpha=32$; BOFT uses butterfly depth $m=3$; Hybrid applies a dynamic gradient weight α_t.

Benchmarks and metrics. We use GLUE (macro accuracy across MNLI, QQP, SST-2, and QNLI) [7], GSM8K (math reasoning accuracy) [8], MT-Bench (BLEU used as a multilingual translation proxy) [9], and HumanEval for code generation measured by *pass@1* [10]. For multilingual low-resource evaluation we report XNLI (accuracy on English, Chinese, and Hindi) [11] and FLORES (detokenized BLEU for EN→ZH and EN→ES) [12]. Cross-language parity is summarized by the Parity Gap Δ, defined as the maximum minus the minimum score across languages, where smaller is better. Calibration is measured by the average Expected Calibration Error (Avg-ECE, in percentage), where smaller is better.

Compute. Experiments run on nodes with dual AMD EPYC 7742 CPUs, 1 TB RAM, and 8×NVIDIA A100 GPUs connected by NVLink and InfiniBand. The software stack uses PyTorch 2.0, CUDA 11.8, and NVIDIA Apex for mixed-precision training.

B. General-Purpose Results

Table I compares all methods on GLUE, GSM8K, MT-Bench, and HumanEval. Hybrid tracks Full FT closely while keeping the number of trainable parameters small. On GLUE, Hybrid averages 92.3% on the 405B model, which is 0.2 points below Full FT and 1.0 point above LoRA. On GSM8K, Hybrid reaches 55.9% on 405B, slightly higher than Full FT and 1.7 points above LoRA. MT-Bench shows consistent BLEU gains over LoRA, BOFT, and LoRA-GA across sizes; on the 70B model Hybrid achieves 28.1 BLEU. On HumanEval, Hybrid attains 62.5% *pass@1* on 405B and 40.5% on BloomZ-7B1, narrowing the gap to Full FT while remaining parameter-efficient. These findings suggest that combining gradient alignment with structural constraints yields robust generalization without full-model updates.

Trends are stable across scale. Relative to LoRA, the Hybrid advantage on GLUE is 1.0 at 405B, 1.1 at 70B, 0.9 at 30B, and 1.1 at 7B1. On GSM8K the margins are 1.7, 1.5, 1.3, and 1.3 points. MT-Bench follows the same pattern; for example, at 70B the averages are 27.8 for Hybrid and 27.0 for LoRA. On HumanEval, the edge is consistent and moderate; at 405B the averages are 62.5 for Hybrid and 61.0 for LoRA, and at 70B the averages are 58.5 and 56.9. BOFT and LoRA-GA reduce part of the gap relative to LoRA on several tasks, but Hybrid remains the most uniform alternative to Full FT under the shared protocol.

C. Multilingual and Low-resource Evaluation

We evaluate few-shot multilingual adaptation with 32 labeled examples per language. For XNLI [11], we report per-language accuracy on English (EN), Chinese (ZH), and Hindi (HI), as well as Macro, Parity Gap Δ (max–min across languages; lower is better), and Avg-ECE. For FLORES [12], we report detokenized BLEU for EN→ZH and EN→ES, their average, and the direction gap Δ.

Table II shows consistent gains for *Hybrid* across both backbones. Relative to LoRA, macro accuracy increases by +0.8 to +1.6 points; relative to BOFT, by +0.6 to +1.0. Improvements are not concentrated in a single language: gains are largest on HI (e.g., +1.7 over LoRA on BloomZ and +1.7 on Wizard-Vicuna), moderate on ZH, and smaller but steady on EN. This pattern suggests that the method transfers better to lower-resource or morphologically distinct conditions without sacrificing high-resource performance.

TABLE I: **Performance across GLUE, GSM8K, MT-Bench, and HumanEval (*pass@1*).** GLUE/GSM8K/MT-Bench report three runs and the average (Avg). HumanEval cells list three *pass@1* runs and Avg.

