MULTILINGUAL FINE-TUNING VIA LOCALIZED GRADIENT CONFLICT RESOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-lingual versatility has become a defining capability of Large Language Models (LLMs). However, fine-tuning these models often induces *negative interference*, where gains in one language come at another's expense. To address this, we cast multilingual fine-tuning as a multi-objective optimization (MOO) problem. Specifically, we introduce **Bucket-Level MOO**, a scalable distributed framework that applies gradient-based MOO algorithms locally, within the parameter buckets of distributed training. This enables conflict-aware updates without ever materializing a full gradient vector for any objective, and improves multilingual performance over standard fine-tuning by up to 4.3 points. Theoretically, we define *bucketwise Pareto stationarity*, show it is necessary for Pareto optimality under any fixed partition, and prove descent and convergence for representative Bucket-Level variants. Our work brings conventional MOO within reach for multilingual LLMs at a far lower cost. More broadly, Bucket-Level MOO is agnostic to both the objectives and the update rule, readily extending to other multi-task learning at scale.[1]

1 INTRODUCTION

The multilingual capabilities of Large Language Models (LLMs) have advanced rapidly, driven largely by data-centric strategies such as translating English corpora or distilling knowledge from high-resource languages (Muennighoff et al., 2023; Pan et al., 2024; Chen et al., 2024). While these approaches improve average cross-lingual performance, they implicitly assume that a single set of shared parameters can accommodate diverse and often incompatible linguistic patterns. In practice, this assumption frequently breaks down during fine-tuning: updates that benefit one language can adversely affect others, leading to *negative interference*, a phenomenon where optimizing for one language degrades performance on another (Conneau et al., 2020; Wang et al., 2020). As multilingual training scales to more languages and tasks, mitigating such interference becomes a central challenge for achieving robust and balanced cross-lingual performance.

From an optimization perspective, negative interference is fundamentally driven by conflicting gradients across language-specific objectives (Sener & Koltun, 2018; Yu et al., 2020; Shi et al., 2023). As illustrated in Figure 1 (Left), models with limited initial multilingual alignment (e.g., `Meta-Llama-3-8B`) exhibit severe, sustained gradient volatility throughout fine-tuning. Their pairwise cosine similarities frequently plunge into negative regimes with massive variance, severely constraining downstream progress. Conversely, stronger base multilingual models (e.g., `Qwen3-8B-Base`) are significantly more stable and maintain a largely positive average alignment. However, they still experience occasional, sharp gradient conflicts. These observations suggest the inherent instability of optimizing for diverse languages simultaneously.

A natural approach to mitigating such conflicts is to cast multilingual fine-tuning as a Multi-Objective Optimization (MOO) problem (Wang et al., 2021; Mao et al., 2022). In this framework, each language defines a separate objective, and the ultimate goal is to achieve *Pareto optimality*–a state where no language's performance can be improved without degrading another's. A practical MOO method therefore seeks an update that improves all language objectives when possible.

Conventional gradient-based MOO methods, such as MGDA (Sener & Koltun, 2018), CAGrad (Liu et al., 2021), and PCGrad (Yu et al., 2020), offer principled ways to mitigate conflicting

[1] Our code will be released publicly upon acceptance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

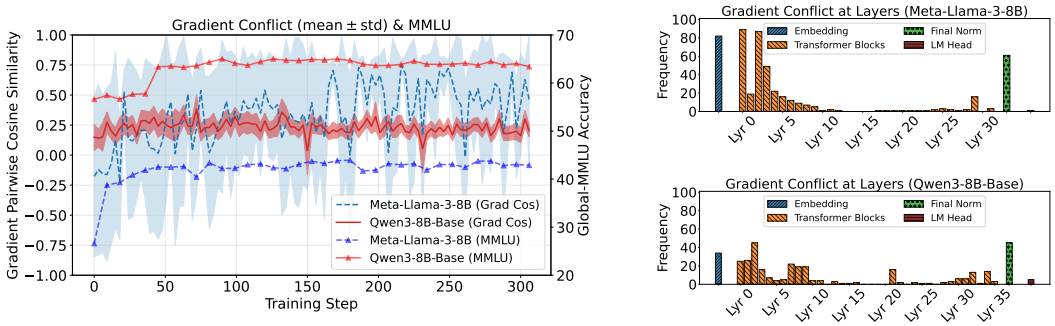

Figure 1: Gradient conflict during multilingual fine-tuning with two base models on eight languages (`en, zh, it, ar, ko, id, bn, sw`). **Left:** Mean ± standard deviation of pairwise gradient cosine similarity across all language pairs, with accuracy scores on Global-MMLU dataset. **Right:** Per-layer frequency of conflicts, defined as the fraction of training steps where at least one language pair exhibits a negative gradient cosine similarity.

objectives and find possible common descent directions. However, these methods require access to full gradient vectors across all objectives, incurring substantial memory overhead in distributed training settings. Beyond scalability, they also treat the model as a monolithic parameter space, implicitly assuming conflicts are globally uniform. In practice, linguistic processing within LLMs is structurally heterogeneous, e.g., various layers perform distinct roles (Zhao et al., 2024; Bandarkar et al., 2026). As shown in Figure 1 (Right), gradient conflicts are highly localized, often concentrating in specific layers (e.g., early or late layers). Resolving such conflicts where they arise is therefore a necessity rather than a preference at LLM scale: it matches the structure of the conflicts themselves and avoids assembling a full gradient for every language in order to act on them.

We therefore introduce *Bucket-Level MOO*, a scalable framework that turns the parameter partitions already used by distributed training into buckets of multi-objective optimization. Rather than introducing a new MOO solver, it provides a protocol that allows a broad class of MOO and multi-task learning methods to operate locally without reconstructing full-model objective gradients. This partitioned formulation also yields a natural first-order characterization: we introduce *bucketwise Pareto stationarity*, prove that it is necessary for Pareto optimality under any fixed parameter partitions, and establish descent and convergence guarantees for the MGDA and CAGrad variants.

Our main contributions are summarized as follows:

- *Practically,* we characterize the layer-wise heterogeneity of negative interference in LLMs, motivating localized conflict resolution as an efficient, effective alternative to the global one.
- *Methodologically,* we propose *Bucket-Level MOO*, a solver-agnostic framework that integrates into distributed training, enabling conflict resolution at any granularity while bounding the memory it adds by a single bucket.
- *Theoretically,* we introduce *bucketwise Pareto stationarity* for arbitrary fixed parameter partitions as a necessary condition for Pareto optimality. We further establish descent and convergence guarantees for the MGDA and CAGrad variants of Bucket-Level MOO.
- *Empirically*, we demonstrate through extensive experiments that Bucket-Level MOO significantly improves both in-distribution and out-of-distribution multilingual capabilities.

2 Methodology

2.1 Multilingual MOO and Localized Gradient Conflicts

Given target languages $\mathcal{T} := \{1, 2, \ldots, T\}$, we formulate multilingual fine-tuning of a model parameterized by $\boldsymbol{\theta}$ as the multi-objective problem

$$\min_{\boldsymbol{\theta}} \mathbf{L}(\boldsymbol{\theta}), \quad \text{where} \quad \mathbf{L}(\boldsymbol{\theta}) = [\mathcal{L}_1(\boldsymbol{\theta}), \ldots, \mathcal{L}_T(\boldsymbol{\theta})]^\top, \tag{1}$$

where $\mathcal{L}_t$ is the loss for language t on its dataset $\mathcal{D}_t$. Two language gradients conflict when $\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_i, \nabla_{\boldsymbol{\theta}} \mathcal{L}_j \rangle < 0$. Indeed, a first-order approximation of an update for language i gives

$$\Delta \mathcal{L}_j \approx -\eta \, \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_j, \nabla_{\boldsymbol{\theta}} \mathcal{L}_i \rangle \,, \tag{2}$$

so a conflicting update increases $\mathcal{L}_j$ and causes negative interference.

Conventional Gradient-Based MOO is therefore used to construct an update that explicitly accounts for the competing language objectives. **MGDA** (Fliege & Svaiter, 2000; Désidéri, 2012) finds a minimum-norm convex combination of objective gradients, **CAGrad** (Liu et al., 2021) optimizes worst-case improvement near their average, and **PCGrad** (Yu et al., 2020) projects away pairwise conflicts. Appendix A presents a detailed overview of these methods and the relevant background.

These methods presume that a full gradient per objective is readily available, an assumption that modern LLM training breaks in two ways: conflicts concentrate in a few layers rather than spreading uniformly across the model (Figure 1), and model states and gradients are sharded across devices. This leads to two practical weaknesses. *Memory Cost:* holding a full gradient per objective together with the solver workspace is prohibitively expensive at LLM scale. *Deferred reduction:* global MOO cannot reduce any part of the gradient until the full gradient of every objective is available, so each GPU must keep its entire unreduced gradient until the backward pass ends.

Distributed training (Rajbhandari et al., 2020; Zhao et al., 2023) already partitions parameters and gradients into fixed-size units, called *communication buckets*, so that a large model can be processed incrementally. We reuse this existing structure for conflict resolution and propose **Bucket-Level MOO**: as soon as the gradients of a bucket become available, the selected MOO rule is applied to that bucket and the resolved update is passed to the usual cross-device communication step before training moves to the next one (Figure 2). Conflicts are thus resolved at the granularity at which they arise, no complete language gradient is ever materialized, and the additional memory is bounded by the size of a single bucket, independent of the number of objectives T.

2.2 Bucket-Level Multi-Objective Optimization

Let $\mathcal{P} = \{\boldsymbol{\theta}_{(1)}, \ldots, \boldsymbol{\theta}_{(K)}\}$ be a fixed partition of the model parameters into K disjoint buckets. Bucket-Level MOO applies a conflict-resolution rule independently within each bucket, so that the granularity of optimization matches both the locality of the conflicts and the communication structure of distributed training.

For clarity, suppose T GPUs each process a micro-batch from a distinct language, so that GPU t computes $\mathcal{L}_t$. Backpropagation makes $\nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$ available on GPU t as soon as bucket k is complete, and Bucket-Level MOO exchanges only these bucket-local gradients:

$$\mathbf{G}_{(k)} = \left\{ \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_1, \ldots, \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_T \right\}.$$

The selected MOO rule maps $\mathbf{G}_{(k)}$ to a single update for bucket k, which is immediately reduced and distributed across the GPUs before training proceeds to the next bucket. A complete model gradient is therefore never formed for any language.

We instantiate the per-bucket rule with the three solvers introduced in Section 2.1. The equations below make explicit that each acts only on the current bucket.

Bucket-Level MGDA. For each bucket k we solve the local quadratic program

$$\min_{\boldsymbol{\alpha}^{(k)} \in \Delta} \left\| \sum_{t=1}^{T} \alpha_t^{(k)} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t \right\|_2^2, \qquad \Delta = \left\{ \boldsymbol{\alpha} \geq \mathbf{0} : \sum_{t=1}^{T} \alpha_t = 1 \right\}. \tag{3}$$

Its solution defines the update $\mathbf{d}_{(k)}^{\text{MGDA}} = \sum_{t=1}^{T} \alpha_t^{(k)} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$, the minimum-norm element of the convex hull of the bucket gradients. The weights $\boldsymbol{\alpha}^{(k)}$ are free to differ across buckets.

