# BEYOND GRADIENT CONFLICT: MULTI-TASK LANDSCAPE-BASED OPTIMIZATION VIA GEOMETRIC INDEPENDENT-TASK CONFLUENCE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-task learning (MTL) is largely dominated by gradient-based methods that reconcile conflicting objectives through a single shared update direction. However, under strong task conflict, this shared-trajectory constraint often leads to unstable optimization and degraded performance. We introduce **LOGIC**, **L**andscape-based **O**ptimization via **G**eometric **I**ndependent-task **C**onfluence, a conceptual shift in MTL that replaces a single shared gradient compromise with independently optimized task trajectories that are later coordinated in parameter space. Rather than enforcing agreement at every optimization step, LOGIC allows each task to evolve along its own loss landscape before geometrically aligning task-adapted solutions through a curvature-weighted centroid, with periodic re-anchoring of the task trajectories. By decoupling task-specific optimization from cross-task coordination, LOGIC stabilizes training and enables consistent positive transfer under strong task conflict. Across four benchmarks, three backbones, and both vision and language tasks, LOGIC consistently outperforms thirteen strong gradient-based baselines and frequently surpasses single-task learning, while exhibiting substantially smoother and more stable convergence dynamics.

## 1 INTRODUCTION

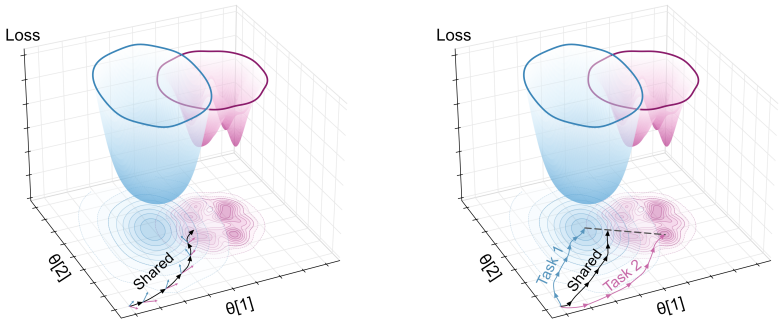
Multi-task learning (MTL) aims to train a single model that solves multiple tasks simultaneously, leveraging shared representations to improve performance (Caruana, 1997; Ruder, 2017). Despite its promise, MTL often fails in practice. As task diversity increases, optimization becomes unstable, performance deteriorates, and jointly trained models frequently underperform their single-task counterparts (Lin et al., 2019; Liu et al., 2021a; 2023; Navon et al., 2022; Zamir et al., 2018). This phenomenon has motivated research on multi-task optimization.

The dominant concept in MTL addresses task interference at the level of gradients. A standard formulation minimizes the average loss across tasks; however, this objective inherently suffers from gradient conflict, where task gradients differ in magnitude or direction and obstruct effective optimization (Liu et al., 2021a). As a result, common MTL methods attempt to reconcile these conflicts by constructing a single shared update direction. While effective in some cases, this strategy fundamentally restricts optimization to shared compromise solutions under conflicting task objectives. For example, Conflict-Averse Gradient Descent (CAGrad) selects an update that maximizes the worst-case local improvement across tasks while remaining close to the average gradient, balancing objectives with convergence guarantees (Liu et al., 2021a). More broadly, gradient reweighting (Chen et al., 2018; Kendall et al., 2018; Liu et al., 2019), gradient projection (Yu et al., 2020), and multi-objective optimization (Sener & Koltun, 2018; Liu et al., 2021a;b; Navon et al., 2022) attempt to resolve task interference through increasingly sophisticated gradient aggregation strategies.

Existing MTL methods operate in gradient space, relying on local, first-order updates that use instantaneous information and overlook curvature and long-term dynamics. More fundamentally, they enforce a single shared trajectory, forcing tasks to agree *at every step* and reducing optimization to a sequence of compromises under conflict, often leading to instability and suboptimal solutions. To overcome this limitation, we introduce **LOGIC**, **L**andscape-based **O**ptimization via **G**eometric **I**ndependent-task **C**onfluence, which operates directly in parameter space. Rather than combining



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.




(a) (b)

Figure 1: Conceptual illustration of multi-task optimization in a shared parameter space with two task-specific loss landscapes. **(a)** Gradient-aggregation MTL methods optimize a single shared parameter vector in $\theta$, limiting exploration to a confined trajectory through the joint loss landscape. **(b)** LOGIC maintains per-task copies that independently traverse each task's loss surface, exposing the geometric structure across landscapes. The shared model is guided toward an inverse-curvature-weighted centroid within the dashed line, steering it toward robust, better-generalized regions.

gradients, LOGIC coordinates task-adapted solutions using trajectory-level and curvature-aware information. Each task evolves independently, and coordination is deferred to remove the "agree-now" constraint and enable geometry-aware alignment that suppresses conflicting directions and promotes stable, well-conditioned regions (See Figure 1). Our contributions are as follows:

- **A conceptual shift: from local first-order gradients to geometry-aware trajectories.** By leveraging trajectory-level information in parameter space, LOGIC captures global landscape geometry invisible to single-step updates, bypassing local gradient compromise. In essence, LOGIC aligns solutions rather than reconciling gradients.
- **Decoupling task optimization from cross-task merging.** LOGIC reverses the merge-then-optimize order of existing MTL methods: rather than merging task gradients and optimizing the shared result, it optimizes each task independently along its own trajectory to accumulate meaningful task-specific information before geometric alignment.
- **Robust positive transfer: exceeding both single-task and multi-task baselines.** LOGIC substantially surpasses single-task performance across four benchmarks, three backbones, and both vision and language tasks while outperforming gradient-based methods, with stable convergence even under severe task conflict.

## 2 Related Work

**Gradient-based Centralized Multi-Task Optimization.** Centralized multi-task optimization (MTO) methods, in which all task losses are jointly available at every step, predominantly handle task interactions by constructing a single shared update compromise. Despite a wide range of techniques, these methods share a common structural assumption: all tasks must be optimized along a single trajectory. Existing approaches resolve task interference at the gradients level.

*Loss weighting methods* adjust each task's relative contribution to balance optimization dynamics. GradNorm (Chen et al., 2018) rescales task losses to equalize gradient magnitudes and align training rates. Uncertainty weighting (UW) (Kendall et al., 2018) derives principled weights from task-dependent homoscedastic uncertainty. DWA (Liu et al., 2019) dynamically adjusts weights based on the rate of loss change over time. RLW (Lin et al., 2022) introduces stochastic weighting as an implicit regularizer, while FAMO (Liu et al., 2023) reweights online to prioritize the worst-performing task. However weights are computed, all combine task gradients into a single update.

*Gradient manipulation methods* modify gradient vectors to reduce task interference. PCGrad (Yu et al., 2020) projects conflicting gradients. MGDA (Sener & Koltun, 2018) computes a minimum-norm solution within the convex hull of task gradients, ensuring Pareto optimality. CAGrad (Liu et al., 2021a) balances average and worst-case improvement. IMTL-G (Liu et al., 2021b) enforces alignment across task gradients, Nash-MTL (Navon et al., 2022) casts aggregation as a bargaining game, MoCo (Fernando et al., 2023) corrects bias in multi-objective gradient estimation, and FairGrad (Ban & Ji, 2024) reallocates gradient contributions to enforce fairness across tasks. More

broadly, MTL is often cast as multi-objective optimization, seeking Pareto-optimal solutions through aggregation or scalarization (Sener & Koltun, 2018; Lin et al., 2019). Despite these differences, all gradient-based approaches share a limitation: they compress task-specific information into one update direction per step. When task optima are misaligned, any such direction necessarily induces a trade-off between objectives (Sener & Koltun, 2018; Bai et al., 2024; Standley et al., 2020; Fifty et al., 2021). Optimization is therefore confined to compromises along the Pareto front, often yielding unstable training and degraded performance under strong conflict.

**Multi-Trajectory and Federated MTL.** Maintaining parameter copies is a recurring idea across several lines of work, but prior uses differ from LOGIC in both motivation and algorithm. Lookahead (Zhang et al., 2019a) stabilizes single-objective training, pulling slow weights toward fast weights of the same objective. Reptile (Nichol et al., 2018) and MAML (Finn et al., 2017) target fast adaptation, combining tasks into a shared initialization for later fine-tuning and thus deferring task conflict rather than resolving it. Distributed methods are motivated by communication and data-locality constraints rather than gradient conflict, and this shapes their algorithms: Local SGD, FedAvg, and its outer-optimizer variants FedOpt and DiLoCo (McMahan et al., 2017; Stich, 2019; Reddi et al., 2021; Douillard et al., 2024) train copies of one objective on data shards and suppress their drift as error (Karimireddy et al., 2020; Patel et al., 2024), and Federated MTL (FMTL) (Smith et al., 2017) is likewise shaped by data locality. Although FedOpt and DiLoCo also apply an outer optimizer to the copies' averaged displacement, LOGIC inverts their premise: without a canonical scalarization, MTL has no target trajectory to deviate from, so the displacement between copies measures how task landscapes differ, not how far optimization has strayed. This displacement carries a curvature–gradient covariance term that vanishes in expectation under a shared objective, and which LOGIC exploits as a guidance signal. Model merging (Matena & Raffel, 2022; Ilharco et al., 2023) fuses converged checkpoints offline, whereas LOGIC coordinates online throughout training. Finally, ForkMerge (Jiang et al., 2023) targets auxiliary-task learning (ATL): it merges branches using weights tuned on a designated target task's validation error, yielding one model per target, whereas LOGIC trains a single symmetric MTL model for all tasks without validation feedback.

## 3 The Proposed LOGIC

When gradient conflict occurs, forcing a single shared update limits joint progress, as no direction can advance all tasks at their full rate (see Appendix A.1). LOGIC therefore relaxes not the step but the trajectory: it removes the single-trajectory constraint and operates in parameter space. It maintains a parameter copy per task, each optimized under its own objective and loss landscape. The shared model is updated toward the curvature-weighted centroid of these task-adapted parameters, a geometric coordination step that preserves task-specific differences while guiding a stable update.

**Independent Task Updates.** LOGIC maintains $K$ task-specific parameter copies $\{\theta_k\}_{k=1}^{K}$, initialized from the shared model $\theta$. Each copy is updated independently on its own task loss:

$$\theta_k \leftarrow \mathrm{Opt}_k(\theta_k,\ \nabla_{\theta_k}\mathcal{L}_k(\theta_k))\,. \tag{1}$$

$\mathcal{L}_k$ is the loss function of task $k$. The gradient $\nabla_{\theta_k}\mathcal{L}_k(\theta_k)$ is computed with respect to $\theta_k$ using supervision from task $k$ alone, and the operator $\mathrm{Opt}_k(\cdot)$ denotes the optimizer assigned to that task. All individual tasks' optimizers use the same configuration and hyperparameters without task-specific tuning. Importantly, these task-specific parameters serve as transient optimization states prior to integration into the shared model, rather than as fully independent standalone models. Figure 2 provides an algorithm overview, and the full algorithm can be found in Appendix A.6.

**Curvature Estimation and Scale-Invariant Weighting.** To determine the relative influence of each task on the shared model, we assess the geometry of task landscapes via the Hessian trace:

$$\tau_k = \mathrm{tr}(\nabla^2_{\theta_k}\mathcal{L}_k(\theta_k)). \tag{2}$$

$\tau_k$ summarizes the local curvature of each task's loss; we use it not as a flatness certificate but as a signal of how stable a task's local geometry is relative to its own history. We compute $\tau_k$ efficiently using Hutchinson's stochastic trace estimator (Hutchinson, 1989; Avron & Toledo, 2011). To minimize overhead, these curvature estimates are updated periodically every few steps (a step

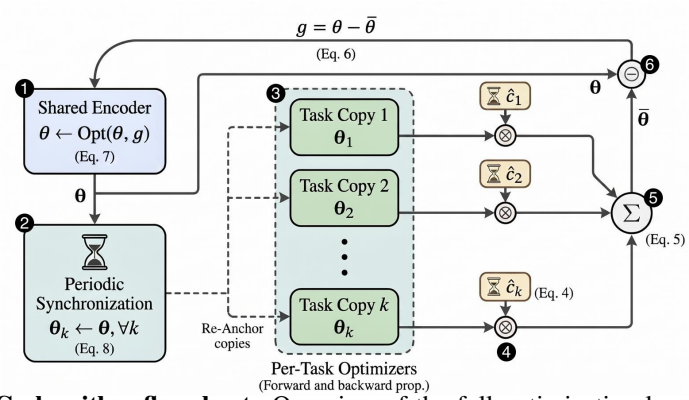



Figure 2: **LOGIC algorithm flowchart.** Overview of the full optimization loop: **(1)** the shared encoder $\theta$ is updated via pseudo-gradient $g$; **(2)** periodic synchronization re-anchors all copies to $\theta$, allowing independent task evolution between syncs; **(3)** per-task optimizers advance each $\theta_k$ via independent STL (single task learning) passes; **(4)** curvature weights $\hat{c}_k$ are re-estimated periodically and held fixed between updates; **(5)** the curvature-weighted centroid $\bar{\theta} = \sum_k \hat{c}_k \theta_k$ is formed; and **(6)** the pseudo-gradient $g = \theta - \bar{\theta}$ is injected into the shared encoder.

denotes a single mini-batch gradient update) and held fixed between them. Crucially, unlike single-task optimization where flatter minima tend to yield increased robustness to parameter perturbations (Hochreiter & Schmidhuber, 1997; Foret et al., 2021; Keskar et al., 2017; Li et al., 2018), multi-task objectives exhibit heterogeneous loss scales and fundamentally different Hessian spectra. To account for scale differences across tasks, LOGIC introduces a *task-relative curvature normalization*. The instantaneous curvature is normalized using a task-specific exponential moving average (EMA) baseline $\bar{\tau}_k$, with smoothing coefficient $\alpha$ and stabilization constant $\varepsilon$:

$$\bar{\tau}_k \leftarrow \alpha\, \bar{\tau}_k + (1-\alpha)\, |\tau_k|\,, \quad \rho_k = \frac{|\tau_k|}{\bar{\tau}_k + \varepsilon}. \tag{3}$$

The resulting metric, $\rho_k$, captures the relative curvature volatility of task $k$ with respect to its optimization history. This scale-invariant formulation prioritizes trajectories evolving in locally stable regions while suppressing transient curvature spikes and noisy updates. Accordingly, each task is assigned a weight $\hat{c}_k$ inversely proportional to $\rho_k$, smoothed through an exponential moving average:

$$\hat{c}_k \leftarrow \beta_c \hat{c}_k + (1-\beta_c)\frac{1/(\rho_k+\varepsilon)}{\sum_j 1/(\rho_j+\varepsilon)} \quad \Rightarrow \quad \sum_{k=1}^{K} \hat{c}_k = 1. \tag{4}$$

where $\beta_c$ controls the temporal smoothing of the exponential moving average, the summation over $j$ spans all $K$ tasks to enforce normalized coefficients satisfying $\sum_k \hat{c}_k = 1$, and $\varepsilon$ denotes the same numerical stabilization constant introduced in Equation (3).