Benchmark	Method	Llama3.1 405B	Llama3.3 70B	Wizard-Vicuna-30B	BloomZ 7B1
GLUE (%)					
	Full FT	91.0 / 94.0 / 92.5 **Avg 92.5**	90.0 / 91.0 / 91.1 **Avg 90.7**	87.8 / 90.0 / 89.0 **Avg 88.9**	84.9 / 86.2 / 86.3 **Avg 85.8**
	LoRA	90.2 / 91.5 / 92.2 **Avg 91.3**	88.9 / 89.2 / 89.3 **Avg 89.1**	87.3 / 87.4 / 87.8 **Avg 87.5**	83.7 / 84.2 / 84.1 **Avg 84.0**
	BOFT	91.5 / 92.0 / 91.6 **Avg 91.7**	89.1 / 89.8 / 89.3 **Avg 89.4**	87.7 / 87.9 / 87.8 **Avg 87.8**	84.0 / 84.5 / 84.4 **Avg 84.3**
	LoRA-GA	91.4 / 92.3 / 92.0 **Avg 91.9**	89.4 / 89.8 / 89.6 **Avg 89.6**	87.6 / 87.9 / 88.2 **Avg 87.9**	84.1 / 84.3 / 84.8 **Avg 84.4**
	uRNN	90.1 / 91.5 / 91.1 **Avg 90.9**	88.0 / 88.8 / 88.7 **Avg 88.5**	86.0 / 86.7 / 87.5 **Avg 86.7**	82.6 / 83.9 / 84.0 **Avg 83.5**
	Hybrid	91.7 / 93.1 / 92.1 **Avg 92.3**	89.8 / 90.3 / 90.4 **Avg 90.2**	87.9 / 88.6 / 88.7 **Avg 88.4**	84.6 / 85.2 / 85.4 **Avg 85.1**
GSM8K (%)					
	Full FT	55.6 / 56.0 / 55.5 **Avg 55.7**	53.0 / 53.5 / 52.8 **Avg 53.1**	51.3 / 51.9 / 51.2 **Avg 51.5**	48.7 / 49.0 / 49.0 **Avg 48.9**
	LoRA	53.8 / 54.4 / 54.3 **Avg 54.2**	51.1 / 51.9 / 51.5 **Avg 51.5**	50.6 / 51.0 / 50.8 **Avg 50.8**	47.8 / 48.2 / 48.0 **Avg 48.0**
	BOFT	54.7 / 55.0 / 54.7 **Avg 54.8**	51.9 / 52.1 / 52.0 **Avg 52.0**	51.0 / 51.2 / 51.4 **Avg 51.2**	48.4 / 48.5 / 48.6 **Avg 48.5**
	LoRA-GA	54.2 / 54.8 / 54.8 **Avg 54.6**	52.1 / 52.4 / 52.0 **Avg 52.2**	51.3 / 51.6 / 51.2 **Avg 51.4**	48.5 / 48.8 / 48.7 **Avg 48.7**
	uRNN	54.2 / 54.7 / 54.6 **Avg 54.5**	51.7 / 51.9 / 51.8 **Avg 51.8**	50.7 / 51.1 / 51.2 **Avg 51.0**	48.0 / 48.4 / 48.2 **Avg 48.2**
	Hybrid	55.3 / 56.0 / 56.4 **Avg 55.9**	52.8 / 53.1 / 53.0 **Avg 53.0**	51.7 / 52.1 / 52.5 **Avg 52.1**	48.7 / 49.4 / 49.8 **Avg 49.3**