Bucket-Level CAGrad. Writing $\mathbf{g}_{(k)}^{avg} = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$ for the local average gradient, we solve the CAGrad subproblem on bucket k:

$$\mathbf{d}_{(k)}^{\text{CAGrad}} = \arg\max_{\mathbf{d}_{(k)}} \min_{t} \langle \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t, \mathbf{d}_{(k)} \rangle \quad \text{s.t.} \quad \|\mathbf{d}_{(k)} - \mathbf{g}_{(k)}^{avg}\| \leq c \|\mathbf{g}_{(k)}^{avg}\|. \tag{4}$$

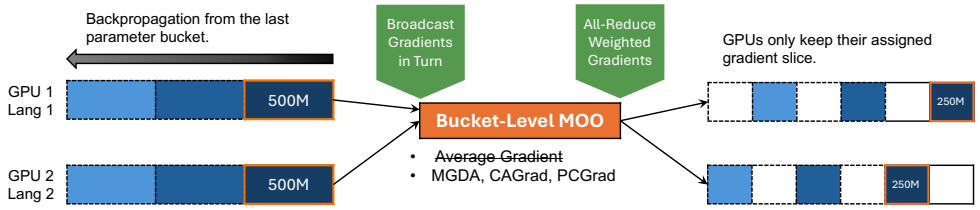

Figure 2: Bucket-Level MOO for two languages. Backpropagation produces gradients one fixed-size parameter bucket at a time. For each bucket, the GPUs broadcast their language gradients in turn through one reusable buffer, an MOO solver turns their inner products into combination weights, and the weighted gradients are all-reduced, with each GPU keeping only its assigned slice.

The update therefore maximizes the worst-case alignment with the bucket gradients while staying within a ball around the local average, whose radius is set by $c \in [0, 1)$.

Bucket-Level PCGrad. Writing $\mathbf{g}_t^{(k)} = \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$, each conflicting pair is deconflicted using only inner products computed within the bucket:

$$\langle \mathbf{g}_i^{(k)}, \mathbf{g}_j^{(k)} \rangle < 0 \quad \Longrightarrow \quad \mathbf{g}_i^{(k)} \leftarrow \mathbf{g}_i^{(k)} - \frac{\langle \mathbf{g}_i^{(k)}, \mathbf{g}_j^{(k)} \rangle}{\|\mathbf{g}_j^{(k)}\|_2^2} \mathbf{g}_j^{(k)}. \tag{5}$$

After all conflicting pairs are projected, the bucket update averages the deconflicted gradients, $\mathbf{d}_{(k)}^{\text{PCGrad}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{g}_t^{(k)}$, which keeps its scale comparable to the MGDA and CAGrad updates.

In every case, the bucket updates concatenate into $\mathbf{d} = [\mathbf{d}_{(1)}, \ldots, \mathbf{d}_{(K)}]$. Figure 2 traces one bucket from gradient arrival to reduced update, and Appendix D gives pseudocode.

2.3 Bucketwise Pareto Stationarity and Convergence

Having defined the bucket-level variants, we next ask what condition should hold when no further update is produced. The ultimate goal remains Pareto optimality, where no language can improve without degrading another. However, resolving conflicts bucket by bucket changes the first-order condition that characterizes a fixed point. We say that $\boldsymbol{\theta}^*$ is *bucketwise Pareto stationary* (BPS) if

$$\mathbf{0} \in \prod_{k=1}^{K} \operatorname{conv} \left\{ \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_1(\boldsymbol{\theta}^*), \ldots, \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_T(\boldsymbol{\theta}^*) \right\}. \tag{6}$$

Equivalently, the language gradients within each bucket admit a vanishing convex combination, $\sum_{t=1}^{T} \alpha_t^{(k)} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t(\boldsymbol{\theta}^*) = \mathbf{0}$ for some $\boldsymbol{\alpha}^{(k)} \in \Delta$. Under any fixed grouping, every Pareto-optimal solution is BPS (Theorem A.4); Bucket-Level MOO therefore never discards a solution that the conventional one would retain. The next two results show that the MGDA and CAGrad variants converge to such points.

Theorem 2.1. *Let $\boldsymbol{\alpha}^{(k)} \in \Delta$ solve Equation 3, let $\mathbf{d}_{(k)}^{\text{MGDA}} = \sum_{t=1}^{T} \alpha_t^{(k)} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$, and concatenate $\mathbf{d}^{\text{MGDA}} = [\mathbf{d}_{(1)}^{\text{MGDA}}, \ldots, \mathbf{d}_{(K)}^{\text{MGDA}}]$. Then:*

- *(a) $\mathbf{d}^{\text{MGDA}} = \mathbf{0}$ if and only if the current parameters are BPS with respect to $\mathcal{P}$.*
- *(b) If $\mathbf{d}^{\text{MGDA}} \neq \mathbf{0}$, then $-\mathbf{d}^{\text{MGDA}}$ is a common descent direction and $\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_t, \mathbf{d}^{\text{MGDA}} \rangle \geq \|\mathbf{d}^{\text{MGDA}}\|_2^2 > 0$ for every t.*
- *(c) Suppose each $\mathcal{L}_t$ is lower bounded and has an L_t-Lipschitz gradient. For exact updates $\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n - \eta \mathbf{d}^n$ with $0 < \eta < 2/L$, where $L = \max_t L_t$, we have $\sum_{n=0}^{\infty} \|\mathbf{d}^n\|_2^2 < \infty$ and hence $\|\mathbf{d}^n\|_2 \to 0$. Every accumulation point is BPS.*

BPS is therefore the exact convergence certificate of Bucket-Level MGDA: the update vanishes only at BPS points, and every nonzero update decreases all language losses simultaneously.

Theorem 2.2. *Let* $\mathbf{d}_{(k)}^{\mathrm{CAGrad}}$ *solve the local CAGrad problem with* $0 \le c < 1$*, local average gradient* $\mathbf{g}_{(k)}^{avg}$*, and global concatenation* $\mathbf{d}^{\mathrm{CAGrad}}$*. Then:*

(a) If $\nabla \mathcal{L}_{avg} \neq \mathbf{0}$*, then* $-\mathbf{d}^{\mathrm{CAGrad}}$ *is a descent direction for* $\mathcal{L}_{avg}$*, with* $\langle \nabla \mathcal{L}_{avg}, \mathbf{d}^{\mathrm{CAGrad}} \rangle \ge \frac{1-c^2}{2} \|\nabla \mathcal{L}_{avg}\|_2^2 > 0$.

(b) $\mathbf{d}^{\mathrm{CAGrad}} = \mathbf{0}$ *if and only if* $\nabla \mathcal{L}_{avg} = \mathbf{0}$*. Thus a fixed point is standard Pareto stationary with the single uniform global weight* $\alpha_t = 1/T$*, and is consequently also BPS.*

(c) If $\mathcal{L}_{avg}$ *is lower bounded and has an* L_F*-Lipschitz gradient, then exact updates with* $0 < \eta \le 1/L_F$ *satisfy* $\sum_n \|\nabla \mathcal{L}_{avg}(\boldsymbol{\theta}^n)\|_2^2 < \infty$*; thus* $\|\nabla \mathcal{L}_{avg}(\boldsymbol{\theta}^n)\|_2 \to 0$ *and* $\|\mathbf{d}^n\|_2 \to 0$.

Bucket-Level CAGrad provides a guarantee of a different kind: the updates decrease the average language loss and drive its gradient to zero. The points at which the update vanishes are again BPS.

3 Experiments

3.1 Experimental Setup

Models and Baselines. Four base LLMs are evaluated: `Meta-Llama-3-8B`, `Llama-3.1-8B`, `Qwen3-4B-Base`, and `Qwen3-8B-Base`. We compare the pretrained base model, Vanilla SFT, Vanilla SFT with a language-balanced data loader, and the three Bucket-Level MOO variants.

LIMA–s1 Training Data and Evaluation. We construct an English corpus comprising 1,000 LIMA conversational pairs (Zhou et al., 2023) and 630 s1k-1.1 reasoning samples (Muennighoff et al., 2025) distilled from DeepSeek-R1.[2] We use `Gemini 2.0 Flash` to translate this corpus into eight languages (`en, zh, it, ar, ko, id, bn, sw`) spanning the high- to low-resource spectrum. We evaluate three complementary competencies:

- **Natural language understanding:** BELEBELE (Bandarkar et al., 2024) measures parallel reading comprehension.
- **Technical reasoning:** Multilingual ARC-E (Han et al., 2025) and the low-medium subsets of PolyMath (Wang et al., 2025) assess scientific and mathematical reasoning.
- **Knowledge-based question answering:** Global-MMLU-Lite (Singh et al., 2024) measures world-knowledge retention and transfer.

For a consistent comparison across tasks, we use the 13 languages shared by all four benchmarks. We report results separately for the eight *seen* training languages and five *unseen* languages (`de, es, fr, ja, pt`), the latter measuring zero-shot cross-lingual generalization.

WMT24++ Training Data and Evaluation. For multilingual machine translation, we use the WMT24++ dataset (Deutsch et al., 2025) and jointly fine-tune each backbone on four languages: Vietnamese (`vi`), Indonesian (`id`), Thai (`th`), and Tamil (`ta`). We evaluate the fine-tuned models on the corresponding FLORES-200 subsets included in SEA-HELM (Susanto et al., 2025). Following SEA-HELM, we measure translation quality using MetricX-wmt24, normalized to the 0–100 scale, where higher values indicate better translation quality (Juraska et al., 2024).

Fine-tuning details and computational costs are provided in Appendices E and F, respectively.

3.2 Multilingual Instruction Fine-Tuning on LIMA–s1

Seen-Language Performance. Table 1 shows that every Bucket-Level MOO variant improves the average seen-language performance over both Vanilla SFT and its balanced-loader control across all four base models. Across all 12 model–method pairs, the gains over Vanilla SFT range from $+1.6$ to $+2.9$ points; against the balanced-loader control, which is the stronger of the two baselines

[2] We filter s1k-1.1 to include only samples with their short CoT traces marked as correctly.

Table 1: Main results (higher is better) after fine-tuning on the multilingual LIMA–s1 dataset. Per-language results in Appendix G show consistent gains from Bucket-Level MOO across languages.