**Shared Space Update Through Geometric Coordination.** The shared model is updated by geometrically coordinating the task-adapted parameters via a curvature-weighted centroid (Figure 3):

$$\bar{\theta} \;=\; \sum_{k=1}^{K} \hat{c}_k \theta_k. \tag{5}$$

The displacement from current shared parameters $\theta$ to the centroid $\bar{\theta}$ defines the pseudo-gradient $g$:

$$g \;=\; \theta - \bar{\theta}, \tag{6}$$

which provides a geometry-driven update direction in parameter space, replacing the conventional gradient aggregation. Proposition 2 (Appendix A.1.2) characterizes this displacement relative to the synchronization anchor; the injected update also carries the encoder's own motion. Either way, $g$ steers the shared model toward regions reflecting agreement among independently optimized task solutions. It is used to update the shared encoder, and the optimizer applies its standard update rule (including momentum and adaptive scaling) to move the shared model toward the centroid:

$$\theta \;\leftarrow\; \mathrm{Opt}(\theta,\; g)\,. \tag{7}$$

While task-specific heads (i.e., task decoders) are optimized through standard backpropagation, the shared encoder is entirely decoupled from direct task-gradient aggregation and updated exclusively

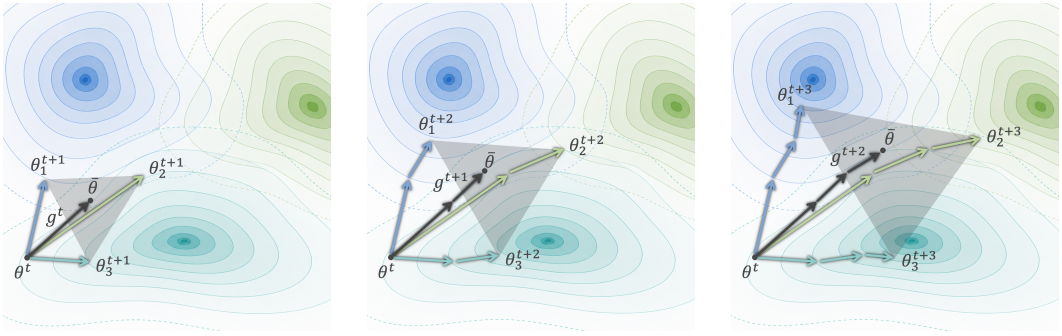



Figure 3: **Geometric illustration of task confluence (simplified).** Panels **(a)–(c)** show three update steps in which the pseudo-gradient $g$ progressively steers the shared model into a multi-task compatible region. At each step, task copies $\{\theta_k\}$ drift toward task-specific optima, while the curvature-weighted centroid $\bar{\theta}$ balances all task-adapted solutions: tasks more stable relative to their own optimization history exert stronger influence, shielding the shared model from curvature spikes and gradient noise. For clarity, optimizer effects such as momentum and adaptive scaling are omitted.

via the pseudo-gradient $g$. This separation introduces a unique optimization mechanism: task decoders learn from task-specific supervision, whereas the shared encoder is updated according to the geometric agreement among independently optimized task trajectories in parameter space.

**Task Trajectory Synchronization.** As task-specific parameters evolve, they drift apart (see Figure 3). To maintain coherence, all task copies are periodically synchronized:

$$\theta_k \leftarrow \theta \quad \forall k \in \{1, \ldots, K\}. \tag{8}$$

Synchronization enforces locality, ensuring that task-specific solutions remain comparable and their coordination remains meaningful. The synchronization interval grows linearly throughout training. In the early stages, frequent resynchronization constrains task trajectories and prevents premature divergence. As optimization progresses and the loss landscape stabilizes, the synchronization interval is gradually increased, allowing richer task-specific exploration and overcoming early-stage noise. Without synchronization, task trajectories drift apart and degrade the quality of the shared update, as confirmed by our ablation study (Section 4.7).

*Overall,* LOGIC *coordinates task-specific solutions through locality and curvature-based weighting, enabling stable multi-task optimization in which cross-task coordination acts on task trajectories rather than on per-step gradients (Propositions 1–4; see Appendix A.1 for their precise scope).*

# 4 Experiments

## 4.1 Datasets, Architectures, and Baselines

To assess the effectiveness of LOGIC under varying degrees of task conflict, we conducted an extensive set of experiments on two widely-used dense prediction benchmarks: **NYUv2** (Silberman et al., 2012), featuring semantic segmentation, depth estimation, and surface normals; and **Cityscapes** (Cordts et al., 2016), focused on segmentation and depth. Following established MTL evaluation protocols commonly adopted in prior work (Vandenhende et al., 2022), we evaluated our method using a shared ResNet-50 backbone across two widely-used yet fundamentally different multi-task learning architectures: **HPS** (Caruana, 1997), a classic hard-parameter-sharing encoder–decoder framework, and **MTAN** (Liu et al., 2019), which incorporates task-specific attention mechanisms over shared feature representations. This setup tests whether the framework generalizes across distinct feature-sharing and task-interaction mechanisms rather than relying on a particular architectural design. The consistent improvements across both architectures suggest that LOGIC enhances optimization at the task-coordination level, independent of the sharing strategy. Beyond the main text, we further evaluate LOGIC on PASCAL-Context (five tasks, Appendix A.7), PAWS-X (four NLP tasks, following the multilingual MTL protocol of Lin et al. (2022), Appendix A.9), a ViT-B/16 backbone (Appendix A.11), and training from scratch (Appendix A.10).

**Scope of Comparison.** As shown in Section 2, centralized MTO and FMTL algorithms address different MTL challenges and are rarely cross-evaluated in the literature. Consequently, we follow standard convention and restrict our evaluation to centralized MTO methods that share our motivation (gradient conflicts challenge) and setting. We compared LOGIC against 13 state-of-the-art centralized MTO baselines spanning loss weighting (Equal Weighting - EW, UW (Kendall et al., 2018), GradNorm (Chen et al., 2018), DWA (Liu et al., 2019), RLW (Lin et al., 2022), FAMO (Liu et al., 2023)) and gradient manipulation (PCGrad (Yu et al., 2020), MGDA (Sener & Koltun, 2018), CAGrad (Liu et al., 2021a), IMTL-G (Liu et al., 2021b), Nash-MTL (Navon et al., 2022), FairGrad (Ban & Ji, 2024), MoCo (Fernando et al., 2023)). All methods, including LOGIC, share an identical training protocol with no task-specific tuning. All baselines use publicly available implementations with their default hyperparameters, under an identical training protocol.

### 4.2 Evaluation Metrics

We reported per-task metrics and summarized performance using $\Delta m\%$ (Liu et al., 2021a), the average relative gap to single-task baselines (lower is better; negative values indicate that a method surpasses STL), and Mean Rank (MR), the average rank across all metrics (lower is better). Results were averaged over three shared random seeds across all experiments. We also report $\Delta m\%$ standard deviations; LOGIC achieves the lowest among compared methods in every table.

### 4.3 NYUv2 Benchmark

Table 1: **NYUv2: three-task scene understanding under HPS.** Results are averaged over three random seeds (but the same for all methods), each trained for 200 epochs. The backbone was the widely-used ResNet-50 (He et al., 2016), pretrained on ImageNet (Deng et al., 2009). $\uparrow$ / $\downarrow$ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all NYUv2 metrics (lower is better). **Bold** denotes the best result per column; underline denotes the second best.

| Method | Segmentation | | Depth | | Surface Normal | | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | mIoU↑ | PixAcc↑ | AbsErr↓ | RelErr↓ | Mean↓ | Median↓ | $< 11.25^\circ \uparrow$ | $< 22.5^\circ \uparrow$ | $< 30^\circ \uparrow$ | MR↓ | $\Delta m\% \downarrow$ |
| STL | 54.43 | 76.06 | 0.3668 | 0.1493 | 21.1760 | 14.4351 | 40.44 | 67.00 | 76.91 | – | – |
| EW | 55.52 | 76.74 | 0.3595 | 0.1454 | 22.2162 | 15.7010 | 37.31 | 64.18 | 74.82 | 9.22 | 2.32±1.690 |
| UW (Kendall et al., 2018) | 56.14 | 77.16 | 0.3578 | 0.1449 | 22.0101 | 15.4920 | 37.78 | 64.67 | 75.22 | 6.17 | 1.50±2.007 |
| GradNorm (Chen et al., 2018) | 55.69 | 76.90 | 0.3587 | 0.1454 | 21.8366 | 15.3207 | 38.19 | 65.09 | 75.59 | 6.44 | 1.24 ±1.140 |
| DWA (Liu et al., 2019) | 55.55 | 76.77 | 0.3593 | 0.1455 | 22.2440 | 15.7062 | 37.37 | 64.11 | 74.73 | 9.44 | 2.33 ±1.667 |
| MGDA (Sener & Koltun, 2018) | 50.97 | 73.91 | 0.3858 | 0.1553 | 21.2197 | **14.4734** | **40.34** | 66.95 | 76.87 | 6.56 | 2.14 ±2.848 |
| MoCo (Fernando et al., 2023) | 46.59 | 70.47 | 0.4111 | 0.1657 | 24.2308 | 17.4831 | 33.74 | 59.84 | 70.82 | 14.00 | 12.84 ±0.999 |
| PCGrad (Yu et al., 2020) | 55.30 | 76.34 | 0.3601 | 0.1468 | 22.3560 | 15.8864 | 36.92 | 63.78 | 74.54 | 11.00 | 2.97 ±1.950 |
| CAGrad (Liu et al., 2021a) | 55.96 | 77.25 | 0.3566 | 0.1440 | 21.3992 | 14.8419 | 39.31 | 66.25 | 76.50 | 3.89 | −0.27 ±1.655 |
| IMTL-G (Liu et al., 2021b) | 55.57 | 76.85 | 0.3608 | 0.1454 | 21.4872 | 14.9271 | 39.12 | 66.03 | 76.32 | 6.56 | 0.33 ±0.871 |
| Nash-MTL (Navon et al., 2022) | 55.48 | 76.81 | 0.3562 | 0.1444 | 22.0277 | 15.4653 | 37.92 | 64.65 | 75.16 | 7.11 | 1.57 ±0.748 |
| RLW (Lin et al., 2022) | 54.47 | 75.75 | 0.3649 | 0.1473 | 22.4117 | 15.9150 | 36.81 | 63.68 | 74.42 | 12.11 | 3.53 ±2.038 |
| FAMO (Liu et al., 2023) | 52.93 | 75.10 | 0.3825 | 0.1557 | 21.3582 | 14.7150 | 39.67 | 66.44 | 76.57 | 7.33 | 2.06 ±2.326 |
| FairGrad (Ban & Ji, 2024) | 55.78 | 77.10 | 0.3583 | 0.1449 | 21.3554 | 14.7848 | 39.41 | 66.40 | 76.60 | 3.94 | −0.22 ±0.658 |
| **LOGIC (Ours)** | **57.16** | **78.18** | **0.3380** | **0.1338** | **21.0387** | 14.5507 | 39.99 | **67.00** | **77.12** | **1.22** | **−2.78 ±0.349** |

As shown in Table 1, LOGIC frequently exceeded STL performance across multiple task groups and achieved the best overall results on NYUv2, obtaining the leading $\Delta m\%$ ($-2.78$) and Mean Rank (1.22). Although LOGIC achieved performance comparable to STL on two surface normal metrics, it clearly outperformed STL across all remaining tasks and metrics, demonstrating that geometric coordination enables substantial overall gains without compromising performance on challenging objectives. In contrast, even strong baselines such as CAGrad (Liu et al., 2021a) and FairGrad (Ban & Ji, 2024) delivered only marginal improvements over STL, whereas several competing methods deteriorated substantially under task conflict (performance of competing methods has been consistently validated in prior literature). Notably, the largest gains of LOGIC were observed in depth estimation, a task known to exhibit severe interference under shared gradient optimization (Yu et al., 2020; Standley et al., 2020). These results highlight the key advantage of allowing task trajectories to evolve independently and accumulate task-specific optimization structure before being coordinated geometrically in parameter space, particularly under strong inter-task conflict. LOGIC also achieves the lowest cross-seed standard deviation ($\pm 0.349$) among all methods.

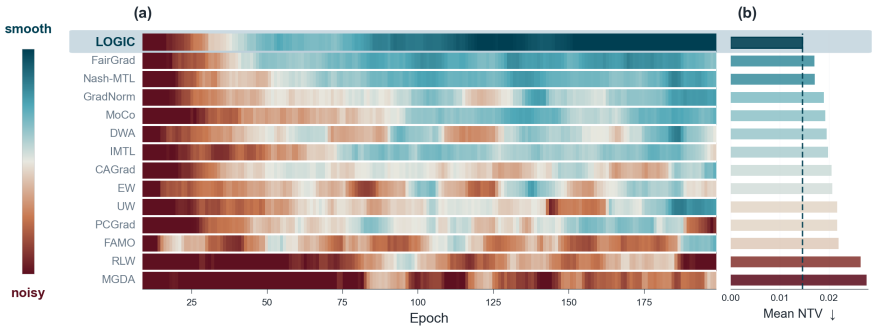



Figure 4: **Training stability comparison across MTL weighting methods on NYUv2.** We define Normalized Total Variation (NTV) as per-epoch oscillation: NTV $= \frac{1}{T-1}\sum_{t=1}^{T-1}\frac{|y_{t+1}-y_t|}{\bar{y}}$, where $T$ is the total number of training epochs, $y_t$ denotes the metric value at epoch $t$, and $\bar{y} = \frac{1}{T}\sum_t |y_t|$ is the mean absolute value of the metric. We render NTV, scale-free across tasks; all scores were averaged over three tasks and three random seeds. **(a)** Sliding-window NTV over training epochs, color-coded from smooth (dark teal) to noisy (dark red). **(b)** Full-training mean NTV per method (lower is better). **LOGIC** achieves the lowest NTV by a clear margin, demonstrating consistently smooth convergence where competing methods exhibit persistent oscillations.

LOGIC's gains in Table 1 are accompanied by the smoothest training dynamics of all methods, with the lowest mean NTV (Figure 4). LOGIC is the only method that achieves both the best $\Delta m\%$ and the most stable optimization, so its reported results are less sensitive to the choice of evaluation epoch. Appendix A.4 distinguishes the smoothness reported here from generic slow-weight averaging. Under strong task conflict, gradient-based methods exhibited pronounced oscillations and unstable convergence, whereas LOGIC maintained smooth and well-behaved training trajectories throughout optimization. Figure 4(a) further shows that this stability persists consistently across training epochs rather than appearing only near convergence. Consequently, LOGIC achieved steady and predictable performance improvements over time, remaining largely insensitive to the choice of stopping epoch, in contrast to the brittle behavior observed in competing approaches.

To isolate LOGIC's pure contribution, we additionally evaluated NYUv2 under HPS, trained from scratch. Even in this challenging setting, LOGIC outperformed (Table 8 of Appendix A.10).