MT-Bench (BLEU)					
	Full FT	28.7 / 29.8 / 29.4 **Avg 29.3**	27.5 / 28.2 / 27.9 **Avg 27.9**	26.0 / 26.8 / 26.4 **Avg 26.4**	24.5 / 25.0 / 24.8 **Avg 24.8**
	LoRA	28.4 / 29.1 / 28.5 **Avg 28.7**	27.0 / 27.6 / 27.3 **Avg 27.3**	25.4 / 26.0 / 25.9 **Avg 25.8**	23.8 / 24.4 / 24.7 **Avg 24.3**
	BOFT	28.6 / 29.2 / 29.0 **Avg 28.9**	27.1 / 27.8 / 27.5 **Avg 27.5**	25.6 / 26.2 / 26.0 **Avg 26.0**	24.0 / 24.6 / 24.9 **Avg 24.5**
	LoRA-GA	28.7 / 29.3 / 29.1 **Avg 29.0**	27.2 / 27.8 / 27.7 **Avg 27.7**	25.9 / 26.4 / 26.2 **Avg 26.2**	24.2 / 24.7 / 25.2 **Avg 24.7**
	uRNN	28.2 / 29.1 / 28.5 **Avg 28.6**	26.7 / 27.5 / 27.1 **Avg 27.1**	25.2 / 26.1 / 25.7 **Avg 25.7**	23.7 / 24.5 / 24.4 **Avg 24.2**
	Hybrid	29.0 / 29.6 / 29.7 **Avg 29.4**	27.8 / 28.3 / 28.1 **Avg 28.1**	26.4 / 26.8 / 27.0 **Avg 26.7**	25.1 / 25.7 / 25.4 **Avg 25.4**
HumanEval (*pass@1*, %)					
	Full FT	61.8 / 62.2 / 62.0 **Avg 62.0**	**57.9 / 58.8 / 58.6** **Avg 58.4**	54.0 / 54.3 / 54.1 **Avg 54.1**	41.1 / 41.5 / 41.3 **Avg 41.3**
	LoRA	60.8 / 61.2 / 61.0 **Avg 61.0**	**56.4 / 57.3 / 56.9** **Avg 56.9**	51.9 / 52.2 / 52.0 **Avg 52.0**	39.0 / 39.7 / 39.5 **Avg 39.5**
	BOFT	61.3 / 61.6 / 61.4 **Avg 61.4**	**56.8 / 57.6 / 57.2** **Avg 57.2**	52.3 / 52.7 / 52.5 **Avg 52.5**	39.6 / 40.1 / 39.9 **Avg 39.9**
	LoRA-GA	61.4 / 61.7 / 61.5 **Avg 61.5**	**57.1 / 57.9 / 57.5** **Avg 57.5**	52.6 / 52.8 / 52.7 **Avg 52.7**	39.5 / 39.9 / 39.7 **Avg 39.7**
	uRNN	60.5 / 61.1 / 60.8 **Avg 60.8**	**55.8 / 56.7 / 56.3** **Avg 56.3**	51.6 / 52.2 / 51.9 **Avg 51.9**	38.6 / 39.4 / 39.0 **Avg 39.0**
	Hybrid	62.1 / 62.9 / 62.5 **Avg 62.5**	**58.0 / 58.9 / 58.6** **Avg 58.5**	53.2 / 53.8 / 53.5 **Avg 53.5**	40.2 / 40.8 / 40.5 **Avg 40.5**