Method	*8 Seen Languages*					*5 Unseen Languages*				
	BELE	**ARC-E**	**PolyMath**	**MMLU**	**Avg.**	**BELE**	**ARC-E**	**PolyMath**	**MMLU**	**Avg.**
Meta-Llama-3-8B										
Base	51.8	54.2	12.0	37.2	38.8	55.9	65.2	13.5	42.6	44.3
V. SFT	53.5	61.5	18.3	42.5	44.0	60.7	71.5	21.5	48.9	50.7
+ *bal-loader*	54.9	62.4	19.2	44.6	45.3	60.2	72.3	22.0	50.5	51.3
MGDA	56.4	63.7	21.6	45.5	46.8	63.4	72.7	24.3	50.6	52.8
CAGrad	56.6	63.9	20.2	43.9	46.2	62.4	72.8	23.8	50.5	52.4
PCGrad	56.8	64.3	21.4	45.2	46.9	63.7	73.6	24.5	49.0	52.7
Llama-3.1-8B										
Base	59.1	60.0	13.2	41.8	43.5	67.8	70.0	13.8	48.9	50.1
V. SFT	53.1	61.3	20.3	44.2	44.7	61.3	72.6	22.8	50.5	51.8
+ *bal-loader*	56.6	63.1	20.5	44.1	46.1	62.1	73.5	22.7	52.3	52.6
MGDA	57.1	63.9	22.4	45.2	47.2	65.0	74.1	25.8	51.7	54.2
CAGrad	57.2	63.7	22.9	45.6	47.3	63.9	74.4	26.6	52.1	54.3
PCGrad	57.0	63.5	22.4	44.8	46.9	63.2	74.4	26.1	52.6	54.1
Qwen3-4B-Base										
Base	75.5	81.7	39.4	56.9	63.4	83.8	93.3	44.0	63.6	71.2
V. SFT	69.8	78.9	43.4	55.9	62.0	76.0	88.8	45.8	62.3	68.2
+ *bal-loader*	70.3	77.8	43.5	57.2	62.2	75.5	88.1	45.2	63.7	68.1
MGDA	72.2	80.1	44.4	57.7	63.6	79.3	90.5	47.4	66.0	70.8
CAGrad	72.6	80.7	44.9	58.0	64.0	79.4	90.6	47.2	65.5	70.7
PCGrad	73.0	80.1	44.6	58.4	64.0	79.4	90.6	46.1	64.0	70.0
Qwen3-8B-Base										
Base	81.8	85.7	44.5	61.7	68.4	85.0	95.1	47.6	70.0	74.4
V. SFT	75.3	84.8	46.8	63.4	67.6	80.6	93.4	49.0	69.1	73.0
+ *bal-loader*	76.6	84.6	47.1	64.1	68.1	82.2	93.0	48.6	69.5	73.4
MGDA	77.9	85.3	50.0	64.2	69.3	83.9	94.0	50.5	70.8	74.8
CAGrad	78.4	85.2	49.0	65.2	69.5	83.6	94.0	49.8	71.9	74.8
PCGrad	78.2	85.4	48.4	64.6	69.2	83.8	94.0	49.1	70.3	74.3

in this setting, MOO consistently adds $+0.8$ to $+1.8$ points. Thus, balanced sampling alone does not account for the improvements. The benefits also extend across capabilities: compared with the balanced-loader baseline, MGDA improves PolyMath by $+2.9$ on `Qwen3-8B-Base`, while PCGrad improves BELEBELE by $+2.7$ on `Qwen3-4B-Base`.

Zero-Shot Transfer to Unseen Languages. Although the balanced loader changes the Vanilla-SFT averages by only -0.1 to $+0.8$ points, all Bucket-Level MOO variants outperform this control, with gains ranging from $+0.9$ to $+2.7$. The largest improvement is obtained by MGDA on `Qwen3-4B-Base`, increasing the unseen-language average from 68.1 to 70.8. These gains span both knowledge and reasoning: MGDA improves MMLU by $+2.3$ on `Qwen3-4B-Base`, while CAGrad improves PolyMath by $+3.9$ on `Llama-3.1-8B`. Conflict-aware optimization therefore promotes transfer beyond the languages observed during fine-tuning.

Mitigating Catastrophic Forgetting. The results further reveal that Vanilla SFT can trade reasoning gains for substantial losses in pretrained reading comprehension. For example, the seen-language BELEBELE score of `Llama-3.1-8B` falls from 59.1 to 53.1 after Vanilla SFT. Balanced loading partially restores it to 56.6, whereas CAGrad further raises it to 57.2. Similarly, on unseen BELEBELE, `Qwen3-8B-Base` drops from 85.0 to 80.6; balanced loading recovers 82.2, and MGDA reaches 83.9. This preservation is also visible in the aggregate: the seen-language average of `Qwen3-8B-Base` decreases from 68.4 to 67.6 with Vanilla SFT, recovers to 68.1 with balanced loading, and reaches 69.5 with CAGrad. Recovery is not always complete: on unseen languages, `Qwen3-4B-Base` reaches 70.8 with MGDA against a pretrained average of 71.2. Bucket-Level

Table 2: Results on the SEA-HELM FLORES-200 benchmark (higher is better).

Method	*X → English*					*English → X*				
	VI	**ID**	**TH**	**TA**	**Avg.**	**VI**	**ID**	**TH**	**TA**	**Avg.**
Meta-Llama-3-8B										
Base	64.5	42.1	40.5	29.0	44.0	37.5	68.0	52.3	27.4	46.3
V. SFT	85.3	85.9	85.7	72.9	82.4	88.1	88.4	81.5	68.5	81.6
+ *bal-loader*	85.4	84.9	84.8	72.9	82.0	88.2	88.5	81.9	69.0	81.9
MGDA	88.5	89.1	88.4	77.2	85.8	88.3	88.5	82.6	70.7	82.5
CAGrad	88.2	88.7	88.0	76.0	85.2	88.4	88.8	82.7	68.9	82.2
PCGrad	88.3	88.4	87.9	75.8	85.1	89.1	89.4	82.3	68.9	82.4
Llama-3.1-8B										
Base	80.7	37.2	39.1	25.0	45.5	39.8	72.3	52.6	31.1	48.9
V. SFT	85.9	85.7	85.1	74.2	82.7	88.8	89.1	81.1	68.5	81.9
+ *bal-loader*	85.6	85.5	84.8	73.7	82.4	88.4	88.8	80.7	68.6	81.6
MGDA	88.3	89.2	88.2	77.7	85.8	88.8	89.6	81.4	71.0	82.7
CAGrad	88.7	88.9	88.3	76.8	85.6	89.3	89.7	81.6	71.0	82.9
PCGrad	88.3	88.5	87.9	76.9	85.4	89.6	89.8	81.8	69.1	82.6
Qwen3-4B-Base										
Base	89.8	90.3	70.7	75.4	81.5	85.2	83.1	80.8	49.5	74.7
V. SFT	84.6	84.8	83.9	66.4	79.9	86.4	85.7	79.2	62.4	78.4
+ *bal-loader*	84.6	85.1	83.0	67.5	80.1	86.3	85.1	79.5	63.0	78.5
MGDA	88.3	88.5	87.4	72.6	84.2	87.4	86.2	80.7	61.7	79.0
CAGrad	87.8	88.2	87.4	70.7	83.5	87.6	86.4	81.6	62.6	79.5
PCGrad	87.7	88.2	86.8	71.3	83.5	88.2	87.3	81.2	63.4	80.0
Qwen3-8B-Base										
Base	90.5	90.7	85.1	75.2	85.4	88.9	88.3	85.6	68.6	82.9
V. SFT	86.7	86.8	86.7	73.2	83.4	89.2	88.8	83.2	69.8	82.8
+ *bal-loader*	86.4	86.6	86.8	72.4	83.1	89.0	88.9	83.2	70.1	82.8
MGDA	89.4	89.9	89.7	78.0	86.7	90.1	90.0	84.5	72.0	84.1
CAGrad	88.8	89.6	89.6	77.0	86.3	90.1	90.1	84.7	71.3	84.1
PCGrad	88.9	89.7	89.0	77.1	86.2	90.4	90.4	84.7	71.4	84.2

MOO therefore mitigates destructive overwriting more effectively than balanced sampling alone, while retaining the benefits of multilingual adaptation.

3.3 Multilingual Machine Translation on WMT24++

Translation Quality. Table 2 shows that Bucket-Level MOO consistently improves multilingual translation over Vanilla SFT across all four backbones and both translation directions. The improvement is especially clear for *X→English*, where the best MOO variant gains $+3.1$ to $+4.3$ average points over Vanilla SFT. The corresponding best-variant gains for *English→X* are smaller but still consistent, ranging from $+0.9$ to $+1.6$ points, and every variant improves over Vanilla SFT in both directions. In contrast, the balanced loader performs nearly identically to Vanilla SFT.

Preserving Pretrained Translation Ability. Vanilla SFT can degrade the strong pretrained translation ability of Qwen3 models. For instance, the *X→English* average drops from 81.5 to 79.9 on `Qwen3-4B-Base` and from 85.4 to 83.4 on `Qwen3-8B-Base`. Bucket-Level MOO reverses this trend, with MGDA improving the corresponding scores to 84.2 and 86.7. These results indicate that conflict-aware optimization not only improves weaker backbones after SFT, but also better preserves the multilingual capabilities already present in stronger pretrained models.

4 Ablations: Partition Schemes and Bucket Sizes

Using models fine-tuned on LIMA–s1, we study how the granularity of conflict resolution affects quality and cost. Global (conventional) MOO is Bucket-Level MOO on the trivial partition $\mathcal{P} = \{\boldsymbol{\theta}\}$.

Table 3: Effects of partition schemes and bucket sizes on LIMA–s1 performance.

Method	*8 Seen Languages*						*5 Unseen Languages*					
	Layer-Aligned	**100M**	**250M**	**500M**	**1B**	**Global**	**Layer-Aligned**	**100M**	**250M**	**500M**	**1B**	**Global**
						Qwen3-4B-Base						
MGDA	63.8	63.9	63.3	63.6	64.1	63.3	71.5	71.1	71.0	70.8	70.9	70.2
CAGrad	63.9	63.3	63.7	64.0	63.1	63.1	71.2	70.5	71.2	70.7	70.6	70.0
PCGrad	63.6	63.2	63.8	64.0	63.8	63.6	71.0	70.3	71.2	70.0	70.7	70.8
						Qwen3-8B-Base						
MGDA	68.9	69.5	70.0	69.3	69.2	69.2	74.9	75.0	75.3	74.8	75.3	75.0
CAGrad	69.3	69.5	69.2	69.5	69.2	69.4	74.9	75.3	74.9	74.8	74.8	75.2
PCGrad	69.2	69.8	69.6	69.2	69.7	69.3	75.1	75.4	75.3	74.3	74.5	75.1

Table 4: Peak VRAM (GB) of training under different partition schemes and bucket sizes. All three Bucket-Level MOO variants share an identical footprint.

Model	Bucket-Level MOO					Global MOO	V. SFT
	100M	**250M**	**500M**	**1B**	**Layer-Aligned**		
`Qwen3-4B-Base`	34.9	34.5	39.1	39.3	36.1	55.2	36.3
`Qwen3-8B-Base`	60.7	60.9	61.3	61.7	61.4	85.9	60.5

In our implementation, Global MOO therefore inherits the memory-efficient single-buffer exchange of Bucket-Level MOO instead of naively gathering all T gradients (Appendix D).

Partition Schemes. On `Qwen3-4B-Base` and `Qwen3-8B-Base`, we compare layer-aligned partitions, system-oriented fixed-size buckets containing 100M–1B gradient elements, and a global partition. Layer-aligned partitions follow architectural boundaries, whereas fixed-size buckets reflect the contiguous gradient buffers used by distributed training systems. We adopt 500M as the default, matching DeepSpeed's default reduction bucket size, and do not tune it further. Smaller buckets reduce temporary memory but introduce more collective operations; larger buckets amortize per-collective overhead at a higher memory cost.