### 4.4 CITYSCAPES BENCHMARK

Table 2: **Cityscapes: two-task scene understanding under HPS.** Results are averaged over three random seeds, each trained for 200 epochs. The backbone was the widely-used ResNet-50 (He et al., 2016), pretrained on ImageNet (Deng et al., 2009). ↑ / ↓ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all Cityscapes metrics (lower is better). **Bold** denotes the best result per column; underline denotes the second best.

| Method | Segmentation | | Depth | | Overall | |
|---|---|---|---|---|---|---|
| | mIoU↑ | PixAcc↑ | AbsErr↓ | RelErr↓ | MR↓ | $\Delta m\%$ ↓ |
| STL | 69.84 | 91.91 | 0.0136 | 50.7379 | – | – |
| EW | 69.85 | 91.85 | 0.0131 | 48.9027 | 9.25 | $-1.81 \pm 1.061$ |
| UW (Kendall et al., 2018) | 70.19 | 91.99 | 0.0128 | 48.9324 | 4.13 | $-2.51 \pm 0.989$ |
| GradNorm (Chen et al., 2018) | 69.90 | 91.88 | 0.0128 | 48.2670 | 6.13 | $-2.70 \pm 1.214$ |
| DWA (Liu et al., 2019) | 70.02 | 91.94 | 0.0129 | 48.8359 | 6.00 | $-2.30 \pm 1.214$ |
| MGDA (Sener & Koltun, 2018) | 67.68 | 91.04 | 0.0134 | 48.0360 | 10.75 | $-0.69 \pm 1.637$ |
| MoCo (Fernando et al., 2023) | 67.83 | 91.05 | 0.0146 | 51.8077 | 13.00 | $3.32 \pm 1.687$ |
| PCGrad (Yu et al., 2020) | 70.13 | 91.98 | 0.0130 | 49.2960 | 6.00 | $-1.94 \pm 1.458$ |
| CAGrad (Liu et al., 2021a) | 69.98 | 91.92 | 0.0126 | 51.1223 | 6.75 | $-1.70 \pm 1.497$ |
| IMTL-G (Liu et al., 2021b) | 70.13 | 91.94 | **0.0125** | 48.3689 | 3.75 | $-3.30 \pm 0.697$ |
| Nash-MTL (Navon et al., 2022) | 70.02 | 91.95 | 0.0130 | 51.8662 | 7.75 | $-0.62 \pm 1.611$ |
| RLW (Lin et al., 2022) | 69.68 | 91.75 | 0.0133 | 50.5493 | 11.25 | $-0.54 \pm 1.556$ |
| FAMO (Liu et al., 2023) | 69.89 | 91.83 | 0.0128 | 47.8937 | 6.50 | $-2.87 \pm 1.122$ |
| FairGrad (Ban & Ji, 2024) | 69.88 | 91.83 | 0.0142 | 59.4049 | 11.88 | $5.38 \pm 0.751$ |
| **LOGIC (Ours)** | **70.63** | **92.12** | 0.0128 | **47.5183** | **1.88** | $\mathbf{-3.40 \pm 0.348}$ |

Table 2 shows that LOGIC maintains superior performance on Cityscapes, achieving the best performance across all metrics (except for AbsErr, where it shows competitive results to other baselines). Task conflict in Cityscapes is known to be substantially lower than in NYUv2 (Wang et al., 2021); nevertheless, LOGIC consistently surpassed strong baselines. Appendix A.2 compares the task geometry of the two benchmarks. These results demonstrate that the benefits of LOGIC are not confined to highly conflicting regimes such as NYUv2, but extend robustly to other multi-task settings. In contrast, competing gradient-based methods exhibited noticeable sensitivity to the underlying benchmark and conflict regime: methods such as CAGrad (Liu et al., 2021a) and FairGrad (Ban & Ji, 2024) performed strongly on NYUv2, whereas IMTL-G (Liu et al., 2021b) was more competitive on Cityscapes. LOGIC, however, remained consistently robust across both high-conflict and low-conflict scenarios, suggesting greater stability under varying task geometries and optimization conditions. This is further supported by the lowest cross-seed standard deviation ($\pm 0.348$).

### 4.5 NYUv2 - MTAN Architecture

Table 3: **NYUv2: three-task scene understanding under MTAN.** Results are averaged over three random seeds, each trained for 300 epochs, using ResNet-50 (He et al., 2016), pretrained on ImageNet (Deng et al., 2009). We evaluate the 9 top-performing baselines for comparison. $\uparrow$ / $\downarrow$ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all NYUv2 metrics (lower is better). **Bold** denotes the best result per column; underline the second best.

| Method | Segmentation | | Depth | | Surface Normal | | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | mIoU↑ | PixAcc↑ | AbsErr↓ | RelErr↓ | Mean↓ | Median↓ | $< 11.25^\circ \uparrow$ | $< 22.5^\circ \uparrow$ | $< 30^\circ \uparrow$ | MR↓ | $\Delta m\% \downarrow$ |
| STL | 53.47 | 75.16 | 0.3758 | 0.1513 | 21.8267 | 14.8194 | 39.54 | 65.94 | 75.87 | – | – |
| PCGrad (Yu et al., 2020) | 54.89 | 76.35 | 0.3563 | 0.1426 | 22.1773 | 15.5964 | 37.68 | 64.32 | 74.86 | 8.06 | 0.02 ±1.414 |
| CAGrad (Liu et al., 2021a) | 54.86 | 76.47 | 0.3580 | 0.1455 | 21.7536 | 15.1240 | 38.65 | 65.48 | 75.80 | 6.22 | −0.91 ±1.889 |
| IMTL-G (Liu et al., 2021b) | 55.57 | 76.88 | 0.3527 | 0.1421 | 21.5395 | 14.9801 | 38.89 | 65.96 | 76.29 | 3.56 | −1.96 ±0.516 |
| Nash-MTL (Navon et al., 2022) | 55.02 | 76.46 | 0.3526 | 0.1431 | 22.0867 | 15.4286 | 38.05 | 64.68 | 75.10 | 6.44 | −0.47 ±0.646 |
| FairGrad (Ban & Ji, 2024) | **55.60** | 76.98 | 0.3503 | 0.1426 | 21.4016 | 14.7514 | 39.46 | 66.49 | 76.66 | 2.50 | −2.56 ±0.488 |
| FAMO (Liu et al., 2023) | 50.70 | 73.45 | 0.3961 | 0.1570 | 21.7997 | 14.9919 | 39.11 | 65.69 | 75.83 | 7.00 | 2.13 ±2.974 |
| GradNorm (Chen et al., 2018) | 54.29 | 76.20 | 0.3554 | 0.1432 | 21.9567 | 15.4015 | 38.07 | 64.86 | 75.37 | 7.00 | −0.35 ±1.019 |
| MGDA (Sener & Koltun, 2018) | 49.83 | 72.87 | 0.3873 | 0.1569 | 21.3842 | 14.5128 | **40.28** | 66.69 | 76.59 | 5.33 | 0.95 ±2.431 |
| UW (Kendall et al., 2018) | 55.28 | 76.44 | 0.3543 | 0.1443 | 22.1957 | 15.4581 | 38.01 | 64.54 | 74.93 | 7.44 | −0.24 ±1.903 |
| **LOGIC (Ours)** | 55.25 | **77.10** | **0.3295** | **0.1327** | **21.0494** | **14.5007** | 40.16 | **67.07** | **77.09** | **1.44** | **−4.57 ± 0.388** |

Table 3 shows that LOGIC preserved the same superiority observed under HPS also under the stronger MTAN architecture: it consistently delivered the best overall performance (MR and $\Delta m\%$) while maintaining improvements across all task groups. These observations indicate that the advantages of LOGIC do not depend on a specific sharing architecture/mechanism, but stem from its ability to resolve cross-task conflicts at the optimization level. Even in MTAN, where task-specific attention mitigates interference at the feature level, the core principles of LOGIC (e.g., individual trajectories and geometric coordination in parameter space) yield complementary benefits, leading to superior, well-balanced performance. Cross-seed standard deviation remains the lowest ($\pm 0.388$).

### 4.6 Sensitivity to Hyperparameter Tuning

LOGIC uses a single fixed configuration across all experiments, identical across architectures and benchmarks without task-specific tuning. Four hyperparameters control the framework: Hutchinson probes (number of random vectors for curvature estimation), curvature update interval (frequency of Hessian trace refresh), task-copy learning rate (scaled from the main learning rate), and synchronization schedule (re-anchoring frequency of task copies). All hyperparameters demonstrated highly stable behavior, with maximal variation remaining below 0.5 mIoU, while depth and surface normal metrics exhibited similarly negligible sensitivity across configurations (see details in Appendix A.5).

### 4.7 Ablation Study

Table 4 isolates the contribution of each component of LOGIC. *Synchronization* and *curvature-aware weighting* play complementary roles. Without synchronization, independently trained copies

Table 4: Ablation study of LOGIC on NYUv2, averaged over three random seeds. Each row activates a subset of LOGIC's components (✓ = enabled). We report mean IoU (segmentation), absolute error (depth), and mean angle distance (surface normals). $\Delta m\%$ is the average per-task gap relative to single-task baselines (lower is better; negative means the multi-task model surpasses single-task). **Bold** denotes best per column; underline denotes second best.

| | Components | | | | Per-task metrics | | | |
|---|---|---|---|---|---|---|---|---|
| **Configuration** | Pseudo Gradient | Sync | Curv | Rel | Seg. mIoU ↑ | Depth Abs Err ↓ | Normals Mean ↓ | $\Delta m\%$ ↓ |
| Joint training (EW) | | | | | 55.52 | 0.3595 | 22.2162 | 2.32 ±1.690 |
| Equal-weight centroid | ✓ | | | | 31.81 | 0.5405 | 25.5352 | 29.52 ±2.056 |
| Absolute curvature (no sync) | ✓ | | ✓ | | 51.73 | 0.3696 | 23.6437 | 8.71 ±1.387 |
| Relative curvature (no sync) | ✓ | | ✓ | ✓ | 53.26 | 0.3908 | 23.5201 | 8.74 ±1.238 |
| Synchronization | ✓ | ✓ | | | <u>56.85</u> | 0.3404 | 21.1484 | −2.09 ±0.353 |
| Absolute curvature | ✓ | ✓ | ✓ | | 56.21 | <u>0.3389</u> | **20.9482** | <u>−2.45 ±0.311</u> |
| **LOGIC (full)** | ✓ | ✓ | ✓ | ✓ | **57.16** | **0.3380** | <u>21.0387</u> | **−2.78 ± 0.349** |

drift apart, and averaging them with equal weights degrades performance sharply ($\Delta m\% = +29.52$). Curvature-aware weighting counteracts the effect of this drift, reducing the degradation to $+8.71$ by down-weighting copies with higher curvature. Synchronization keeps the copies close enough for their centroid to be informative: every synchronized configuration surpasses single-task learning, while no unsynchronized one does, and with uniform weights it already reaches $-2.09$. Combining the two yields the best result ($-2.78$): synchronization bounds the drift, and curvature weighting decides how the copies are combined within each window. This additional $0.69$ is far from a minor refinement: it is more than twice the $0.27$ gain over single-task learning achieved by CAGrad, the strongest baseline in Table 1, and it comes where further gains are hardest to obtain.

### 4.8 Computational and Memory Considerations

While LOGIC maintains $K$ copies of the encoder's parameters alongside the shared encoder, its computational overhead remains tightly bounded. Because both LOGIC and gradient-manipulation baselines execute $K$ backward passes per iteration (one per task), LOGIC's additional runtime is confined to the forward pass ($K+1$ passes vs. 1), with infrequent curvature estimation contributing negligible runtime. On the 3-task NYUv2, LOGIC requires $2.7\times$ the epoch time of equal-weighting against $\approx 3.1\times$ for gradient-manipulation baselines, establishing it as the fastest conflict-resolution method while also attaining the largest improvement over single-task baselines, and placing it on the cost–accuracy Pareto front (Appendix A.3). This advantage expands with task count: across 1 to 5 tasks on PASCAL-Context, LOGIC scales to only $3.45\times$ at 5 tasks, whereas gradient-manipulation methods surge to $\approx 5\times$ (Appendix A.8). Regarding footprint, parameter copies do not translate to a naive $(1+K)\times$ memory penalty; because peak training memory is dominated by feature activations rather than weights or optimizer states, LOGIC incurs a $1.79\times$ peak memory overhead on NYUv2 that remains virtually flat as tasks scale. Crucially, all trade-offs are restricted to training; at inference, LOGIC collapses to a single standard encoder with no added latency or memory overhead.

## 5 Discussion

Multi-task learning has long been framed as an inevitability of compromise. Because competing tasks pull shared parameters in conflicting directions, traditional methods settle for a direction that fully satisfies none. LOGIC challenges this foundational assumption. By letting each task freely navigate its own loss landscape before coordinating the resulting solutions, LOGIC reaches a shared model unattainable through gradient compromise. Disagreement between tasks ceases to be an obstacle and becomes the primary signal guiding the update: what tasks reveal by diverging is where their landscapes are mutually well-conditioned. This shift enables a single model to consistently surpass its single-task baselines without per-task tuning. It incurs a memory cost strictly during training, leaving deployment overhead untouched. More broadly, it suggests that the performance ceiling in multi-task learning has been imposed by our optimization constraints rather than the tasks themselves. Across seven settings spanning vision and language benchmarks and three convolutional and transformer-based backbones, LOGIC consistently attains the best performance, smoothest training, and lowest cross-seed variance.

### AI USE STATEMENT

Generative AI tools were used only to draft and edit text for readability and to help identify related literature. They were not used for any subtask requiring disclosure under the ICLR policy: theoretical development, mathematical claims or proofs, hypotheses, methodology or experiment design, implementation, data processing, and interpretation of results. The authors reviewed all AI-assisted text and take full responsibility for the content of this paper.

### ETHICS STATEMENT

This work uses only established, publicly available benchmarks (NYUv2, Cityscapes, PASCAL-Context and PAWS-X), with no human subjects, personally identifiable information, or newly collected data. We are aware of no ethical concerns specific to the proposed method.

### REPRODUCIBILITY STATEMENT

The full algorithm is given in Appendix A.6, all assumptions and proofs in Appendix A.1, and the hyperparameter configuration, fixed across all experiments, in Appendix A.5. Datasets, backbones, baselines and metrics are described in Section 4. All reported numbers are averaged over three shared random seeds with the corresponding standard deviations. The source code will be released publicly upon acceptance.

## REFERENCES

Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2):8:1–8:17, 2011. doi: 10.1145/1944345.1944349.

Lu Bai, Abhishek Gupta, and Yew-Soon Ong. Multi-task learning with multi-task optimization, 2024. arXiv:2403.16162.

Hao Ban and Kaiyi Ji. Fair resource allocation in multi-task learning. In *Int. Conf. Mach. Learn.*, pp. 2715–2731, 2024.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient Normalization for adaptive loss balancing in deep multitask networks. In *Int. Conf. Mach. Learn.*, 2018.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Adv. Neural Inform. Process. Syst.*, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conf. North Am. Chapter Assoc. Comput. Linguistics*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models, 2024. URL https://arxiv.org/abs/2311.08105.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Heshan Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *Int. Conf. Learn. Represent.*, 2023.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *Adv. Neural Inform. Process. Syst.*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Int. Conf. Mach. Learn.*, pp. 1126–1135, 2017.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Int. Conf. Learn. Represent.*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Int. Conf. Mach. Learn.*, pp. 4411–4421, 2020.