TABLE II: XNLI, 32-shot per language. Accuracy (%), Macro, Parity Gap Δ (%), Avg-ECE (%).

Method	BloomZ-7B1						Wizard-Vicuna-30B					
	EN	ZH	HI	Macro	Δ	Avg-ECE	EN	ZH	HI	Macro	Δ	Avg-ECE
LoRA	80.6	77.9	73.5	77.3	7.1	2.8	84.7	81.9	77.2	81.3	7.5	2.2
BOFT	81.2	78.4	74.1	77.9	7.1	2.5	85.1	82.4	77.8	81.8	7.3	2.0
LoRA-GA	81.4	78.6	74.3	78.1	7.1	2.4	85.4	82.7	78.0	82.0	7.4	1.9
Hybrid	82.1	79.3	75.2	78.9	6.9	2.1	86.0	83.4	78.9	82.8	7.1	1.7

TABLE III: FLORES, 32-shot per language. Detokenized BLEU (%), average, and direction gap Δ (%).

Method	BloomZ-7B1				Wizard-Vicuna-30B			
	EN→ZH	EN→ES	Avg	Δ	EN→ZH	EN→ES	Avg	Δ
LoRA	24.6	32.3	28.5	7.7	27.4	35.8	31.6	8.4
BOFT	25.1	32.8	29.0	7.7	27.9	36.2	32.0	8.3
LoRA-GA	25.3	33.0	29.2	7.7	28.1	36.4	32.2	8.3
Hybrid	25.9	33.6	29.8	7.7	28.7	36.9	32.8	8.2

Cross-language balance slightly improves. On BloomZ-7B1, the Parity Gap narrows from 7.1 to 6.9; on Wizard-Vicuna-30B, from 7.5 to 7.1. Calibration also improves: Avg-ECE drops from 2.8% to 2.1% on BloomZ and from 2.2% to 1.7% on Wizard-Vicuna, indicating that probability estimates become more reliable while accuracy rises.

Table III shows that translation trends mirror classification. *Hybrid* yields average BLEU gains of about $+0.6$ over LoRA on BloomZ and about $+0.6$ over LoRA-GA on Wizard-Vicuna, while the direction gap Δ remains essentially unchanged (e.g., 7.7 on BloomZ) or slightly shrinks (from 8.4 to 8.2 on Wizard-Vicuna). Quality gains therefore do not come at the expense of directional balance. Taken together, the results indicate that under the same 32-shot budget, *Hybrid* improves accuracy and calibration uniformly across languages and tasks.

D. Robustness to Orthographic Variants and Data Curation Effects

We evaluate two practical axes on XNLI. First, robustness to lightweight orthographic variants where each example is perturbed by exactly one operation (punctuation normalization, diacritics removal, or whitespace compaction). The metric is the macro-accuracy drop relative to clean input (lower is better). Second, governance-oriented data curation for BloomZ-7B1 under a 32-shot setting with three stages: C0 (raw crawl), C1 (+language identification and near-duplicate removal), and C2 (+light perplexity- and length-based filtering). We report macro accuracy, Parity Gap Δ (max–min across languages; lower is better), and Avg-ECE.

TABLE IV: XNLI robustness to lightweight orthographic variants (macro accuracy drop, $\downarrow$ %). Lower is better.

Method	BloomZ-7B1	Wizard-30B
LoRA	3.2	2.7
BOFT	2.6	2.3
LoRA-GA	2.5	2.2
Hybrid	2.1	1.9

Across both backbones, Hybrid shows the smallest degradation (2.1/1.9), improving over LoRA by 1.1/0.8 points and remaining ahead of BOFT and LoRA-GA. The consistent ordering indicates higher invariance to frequent, low-severity input variations.

Curation produces monotonic gains: macro accuracy rises by $+1.5$ points from C0 to C2, while Δ and Avg-ECE drop by 0.6 and 0.8, respectively. Together with Table IV, these results suggest that the hybrid method reduces sensitivity to orthographic variation and that light, label-free governance improves cross-language parity and calibration under the same adaptation budget.

TABLE V: Data curation ablation on XNLI (BloomZ-7B1, 32-shot).

Stage	Macro (%) ↑	Δ (%) ↓	Avg-ECE (%) ↓
C0: Raw	77.4	7.5	2.9
C1: +LID +Dedup	78.2	7.2	2.5
C2: +Quality Filtering	78.9	6.9	2.1

E. Few-shot Scaling and Training Footprint

We study 0, 8, 32, and 128-shot scaling on XNLI with BloomZ-7B1 (Figure 1). At zero-shot, *Hybrid* has a small advantage; the margin widens to about $+1.6$ points at 32-shot and $+1.9$ points at 128-shot over LoRA.
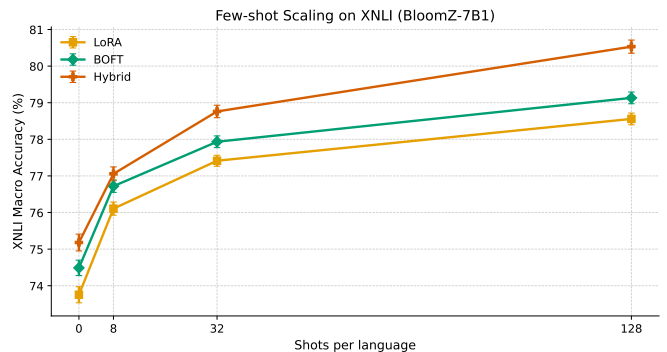

Fig. 1: Few-shot scaling on XNLI (BloomZ-7B1). Macro accuracy at 0, 8, 32, and 128 shots.