Insensitive Quality to Granularity. Table 3 varies the partition over an order of magnitude, from single decoder blocks to the entire model, and the outcome barely moves. Across all methods and both language distributions, performance spans at most 1.3 points on `Qwen3-4B-Base` (median 1.1) and 1.1 points on `Qwen3-8B-Base` (median 0.6). Layer-aligned partitions fall inside the same range as the fixed-size buckets at both scales, so localized resolution needs no alignment with Transformer-layer boundaries and can run directly on the contiguous gradient buffers distributed training already produces. Crucially, this variation is smaller than the effect being measured: all 72 configurations in the table exceed the balanced-loader baseline of Table 1, by at least 0.8 points. Which partition one picks therefore never decides whether Bucket-Level MOO helps.

Granularity versus Global MOO. The invariance extends to the trivial partition. Averaged over the three solvers and the five partitioned settings, partitioned resolution leads on both splits of `Qwen3-4B-Base`, by 0.3 points on the seen languages and 0.5 on the unseen; on `Qwen3-8B-Base` the two become indistinguishable, at 0.1 and -0.1. Conflict resolution is therefore granularity-invariant: confining it to buckets matches a full-model decision at both scales, and the match tightens as the backbone grows.

Memory Efficiency. By maintaining a single bucket rather than full-model gradients per objective, Bucket-Level MOO drastically curtails memory overhead while keeping peak consumption virtually invariant to bucket size (Table 4). Across a tenfold range of bucket sizes and layer-aligned partitions, peak VRAM remains stable at 34.5–39.3 GB on `Qwen3-4B-Base` and 60.7–61.7 GB on `Qwen3-8B-Base`. Relative to Global MOO, the 500M default cuts peak VRAM by 29.2% on 4B (39.1 vs. 55.2 GB) and by 28.6% on 8B (61.3 vs. 85.9 GB), with all three solvers sharing the exact

same footprint. Crucially, Bucket-Level MOO achieves near-parity with Vanilla SFT (36.3 GB on 4B, 60.5 GB on 8B). The small gap is cached memory for the transient bucket buffers, not live tensors.

5 Related Work

Multilingual LLMs and Negative Interference. Modern LLMs project diverse languages into a shared parameter space (Chen et al., 2025; Yang et al., 2025), where these languages occupy distinct representational regions (Chang et al., 2022; Xu & Zhang, 2026). Structurally, models often exhibit a division of labor: early and late layers handle multilingual specializations, while middle layers serve as language-universal mechanisms (Zhao et al., 2024; Bandarkar et al., 2026). While targeted activation interventions improve reasoning (Zhao et al., 2026), cross-lingual generalization is predominantly driven by data-centric strategies like translating instruction corpora (Pan et al., 2024; Chen et al., 2024), self-distillation (Zhang et al., 2024), and multi-task tuning (Muennighoff et al., 2023). However, forcing disparate linguistic distributions into a static parameter space inherently triggers *negative interference*, where optimizing for one language disproportionately degrades others (Conneau et al., 2020; Wang et al., 2020).

Gradient Conflicts and Scalable Optimization in Distributed LLMs. From an optimization perspective, negative interference stems from conflicting gradients across language-specific objectives, which vary across the model: GradVac (Wang et al., 2021) manipulates them per layer and per weight matrix, and Recon (Shi et al., 2023) makes the most conflicting layers task-specific. Loss-weighting schemes (Kendall et al., 2018; Chen et al., 2018) reweight objectives on the fly; others modify the architecture (Zhang et al., 2025) or stage the fine-tuning (Ye et al., 2025). Gradient-based MOO acts on the gradients themselves: MGDA (Désidéri, 2012; Sener & Koltun, 2018), PCGrad (Yu et al., 2020), GradDrop (Chen et al., 2020), CAGrad (Liu et al., 2021), and Nash-MTL (Navon et al., 2022), with multilingual uses that curb forgetting (Mao et al., 2022) or filter preference data (Li et al., 2026), while RP-MGDA (Hu & Yu, 2025) refines Pareto stationarity over sparsity-derived partitions. At LLM scale, however, FSDP (Zhao et al., 2023) and DeepSpeed ZeRO (Rajbhandari et al., 2020) shard gradients, so conventional MOO must hold each full unreduced objective gradient until the update is formed. Bucket-Level MOO instead applies such methods to the gradient buckets the system already creates, with a matching theoretical analysis of the resulting updates.

6 Conclusion

We introduced Bucket-Level MOO, a protocol that resolves gradient conflicts within the parameter partitions distributed training already creates; its novelty lies not in a new MOO solver, but in the granularity at which existing ones are applied. Theoretically, we formalize *bucketwise Pareto stationarity*, prove its necessity for Pareto optimality under arbitrary partitions, and establish descent and convergence guarantees for representative solvers. Empirically, the three-variant average exceeds both baselines in all 16 backbone–split cells, gaining $+0.8$ to $+3.8$ points over Vanilla SFT while mitigating forgetting of pretrained capabilities. Conflict resolution also proves granularity-invariant, with quality varying by at most 1.3 points across partitions and scales. Bucketing thus matches Global (conventional) MOO quality while reducing peak memory by 29% (16.1 GB on 4B and 24.6 GB on 8B) with zero live allocated memory overhead over Vanilla SFT.

Several limitations bound this reading. Our guarantees cover exact deterministic updates for the MGDA and CAGrad variants, leaving PCGrad and the stochastic regime used in practice outside the analysis. We also instantiate the framework with only three solvers; others, such as GradDrop (Chen et al., 2020) and Nash-MTL (Navon et al., 2022), should be explored. Furthermore, all experiments fine-tune pre-trained models, so the behavior of Bucket-Level MOO when training multilingually from scratch remains unexamined. Our objectives are languages throughout; extending the framework to other multi-task learning settings, such as multi-domain, multi-modal, or preference-based post-training, remains open. Nevertheless, we remark that these qualifications bound the scope of our analysis rather than the protocol itself. Finally, we hope that treating conflict-resolution granularity as both an optimization and a systems choice, rather than an implementation detail, proves useful wherever objectives are trained jointly at scale.

AI USE STATEMENT

We used generative AI tools for language editing, LaTeX formatting, and assistance with code and experimental analysis. We have reviewed all AI-assisted text, claims, and artifacts. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our code will be released publicly upon acceptance. Training data sources, evaluation benchmarks, and implementation details are described in Section 2 and Section 3, with additional definitions, proofs, and experimental details in the appendix.

REFERENCES

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 749–775, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.44. URL https://aclanthology.org/2024.acl-long.44/.

Lucas Bandarkar, Chenyuan Yang, Mohsen Fayyaz, Junlin Hu, and Nanyun Peng. Multilingual routing in mixture-of-experts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZoZR0x7tTD.

Tyler A. Chang, Zhuowen Tu, and Benjamin K. Bergen. The geometry of multilingual language model representations. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 119–136, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.9. URL https://aclanthology.org/2022.emnlp-main.9/.

Pinzhen Chen, Shaoxiong Ji, Nikolay Bogoychev, Andrey Kutuzov, Barry Haddow, and Kenneth Heafield. Monolingual or multilingual instruction tuning: Which makes a better alpaca. In Yvette Graham and Matthew Purver (eds.), *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1347–1356, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-eacl.90/.

Yuxin Chen, Yiran Zhao, Yang Zhang, An Zhang, Kenji Kawaguchi, Shafiq Joty, Junnan Li, Tat-Seng Chua, Michael Qizhe Shieh, and Wenxuan Zhang. The emergence of abstract thought in large language models beyond any language. In *Advances in Neural Information Processing Systems 38*, 2025. URL https://openreview.net/forum?id=ZE9cxnEBpy.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747/.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Daniel Deutsch, Eleftheria Briakou, Isaac Rayburn Caswell, Mara Finkelstein, Rebecca Galor, Juraj Juraska, Geza Kovacs, Alison Lui, Ricardo Rei, Jason Riesa, Shruti Rijhwani, Parker Riley, Elizabeth Salesky, Firas Trabelsi, Stephanie Winkler, Biao Zhang, and Markus Freitag. WMT24++: Expanding the language coverage of WMT24 to 55 languages & dialects. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12257–12284, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.634. URL https://aclanthology.org/2025.findings-acl.634/.

Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 51(3):479–494, 2000.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Wenhan Han, Yifan Zhang, Zhixun Chen, Binbin Liu, Haobin Lin, Bingni Zhang, Taifeng Wang, Mykola Pechenizkiy, Meng Fang, and Yin Zheng. Mubench: Assessment of multilingual capabilities of large language models across 61 languages, 2025. URL https://arxiv.org/abs/2506.19468.

Zeou Hu and Yaoliang Yu. Leveraging variable sparsity to refine pareto stationarity in multi-objective optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Bl3e8HV9xW.

Juraj Juraska, Daniel Deutsch, Mara Finkelstein, and Markus Freitag. MetricX-24: The Google submission to the WMT 2024 metrics shared task. In Barry Haddow, Tom Kocmi, Philipp Koehn, and Christof Monz (eds.), *Proceedings of the Ninth Conference on Machine Translation*, pp. 492–504, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.wmt-1.35. URL https://aclanthology.org/2024.wmt-1.35/.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Jiangnan Li, Thuy-Trang Vu, Christian Herold, Amirhossein Tebbifakhr, Shahram Khadivi, and Gholamreza Haffari. CONGRAD: Conflicting gradient filtering for multilingual preference alignment. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6371–6387, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.299. URL https://aclanthology.org/2026.eacl-long.299/.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 18878–18890. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9d27fdf2477ffbff837d73ef7ae23db9-Paper.pdf.

Yuren Mao, Yaobo Liang, Nan Duan, Haobo Wang, Kai Wang, Lu Chen, and Yunjun Gao. Less-forgetting multi-lingual fine-tuning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=7vmyjUHgm9_.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. Crosslingual generalization through multitask finetuning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15991–16111, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.891. URL https://aclanthology.org/2023.acl-long.891/.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

Xingyuan Pan, Luyang Huang, Liyan Kang, Zhicheng Liu, Yu Lu, and Shanbo Cheng. G-DIG: Towards gradient-based DIverse and hiGh-quality instruction data selection for machine translation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15395–15406, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.821. URL https://aclanthology.org/2024.acl-long.821/.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 525–536. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization.pdf.

Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. Recon: Reducing conflicting gradients from the root for multi-task learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ivwZO-HnzG_.

Shivalika Singh, Angelika Romanou, Clémentine Fourrier, David I. Adelani, Jian Gang Ngui, Daniel Vila-Suero, Peerat Limkonchotiwat, Kelly Marchisio, Wei Qi Leong, Yosephine Susanto, Raymond Ng, Shayne Longpre, Wei-Yin Ko, Madeline Smith, Antoine Bosselut, Alice Oh, Andre F. T. Martins, Leshem Choshen, Daphne Ippolito, Enzo Ferrante, Marzieh Fadaee, Beyza Ermis, and Sara Hooker. Global mmlu: Understanding and addressing cultural and linguistic biases in multilingual evaluation, 2024. URL https://arxiv.org/abs/2412.03304.

Yosephine Susanto, Adithya Venkatadri Hulagadri, Jann Railey Montalan, Jian Gang Ngui, Xianbin Yong, Wei Qi Leong, Hamsawardhini Rengarajan, Peerat Limkonchotiwat, Yifan Mai, and William Chandra Tjhi. SEA-HELM: Southeast Asian holistic evaluation of language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12308–12336, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.636. URL https://aclanthology.org/2025.findings-acl.636/.