M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics – Simulation and Computation*, 18(3):1059–1076, 1989.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *Int. Conf. Learn. Represent.*, 2023.

Junguang Jiang, Baixu Chen, Junwei Pan, Ximei Wang, Dapeng Liu, Jie Jiang, and Mingsheng Long. ForkMerge: Mitigating negative transfer in auxiliary-task learning. In *Adv. Neural Inform. Process. Syst.*, 2023.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Int. Conf. Mach. Learn.*, 2020.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning. In *Adv. Neural Inform. Process. Syst.*, 2022.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Adv. Neural Inform. Process. Syst.*, 2018.

Zhenyu Liao and Michael W. Mahoney. Hessian eigenspectra of more realistic nonlinear models. In *Adv. Neural Inform. Process. Syst.*, 2021.

Baijiong Lin and Yu Zhang. LibMTL: A Python library for deep multi-task learning. *Journal of Machine Learning Research*, 24(209):1–7, 2023.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W.H. Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qingfu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, 2019.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Adv. Neural Inform. Process. Syst.*, 2021a.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. In *Adv. Neural Inform. Process. Syst.*, 2023.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *Int. Conf. Learn. Represent.*, 2021b.

Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1851–1860, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, 2015.

Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging. In *Adv. Neural Inform. Process. Syst.*, 2022.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Int. Conf. Artif. Intell. Stat.*, 2017.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 891–898, 2014.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *Int. Conf. Mach. Learn.*, 2022.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018. arXiv:1803.02999.

Kumar Kshitij Patel, Margalit Glasgow, Ali Zindari, Lingxiao Wang, Sebastian U. Stich, Ziheng Cheng, Nirmit Joshi, and Nathan Srebro. The limits and potentials of local sgd for distributed heterogeneous learning with intermittent communication. In *Conference on Learning Theory (COLT)*, pp. 4115–4157, 2024.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. arXiv:1706.05098.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Adv. Neural Inform. Process. Syst.*, 2018.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Eur. Conf. Comput. Vis.*, 2012.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4424–4434, 2017.

Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *Int. Conf. Mach. Learn.*, 2020.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *Int. Conf. Learn. Represent.*, 2019.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3614–3633, 2022. doi: 10.1109/TPAMI.2021.3054719.

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 (ICLR)*, 2021.

Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? In *Adv. Neural Inform. Process. Syst.*, 2022.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *Conf. Empirical Methods Nat. Lang. Process.*, pp. 3687–3692, 2019. doi: 10.18653/v1/D19-1382.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Adv. Neural Inform. Process. Syst.*, 2020.

Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: $k$ steps forward, 1 step back. In *Adv. Neural Inform. Process. Syst.*, 2019a.

Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In *Conf. North Am. Chapter Assoc. Comput. Linguistics*, pp. 1298–1308, 2019b.

# A Appendix

## A.1 Theoretical Analysis

**Purpose and roadmap.** This section identifies precisely what LOGIC computes and what distinguishes it from gradient aggregation. It establishes five results. First, the best worst-case descent available to *any* single shared direction equals the distance from the origin to the convex hull of the task gradients (Proposition 1); this ceiling is met by existing aggregation methods, and it applies to LOGIC's own encoder step, which locates the remaining freedom in the trajectory rather than in the step. Second, the pseudo-gradient is a curvature-weighted aggregate perturbed by an error bounded non-asymptotically, and at a synchronization interval of $T = 1$ it coincides with weighted gradient aggregation exactly (Proposition 2); the interval is therefore an explicit dial between aggregation and trajectory coordination. Third, and centrally, for $T > 1$ the update acquires a component

$$D = \mathrm{Cov}_{\hat{c}}(H, g) = \sum_k \hat{c}_k (H_k - \bar{H})(g_k - \bar{g}),$$

a weighted covariance between task curvature and task gradient that *no* single-trajectory method can reproduce, with fixed weights (Proposition 3) or with weights re-solved at every step (Proposition 4). This is the formal content of the claim that LOGIC coordinates trajectories rather than gradients: the term vanishes identically when tasks agree in gradient or in curvature, and is available only to a method that evaluates gradients at $K$ distinct points. Fourth, the noise in the anchor-relative direction decreases as $\sigma^2/T$ while the drift bound increases, which identifies the trade-off the synchronization schedule navigates (Proposition 5). Fifth, the curvature weights vary slowly enough within a window to license treating them as fixed (Proposition 6).

**What we do not claim.** We prove no optimality result and no unconditional convergence result. Multi-task learning has no canonical scalar objective: convergence statements are relative to a chosen scalarization, and different scalarizations yield incomparable solutions. The scalarization-free alternative, convergence to Pareto-stationary points, is not established here: the weights $\hat{c}$ evolve during training, so the update does not descend a fixed objective, and we do not assume a common descent direction across tasks. Accordingly, the weighted aggregate $\mathcal{L}_{\hat{c}}$ appears below only as a proxy whose per-step descent (for the current $\hat{c}$) is a consistency property of the update, not as the objective of multi-task learning. Similarly, we characterize the term $D$, bound it, and identify when it vanishes, but do not claim it improves solution quality: the propositions show that this capacity exists and is unavailable to single-trajectory methods with nonnegative task weights, and Appendix A.2.5 shows that it forms a substantial part of the update throughout training. The empirical case for LOGIC rests on the experiments: across seven settings spanning four benchmarks, three backbones, and both vision and language tasks, LOGIC attains the best overall performance and the lowest cross-seed variance among the compared methods (Section 4 and Appendices A.7, A.9, A.10 and A.11), and its gains are not reproduced by generic slow-weight averaging (Appendix A.4). Appendix A.1.6 lists every point at which our assumptions and the implemented method diverge.

**Setup and notation.** Let $\{\mathcal{L}_k\}_{k=1}^K$ be task losses on shared parameters $\theta \in \mathbb{R}^n$, and let $\eta > 0$ be the step size used by each per-task optimizer. Write $g_k = \nabla \mathcal{L}_k(\theta)$ and $H_k = \nabla^2 \mathcal{L}_k(\theta)$. For weights $w$ in the simplex $\Delta^{K-1} = \{w \in \mathbb{R}^K_{\geq 0} : \sum_k w_k = 1\}$ set

$$\mathcal{L}_w = \sum_k w_k \mathcal{L}_k, \qquad g_w = \nabla \mathcal{L}_w(\theta) = \sum_k w_k g_k, \qquad H_w = \nabla^2 \mathcal{L}_w(\theta) = \sum_k w_k H_k.$$

The curvature weights of Section 3 are $\hat{c} \in \Delta^{K-1}$, with $\hat{c}_k > 0$ for every $k$ since $1/(\rho_k + \varepsilon) > 0$ and the outer EMA starts uniform. We abbreviate $\bar{g} = g_{\hat{c}}$, $\bar{H} = H_{\hat{c}}$. Let $\theta_k(t)$ be the $k$-th task copy $t$ steps after a synchronization, so $\theta_k(0) = \theta$; let $\bar{\theta}(T) = \sum_k \hat{c}_k \theta_k(T)$ and $g(T) = \theta - \bar{\theta}(T)$. Finally

$$\zeta^2_{\hat{c}} = \sum_k \hat{c}_k \|g_k - \bar{g}\|^2$$

measures weighted inter-task gradient disagreement at $\theta$.

**Assumptions.**

(A1) Each $\mathcal{L}_k$ is twice continuously differentiable and $\beta$-smooth: $\|\nabla^2 \mathcal{L}_k(\cdot)\|_{\mathrm{op}} \le \beta$. Every $\mathcal{L}_w$, $w \in \Delta^{K-1}$, is then $\beta$-smooth as well.

(A2) Gradients are bounded by $G$ in the region of interest.

(A3) Each $\nabla^2 \mathcal{L}_k$ is $L$-Lipschitz in the region of interest. This is used only to quantify the remainders in Propositions 3 and 4.

(A4) Propositions 1–4 are stated for deterministic gradients; Proposition 5 treats sampling noise explicitly and is the only result in which it appears.

(A5) Propositions 2–4 are stated for plain gradient steps. Appendix A.1.6 records which conclusions survive under the optimizers actually used and which do not; we do not assert that adaptivity and momentum leave the analysis unchanged.

(A6) The weights $\hat{c}$ are treated as constant within a window of length $T$. Proposition 6 bounds the error this introduces rather than assuming it away.

### A.1.1 The single-step ceiling, and why it applies to us too

**Proposition 1 (worst-case descent of a shared direction).** *Let $C = \mathrm{conv}\{g_1, \dots, g_K\}$ and let $g^\star$ be its minimum-norm element. Then*

$$\sup_{\|d\| \le 1} \min_{k \in [K]} \big(-\langle g_k, d\rangle\big) = \|g^\star\| = \mathrm{dist}(0, C),$$

*attained for $g^\star \neq 0$ at $d^\star = -g^\star/\|g^\star\|$.*

**Proof.** *Upper bound.* Pick $\lambda \in \Delta^{K-1}$ with $\sum_k \lambda_k g_k = g^\star$. For $\|d\| \le 1$,

$$\min_k \big(-\langle g_k, d\rangle\big) \le \sum_k \lambda_k \big(-\langle g_k, d\rangle\big) = -\langle g^\star, d\rangle \le \|g^\star\|.$$

*Attainment.* For $g^\star \neq 0$, $g^\star$ is the Euclidean projection of $0$ onto the closed convex set $C$, so $\langle g^\star, y - g^\star\rangle \ge 0$ for all $y \in C$; taking $y = g_k$ gives $\langle g^\star, g_k\rangle \ge \|g^\star\|^2$. Hence for $d^\star = -g^\star/\|g^\star\|$ and every $k$, $-\langle g_k, d^\star\rangle = \langle g_k, g^\star\rangle/\|g^\star\| \ge \|g^\star\|$, matching the bound. If $g^\star = 0$ then $0 \in C$ and the supremum is $0$. □

**Corollary 1.1 (two tasks).** For $K = 2$ with $\|g_1\| = \|g_2\| = G$ at angle $\omega \in (\pi/2, \pi)$, the minimizer is $g^\star = \frac{1}{2}(g_1 + g_2)$ and the ceiling is $G\cos(\omega/2)$.

**Proof.** On the segment, $h(t) = \|t g_1 + (1-t) g_2\|^2 = G^2[t^2 + (1-t)^2 + 2t(1-t)\cos\omega]$ has $h'(t) = 2G^2(2t-1)(1-\cos\omega)$, vanishing only at $t = \frac{1}{2}$ and a minimum since $\cos\omega < 1$; then $\|g^\star\|^2 = \frac{G^2}{2}(1 + \cos\omega) = G^2\cos^2(\omega/2)$. □

**Reading.** The proposition concerns *any* single direction taken from $\theta$, and this includes LOGIC's own encoder update $\theta \leftarrow \theta - \gamma g$. Three consequences follow.

The ceiling is not a deficiency of gradient-aggregation methods. MGDA (Sener & Koltun, 2018) computes precisely the minimizer $g^\star$, and CAGrad (Liu et al., 2021a) solves a regularized relaxation of the same problem; these methods operate *at* the ceiling, and no reweighting or projection scheme can pass it. LOGIC does not pass it either: its encoder step is a single direction subject to the same bound, and we make no claim that it escapes gradient conflict within a single update.

What the proposition bounds is *greedy, instantaneous, worst-case* descent. Greedy per-step optimality does not imply optimality of a trajectory, a gap familiar already in single-objective optimization, where steepest descent is greedily optimal at every step yet traverses ill-conditioned landscapes poorly. Proposition 1 therefore localizes the remaining freedom exactly: not in the choice of a step, which is settled, but in the sequence of points at which task gradients are evaluated. Propositions 2 through 4 show that this freedom is real and quantify it.

### A.1.2 WHAT THE PSEUDO-GRADIENT IS

**Proposition 2 (non-asymptotic characterization of $g(T)$).** *Under (A1)–(A2), write $g(T)/(\eta T) = \bar{g} + e(T)$. Then*

$$\|e(T)\| \le G\,\Phi(T), \qquad \Phi(T) := \frac{(1+\eta\beta)^T - 1}{\eta\beta T} - 1.$$

*Moreover:*

- *(a) $\Phi(1) = 0$, so at $T = 1$ the identity $g(1) = \eta \sum_k \hat{c}_k g_k$ holds exactly, with no remainder.*
- *(b) $\Phi$ is strictly increasing in $T$, with $\Phi(T) = \frac{1}{2}\eta\beta(T-1) + O((\eta\beta T)^2)$.*
- *(c) If $G\Phi(T) < \|\bar{g}\|$ then $\langle g(T), \nabla\mathcal{L}_{\hat{c}}(\theta)\rangle > 0$, and for every $0 < \gamma < \big(\|\bar{g}\| - G\Phi(T)\big)/\big(2\beta\eta T\|\bar{g}\|\big)$ the update $\theta^+ = \theta - \gamma g(T)$ satisfies $\mathcal{L}_{\hat{c}}(\theta^+) < \mathcal{L}_{\hat{c}}(\theta)$.*

**Proof.** Unrolling $\theta_k(t+1) = \theta_k(t) - \eta\nabla\mathcal{L}_k(\theta_k(t))$ from $\theta_k(0) = \theta$ gives $\theta - \theta_k(T) = \eta\sum_{t=0}^{T-1}\nabla\mathcal{L}_k(\theta_k(t))$, so

$$\frac{g(T)}{\eta T} = \sum_k \hat{c}_k \frac{1}{T}\sum_{t=0}^{T-1}\nabla\mathcal{L}_k(\theta_k(t)), \qquad e(T) = \sum_k \hat{c}_k \frac{1}{T}\sum_{t=0}^{T-1}\big(\nabla\mathcal{L}_k(\theta_k(t)) - g_k\big). \tag{9}$$

Let $a_k(t) = \|\theta_k(t) - \theta\|$. By (A1)–(A2), $\|\nabla\mathcal{L}_k(\theta_k(t))\| \le G + \beta a_k(t)$, hence $a_k(t+1) \le (1+\eta\beta)a_k(t) + \eta G$ with $a_k(0) = 0$, giving $a_k(t) \le \frac{G}{\beta}[(1+\eta\beta)^t - 1]$. Using $\|\nabla\mathcal{L}_k(\theta_k(t)) - g_k\| \le \beta a_k(t)$ in (9),

$$\|e(T)\| \le \frac{\beta}{T}\sum_{t=0}^{T-1}\frac{G}{\beta}\big[(1+\eta\beta)^t - 1\big] = \frac{G}{T}\left[\frac{(1+\eta\beta)^T - 1}{\eta\beta} - T\right] = G\Phi(T).$$

*(a)* $\Phi(1) = 0$ by direct substitution, and (9) at $T = 1$ gives the exact identity. *(b)* $\Phi(T) + 1$ is the arithmetic mean of the strictly increasing sequence $\{(1+\eta\beta)^t\}_{t=0}^{T-1}$, hence strictly increasing in $T$; expanding and averaging gives the stated asymptotic. *(c)* Since $\bar{g} = \nabla\mathcal{L}_{\hat{c}}(\theta)$,

$$\langle g(T), \nabla\mathcal{L}_{\hat{c}}(\theta)\rangle = \eta T\big(\|\bar{g}\|^2 + \langle e(T), \bar{g}\rangle\big) \ge \eta T\|\bar{g}\|\big(\|\bar{g}\| - G\Phi(T)\big) > 0,$$

and $\|g(T)\| \le 2\eta T\|\bar{g}\|$ under the same condition. As $\mathcal{L}_{\hat{c}}$ is $\beta$-smooth, the descent lemma gives $\mathcal{L}_{\hat{c}}(\theta - \gamma g) \le \mathcal{L}_{\hat{c}}(\theta) - \gamma\langle\nabla\mathcal{L}_{\hat{c}}(\theta), g\rangle + \frac{\beta}{2}\gamma^2\|g\|^2$, whose right-hand side is smaller than $\mathcal{L}_{\hat{c}}(\theta)$ whenever $\frac{\beta}{2}\gamma\|g\|^2 < \langle\nabla\mathcal{L}_{\hat{c}}(\theta), g\rangle$; the stated range for $\gamma$ suffices. □

**Reading, and the limits of part (c).** Part (a) deserves emphasis: *at $T = 1$ the encoder step of* LOGIC *is exactly curvature-weighted gradient aggregation*. The synchronization interval is therefore the parameter that interpolates between gradient aggregation and trajectory coordination, and everything separating LOGIC from a reweighting scheme is a $T > 1$ effect, quantified next.