We then measure training footprint under the multilingual 32-shot protocol on Wizard-Vicuna-30B and BloomZ-7B1 (Table VI) and summarize the cost–quality frontier (Figure 2). *Hybrid* introduces modest overhead: approximately 0.6–0.7 GB higher peak memory and 0.1–0.2 s longer step time at comparable scales, with only tens of millions of additional tunable parameters. Given the accuracy and calibration gains, its points lie near the empirical Pareto front, indicating improved quality at nearly the same budget.

F. Training Dynamics and Stability

Figure 3 reports per-epoch time and peak memory across scales, and Figure 4 tracks gradient norms and validation loss over epochs on Wizard-Vicuna-30B. *Hybrid* suppresses early spikes in gradient norm and descends faster in the first few epochs, narrowing the gap to Full FT by the mid-stage. Final validation loss reaches 0.88 versus 0.83 for Full FT, while using far fewer tunable parameters, indicating that most of the quality of full adaptation is recovered at a fraction of the cost.

TABLE VI: Training footprint in the multilingual 32-shot setting. Peak memory (GB), step time (s), and tunable parameters (M).

	Wizard-Vicuna-30B			BloomZ-7B1		
Method	Peak mem.	Step time	Tunable	Peak mem.	Step time	Tunable
LoRA	14.8	1.35	93.2	12.3	0.91	58.7
BOFT	15.4	1.41	98.7	12.8	0.95	62.1
LoRA-GA	15.5	1.44	98.7	12.9	0.97	62.1
Hybrid	16.1	1.51	104.3	13.4	1.02	66.0

Fig. 2: Cost–quality frontier across methods. Upper-left is better.

Fig. 3: Per-epoch training time and peak GPU memory per method across model scales.

LoRA and LoRA-GA start with larger gradients and exhibit a short plateau before resuming improvement, matching the smaller few-shot gains observed in the scaling study. BOFT and uRNN maintain tighter gradients but converge to slightly higher losses, consistent with their weaker macro accuracy. The fused design of *Hybrid* combines rapid early progress with stable late-phase optimization, which aligns with its robustness and parity improvements: the method learns quickly enough to exploit limited supervision while avoiding destabilizing updates later in training.

V. CONCLUSIONS AND OUTLOOK

This work presents a *governance-aware* recipe for multilingual, low-resource adaptation of Large Language Models through a *Hybrid* Parameter-Efficient Fine-Tuning (PEFT) scheme with selective unitary stabilization. Rather than introducing yet another isolated adapter, we studied how a single-step, per-layer fusion behaves under realistic constraints in which accuracy must be accompanied by calibration, cross-language parity, robustness to routine input variants, and a budgeted training footprint. Across general-purpose tasks, *Hybrid* tracks full-model fine-tuning while maintaining a small trainable set, and on code generation reaches 62.5% *pass@1* at 405B. In 32-shot multilingual adaptation, it improves XNLI macro accuracy by +0.8–1.6 over LoRA and +0.6–1.0 over BOFT, and increases FLORES average BLEU by +0.6 without widening direction gaps. It further exhibits the smallest degradation under lightweight orthographic variants (2.1/1.9 pp drops on BloomZ-7B1/Wizard-Vicuna-30B), and stays near a favorable cost–quality frontier with only modest overhead (+0.6–0.7 GB peak memory and +0.1–0.2 s step time). Together with reduced calibration error and narrower parity gaps under simple curation, these findings indicate that mixing gradient-aligned low-rank updates with structured orthogonal adjustments—and reinforcing deep stacks via selective unitary constraints—yields a stable, calibrated, and resource-efficient path for multilingual fine-tuning under tight budgets.

Beyond headline metrics, our analysis highlights two practical aspects. First, the fused update suppresses early gradient spikes and accelerates mid-stage descent while preserving late-epoch stability, which helps recover much of the full-tuning quality at a fraction of the trainable parameters. Second, the evaluation protocol—pairing accuracy with Expected Calibration Error, parity gaps, orthographic robustness, and light, label-free data stewardship—offers a compact but actionable lens for deployment readiness, particularly when practitioners operate under fixed memory and time budgets. The overall recipe remains simple: a per-layer, single-step combination that is compatible with standard PEFT pipelines and amenable to existing mixed-precision and scheduling stacks.