Yiming Wang, Pei Zhang, Jialong Tang, Hao-Ran Wei, Baosong Yang, Rui Wang, Chenshu Sun, Feitong Sun, Jiran Zhang, Junxuan Wu, Qiqian Cang, Yichang Zhang, Fei Huang, Junyang Lin, Fei Huang, and Jingren Zhou. Polymath: Evaluating mathematical reasoning in multilingual contexts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=B1vCImy6yI.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. On negative interference in multilingual models: Findings and a meta-learning treatment. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang

Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4438–4450, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.359. URL https://aclanthology.org/2020.emnlp-main.359/.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=F1vEjWK-lH_.

Shaoyang Xu and Wenxuan Zhang. Language of thought shapes output diversity in large language models, 2026. URL https://arxiv.org/abs/2601.11227.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hua Ye, Siyuan Chen, Haoliang Zhang, Weihao Luo, Yanbin Li, and Xuan Zhang. Synergy over discrepancy: A partition-based approach to multi-domain LLM fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2jzGEudVdS.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Yufeng Chen, Jinan Xu, and Jie Zhou. Less, but better: Efficient multilingual expansion for LLMs via layer-wise mixture-of-experts. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17948–17963, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.878. URL https://aclanthology.org/2025.acl-long.878/.

Yuanchi Zhang, Yile Wang, Zijun Liu, Shuo Wang, Xiaolong Wang, Peng Li, Maosong Sun, and Yang Liu. Enhancing multilingual capabilities of large language models through self-distillation from resource-rich languages. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11189–11204, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.603. URL https://aclanthology.org/2024.acl-long.603/.

Weixiang Zhao, Jiahe Guo, Yang Deng, Tongtong Wu, Wenxuan Zhang, Yulin Hu, Xingyu Sui, Yanyan Zhao, Wanxiang Che, Bing Qin, Tat-Seng Chua, and Ting Liu. When less language is more: Language-reasoning disentanglement makes LLMs better multilingual reasoners. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=fleQlZ2VTx.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. How do large language models handle multilingualism? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 15296–15319. Curran Associates, Inc., 2024. doi: 10.52202/079017-0489. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1bd359b32ab8b2a6bbafa1ed2856cf40-Paper-Conference.pdf.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55006–55021. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ac662d74829e4407ce1d126477f4a03a-Paper-Conference.pdf.

A Definitions and Related MOO Results

A.1 Conventional Gradient-Based MOO Methods

Let $\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} \mathcal{L}_t$ denote the full-model gradient of objective t, let $\Delta = \{\boldsymbol{\alpha} \geq \mathbf{0} : \sum_{t=1}^{T} \alpha_t = 1\}$, and let $\mathbf{g}_{avg} = T^{-1} \sum_{t=1}^{T} \mathbf{g}_t$.

Multiple Gradient Descent Algorithm (MGDA). MGDA (Fliege & Svaiter, 2000; Désidéri, 2012) selects the minimum-norm convex combination of objective gradients:

$$\boldsymbol{\alpha}^* \in \arg\min_{\boldsymbol{\alpha} \in \Delta} \left\| \sum_{t=1}^{T} \alpha_t \mathbf{g}_t \right\|_2^2, \qquad \mathbf{d}^{\mathrm{MGDA}} = \sum_{t=1}^{T} \alpha_t^* \mathbf{g}_t. \tag{7}$$

A nonzero $-\mathbf{d}^{\mathrm{MGDA}}$ is a common descent direction; a zero direction certifies Pareto stationarity.

Conflict-Averse Gradient Descent (CAGrad). CAGrad (Liu et al., 2021) maximizes worst-case alignment within a trust region around the average gradient:

$$\mathbf{d}^{\mathrm{CAGrad}} = \arg\max_{\mathbf{d}} \min_{t} \langle \mathbf{g}_t, \mathbf{d} \rangle \quad \text{s.t.} \quad \|\mathbf{d} - \mathbf{g}_{avg}\|_2 \leq c \|\mathbf{g}_{avg}\|_2, \qquad 0 \leq c < 1. \tag{8}$$

The trust region balances conflict aversion with progress on the average objective.

Projected Conflicting Gradients (PCGrad). PCGrad (Yu et al., 2020) projects away pairwise conflicts. When $\langle \mathbf{g}_i, \mathbf{g}_j \rangle < 0$, it replaces

$$\mathbf{g}_i^{\mathrm{PC}} = \mathbf{g}_i - \frac{\langle \mathbf{g}_i, \mathbf{g}_j \rangle}{\|\mathbf{g}_j\|_2^2} \mathbf{g}_j, \qquad \mathbf{d}^{\mathrm{PCGrad}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{g}_i^{\mathrm{PC}}. \tag{9}$$

For multiple objectives, PCGrad applies these projections successively, typically in shuffled order, and then aggregates the adjusted gradients.

A.2 Optimality and Stationarity

Definition A.1 (Pareto Optimality)**.** A point $\boldsymbol{\theta}^*$ is **Pareto optimal** if no $\boldsymbol{\theta}$ satisfies $\mathcal{L}_t(\boldsymbol{\theta}) \leq \mathcal{L}_t(\boldsymbol{\theta}^*)$ for every $t \in [T]$ with strict inequality for at least one objective.

Definition A.2 (Pareto Stationarity)**.** A point $\boldsymbol{\theta}^*$ is **Pareto stationary** (PS) if

$$\mathbf{0} \in \mathrm{conv}\left\{ \nabla_{\boldsymbol{\theta}} \mathcal{L}_t(\boldsymbol{\theta}^*) : t \in [T] \right\}. \tag{10}$$

Equivalently, some $\boldsymbol{\alpha} \in \Delta$ satisfies $\sum_{t=1}^{T} \alpha_t \nabla_{\boldsymbol{\theta}} \mathcal{L}_t(\boldsymbol{\theta}^*) = \mathbf{0}$.

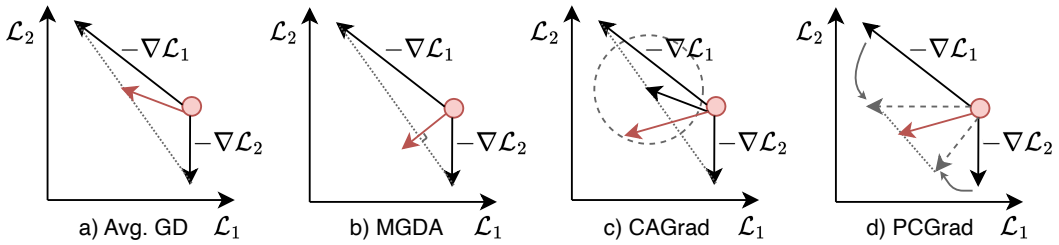

Figure 3: Gradient update rules under conflict. Averaging may reduce one loss at the expense of another; MGDA selects the minimum-norm point in the gradients' convex hull; CAGrad optimizes worst-case alignment within a trust region around the average gradient; and PCGrad removes pairwise conflicting components by projection.

Bucketwise stationarity. Fix a partition $\mathcal{P} = \{\boldsymbol{\theta}_{(1)}, \ldots, \boldsymbol{\theta}_{(K)}\}$ of the trainable parameters into nonempty, disjoint buckets.

Definition A.3 (Bucketwise Pareto Stationarity)**.** A point $\boldsymbol{\theta}^*$ is **bucketwise Pareto stationary** (BPS) with respect to $\mathcal{P}$ if

$$\mathbf{0} \in \prod_{k=1}^{K} \operatorname{conv}\left\{\nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t(\boldsymbol{\theta}^*) : t \in [T]\right\}. \tag{11}$$

Equivalently, each bucket admits weights $\boldsymbol{\alpha}^{(k)} \in \Delta$ such that

$$\sum_{t=1}^{T} \alpha_t^{(k)} \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t(\boldsymbol{\theta}^*) = \mathbf{0}. \tag{12}$$

Theorem A.4 (Necessity of BPS)**.** *For an unconstrained parameter domain, if all objectives are continuously differentiable, every Pareto-optimal point is BPS with respect to any fixed partition $\mathcal{P}$.*

Proof. Fix k and hold every bucket $\ell \neq k$ at $\boldsymbol{\theta}^*_{(\ell)}$. This leaves a T-objective problem in $\boldsymbol{\theta}_{(k)}$. If $\boldsymbol{\theta}^*_{(k)}$ were not Pareto optimal for this restricted problem, replacing bucket k would weakly improve every objective and strictly improve at least one, contradicting Pareto optimality of $\boldsymbol{\theta}^*$. The first-order necessary condition for the restricted problem therefore gives $\mathbf{0} \in \operatorname{conv}\{\nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t(\boldsymbol{\theta}^*) : t \in [T]\}$. Applying this argument to every bucket proves BPS. □

Refined Pareto stationarity. Write $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ and define the objective–parameter dependency matrix $A \in \{0,1\}^{T \times d}$ by $A_{tj} = 1$ if and only if $\mathcal{L}_t$ depends on θ_j. Let

$$\mathcal{P}_{\mathrm{RPS}} = \operatorname{RefinedPartition}(A) = \{P_1, \ldots, P_R\} \tag{13}$$

be the partition obtained by repeatedly merging parameter coordinates that occur in a common cycle of the bipartite dependency graph until the contracted graph is acyclic (Hu & Yu, 2025). For each block, let $F(P_r) = \{t \in [T] : A_{tj} = 1 \text{ for some } j \in P_r\}$, let $\boldsymbol{\theta}_{P_r}$ denote the parameter subvector indexed by P_r, and let $\Delta_{F(P_r)}$ denote the probability simplex over the objectives in $F(P_r)$.

Definition A.5 (Refined Pareto Stationarity)**.** A point $\boldsymbol{\theta}^*$ is **refined Pareto stationary** (RPS) if it is blockwise stationary with respect to $\mathcal{P}_{\mathrm{RPS}}$:

$$\mathbf{0} \in \prod_{r=1}^{R} \operatorname{conv}\left\{\nabla_{\boldsymbol{\theta}_{P_r}} \mathcal{L}_t(\boldsymbol{\theta}^*) : t \in F(P_r)\right\}. \tag{14}$$

Equivalently, for each r there exists $\boldsymbol{\lambda}^{(r)} \in \Delta_{F(P_r)}$ such that

$$\sum_{t \in F(P_r)} \lambda_t^{(r)} \nabla_{\boldsymbol{\theta}_{P_r}} \mathcal{L}_t(\boldsymbol{\theta}^*) = \mathbf{0}. \tag{15}$$

Relationships among PO, RPS, PS, and BPS. For the dependency-refined partition above, $\mathrm{PO} \subseteq \mathrm{RPS} \subseteq \mathrm{PS}$ (Hu & Yu, 2025). Moreover, $\mathrm{PS} \subseteq \mathrm{BPS}$ for any fixed bucket partition, because the single global weight vector certifying PS also balances every bucket. Thus,

$$\mathrm{PO} \subseteq \mathrm{RPS} \subseteq \mathrm{PS} \subseteq \mathrm{BPS}. \tag{16}$$

In a fully shared model, the dependency graph is dense and the refinement collapses to one parameter block, so RPS reduces to PS. BPS generally does not imply PS or RPS because it permits different balancing weights across native buckets. For Bucket-Level MGDA, BPS exactly characterizes zero updates.