Part (c) is a sufficient condition and we state its limits rather than leave them to be inferred. (i) It tightens as disagreement grows: large $\zeta_{\hat{c}}$ implies cancellation in $\bar{g}$, shrinking $\|\bar{g}\|$ while $G\Phi(T)$ is unaffected, so the guarantee is weakest in the high-conflict regime. (ii) It is sufficient, not necessary, and the bound is worst-case, treating every per-step deviation as adversarially aligned, whereas in practice the deviations $\nabla\mathcal{L}_k(\theta_k(t)) - g_k$ partially cancel. (iii) $\mathcal{L}_{\hat{c}}$ is a proxy: descent on it is a consistency property, not evidence of optimality. Both sides of the condition are computable during training, and measuring $\|\bar{g}\|$ against $G\Phi(T)$ is the appropriate way to determine how binding it is, in preference to further analysis. Appendix A.2.5 reports $\|\bar{g}\|$ over the course of training on both benchmarks; evaluating $G\Phi(T)$ additionally requires estimates of $G$ and $\beta$, which we do not undertake, so we do not report the comparison itself.

### A.1.3 WHAT A WINDOW OF LENGTH $T > 1$ MAKES REACHABLE

**Proposition 3 (the curvature–conflict term).** *Define*

$$D := \sum_k \hat{c}_k H_k g_k - \bar{H}\bar{g} = \sum_k \hat{c}_k (H_k - \bar{H})(g_k - \bar{g}) = \mathrm{Cov}_{\hat{c}}(H, g).$$

*Under (A1)–(A3):*

*(a) The centroid displacement satisfies*

$$\bar{\theta}(T) - \theta = -\eta T \bar{g} + \tfrac{1}{2}\eta^2 T(T-1) \sum_k \hat{c}_k H_k g_k + R, \qquad \|R\| \le \tfrac{1}{6}\eta^3 T^3 G\big(LG + \beta^2\big),$$

*while $T$ steps of gradient descent on a fixed scalarization $\mathcal{L}_w$ from $\theta$ reach $\psi_w(T) - \theta = -\eta T g_w + \frac{1}{2}\eta^2 T(T-1) H_w g_w + R_w$ with the same remainder bound.*

*(b) If $\{g_k\}$ are linearly independent then $w = \hat{c}$ is the unique element of $\Delta^{K-1}$ with $g_w = \bar{g}$, and*

$$\bar{\theta}(T) - \psi_{\hat{c}}(T) = \tfrac{1}{2}\eta^2 T(T-1)\, D + R', \qquad \|R'\| \le \tfrac{1}{3}\eta^3 T^3 G(LG + \beta^2).$$

*(c) $D = 0$ whenever the task gradients coincide, and $D = 0$ whenever the task Hessians coincide; moreover $\|D\| \le 2\beta\zeta_{\hat{c}}$.*

**Proof.** *(a)* By (A1) and (A3), $\nabla\mathcal{L}_k(\theta_k(t)) = g_k + H_k(\theta_k(t) - \theta) + r_{k,t}$ with $\|r_{k,t}\| \le \frac{L}{2}\|\theta_k(t) - \theta\|^2$. Substituting $\theta_k(t) - \theta = -\eta t g_k + O(\eta^2 t^2 \beta G)$ from the recursion in Proposition 2 yields $\nabla\mathcal{L}_k(\theta_k(t)) = g_k - \eta t H_k g_k + \varrho_{k,t}$ with $\|\varrho_{k,t}\| \le \frac{1}{2}\eta^2 t^2 G(LG + \beta^2)$. Summing over $t = 0, \dots, T-1$ with $\sum_{t<T} t = T(T-1)/2$ and $\sum_{t<T} t^2 \le T^3/3$, then multiplying by $\eta$ and taking the $\hat{c}$-weighted combination, gives the claim; the same computation applied to the single loss $\mathcal{L}_w$, whose gradient and Hessian at $\theta$ are $g_w$ and $H_w$, gives the second identity.

*(b)* If $g_w = \bar{g}$ then $\sum_k (w_k - \hat{c}_k) g_k = 0$, so linear independence forces $w = \hat{c}$. Subtracting the two expansions at $w = \hat{c}$ leaves $\frac{1}{2}\eta^2 T(T-1)[\sum_k \hat{c}_k H_k g_k - \bar{H}\bar{g}]$ plus the two remainders. Any $w \neq \hat{c}$ already differs at first order in $\eta T$, which dominates.

*(c)* Expanding $\sum_k \hat{c}_k (H_k - \bar{H})(g_k - \bar{g})$ and using $\sum_k \hat{c}_k (g_k - \bar{g}) = 0$ and $\sum_k \hat{c}_k (H_k - \bar{H}) = 0$ gives the second expression. If all $g_k$ agree then $g_k - \bar{g} = 0$; if all $H_k$ agree then $H_k - \bar{H} = 0$; either annihilates the sum. Since $\|H_k - \bar{H}\|_{\mathrm{op}} \le 2\beta$,

$$\|D\| \le \sum_k \hat{c}_k \|H_k - \bar{H}\|_{\mathrm{op}} \|g_k - \bar{g}\| \le 2\beta \sum_k \hat{c}_k \|g_k - \bar{g}\| \le 2\beta \Big(\sum_k \hat{c}_k \|g_k - \bar{g}\|^2\Big)^{1/2} = 2\beta\zeta_{\hat{c}},$$

the last step by Jensen's inequality. □

**Remark (range of validity).** The separation in part (b) is a second-order statement, and (A3) makes its range explicit: the $D$ term dominates the remainder as soon as $\frac{1}{2}\eta^2 T^2 \|D\| > \frac{1}{3}\eta^3 T^3 G(LG + \beta^2)$, that is

$$\eta T \;<\; \frac{3\,\|D\|}{2\,G\big(LG + \beta^2\big)}.$$

The comparison is therefore informative for windows short enough that the copies remain in the region where the quadratic model is accurate, which the synchronization schedule keeps the copies in during early training; Appendix A.2.5 shows the runs leave this range later, and it is uninformative for windows long enough that all three terms are comparable. We regard this as the honest scope of the result rather than a limitation to be minimized.

**Proposition 4 (unreachability under freely adaptive weights).** *Consider any single-trajectory method of the form*

$$\psi(0) = \theta, \qquad \psi(t+1) = \psi(t) - \eta \sum_{k=1}^{K} w_k(t)\, \nabla\mathcal{L}_k(\psi(t)),$$

*where the coefficients $w_k(t) \ge 0$ may be chosen freely and independently at every step, with arbitrary nonnegative magnitudes. Assume $\{g_k\}$ and $\{H_k g_j\}_{k,j\in[K]}$ are each linearly independent. If $K \ge 2$, $T \ge 2$ and $\hat{c}$ has at least two nonzero entries, then no such method reproduces $\bar{\theta}(T) - \theta$ to second order in $\eta$.*

**Proof.** Expanding as in Proposition 3(a), and writing $H_{w(t)} = \sum_k w_k(t) H_k$,

$$\begin{aligned}\psi(T) - \theta &= -\eta \sum_{t=0}^{T-1} g_{w(t)} + \eta^2 \sum_{t=0}^{T-1} H_{w(t)} \sum_{s<t} g_{w(s)} + O(\eta^3 T^3) \\ &= -\eta \sum_k W_k g_k + \eta^2 \sum_{k,j} M_{kj} H_k g_j + O(\eta^3 T^3),\end{aligned}$$

where $W_k = \sum_{t<T} w_k(t)$ and $M_{kj} = \sum_{t<T} \sum_{s<t} w_k(t) w_j(s)$. By Proposition 3(a), matching $\bar{\theta}(T) - \theta$ requires

$$\text{(first order)} \quad \sum_k W_k g_k = T\bar{g}, \qquad \text{(second order)} \quad \sum_{k,j} M_{kj} H_k g_j = \tfrac{T(T-1)}{2} \sum_k \hat{c}_k H_k g_k.$$

Linear independence of $\{g_k\}$ turns the first condition into $W_k = T\hat{c}_k$ for every $k$; linear independence of $\{H_k g_j\}$ turns the second into $M = \frac{T(T-1)}{2} \mathrm{diag}(\hat{c})$, so in particular $M_{kj} = 0$ for all $k \neq j$.

Since every $w_k(t) \geq 0$, each summand of $M_{kj}$ is nonnegative, so $M_{kj} = 0$ forces

$$w_k(t)\, w_j(s) = 0 \qquad \text{for all } k \neq j \text{ and all } s < t. \tag{10}$$

Let $S_t = \{k : w_k(t) > 0\}$. Condition (10) says that for $s < t$ there is no pair $k \in S_t$, $j \in S_s$ with $k \neq j$. If some $S_t$ with $t \geq 1$ contained two distinct indices $a \neq b$, then any $j \in S_s$ for $s < t$ would differ from at least one of them, a contradiction unless $S_s = \emptyset$ for all $s < t$; and if $S_t = \{a\}$, $S_s = \{b\}$ with $a \neq b$ and $s < t$, that is a direct violation. Hence all nonempty $S_t$ are the same singleton $\{a\}$, so $W_k = 0$ for every $k \neq a$. This contradicts $W_k = T\hat{c}_k > 0$ for the at least two indices with $\hat{c}_k > 0$. □

**Corollary 4.1 (fixed weights).** Taking $w(t) \equiv w$ gives $M = \frac{T(T-1)}{2} w w^\top$, a rank-one matrix, which is diagonal only when $w$ is a vertex of $\Delta^{K-1}$. This recovers Proposition 3(b) as a special case.

**Remark (scope of Proposition 4).** The nonnegativity of the coefficients is what converts $M_{kj} = 0$ into the support condition (10), and it holds for every method that forms a convex or conic combination of task gradients, including MGDA (Sener & Koltun, 2018), CAGrad (Liu et al., 2021a), IMTL-G (Liu et al., 2021b), Nash-MTL (Navon et al., 2022) and all loss-weighting schemes. Methods that may assign negative coefficients, such as the projection step of PCGrad (Yu et al., 2020), are not covered: cancellation between terms of opposite sign could in principle produce a vanishing off-diagonal entry, and we do not claim otherwise. The linear-independence hypotheses concern $K$ and $K^2$ vectors respectively in $\mathbb{R}^n$ with $n$ of order $10^7$ in our experiments.

**Reading: freedom, and what freedom is not.** Propositions 3 and 4 give the precise sense in which LOGIC has room a single-trajectory method does not. Over a window, a single trajectory observes the landscape along *one* curve, evaluating every task gradient at a common iterate; LOGIC observes it along $K$ curves. The consequence is structural: a single trajectory can only assemble *products of aggregates*, $H_w g_w$, because the coefficients enter multiplicatively through one shared gradient, whereas the centroid assembles an *aggregate of products*, $\sum_k \hat{c}_k H_k g_k$. Proposition 4 shows the gap survives even when the coefficients are re-solved at every step with full knowledge of the landscape: matching at second order would force all weight onto a single task, which contradicts matching at first order.

Part (c) of Proposition 3 identifies what $D$ measures. As a weighted covariance between task curvature and task gradient, it vanishes if the tasks agree in gradient, and vanishes if they share the same local geometry; it is nonzero only when they differ in *both*, and its size is controlled by the same disagreement quantity $\zeta_{\hat{c}}$ that governs conflict. Since $H_k g_k$ need not lie in $\mathrm{span}\{g_k\}$, the update leaves the span of the task gradients without any Hessian–vector product ever being formed: this curvature is a consequence of trajectory geometry, distinct from the Hessian-trace estimates of Section 3, which serve only to set $\hat{c}$.

Three limitations are worth stating in the same breath. We do not claim $D$ is beneficial; the analysis fixes its identity, its magnitude and its vanishing conditions, not its sign or its effect on generalization. A larger reachable set is necessary but not sufficient for reaching better solutions: it enlarges what the update can express and says nothing about orientation toward useful regions. And unconstrained freedom is actively harmful here, which is the clearest evidence that the mechanism is not automatic: Table 4 shows that removing synchronization entirely yields $\Delta m\% = 29.52$, far worse than plain joint training, because copies that drift without bound leave the region in which the expansion of Proposition 3(a) is accurate and their positions cease to carry information about the landscape near $\theta$. Synchronization is what keeps the additional reach usable, by bounding copy drift (Figure 7); the second-order characterization itself holds only over the early part of training (Appendix A.2.5). The division of labor is thus explicit: theory establishes that the capacity exists and that single-trajectory methods lack it; the experiments establish that LOGIC realizes it.

**Why aggregation methods admit no confluence variant.** Existing methods like MGDA (Sener & Koltun, 2018), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), IMTL-G (Liu et al., 2021b), and Nash-MTL (Navon et al., 2022) cannot be directly combined with confluence. These frameworks map gradients evaluated at a *single* parameter point to a unified direction; key quantities like pairwise gradient angles or the convex hull of $\{g_k\}$ require gradients to share a common tangent space. Evaluating task copies at distinct points $\theta_k$ renders these angles meaningless and the convex hull undefined. Any attempt to combine them reduces to either applying the baseline strictly at synchronization points, which Proposition 2(a) shows is LOGIC at $T = 1$, or substituting the aggregation rule for curvature weights $\hat{c}$, an option already ablated in Table 4.

### A.1.4 Noise in the anchor-relative direction

The results most characteristic of LOGIC concern stability: the lowest normalized total variation in Figure 4 and the lowest cross-seed standard deviation among the compared methods in every benchmark table. The following gives one mechanism consistent with them.