Our findings point to several directions. *(i) Learning the mixer.* The current gradient-norm mixing is simple and effective; learning the per-layer schedule (e.g., via meta-gradients or small controllers) under a joint accuracy–calibration–parity objective could further improve the frontier. *(ii) Hardware-aware integration.* Combining *Hybrid* with quantization-aware training (e.g., low-bit activations) [42] and memory schedulers may reduce the residual overhead while preserving stability in deep stacks. *(iii) Broader multilingual stress-tests.* Extending beyond EN/ZH/HI to more scripts and families, and jointly
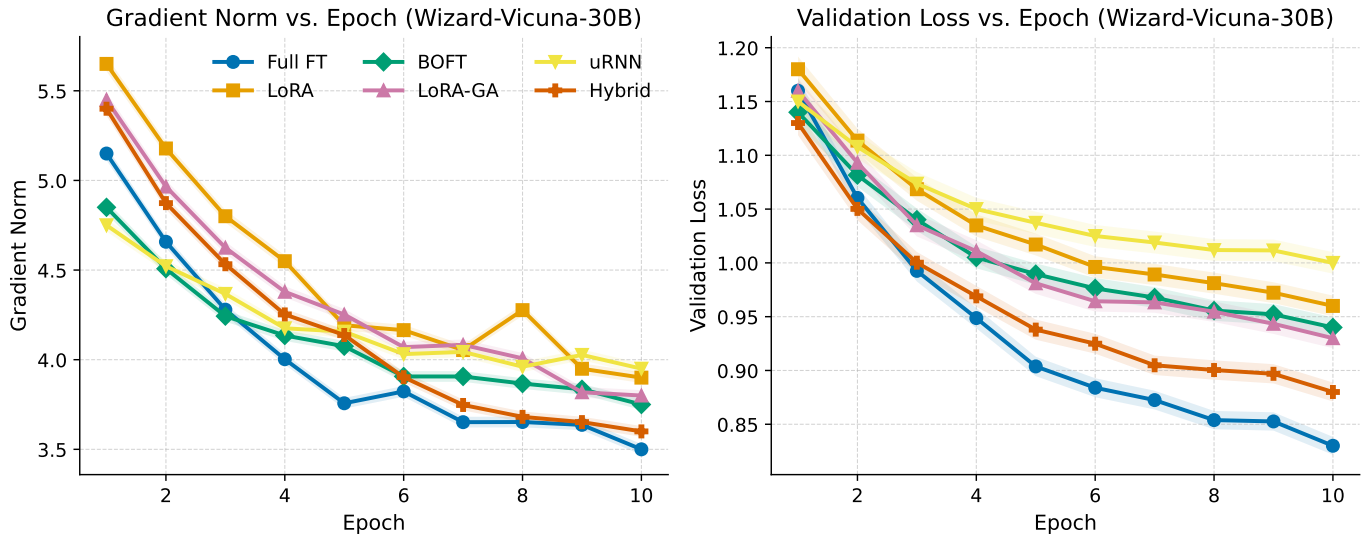
Fig. 4: Training stability on Wizard-Vicuna-30B across ten epochs: gradient norm and validation loss.

measuring fairness-adjacent disparity metrics alongside ECE, would clarify when parity gains persist at scale [43]. *(iv) Robustness under real noise.* Beyond orthographic tweaks, evaluating resilience to tokenization drift [44], locale-specific normalization, and web-crawl artifacts can better align with deployment realities.

REFERENCES

[1] 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

[2] 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 fine-tuning via butterfly factorization," in *ICLR*, 2024.