Definition A.6 (Conflicting Gradients)**.** Gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_j$ **conflict** if $\langle \nabla_{\boldsymbol{\theta}}\mathcal{L}_i, \nabla_{\boldsymbol{\theta}}\mathcal{L}_j \rangle < 0$, equivalently, if their cosine similarity is negative.

Theorem A.7 (Désidéri (2012))**.** *Let $\boldsymbol{\alpha}$ solve Equation 7, and set $\mathbf{d}^{\mathrm{MGDA}} = \sum_{t=1}^{T} \alpha_t \nabla_{\boldsymbol{\theta}}\mathcal{L}_t$. Then either*

(a) $\mathbf{d}^{\mathrm{MGDA}} = \mathbf{0}$ and the point is Pareto stationary; or

(b) $-\mathbf{d}^{\mathrm{MGDA}}$ is a common descent direction.

Theorem A.8 (Liu et al. (2021))**.** *Let $\mathcal{L}_{avg} = T^{-1}\sum_{t=1}^{T}\mathcal{L}_t$ be lower bounded, and suppose the objective gradients are L-Lipschitz. For exact CAGrad updates with $0 \le c < 1$ and fixed step size $0 < \eta \le 1/L$,*

$$\sum_{n=0}^{\infty} \|\nabla \mathcal{L}_{avg}(\boldsymbol{\theta}^n)\|_2^2 < \infty, \tag{17}$$

so $\|\nabla \mathcal{L}_{avg}(\boldsymbol{\theta}^n)\|_2 \to 0$. Consequently, every accumulation point is stationary for $\mathcal{L}_{avg}$ and hence Pareto stationary under uniform weights.

B Proof of Theorem 2.1

Proof. For bucket k, $\mathbf{d}_{(k)}^{\mathrm{MGDA}}$ is the minimum-norm point in the convex hull of its objective gradients. The projection property of this point gives (Désidéri, 2012)

$$\left\langle \nabla_{\boldsymbol{\theta}_{(k)}}\mathcal{L}_t, \mathbf{d}_{(k)}^{\mathrm{MGDA}} \right\rangle \ge \left\| \mathbf{d}_{(k)}^{\mathrm{MGDA}} \right\|_2^2 \qquad \forall t, k. \tag{18}$$

Summing over buckets yields

$$\left\langle \nabla_{\boldsymbol{\theta}}\mathcal{L}_t, \mathbf{d}^{\mathrm{MGDA}} \right\rangle = \sum_{k=1}^{K} \left\langle \nabla_{\boldsymbol{\theta}_{(k)}}\mathcal{L}_t, \mathbf{d}_{(k)}^{\mathrm{MGDA}} \right\rangle \ge \sum_{k=1}^{K} \left\| \mathbf{d}_{(k)}^{\mathrm{MGDA}} \right\|_2^2 = \left\| \mathbf{d}^{\mathrm{MGDA}} \right\|_2^2. \tag{19}$$

Because the squared global norm is the sum of the squared bucket norms, $\mathbf{d}^{\mathrm{MGDA}} = \mathbf{0}$ exactly when every bucket direction is zero. By the local MGDA definition, this is equivalent to BPS. Otherwise, Equation 19 is strictly positive for every objective, so $-\mathbf{d}^{\mathrm{MGDA}}$ is a common descent direction.

For convergence, let $L = \max_t L_t$ and $\gamma = \eta(1 - L\eta/2) > 0$. Smoothness and Equation 19 imply, for every t,

$$\mathcal{L}_t(\boldsymbol{\theta}^{n+1}) \le \mathcal{L}_t(\boldsymbol{\theta}^n) - \eta \langle \nabla \mathcal{L}_t(\boldsymbol{\theta}^n), \mathbf{d}^n \rangle + \frac{L_t \eta^2}{2} \|\mathbf{d}^n\|_2^2 \tag{20}$$

$$\le \mathcal{L}_t(\boldsymbol{\theta}^n) - \gamma \|\mathbf{d}^n\|_2^2. \tag{21}$$

Telescoping and lower boundedness imply

$$\gamma \sum_{n=0}^{\infty} \|\mathbf{d}^n\|_2^2 \le \mathcal{L}_t(\boldsymbol{\theta}^0) - \inf_{\boldsymbol{\theta}} \mathcal{L}_t(\boldsymbol{\theta}) < \infty, \tag{22}$$

so $\|\mathbf{d}^n\|_2 \to 0$. For any convergent subsequence $\boldsymbol{\theta}^{n_j} \to \bar{\boldsymbol{\theta}}$, compactness of the simplices yields a further subsequence on which every $\boldsymbol{\alpha}^{(k),n_j}$ converges. Taking limits in $\mathbf{d}_{(k)}^{n_j} = \sum_t \alpha_t^{(k),n_j} \nabla_{\boldsymbol{\theta}_{(k)}}\mathcal{L}_t(\boldsymbol{\theta}^{n_j}) \to \mathbf{0}$ shows that $\bar{\boldsymbol{\theta}}$ is BPS. □

C Proof of Theorem 2.2

Proof. Let $F = \mathcal{L}_{avg}$, $\mathbf{g}_{(k)} = \nabla_{\boldsymbol{\theta}_{(k)}} F$, and $\mathbf{d}_{(k)} = \mathbf{d}_{(k)}^{\mathrm{CAGrad}}$. The local trust-region constraint is

$$\|\mathbf{d}_{(k)} - \mathbf{g}_{(k)}\|_2^2 \leq c^2 \|\mathbf{g}_{(k)}\|_2^2. \tag{23}$$

Expanding it gives

$$2\langle \mathbf{g}_{(k)}, \mathbf{d}_{(k)} \rangle \geq \|\mathbf{d}_{(k)}\|_2^2 + (1 - c^2)\|\mathbf{g}_{(k)}\|_2^2. \tag{24}$$

After summing over buckets,

$$2\langle \nabla F, \mathbf{d} \rangle \geq \|\mathbf{d}\|_2^2 + (1 - c^2)\|\nabla F\|_2^2. \tag{25}$$

If $\nabla F \neq \mathbf{0}$, the right-hand side is positive because $c < 1$; hence $-\mathbf{d}$ is a descent direction for F.

The global update is zero exactly when every $\mathbf{d}_{(k)}$ is zero. In that case, the trust-region constraint gives $\|\mathbf{g}_{(k)}\|_2 \leq c\|\mathbf{g}_{(k)}\|_2$, so $\mathbf{g}_{(k)} = \mathbf{0}$ because $c < 1$. Conversely, $\mathbf{g}_{(k)} = \mathbf{0}$ forces $\mathbf{d}_{(k)} = \mathbf{0}$. Hence $\mathbf{d} = \mathbf{0}$ if and only if $\nabla F = \mathbf{0}$. The latter condition is PS under uniform global weights and BPS under uniform bucket weights.

For convergence, write $\mathbf{g}^n = \nabla F(\boldsymbol{\theta}^n)$. For $0 < \eta \leq 1/L_F$, smoothness and Equation 25 give

$$F(\boldsymbol{\theta}^{n+1}) \leq F(\boldsymbol{\theta}^n) - \eta\langle \mathbf{g}^n, \mathbf{d}^n \rangle + \frac{L_F \eta^2}{2} \|\mathbf{d}^n\|_2^2 \tag{26}$$

$$\leq F(\boldsymbol{\theta}^n) - \frac{\eta(1 - c^2)}{2} \|\mathbf{g}^n\|_2^2. \tag{27}$$

Lower boundedness of F then implies $\sum_n \|\mathbf{g}^n\|_2^2 < \infty$ and $\|\mathbf{g}^n\|_2 \to 0$. Finally, $\|\mathbf{d}^n - \mathbf{g}^n\|_2 \leq c\|\mathbf{g}^n\|_2$ gives $\|\mathbf{d}^n\|_2 \leq (1 + c)\|\mathbf{g}^n\|_2 \to 0$. □

D Distributed Execution

Algorithm 1 states Bucket-Level MOO as a protocol rather than a solver. As each bucket's gradients become ready during backpropagation, each worker broadcasts that bucket in turn, every other worker reduces the arrival to a scalar inner product through a single reusable buffer, the bucket's conflicts are resolved from the resulting Gram matrix, the resolved update is handed to the standard ZeRO-2 reduction, which leaves each worker only its own partition, and the buffer is released before the next bucket arrives. No stage of the step ever holds a full-model gradient for any objective, and no worker ever holds more than one bucket of gradient beyond its own.

One interface for every rule. MGDA, CAGrad, and PCGrad all return an update that is a linear combination $\sum_t \beta_t \mathbf{g}_t^{(k)}$ whose coefficients depend on the bucket gradients only through their pairwise inner products. Algorithm 2 makes this explicit: each rule consumes the $T \times T$ Gram matrix and returns $\boldsymbol{\beta} \in \mathbb{R}^T$. Three consequences follow. First, the per-bucket solver cost is $O(T^3)$ scalar work plus the $O(T|\boldsymbol{\theta}_{(k)}|)$ inner products accumulated during the exchange, independent of how the rule combines gradients; for PCGrad this replaces the $O(T^2)$ vector projections of the naive implementation and explains the small overhead measured in Appendix F. Second, because M is the only quantity the solver reads, the gradients themselves never need to coexist: they can be consumed one at a time as they arrive, which is what bounds the memory in Algorithm 1. Third, substituting a different MOO or multi-task rule requires only a new Coefficients routine, leaving the distributed protocol unchanged, provided the rule is expressible through the Gram matrix; element-wise rules such as GradDrop (Chen et al., 2020) fall outside this interface.

Cost. Let $|\boldsymbol{\theta}| = \sum_k |\boldsymbol{\theta}_{(k)}|$. Peak memory attributable to conflict resolution is $O(\max_k |\boldsymbol{\theta}_{(k)}|)$, a single buffer, rather than the $O(T|\boldsymbol{\theta}|)$ of a naive global implementation that materializes every objective gradient before solving; in particular it does not grow with the number of objectives. The Global MOO baseline of Section 4 is not naive in this sense: it runs the same exchange with a single model-sized bucket, so its extra memory is $O(|\boldsymbol{\theta}|)$ (see below). Communication volume, by contrast, is *not* reduced: each worker receives $(T - 1)|\boldsymbol{\theta}_{(k)}|$ elements per bucket, the same total an All-Gather would move and the same total a global implementation would move. The closing

Algorithm 1 Bucket-Level MOO: one optimizer step under ZeRO-2 data parallelism. Worker t owns objective $\mathcal{L}_t$.