**Proposition 5 (variance of the pseudo-gradient).** *Suppose the copies are advanced with stochastic gradients $\widehat{\nabla}\mathcal{L}_k(\theta_k(t)) = \nabla\mathcal{L}_k(\theta_k(t)) + \xi_{k,t}$ where $\mathbb{E}[\xi_{k,t} \mid \mathcal{F}_t] = 0$, $\mathbb{E}[\|\xi_{k,t}\|^2 \mid \mathcal{F}_t] \le \sigma^2$ and $\|\widehat{\nabla}\mathcal{L}_k\| \le G$ almost surely, with $\mathcal{F}_t$ the filtration generated by the first $t$ steps. Then*

$$\frac{g(T)}{\eta T} = \bar{g} + e_{\mathrm{drift}}(T) + e_{\mathrm{noise}}(T), \quad \|e_{\mathrm{drift}}(T)\| \le G\Phi(T) \text{ a.s.}, \quad \mathbb{E}\|e_{\mathrm{noise}}(T)\|^2 \le \frac{\sigma^2}{T}.$$

**Proof.** Decompose (9) with stochastic gradients in place of exact ones, giving $e_{\mathrm{noise}}(T) = \frac{1}{T}\sum_{t<T} u_t$ with $u_t = \sum_k \hat{c}_k \xi_{k,t}$. The drift bound is as in Proposition 2, the recursion for $a_k(t)$ being valid pathwise under the almost-sure bound. As $\{u_t\}$ is a martingale difference sequence with respect to $\{\mathcal{F}_t\}$, cross terms vanish and $\mathbb{E}\|e_{\mathrm{noise}}(T)\|^2 = T^{-2}\sum_{t<T}\mathbb{E}\|u_t\|^2$. By Jensen's inequality on the weights, $\|u_t\|^2 \le \sum_k \hat{c}_k \|\xi_{k,t}\|^2$, so $\mathbb{E}\|u_t\|^2 \le \sigma^2$ and the sum is at most $\sigma^2/T$. □

**Reading.** The two error terms move oppositely in $T$: the drift bound $G\Phi(T)$ increases, the noise bound $\sigma^2/T$ decreases. A shared update formed afresh at every step corresponds to $T = 1$, where drift vanishes but noise is largest. The growing synchronization schedule of Section 3 trades one against the other, and the proposition offers a candidate mechanism relevant to Figure 4: the centroid aggregates $T$ stochastic gradients along each task trajectory, so the direction it defines averages per-step sampling noise rather than transmitting it. The scope is narrower than the observation, however: the bound concerns the variance of the anchor-relative direction, not of the encoder iterate, which moves at every step (Appendix A.1.6) and whose variance we do not analyze, and $\sigma$ is not measured here. This is a mechanism consistent with the observations rather than a demonstration that it is the operative one. It does yield a falsifiable prediction: with other hyperparameters fixed, training smoothness should degrade as the synchronization interval is driven toward $T = 1$.

### A.1.5 The weights are slowly varying

**Proposition 6 (weight drift within a window).** *Let $\hat{c}$ be updated as in Section 3 with smoothing coefficient $\beta_c \in (0, 1)$, once every $\Delta_{\mathrm{curv}}$ steps. Then $\hat{c}(t) \in \Delta^{K-1}$ for all $t$ and, over any window*

*of $T$ steps,*

$$\|\hat{c}(t+T) - \hat{c}(t)\| \le \sqrt{2}\,(1-\beta_c)\left\lceil \frac{T}{\Delta_{\mathrm{curv}}} \right\rceil .$$

**Proof.** Write the update as $\hat{c} \leftarrow \beta_c \hat{c} + (1-\beta_c)p$ with $p_k = \big(1/(\rho_k+\varepsilon)\big)/\sum_j \big(1/(\rho_j+\varepsilon)\big) \in \Delta^{K-1}$. Convexity of the simplex gives $\hat{c} \in \Delta^{K-1}$ throughout, and positivity follows from $p_k > 0$ with uniform initialization. Each update moves $\hat{c}$ by $(1-\beta_c)\|p - \hat{c}\| \le \sqrt{2}(1-\beta_c)$, the Euclidean diameter of $\Delta^{K-1}$ being $\sqrt{2}$. There are at most $\lceil T/\Delta_{\mathrm{curv}} \rceil$ updates in the window, and the triangle inequality gives the bound. □

**Reading, and one conditional statement.** With the values used throughout ($\beta_c = 0.99$, $\Delta_{\mathrm{curv}} = 100$) the bound gives $\|\hat{c}(t+T) - \hat{c}(t)\| \le 0.14$ over $T = 1000$ steps: the weights are slowly varying relative to the window rather than constant. This licenses (A6), and it also delimits what Proposition 2(c) can mean. Because $\hat{c}$ changes across windows, successive steps decrease slightly different functions $\mathcal{L}_{\hat{c}(t)}$, and per-step descent on a moving objective is not a convergence proof; we make no convergence claim.

One conditional statement is available. Since $\hat{c} \ge 0$ with $\sum_k \hat{c}_k = 1$, any $\theta^\star$ with $\nabla \mathcal{L}_{\hat{c}}(\theta^\star) = 0$ satisfies $0 \in \mathrm{conv}\{\nabla \mathcal{L}_k(\theta^\star)\}$ and is Pareto-stationary; equivalently, by Proposition 1, the worst-case descent ceiling at $\theta^\star$ is zero. Hence *if* the weights converge and *if* the iterates converge, the limit points are Pareto-stationary. Both antecedents are unproven here. Figure 6 shows the weights stabilizing empirically, which supports the first without establishing either.

### A.1.6 Assumptions versus the implemented algorithm

We list every point at which the analysis departs from the method as run.

**Adaptive per-task optimizers.** The copies use Adam, not plain gradient descent. Proposition 2 extends with modification: writing the step as $-\eta P_k(t)\widehat{\nabla}\mathcal{L}_k(\theta_k(t))$ for a diagonal preconditioner $P_k(t)$, (9) becomes a weighted average of *preconditioned* gradients and part (a) reads $g(1) = \eta \sum_k \hat{c}_k P_k(0) g_k$. The structure of Propositions 3 and 4 is likewise preserved with $H_k g_j$ replaced by the corresponding preconditioned quantities, provided $P_k$ varies slowly within a window. We state this as the natural extension, not as a proved result: the bounds would acquire a dependence on the conditioning of $P_k$, which we have not tracked.

**Stochastic sampling.** Treated explicitly in Proposition 5, where it is load-bearing. Propositions 1–4 concern the deterministic quantities $g_k, H_k$ and are unaffected in expectation.

**Task-specific heads.** Not covered. The analysis treats losses as functions of $\theta$ alone, whereas $\mathcal{L}_k = \mathcal{L}_k(\theta, \phi_k)$ and the heads descend the *unweighted* sum $\sum_k \mathcal{L}_k$ while the encoder descends the $\hat{c}$-weighted aggregate. The two updates target different objectives and the coupled dynamics of $(\theta, \phi)$ are not covered; all propositions should be read as statements about the encoder update at fixed $\phi$.

**The shared encoder moves within a window.** Not covered. The analysis fixes an anchor $\theta$ at synchronization, sets $\theta_k(0) = \theta$, and characterizes $g(T) = \theta - \bar{\theta}(T)$: the displacement of the centroid relative to that anchor. The implementation instead updates the shared encoder at every iteration while the copies continue from the anchor until the next synchronization, so the pseudo-gradient injected at step $t$ of a window is $\theta(t) - \bar{\theta}(t)$ rather than $\theta(0) - \bar{\theta}(t)$. The two coincide at $t = 0$ and separate as the window proceeds, by the accumulated encoder motion $\theta(t) - \theta(0)$, which our expansions do not represent. The curvature term $D$ arises from the copies' trajectories and is unaffected by this, but Propositions 2–4 should be read as statements about the copies' displacement relative to their own anchor rather than as a characterization of the update injected at every step within a window.

Two existing results bear on whether this amounts to an effective step-size effect. A slow-weight control with the same repeated-displacement structure (Appendix A.4) does not improve on equal weighting, and performance is nearly flat across a fourfold range of task-copy learning-rate scales (Appendix A.5).

**Non-smooth losses.** The $L_1$ depth loss is not $\beta$-smooth at its kink, so (A1) fails on a measure-zero set. The results hold for any smoothed surrogate with bounded Hessian; we have not quantified the approximation error.

**Synchronization preserving optimizer state.** The implementation resets $\theta_k \leftarrow \theta$ while retaining momentum and second-moment estimates. The analysis assumes a clean restart and does not cover the retained-state variant.

**Curvature estimation.** The Hutchinson estimator is unbiased but noisy; Proposition 6 treats $\hat{c}$ as given and does not model estimation error.

#### A.1.7 SUMMARY

*Established.* (i) The worst-case descent available to any single shared direction is $\mathrm{dist}(0, \mathrm{conv}\{g_k\})$, a ceiling met by existing aggregation methods and binding on LOGIC's encoder step as well, which places the remaining freedom in the trajectory rather than the step (Proposition 1). (ii) The pseudo-gradient is a weighted aggregate plus an error bounded non-asymptotically by $G\Phi(T)$ and exactly zero at $T = 1$, making the synchronization interval an explicit dial between aggregation and coordination (Proposition 2). (iii) For $T > 1$ the window displacement contains the curvature–conflict covariance $D$, which vanishes exactly when tasks agree in gradient or in curvature and which no single-trajectory method reproduces, whether its coefficients are fixed (Proposition 3) or re-solved at every step (Proposition 4). (iv) Noise in the anchor-relative direction falls as $\sigma^2/T$ while the drift bound rises, identifying the trade-off the synchronization schedule navigates (Proposition 5). (v) The curvature weights vary slowly enough within a window to license treating them as fixed (Proposition 6).

*Deliberately not claimed.* No optimality result, and no privileged status for $\mathcal{L}_{\hat{c}}$. No convergence result: the weights vary across windows, so per-step descent is on a moving objective, and Pareto-stationarity of limit points is stated only conditionally. No claim that $D$ improves solution quality, only that it exists, is quantified, and is inaccessible to single-trajectory methods with nonnegative coefficients. No coverage of the coupled encoder–head dynamics, the retained optimizer state, or the adaptive optimizers actually used, beyond the extensions recorded in Appendix A.1.6. The empirical claims of Section 4 rest on the experiments, and these propositions are offered as an account of the mechanism rather than as a guarantee of the outcome.

### A.2 TASK GEOMETRY ON NYUV2 AND CITYSCAPES: A DATA-CENTRIC VIEW

The propositions of Section A.1 identify the quantities that govern LOGIC's update: the inter-task gradient disagreement $\zeta_{\hat{c}}$, the per-task curvature entering through the covariance $D = \mathrm{Cov}_{\hat{c}}(H, g)$, and the window length $T$ that controls how far the pseudo-gradient departs from a weighted aggregate. This section asks what those quantities correspond to in the two benchmarks and reports per-step measurements alongside them. We stress the status of what follows: none of the propositions predicts a performance ordering between methods, so these are observations consistent with the analysis rather than confirmations of a prediction, and the attributions we offer are interpretations of the measured signals. Appendix A.2.5 then measures $D$ itself and reports where these interpretations are borne out and where they are not.

#### A.2.1 TWO AXES OF DIFFICULTY, AND WHY THEY ARE THE TWO FACTORS OF $D$

NYUv2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) differ along two essentially orthogonal axes: *gradient disagreement* (how far the per-task gradients are from their weighted mean at the shared parameter) and *curvature heterogeneity* (how much the per-task Hessians differ). NYUv2's three structurally heterogeneous tasks (segmentation, depth, and surface normals) enforce distinct inductive biases that routinely conflict at the pixel level. Cityscapes' two tasks are strongly correlated in practice: vehicles, roads, and buildings occupy predictable depth ranges, so segmentation and depth gradients tend to be aligned (Wang et al., 2021). NYUv2 therefore loads the disagreement axis far more heavily.

These are precisely the two factors appearing in Proposition 3(c), which writes $D = \sum_k \hat{c}_k (H_k - \bar{H})(g_k - \bar{g})$ and shows that $D$ vanishes if the task gradients coincide and vanishes if the task Hessians coincide. The term that distinguishes a LOGIC window from any single-trajectory update

is thus non-zero only where *both* axes are loaded. This is worth stating carefully, because the two benchmarks load them differently and a large curvature spread alone is not sufficient. Which of the two factors in fact dominates is an empirical question, and Appendix A.2.5 settles it empirically.

The curvature axis, measured in absolute terms, in fact loads in the opposite direction to the disagreement axis. Different loss families produce systematically different parameter-space Hessian traces (Liao & Mahoney, 2021; Li et al., 2018): the $L_1$ depth loss is piecewise linear with curvature concentrated at the kink, while the cosine-similarity normal loss is bounded and smoothly curved. The absolute traces $\tau_k = \mathrm{tr}(H_k)$ therefore encode loss-family identity rather than task hardness (Li et al., 2018), and this offset is *larger on Cityscapes than on NYUv2*: the cross-entropy-to-regression ratio is $\sim 57\times$ on Cityscapes versus $\sim 2.7\times$ on NYUv2 (Figure 5). Because the Cityscapes gradients are closely aligned, one might expect the factor $(g_k - \bar{g})$ to be small enough there that a large absolute trace spread does not by itself make $D$ large, while on NYUv2 both factors are non-negligible. Appendix A.2.5 tests this reading by measuring $\|D\|$ directly against $\|\eta T \bar{g}\|$, and finds that it holds only during early training: over the full run the curvature factor is the dominant one on both benchmarks.

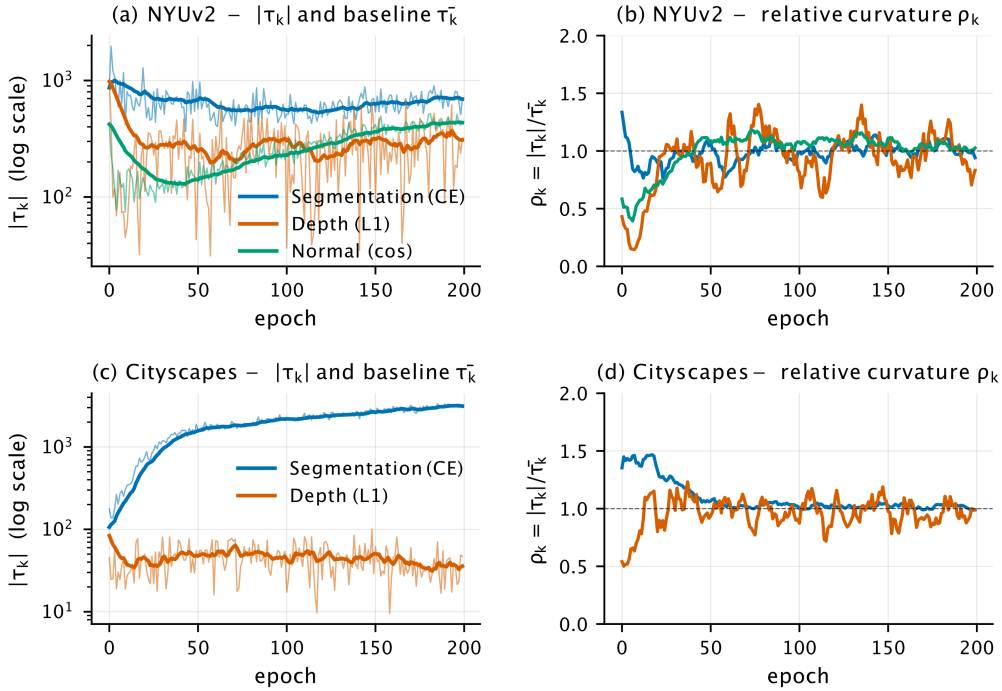



Figure 5: **Hessian-trace heterogeneity across loss families and its removal via the relative formulation.** Panels (a) and (c) plot the per-step absolute Hessian trace $|\tau_k|$ (thin) and its EMA baseline $\bar{\tau}_k$ (bold), estimated via Hutchinson probes (Hutchinson, 1989), on a semi-logarithmic vertical axis. The cross-entropy / regression ratio is moderate on NYUv2 ($\sim 2.7\times$) but nearly two orders of magnitude on Cityscapes ($\sim 57\times$): a structural artifact of the loss family rather than a signal of task hardness (Liao & Mahoney, 2021; Li et al., 2018). Panels (b) and (d) plot the relative curvature $\rho_k = |\tau_k|/\bar{\tau}_k$ used by LOGIC (Section 3), which removes this offset on both benchmarks and leaves only fluctuations around unity (dashed line) relative to each task's own recent history.