[3] Z. Zhang, T. Lin, Y. He, X. Sun, L. Yuan, Y. Liu, Z. Wang *et al.*, "Galore: Memory-efficient llm training by gradient low-rank projection," *arXiv preprint arXiv:2403.03507*, 2024. [Online]. Available: https://arxiv.org/abs/2403.03507

[4] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal RNNs and long-memory tasks with the scaled cayley transform," in *International Conference on Machine Learning (ICML)*, 2018. [Online]. Available: https://proceedings.mlr.press/v80/helfrich18a.html

[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] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning (ICML)*, 2017. [Online]. Available: http://proceedings.mlr.press/v70/guo17a.html

[7] 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 *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=rJ4km2R5t7

[8] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, J. Schulman, J. Kaplan, D. Amodei, I. Sutskever, and W. Zaremba, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[9] L. Zheng, W.-L. Chiang, Y. S. Li, Z. Lin, Z. Huang, S. Zhou, S. Zhuang, Y. Zhong, J. Li, C. Li, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge: Multi-turn MT-Bench and Chatbot Arena," *arXiv preprint arXiv:2306.05685*, 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[10] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, and et al., "Evaluating large language models trained on code," in *NeurIPS Datasets and Benchmarks*, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[11] 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.

[12] N. Goyal, P.-J. Gao, V. Chaudhary, G. Wenzek, D. Ju, S. Krishnan, F. Guzmán, P. Koehn, A. Fan, S. Bhosale *et al.*, "The FLORES-101 evaluation benchmark for low-resource and multilingual machine translation," *arXiv preprint arXiv:2106.03193*, 2021. [Online]. Available: https://arxiv.org/abs/2106.03193

[13] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *LREC*, 2018. [Online]. Available: https://aclanthology.org/L18-1550/

[14] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, 2002, pp. 380–388. [Online]. Available: https://dl.acm.org/doi/10.1145/509907.509965

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] E. Ben Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Annual Meeting of the Association for Computational Linguistics (ACL) Findings*, 2022. [Online]. Available: https://aclanthology.org/2022.findings-acl.174/

[20] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[21] 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.

[22] 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.

[23] M. R. Costa-jussà, J. Cross *et al.*, "No language left behind: Scaling human-centered machine translation," *arXiv preprint arXiv:2207.04672*, 2022. [Online]. Available: https://arxiv.org/abs/2207.04672

[24] Y. Belinkov and Y. Bisk, "Synthetic and natural noise both break neural machine translation," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=H196sainb

[25] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019. [Online]. Available: https://aclanthology.org/P19-1561/

[26] G. Wenzek, M.-A. Lachaux, A. Conneau, and A. Joulin, "CCNet: Extracting high quality monolingual datasets from web crawl data," in *Conference on Language Resources and Evaluation (LREC)*, 2020. [Online]. Available: https://aclanthology.org/2020.lrec-1.494/

[27] H. Qi, Y. Du, L. Zhang, S. C. Liew, K. Chen, and Y. Du, "Verirag: A retrieval-augmented framework for automated rtl testability repair," 2025. [Online]. Available: https://arxiv.org/abs/2507.15664

[28] H. Qi, C.-H. Sin, R. Y.-Y. Chan, and C. M. V. Wong, "Transforming applied behavior analysis therapy: An internet of things-guided, retrieval-augmented large language model framework," *IEEE Access*, vol. 13, pp. 149 679–149 694, 2025.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

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

[38] Meta AI, "Llama 3 technical report," https://ai.meta.com/llama/, 2024, accessed 2025-10-31.

[39] C. Xu, Q. Sun, H. Zhou *et al.*, "Wizardlm: Empowering large language models to follow complex instructions," *arXiv preprint arXiv:2304.12244*, 2023.

[40] W.-L. Chiang, L. Zheng *et al.*, "Vicuna: An open-source chatbot impressing gpt-4," https://lmsys.org/blog/2023-03-30-vicuna/, 2023, accessed 2025-10-31.

[41] N. Muennighoff, T. Wang *et al.*, "Crosslingual generalization through multitask finetuning," *arXiv preprint arXiv:2211.01786*, 2022.

[42] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[43] D. E. Blasi, A. Anastasopoulos, and G. Neubig, "Systematic inequalities in language technology performance across the world's languages," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

[44] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and P. Riley, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Lin-*

guistics, vol. 10, pp. 291–306, 2022.