Require: Bucket partition $\mathcal{P} = \{\boldsymbol{\theta}_{(k)}\}_{k=1}^{K}$ supplied by the distributed runtime
Require: Per-worker datasets $\{\mathcal{D}_t\}_{t=1}^{T}$; rule $\mathcal{M} \in \{\text{MGDA}, \text{CAGRAD}, \text{PCGRAD}\}$
Require: Optimizer OPT; base seed s shared by all workers

on worker t**:** draw a micro-batch from $\mathcal{D}_t$ and evaluate $\mathcal{L}_t(\boldsymbol{\theta})$ ▷ forward pass
for each bucket k **as its gradients become ready during backpropagation do**
  $\mathbf{g}_t^{(k)} \leftarrow \nabla_{\boldsymbol{\theta}_{(k)}} \mathcal{L}_t$ ▷ local to worker t
  allocate one buffer $\mathbf{b}$ of size $|\boldsymbol{\theta}_{(k)}|$ ▷ the only memory conflict resolution adds
  for $j = 1, \ldots, T$ **do**
    if $j = t$ **then**
      $\mathbf{b} \leftarrow \mathbf{g}_t^{(k)}$ ▷ worker j supplies the payload for its own turn
    end if
    $\mathbf{b} \leftarrow \text{BROADCAST}\big(\mathbf{b};\ \text{root} = j\big)$ ▷ one gradient in flight at a time
    $\rho_j \leftarrow \langle \mathbf{g}_t^{(k)}, \mathbf{b} \rangle$ ▷ chunked, `fp32` per chunk into an `fp64` accumulator; $\mathbf{b}$ is consumed on arrival
  end for
  free $\mathbf{b}$ ▷ peak MOO memory stays $O\big(\max_k |\boldsymbol{\theta}_{(k)}|\big)$, independent of T
  $R \leftarrow \text{ALL-GATHER}(\boldsymbol{\rho})$; $M \leftarrow \frac{1}{2}\big(R + R^\top\big)$ ▷ T^2 *scalars*, identical on every worker; symmetrizing makes M exactly symmetric
  $\boldsymbol{\beta} \leftarrow \text{COEFFICIENTS}_{\mathcal{M}}(M;\ s_k)$, $s_k = (s,\ \text{step},\ k)$ ▷ identical on every worker (Alg. 2); $O(T^3)$ scalar work
  $\mathbf{g}_t^{(k)} \leftarrow \beta_t\, \mathbf{g}_t^{(k)}$; $\mathbf{d}_{(k)} \leftarrow \text{ALL-REDUCE}\big(\mathbf{g}_t^{(k)},\ \text{SUM}\big)$ ▷ scaled in place; worker t contributes only its own term
  keep only worker t's ZeRO-2 partition of $\mathbf{d}_{(k)}$ ▷ as for ordinary ZeRO-2 gradients
end for
$\boldsymbol{\theta} \leftarrow \text{OPT}\big(\boldsymbol{\theta},\ \mathbf{d} = [\mathbf{d}_{(1)}, \ldots, \mathbf{d}_{(K)}]\big)$ ▷ per-partition AdamW step, as in ZeRO-2

All-Reduce is the collective that DeepSpeed's ZeRO-2 already uses for ordinary gradients before each worker keeps its own partition, so Vanilla SFT performs it as well and the added traffic is the T broadcasts. Bucketing itself does not change the volume, since the same exchange could be applied to full gradients; what bucketing contributes is that the transfers are issued incrementally during the backward pass, bucket by bucket, and keep the transient buffer independent of both T and model size. The transfers do not overlap with computation in our runs: we keep DeepSpeed's ZeRO-2 default of no communication–computation overlap, and each bucket's solve waits for its Gram matrix on the CPU. Halving the volume is possible in principle, by forwarding each gradient only $\lfloor T/2 \rfloor$ hops around a ring so that every pair of objectives meets exactly once rather than twice, but we do not implement it here. Table 5 reports the resulting wall-clock overhead.

Global MOO baseline. The Global columns of Tables 3 and 4 use the trivial partition $\mathcal{P} = \{\boldsymbol{\theta}\}$ and run through exactly the same code path as every bucketed configuration: the same ZeRO-2 setup, data pipeline, hyperparameters and COEFFICIENTS routines. The only change is the ZeRO-2 reduction bucket, which we set to the number of trainable parameters plus one, so that the whole gradient forms a single bucket ($K = 1$); the parameter all-gather bucket keeps its default size. Algorithm 1 then streams the T full-model language gradients through one model-sized buffer, solves a single MOO problem for the whole model from the global Gram matrix, and all-reduces the weighted gradients, each worker again keeping only its own partition. Global MOO therefore never holds another worker's gradient beyond that one buffer, but its flattened bucket, its broadcast buffer and DeepSpeed's reduction buffer are all model-sized rather than bounded by the bucket size. This accounts for its footprint in Table 4: on `Qwen3-8B-Base`, its peak of 85.9 GB, whereas every bucketed configuration stays below 62 GB.

Memory measurement. Peak VRAM in Table 4 is the largest device memory reported by `nvidia-smi` on any worker during training, in GB of 2^{30} bytes. It includes memory that Py-

Algorithm 2 COEFFICIENTS$_{\mathcal{M}}$. Each rule returns $\boldsymbol{\beta} \in \mathbb{R}^T$ such that the bucket update is $\mathbf{d}_{(k)} = \sum_{t=1}^{T} \beta_t \, \mathbf{g}_t^{(k)}$, and depends on the gradients *only* through the Gram matrix M. This is what lets Algorithm 1 assemble M by streaming exchange instead of materializing the objective gradients.

Require: Gram matrix $M \in \mathbb{R}^{T \times T}$ in `fp64`; simplex $\Delta = \{\boldsymbol{\alpha} \geq \mathbf{0} : \sum_t \alpha_t = 1\}$; uniform weights $\boldsymbol{u} = T^{-1}\mathbf{1}$

function COEFFICIENTS$_{\text{MGDA}}(M)$
  $V \Lambda V^\top \leftarrow \mathrm{eig}(M)$; $P \leftarrow V\sqrt{\max(\Lambda, 0)}$ ▷ $PP^\top = M$, so the rows of P are T-dimensional proxies for the gradients
  return $\arg\min_{\boldsymbol{\alpha} \in \Delta} \left\| P^\top \boldsymbol{\alpha} \right\|_2^2$ ▷ min-norm point (Eq. 3), via Frank–Wolfe (Frank et al., 1956)
end function

function COEFFICIENTS$_{\text{CAGRAD}}(M, c)$
  $\phi \leftarrow c\sqrt{\boldsymbol{u}^\top M \boldsymbol{u}}$ ▷ $\phi = c\|\mathbf{g}_{(k)}^{avg}\|_2$
  $\boldsymbol{w}^* \leftarrow \arg\min_{\boldsymbol{w} \in \Delta} \; \boldsymbol{w}^\top M \boldsymbol{u} + \phi\sqrt{\boldsymbol{w}^\top M \boldsymbol{w}}$ ▷ dual of Equation 4 (Liu et al., 2021), solved by SLSQP
  return $\dfrac{1}{1+c^2}\left(\boldsymbol{u} + \dfrac{\phi}{\sqrt{\boldsymbol{w}^{*\top} M \boldsymbol{w}^*}}\, \boldsymbol{w}^*\right)$ ▷ including the $1/(1+c^2)$ rescaling of the reference implementation
end function

function COEFFICIENTS$_{\text{PCGRAD}}(M, s_k)$
  $C \leftarrow I_T$; $\pi \leftarrow \text{SHUFFLE}\big([T];\, s_k\big)$ ▷ one order, drawn from the shared seed and reused for every t
  for $t = 1, \ldots, T$ **do**
    for $j \in \pi$ **with** $j \neq t$ **do**
      $\rho \leftarrow \big(C_{t,:} M\big)_j$ ▷ $\rho = \langle \mathbf{g}_t^{\text{PC}}, \mathbf{g}_j^{(k)} \rangle$
      if $\rho < 0$ **then**
        $C_{tj} \leftarrow C_{tj} - \rho / M_{jj}$ ▷ row t expands $\mathbf{g}_t^{\text{PC}}$ in the basis $\{\mathbf{g}_j^{(k)}\}_{j=1}^T$
      end if
    end for
  end for
  return $T^{-1} C^\top \mathbf{1}$ ▷ averaged column sums, since $\mathbf{d}_{(k)} = T^{-1} \sum_t \mathbf{g}_t^{\text{PC}}$
end function

Torch's caching allocator reserves but does not use. DeepSpeed also sizes its own reduction buffers by the bucket size, so configurations with 100M or 250M buckets can peak slightly below Vanilla SFT, which keeps the 500M default. Live allocations are tighter: on `Qwen3-4B-Base`, the peak allocated memory (`torch.cuda.max_memory_allocated`) is 25.97 GB for both Vanilla SFT and Bucket-Level MOO at 500M, and 35.84 GB for Global MOO, so the difference between the first two in Table 4 is reserved but unused memory.

Numerical guards. Algorithm 2 omits the small dampers the implementation carries: the CAGrad and PCGrad divisions add 10^{-8} to $\sqrt{\boldsymbol{w}^\top M \boldsymbol{w}}$ and to M_{jj} respectively, so a bucket in which some objective's gradient vanishes cannot produce a division by zero. CAGrad additionally short-circuits to $\boldsymbol{u}$ when $c = 0$, which is the exact limit of its subproblem.

Practical notes. The presentation above assigns one objective per worker, as in all our experiments (eight workers for LIMA–s1 and four for WMT24++), and each optimizer step uses a single micro-batch per worker, without gradient accumulation; with accumulation, $\mathbf{g}_t^{(k)}$ would denote the accumulated bucket gradient for objective t and conflict resolution would be applied once per optimizer step, not once per micro-batch. DeepSpeed's ZeRO-2 reduction divides each bucket by the number of workers before our routine runs; because all three rules are invariant to a common rescaling of

the gradients, $\boldsymbol{\beta}$ is unchanged and the applied update is $\mathbf{d}/T$, a constant factor that AdamW absorbs. Because every worker sees the same Gram matrix M and solves the same subproblem, the solvers must be deterministic across ranks; PCGRAD therefore draws its projection order from a seed shared by all workers. Finally, Theorems 2.1 and 2.2 analyze the raw direction $\mathbf{d}$ under the exact update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\mathbf{d}$; our experiments pass $\mathbf{d}$ to AdamW, whose preconditioning is outside that analysis.

E EXPERIMENT DETAILS

We fine-tune all models using the `TRL`[3] library with a maximum sequence length of 2,048 tokens and a batch size of 128. All runs use `bfloat16` precision, the AdamW optimizer, and a cosine learning-rate schedule with a warmup ratio of 0.03. For the Qwen3 models, we use a peak learning rate of 2×10^{-5} and a weight decay of 0.1; for the Llama 3 models, we use a peak learning rate of 4×10^{-6} and a weight decay of 0.04. We train for three epochs and evaluate the final checkpoint. The LIMA–s1 experiments use eight NVIDIA H200 GPUs, whereas the WMT24++ experiments use four. The + *bal-loader* baseline combines the language gradients with uniform weights and uses a balanced data loader that guarantees that every step loads the language-specific batches in balance. For Bucket-Level CAGrad, we set the conflict-aversion coefficient to $c = 0.5$ for all runs, and rescale the resulting direction by $1/(1 + c^2)$, following the reference CAGrad implementation; this constant factor is absorbed by AdamW's preconditioning and is omitted from Equation 4 and Theorem 2.2 for clarity.

F TRAINING TIME

Table 5 reports wall-clock training time for `Meta-Llama-3-8B` on the LIMA–s1 experiment. Vanilla SFT completes in 0.6 hours, while CAGrad requires 0.7 hours and MGDA and PCGrad each require 0.8 hours. Thus, bucket-level conflict resolution adds only 0.1–0.2 hours to this configuration, indicating a modest computational overhead relative to standard SFT.