### A.2.2 The curvature weighting signal

**The relative formulation normalizes the weighting signal.** The quantity $\rho_k$ that sets the weights is separate from the curvature appearing in $D$: the Hessian-trace estimates enter only through $\hat{c}$, whereas the curvature shaping the update direction arises from the trajectory geometry (Proposition 3). For the weighting signal, the per-task EMA baseline $\bar{\tau}_k$ absorbs the loss-family scale, so $\rho_k = |\tau_k|/(\bar{\tau}_k + \varepsilon)$ varies around unity by construction. Although the absolute traces span up to $\sim 57\times$ across tasks, the relative-curvature means lie between $0.93$ and $1.09$ on both benchmarks (Figure 5(b,d)). The centroid is therefore pulled toward tasks that are currently flatter than their own

typical level, irrespective of loss family. Figure 6 shows the resulting dynamics: weights diverge from their uniform initialization toward the relatively flatter task on each benchmark, depth in both cases ($\rho_{\text{depth}} = 0.93$ on NYUv2 and 0.97 on Cityscapes).

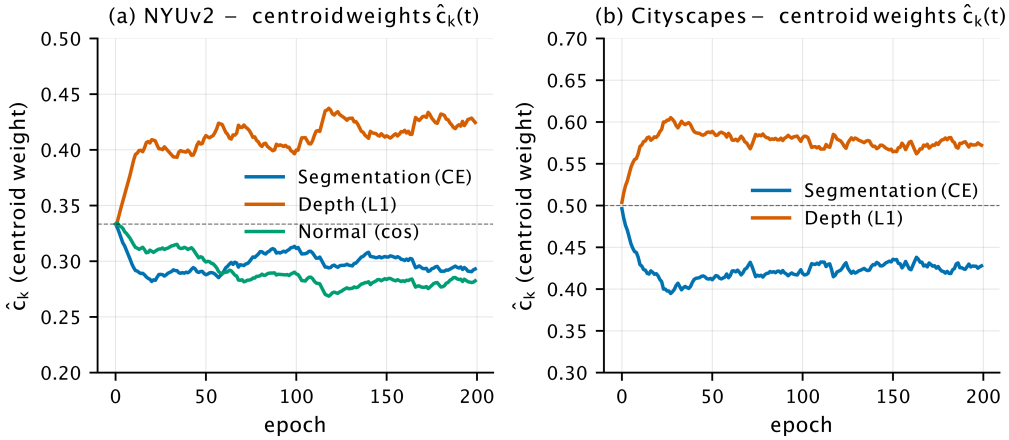



Figure 6: **Centroid weights $\hat{c}_k(t)$ track the relative-curvature signal.** On both benchmarks the inverse-curvature mechanism assigns higher weight to the relatively flatter task: depth dominates on NYUv2 (a) ($\hat{c}_{\text{depth}} \approx 0.43$, against $1/3$ at uniform initialization) and on Cityscapes (b) ($\hat{c}_{\text{depth}} \approx 0.58$, against $1/2$), consistent with the relative-curvature weighting mechanism (Eq. 4) and with the slow-variation bound of Proposition 6.

**Interpreting the centroid-weight dynamics.** A possible source of confusion in Figure 6 is that the weights are not a pointwise inverse of the $\rho_k$ curves: some tasks, depth in particular, hold consistently higher weight even when their instantaneous $\rho_k$ is not the smallest. This is a consequence of the temporal dynamics of the estimator rather than a departure from Eq. (4). Both $\bar{\tau}_k$ and $\hat{c}_k$ are updated by exponential moving averages, so the weights integrate curvature information over time instead of tracking instantaneous fluctuations, and $\rho_k$ is itself a normalized quantity, so short-term deviations do not translate proportionally into weight changes. The correct reading is that $\hat{c}_k$ reflects a history-dependent estimate of relative curvature stability, favoring tasks that are consistently well-conditioned relative to their own history. Proposition 6 quantifies the resulting inertia: with $\beta_c = 0.99$ and $\Delta_{\text{curv}} = 100$, the weights move by at most 0.14 in Euclidean norm over a window of 1000 steps. This smoothing suppresses noise from the Hutchinson estimator and prevents rapid oscillation in task importance, at the cost of the responsiveness that a pointwise inverse would provide.

Freezing $\hat{c}$ at its converged NYUv2 values for the whole run, with curvature estimation switched off, reaches $\Delta m\% = -0.10 \pm 0.618$ on NYUv2, well short of the $-2.78$ obtained when the weights are allowed to evolve, and short also of uniform weighting under synchronization alone ($-2.09$, Table 4). Those converged values are of course unavailable in advance on a new dataset, but that is not the point of the control: the experiment grants them for free and they still do not help. What the mechanism contributes is therefore not a search for a good set of coefficients, but the schedule by which it arrives at them. Early in training the relative-curvature signal has not yet stabilized, and imposing the final split from the first step down-weights tasks before they have contributed to the shared representation. Beginning uniform and separating gradually, as the outer EMA does, is what makes the weights useful.

### A.2.3 Where the margin over baselines is largest

Gradient-aggregation methods (MGDA (Sener & Koltun, 2018), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), Nash-MTL (Navon et al., 2022)) aggregate $\{g_k\}$ at the shared $\theta$ and select a single compromise direction. By Proposition 1, the worst-case descent available to any such direction is $\text{dist}(0, \text{conv}\{g_k\})$, which for two equal-norm gradients at angle $\omega$ is $G\cos(\omega/2)$. We reiterate that this ceiling is not something LOGIC evades: its encoder step is also a single direction and is bounded by the same quantity. What differs is the composition of that direction over a window

of $T > 1$ steps, which by Propositions 3 and 4 contains the component $D$ that no single-trajectory method reproduces, whether its coefficients are fixed or re-solved at every step.

The margins themselves differ markedly between the two benchmarks, and the difference concerns what LOGIC adds beyond what a good aggregation method already achieves rather than absolute performance. On NYUv2 the margin is large: $\Delta m\% = -2.78$ against $-0.27$ for the strongest baseline (Table 1). On Cityscapes it is narrow: $-3.40$ against $-3.30$ for IMTL-G (Table 2), while the absolute improvement over single-task learning is in fact slightly larger there. It would be natural to attribute this pattern to $D$, on the reasoning that Cityscapes' aligned gradients suppress it. Appendix A.2.5 shows that the measurement does not support that attribution: $\|D\|$ is substantial on both benchmarks and ends the run larger on Cityscapes. We therefore report the margins without offering $D$ as their explanation.

### A.2.4 Synchronization keeps the pseudo-gradient informative

Proposition 2 makes the role of the window length explicit: writing $g(T)/(\eta T) = \bar{g} + e(T)$, the deviation obeys $\|e(T)\| \le G\Phi(T)$ with $\Phi(1) = 0$ and $\Phi$ increasing in $T$, so short windows keep the pseudo-gradient close to a weighted aggregate while long windows allow it to drift away. Proposition 5 adds the countervailing effect, the noise in the anchor-relative direction falling as $\sigma^2/T$, and the growing synchronization schedule of Section 3 trades the two against each other. Copy drift is bounded over a window by $\|\theta_k(t) - \theta\| \le (G/\beta)[(1 + \eta\beta)^t - 1]$, established in the proof of Proposition 2.

Figure 7 reports both quantities during training. The injected pseudo-gradient $\|g(t)\| = \|\theta - \bar{\theta}\|$ remains strictly nonzero throughout training (per-step minimum $4.9\times10^{-3}$ on NYUv2, $3.7\times10^{-3}$ on Cityscapes, with no zero entries), and it decays by $1.57$ orders on NYUv2 and $2.10$ orders on Cityscapes. We read this decay as reflecting the shrinking aggregate gradient $\|\bar{g}\|$ together with the decaying learning rate, which enters $g$ multiplicatively; it is not by itself evidence of monotone descent. Proposition 2(c) gives a descent condition, $G\Phi(T) < \|\bar{g}\|$, for the anchor-relative displacement, and does not imply descent of the moving-encoder update. The per-epoch maximum copy drift $\max_t \|\theta_k(t) - \theta\|$ traces an envelope that grows while the synchronization interval expands during warm-up and then contracts as the learning rate decays, so the copies remain confined to a bounded neighborhood of $\theta$ throughout and never exhibit the runaway separation that the no-synchronization ablation produces (Table 4).

### A.2.5 Measuring the curvature–conflict term

The preceding subsections interpret $D$ from the two quantities that compose it. Here we measure it directly.

**Procedure.** Expanding the covariance and using $\sum_k \hat{c}_k(g_k - \bar{g}) = 0$ gives

$$D = \sum_k \hat{c}_k(H_k - \bar{H})(g_k - \bar{g}) = \sum_k \hat{c}_k H_k(g_k - \bar{g}),$$

so $D$ requires $K$ Hessian–vector products and no Hessian is ever formed. We evaluate it at each synchronization event, immediately after the copies are re-anchored and before any of them takes a step, which is the configuration $\theta_k = \theta$ that Proposition 3 assumes. Gradients are taken with respect to the shared encoder parameters on a probe batch fixed once at the start of the run and reused at every measurement, so that successive values are comparable and are not dominated by mini-batch noise; the probe batch never enters an update. The cost is one extra backward pass per task at a few dozen points in a run, and is negligible against the Hutchinson estimation already performed every $\Delta_{\mathrm{curv}}$ steps.

By Proposition 3(a) the window displacement is $-\eta T\bar{g}$ at first order and carries $\frac{1}{2}\eta^2 T(T-1)$ multiplying the curvature term at second order, so the informative quantity is the ratio of the two,

$$\frac{\frac{1}{2}\eta^2 T(T-1)\,\|D\|}{\eta T\,\|\bar{g}\|} = \frac{\eta\,(T-1)\,\|D\|}{2\,\|\bar{g}\|},$$

with $T$ counted in optimizer steps. Figure 8 reports this ratio and the two norms separately on both benchmarks.

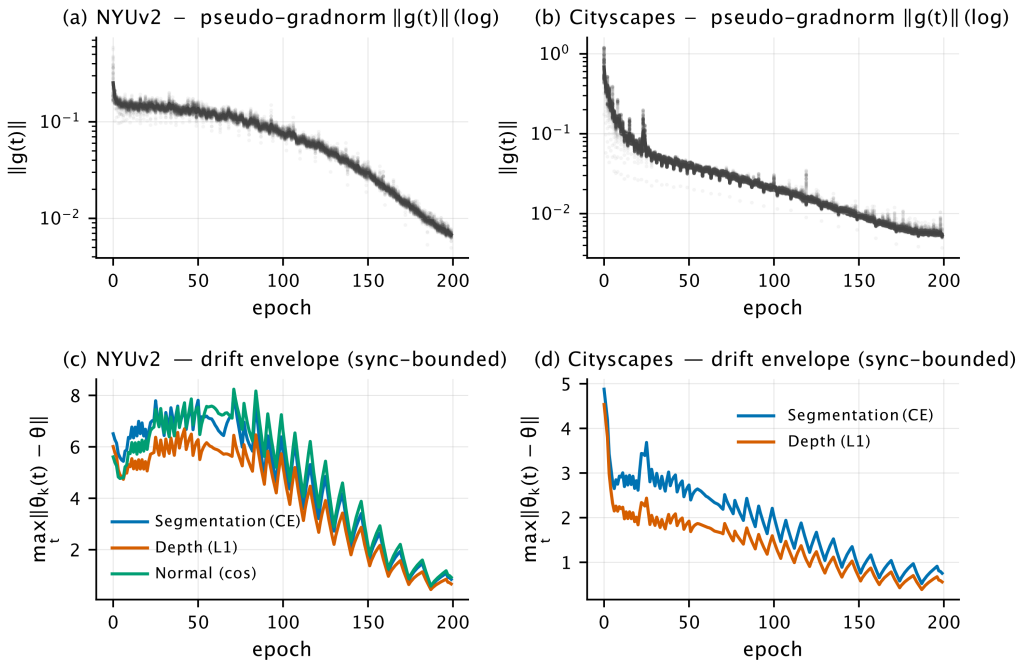



Figure 7: **Pseudo-gradient stays informative; copy drift remains sync-bounded.** Panels (a) and (b) show $\|g(t)\| = \|\theta - \tilde{\theta}\|$ on a semi-logarithmic vertical axis: the magnitude decays by 1.57 orders on NYUv2 and 2.10 orders on Cityscapes but remains strictly positive throughout, so the injected pseudo-gradient remains strictly nonzero throughout training. Panels (c) and (d) report the per-epoch maximum copy drift $\max_t \|\theta_k(t) - \theta\|$, which stays within a bounded envelope across training, consistent with the drift bound established in the proof of Proposition 2.

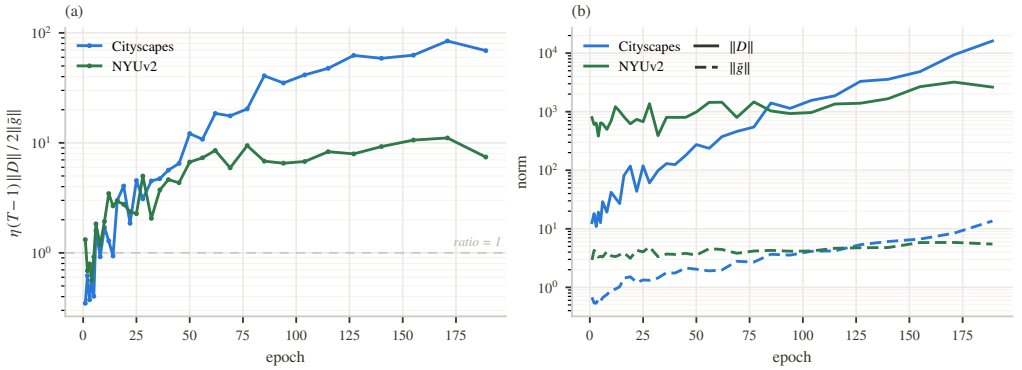



Figure 8: **The curvature–conflict term measured during training.** **(a)** The ratio $\eta(T-1)\|D\|/2\|\bar{g}\|$ of the second-order to the first-order term of Proposition 3(a), on a logarithmic vertical axis. **(b)** $\|D\|$ (solid) and $\|\bar{g}\|$ (dashed) for each benchmark. $D$ is evaluated at each synchronization event via $K$ Hessian–vector products on a fixed probe batch. Both benchmarks leave the regime in which $D$ is a small correction; the ordering between them reverses partway through training.