G PER-LANGUAGE ANALYSIS ON LIMA–S1

Seen-Language Improvements. Table 6 shows that, for every backbone, all three Bucket-Level MOO variants outperform both SFT baselines on average. Relative to balanced SFT, MOO improves 79 of the 96 model–language–method comparisons, although the magnitude varies across languages and backbones. Indonesian (`id`) benefits consistently: CAGrad improves its score by $+5.8$, $+5.5$, $+5.2$, and $+2.7$ points on `Meta-Llama-3-8B`, `Llama-3.1-8B`, `Qwen3-4B-Base`, and `Qwen3-8B-Base`, respectively. The gains also span scripts and resource levels; CAGrad improves Arabic (`ar`) by $+2.8$ points on `Qwen3-8B-Base`, while PCGrad improves Swahili (`sw`) by $+1.7$ points on `Qwen3-4B-Base`. PCGrad improves all eight languages on `Qwen3-4B-Base`, and CAGrad does so on `Qwen3-8B-Base`. Overall, the gains are broadly distributed, but no single gradient-combination strategy dominates for every backbone and language.

Generalization to Unseen Languages. Table 7 shows similarly broad zero-shot transfer. Across the four unseen European languages (`de, es, fr, pt`), Bucket-Level MOO improves 46 of 48 comparisons with balanced SFT. Illustrative gains include $+3.8$ points in Spanish (`es`) from PCGrad on `Meta-Llama-3-8B`, $+3.5$ points in Portuguese (`pt`) from MGDA on `Llama-3.1-8B`, and $+2.8$ points in German (`de`) from CAGrad on `Qwen3-4B-Base`. These results indicate that resolving conflicts among the eight supervised languages can also benefit related languages without direct training data.

Japanese (`ja`) is more architecture dependent. Relative to balanced SFT, the MOO variants change performance by -1.1 to -0.2 points on `Meta-Llama-3-8B` and by -0.8 to $+0.8$ points on `Llama-3.1-8B`. In contrast, every MOO variant improves Japanese on both Qwen models: the gains range from $+1.8$ to $+3.9$ points on `Qwen3-4B-Base` and from $+0.4$ to $+1.0$ points on `Qwen3-8B-Base`. This difference suggests that zero-shot transfer to a typologically and orthographically distant language depends on both conflict resolution and the multilingual capacity of the pretrained backbone.

[3] `https://github.com/huggingface/trl`

Table 5: Training time for `Meta-Llama-3-8B` on LIMA–s1 using eight H200 GPUs.

Method	V. SFT	MGDA	CAGrad	PCGrad
Hours	0.6	0.8	0.7	0.8

Table 6: Per-language results on the eight seen languages after LIMA–s1 fine-tuning.

Method	**EN**	**ZH**	**IT**	**AR**	**KO**	**ID**	**BN**	**SW**	**Avg.**
					Meta-Llama-3-8B				
Base	53.7	39.6	37.1	38.4	40.3	44.5	30.5	26.0	38.8
V. SFT	59.3	45.6	52.5	41.5	42.5	47.7	32.9	29.4	44.0
+ *bal-loader*	61.4	50.3	52.0	43.3	44.1	45.4	34.1	31.4	45.3
MGDA	63.6	50.3	55.2	44.2	44.9	50.9	34.6	30.9	46.8
CAGrad	60.9	48.5	54.3	44.4	45.0	51.2	34.3	30.6	46.2
PCGrad	62.5	50.9	54.2	44.6	45.6	50.8	34.5	32.2	46.9
					Llama-3.1-8B				
Base	59.4	48.6	51.4	41.9	44.1	46.9	30.1	25.5	43.5
V. SFT	60.9	48.9	51.9	42.2	43.4	48.2	34.1	28.1	44.7
+ *bal-loader*	61.9	50.2	54.7	42.7	45.2	46.5	36.5	30.9	46.1
MGDA	64.9	52.7	56.6	44.9	44.1	50.4	35.4	28.3	47.2
CAGrad	63.3	54.1	55.7	44.1	45.1	52.0	35.9	28.4	47.3
PCGrad	62.9	51.4	55.8	44.1	45.7	51.1	34.6	29.8	46.9
					Qwen3-4B-Base				
Base	75.3	72.3	72.1	65.0	65.4	71.6	55.7	29.5	63.4
V. SFT	75.2	70.4	69.9	62.9	63.2	66.9	52.5	35.1	62.0
+ *bal-loader*	75.8	69.7	70.3	62.9	64.3	65.5	54.7	34.1	62.2
MGDA	77.5	71.2	72.3	63.7	66.6	70.6	55.3	31.3	63.6
CAGrad	77.4	71.8	73.8	62.8	65.0	70.7	56.5	34.4	64.0
PCGrad	78.3	71.2	72.8	64.0	65.7	69.2	55.3	35.8	64.0
					Qwen3-8B-Base				
Base	78.4	73.9	75.0	68.7	70.7	75.7	64.6	40.2	68.4
V. SFT	79.6	74.3	73.7	68.5	69.4	73.0	60.1	41.9	67.6
+ *bal-loader*	80.6	73.9	74.9	69.4	69.5	71.2	61.8	43.3	68.1
MGDA	80.9	75.2	76.6	70.9	72.6	73.8	63.0	41.8	69.3
CAGrad	81.1	74.9	76.5	72.2	71.0	73.9	62.0	44.2	69.5
PCGrad	80.5	75.9	76.1	71.1	70.0	73.2	62.4	44.1	69.2

Mitigating Language-Specific Forgetting. The per-language results also show that multilingual fine-tuning can overwrite capabilities already present in stronger base models. Across the two Qwen backbones, Vanilla SFT reduces 22 of the 26 seen- and unseen-language scores relative to the corresponding pretrained models; balanced loading reduces this count only to 19. Bucket-Level MOO recovers many of these losses. On `Qwen3-4B-Base`, Bengali (`bn`) drops from 55.7 to 52.5 under Vanilla SFT but reaches 56.5 with CAGrad, surpassing the pretrained score. Japanese decreases from 67.2 to 62.7 under Vanilla SFT and recovers to 67.1 with MGDA. On `Qwen3-8B-Base`, MGDA raises Korean (`ko`) from 69.4 to 72.6, above the pretrained score of 70.7, while CAGrad raises Arabic (`ar`) from 68.5 to 72.2. These cases suggest that Bucket-Level MOO supports multilingual adaptation both by learning the fine-tuning tasks and by preserving language-specific capabilities that standard SFT can overwrite.

H Multilingual Representation Analysis

We examine whether Bucket-Level MOO changes how multilingual information is organized across model depth. After LIMA–s1 fine-tuning, we extract the final-input-token hidden state at every

Table 7: Zero-shot results on the five unseen languages after LIMA–s1 fine-tuning.

Method	DE	ES	FR	JA	PT	Avg.
Meta-Llama-3-8B						
Base	45.2	46.2	46.1	34.3	49.8	44.3
V. SFT	51.1	54.3	51.2	43.8	53.0	50.7
+ *bal-loader*	51.9	53.0	53.2	43.9	54.4	51.3
MGDA	54.4	56.2	54.1	43.7	55.3	52.8
CAGrad	54.0	55.2	54.5	42.8	55.4	52.4
PCGrad	53.9	56.8	54.5	43.7	54.6	52.7
Llama-3.1-8B						
Base	49.2	50.9	53.2	43.5	53.7	50.1
V. SFT	53.7	53.4	53.1	44.4	54.3	51.8
+ *bal-loader*	54.1	54.8	55.0	45.1	54.2	52.6
MGDA	54.8	57.3	55.1	45.9	57.7	54.2
CAGrad	55.0	57.1	57.0	45.4	56.8	54.3
PCGrad	55.2	56.4	56.9	44.3	57.6	54.1
Qwen3-4B-Base						
Base	70.4	74.0	71.7	67.2	72.6	71.2
V. SFT	68.9	71.1	69.3	62.7	69.1	68.2
+ *bal-loader*	68.2	69.8	68.7	63.2	70.7	68.1
MGDA	70.3	73.2	71.1	67.1	72.2	70.8
CAGrad	71.0	72.5	70.8	66.5	72.7	70.7
PCGrad	70.8	72.5	71.0	65.0	70.7	70.0
Qwen3-8B-Base						
Base	75.2	76.5	74.0	70.9	75.4	74.4
V. SFT	73.9	75.2	73.8	68.0	74.1	73.0
+ *bal-loader*	73.0	74.8	75.1	69.8	74.1	73.4
MGDA	75.7	75.8	76.4	70.8	75.3	74.8
CAGrad	76.2	76.7	75.4	70.2	75.7	74.8
PCGrad	73.7	76.3	74.9	70.8	75.8	74.3

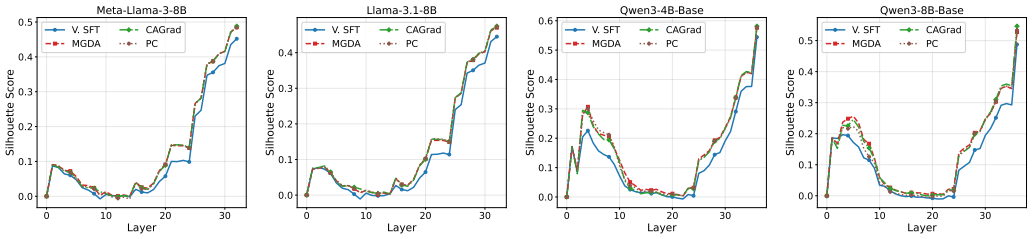

Figure 4: Layer-wise language-clustering Silhouette scores for all four backbones after LIMA–s1 fine-tuning.

layer for all 13,040 training samples (1,630 per seen language), and compute the Silhouette score (Rousseeuw, 1987) using language identity as the cluster label. Higher values indicate representations that are more compact within languages and more distinct across languages. The comparison is against Vanilla SFT; the balanced-loader control is not included here.

Figure 4 reveals a depth-dependent effect that is consistent across all four backbones. All Bucket-Level variants improve early-layer separability over Vanilla SFT. Scores then approach zero in the middle layers for every method, suggesting that those layers still mix languages within a shared space. Separation re-emerges in the network's final third, where the Bucket-Level variants lead across several consecutive layers; on `Llama-3.1-8B`, for example, the advantage holds throughout approximately the final ten layers. Although every method spikes at the output-facing layer, the earlier late-layer gap shows that the effect extends beyond the final prediction interface.

The absolute scale varies by backbone: the Qwen3 models reach peak scores near 0.60, compared with approximately 0.45 for the Llama models. That the improvement appears at both levels of initial multilingual alignment suggests it is not confined to a particular architecture or degree of pretrained separability.

Greater language separability may appear to sit uneasily with the improved zero-shot transfer reported in Section 3.2, since cross-lingual generalization is often attributed to shared representations. The depth profile suggests the two are compatible: separation increases at the input- and output-facing stages, where language-specific encoding and decoding take place, while the middle layers remain mixed and continue to carry shared computation. On this reading, localized conflict resolution sharpens language-conditioned structure at the interfaces without partitioning the network's shared processing. Silhouette scores characterize representational geometry, however, and we treat this evidence as an association rather than a causal explanation of the downstream gains.