**What the measurement shows.** The measurement answers two questions: how large $D$ is relative to the first-order term, and which of its two factors accounts for the difference between the benchmarks.

*$D$ is not a small correction.* On both benchmarks the ratio rises above unity within the first 50 epochs and stays there, ending near 7 on NYUv2 and near 70 on Cityscapes. The term that Propositions 3 and 4 show to be unreachable by any single-trajectory method is therefore a substantial part of the window displacement throughout training, on both benchmarks. This is the empirical content of the distinction the propositions draw: the anchor-relative displacement LOGIC's task trajectories generate is far from a weighted aggregate of the task gradients. Over the epochs where the ratio is

near unity the reading is quantitative; beyond that it is the qualitative statement that the displacement has left the regime in which $D$ is a correction at all, as the caveat below makes precise.

*The two benchmarks reach that regime differently.* On NYUv2 $\|D\|$ is already large in the first epochs and grows only about half an order of magnitude over the run. On Cityscapes it starts one to two orders of magnitude lower, as the alignment of its gradients would suggest, but grows by roughly three orders, crossing NYUv2 near epoch $85$ and ending about six times larger. The ratio crosses earlier, near epoch $45$, since $\eta$, $T$ and $\|\bar{g}\|$ also evolve over training.

*Curvature heterogeneity is the factor that drives that growth.* The Cityscapes $\|D\|$ trajectory follows the shape of its segmentation Hessian trace in Figure 5(c), which climbs steadily throughout training while the NYUv2 traces stay comparatively flat. Both factors of Proposition 3(c) are necessary for $D$ to be non-zero, but it is $(H_k - \bar{H})$ that varies most across these two benchmarks, and the loss-family trace offset documented in Figure 5 is large enough to outweigh the closer gradient alignment on Cityscapes. Two independent instruments, the Hutchinson traces used to set $\hat{c}$ and the Hessian–vector products used here, agree on this, which is some evidence that it reflects the benchmarks rather than either estimator.

**Consequences, including for the reading above.** The measurement supports the claim the propositions actually make and withdraws one that the surrounding discussion had layered on top of it.

What is supported is that LOGIC's update departs substantially from weighted gradient aggregation once $T > 1$, and by a margin that grows through training. This is the empirical counterpart of Proposition 2(a): at $T = 1$ the encoder step is exactly curvature-weighted aggregation, and the measured ratio quantifies how far the synchronization schedule moves it away from that limit.

What is withdrawn is the attribution of the cross-benchmark margin pattern to $D$. On the measured trajectories $\|D\|$ ends larger on Cityscapes, where the margin over the strongest baseline is narrower, so the ordering of $\|D\|$ does not track the ordering of the margins. We record this rather than reconciling it: the propositions never claimed $D$ improves solution quality, and the measurement is consistent with that reservation while ruling out the stronger reading. What governs the margin is not settled by anything reported here.

**A caveat on the second-order reading.** Proposition 3(a) is a truncated expansion, and a ratio exceeding unity means the second-order term is larger than the first-order one, so the truncation is not justified in that regime and the third-order remainder need not be small either. The remark following Proposition 3 states the validity range as $\eta T < 3\|D\|/2G(LG + \beta^2)$; the measurements indicate that the runs leave it partway through training. The quantitative reading of $\|D\|$ as *the* second-order contribution is therefore reliable only over the early epochs, where the ratio is near unity. Beyond that point the honest statement is the qualitative one: the displacement is far from $-\eta T\bar{g}$, and the discrepancy is what $D$ measures at leading order.

### A.3 Empirical Cost-Accuracy Trade-off

To complement the structural analysis of the main text, we measure per-epoch training time and peak memory for all methods under an identical NYUv2 / HPS / ResNet-50 protocol on a single **NVIDIA RTX 4090**. Figure 9 summarizes these in a single view: training time on the horizontal axis, average improvement over single-task learning ($-\Delta m\%$ from Table 1) on the vertical axis, and peak training memory as the marker area, all normalized to the EW baseline ($1\times$). The upper-left region is preferable.

LOGIC attains the largest improvement over single-task learning of any method. Among the methods that actively resolve task conflict, it is also the *fastest*, roughly $2.7\times$ an EW epoch versus $\approx 3.1\times$ for the gradient-manipulation baselines; only the loss-weighting methods are cheaper, and none of them improves on single-task learning. It therefore lies on the Pareto front: no baseline is both faster and more accurate.

A memory footprint above the EW baseline is intrinsic to any method that leaves the single-shared-update regime. The gradient-manipulation baselines already require $1.0$–$1.15\times$ the EW memory to store $K$ per-task gradient buffers, and LOGIC extends this to $1.79\times$ through the $(1+K)\times$ encoder copies maintained during training, a difference of degree rather than of kind. This overhead is

moreover confined to training: at inference LOGIC reduces to a single shared encoder with no additional memory or latency, so its stability and accuracy gains incur no cost at deployment.

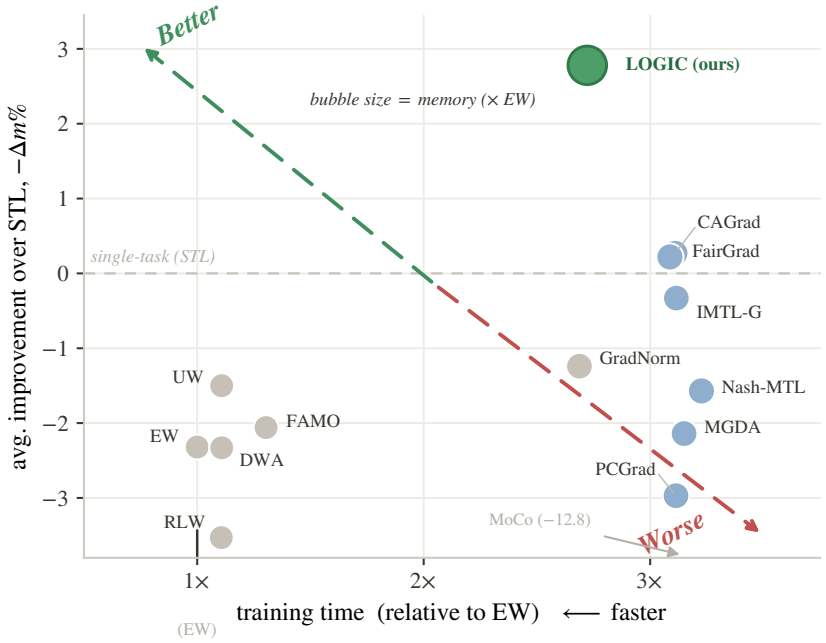



Figure 9: **Cost–accuracy trade-off on NYUv2** (HPS, ResNet-50), on a single NVIDIA RTX 4090. Per-epoch training time ($x$) and peak training memory (marker area) are normalized to the EW baseline ($1\times$); the $y$-axis is the average improvement over single-task learning ($-\Delta m\%$, higher is better), so the upper-left corner is preferable. **Gray** = loss-weighting, **blue** = gradient-manipulation baselines, **green** = LOGIC. LOGIC is the fastest conflict-resolving method and attains the largest improvement over single-task learning of any method, placing it on the Pareto front. The gradient-manipulation baselines already exceed the EW memory (1.0–1.15×); LOGIC's $1.79\times$ reflects its $(1+K)\times$ training-time encoder copies and vanishes at inference. MoCo ($-\Delta m\% = -12.8$) is off-scale.

**The comparison is not confounded by the epoch budget.** Compute-matched comparison is a live concern in this literature: Kurin et al. (2022) and Xin et al. (2022) report that tuned unitary scalarization matches specialized multi-task optimizers once budgets are equalized. Because a LOGIC epoch is more expensive than an EW epoch, one might ask whether allotting every method the same 200 epochs, rather than the same wall-clock budget, favors LOGIC. It does not. For the conflict-resolving baselines the epoch-matched protocol is in fact the *generous* one: at $2.7\times$ EW, LOGIC is cheaper per epoch than the gradient-manipulation family at $\approx 3.1\times$, so 200 LOGIC epochs correspond to roughly 174 epochs of CAGrad, FairGrad, IMTL-G, Nash-MTL, PCGrad, MGDA or MoCo. Equalizing wall-clock would remove epochs from precisely the baselines that compete most closely.

The question therefore concerns only the loss-weighting methods, which are cheap enough (1.0-1.6× EW) that equal wall-clock would grant them 340-540 epochs. Here the training curves show that the additional budget would not be productive: measured on the per-epoch validation trajectories averaged over the three seeds, every method has realized at least 97.9% of its total $\Delta m\%$ improvement by epoch 150 (median 99.8%), and at least 95.2% by epoch 100. The improvement is also clearly decelerating: comparing successive 50-epoch window means, the median shift falls from 1.70 in mid-training to 0.45 over the final window, and the deceleration is uniform across all fourteen methods, with the largest final-window shift of any method being 0.96. Residual movement at that scale (median 0.45, maximum 0.96) is well below the margin of 2.51, separate LOGIC from the strongest baseline in Table 1. We report window means rather than single-epoch values because, as Figure 4 shows, the baselines oscillate substantially from epoch to epoch; that oscillation is what remains at the end of training, not a residual trend.

### A.4 A SLOW-WEIGHT CONTROL

LOGIC displaces a shared model toward parameters that were adapted elsewhere, which is structurally what Lookahead (Zhang et al., 2019a) and Reptile (Nichol et al., 2018) do with their slow weights (Section 3). Slow-weight averaging is known to smooth training curves and reduce run-to-run variance on a single objective. Two of our empirical claims are therefore open to a deflationary reading: the low normalized total variation of Figure 4 and the low cross-seed spread we report in every results table might follow from averaging parameters at all, rather than from coordinating *per-task* trajectories. The concern is of the kind Kurin et al. (2022) and Xin et al. (2022) press for multi-task optimizers generally, and this subsection tests it directly.

We train plain equal weighting with its Adam optimizer wrapped in Lookahead at the published defaults ($k = 5$ inner steps, slow-weight step $\alpha = 0.5$), on NYUv2 / HPS / ResNet-50 for 200 epochs over the same three seeds and with every other setting unchanged. The arm is a single shared model: no task copies, no curvature weighting, no synchronization. If generic slow-weight averaging were responsible for our stability, this arm should approach LOGIC on both stability measures.

Table 5: **Slow-weight control on NYUv2 (HPS, ResNet-50, three seeds).** $\Delta m\%$ is the final-epoch value against the single-task reference of Table 1; SD is across seeds; NTV is the normalized total variation of Figure 4, averaged over the three tasks and the three seeds. Lower is better throughout.

| Configuration | $\Delta m\% \downarrow$ | SD$\downarrow$ | NTV$\downarrow$ |
|---|---|---|---|
| EW | $+2.32$ | 1.690 | 0.0208 |
| EW + Lookahead ($k$=5, $\alpha$=0.5) | $+2.59$ | **0.315** | 0.0172 |
| **LOGIC** | $\mathbf{-2.78}$ | 0.349 | **0.0149** |

**Reading.** Table 5 reports the outcome; the control separates the two claims rather than confirming or refuting both.

On *performance* it is decisive. Lookahead does not improve $\Delta m\%$ at all: $+2.59$ against $+2.32$ for plain equal weighting, a difference within the seed spread and in the wrong direction. The five-point gap between equal weighting and LOGIC is therefore not attributable to slow-weight averaging, and the deflationary reading fails for the result the paper leads with.

On *smoothness* it is partial, and we report it as such. Lookahead does reduce NTV, from $0.0208$ to $0.0172$; LOGIC reaches $0.0149$. Roughly $60\%$ of our reduction relative to equal weighting is thus reproduced by a generic slow-weight scheme, and the remainder is not. Placed against the fourteen methods of Figure 4, this arm would rank third, behind LOGIC and ahead of every gradient-manipulation baseline, which is itself informative about how much of that figure's spread reflects conflict resolution as opposed to update smoothing.

On *cross-seed spread* the result is mixed. Lookahead reaches $0.315$ against our $0.349$ on NYUv2, so a slow-weight wrapper alone can stabilize seed-to-seed behavior. But it gets there at $\Delta m\% = +2.59$, with no multi-task gain at all. Low spread and positive transfer are separable, and LOGIC is the only configuration here with both, as it is on every benchmark we report (Tables 1, 2, 3, 6, 7, 8 and 9).

### A.5 SENSITIVITY TO HYPERPARAMETER TUNING

We evaluate the sensitivity of LOGIC to its key hyperparameters on NYUv2 and Cityscapes using the HPS architecture with a ResNet-50 backbone (commonly-used for HPS). Training protocol hyperparameters (learning rate, epochs, optimizer, backbone) are shared across all methods; LOGIC-specific hyperparameters use identical default values across both benchmarks, without task-specific tuning.

LOGIC introduces four key hyperparameters: **(i)** *Hutchinson probes*: the number of random vectors used to estimate the Hessian trace for curvature weighting (tested: 8, 16, 32). **(ii)** *Curvature update interval*: how often curvature estimates are refreshed during training (tested: every 50 – 500 steps).

**(iii)** *Task-copy learning rate scale*: a multiplicative factor applied to the main learning rate to obtain the task-copy learning rate, scheduled from a start value to an end value over training (tested: start 0.5–2.0, end 0.3–0.7). **(iv)** *Synchronization interval*: the number of epochs between consecutive re-anchoring of task copies to the shared model, growing linearly from a minimum to a maximum over training (tested: minimum 1–5 epochs, maximum 10–20 epochs).

Performance is stable across all four and across all tasks on both benchmarks. Curvature estimation is the least sensitive: varying the number of probes yields $\sim$0.2 mIoU variation, and changing the update interval produces less than 0.3 mIoU degradation. Depth and surface normals on NYUv2 are equally unaffected, with AbsErr varying by only 0.0007 across probe counts and by less than 0.0002 across update intervals, and mean surface normal angle shifting by at most $0.03^\circ$ and $0.10^\circ$ respectively. Cityscapes is even more stable under these two parameters: mIoU varies by only 0.24 across probe counts and 0.22 across update intervals, and depth AbsErr shifts by at most 0.0002 and 0.0001 respectively, consistent with its lower degree of task conflict. The task-copy learning rate scale is similarly robust, with less than 0.5 mIoU variation across all tested configurations; depth and surface normals on NYUv2 follow the same monotone trend as segmentation, with no task deviating from the overall pattern. On Cityscapes, depth AbsErr remains nearly flat across the entire tested range (0.0002 total variation), confirming that the task-copy learning rate acts as a global exploration control rather than a task-specific sensitivity. The synchronization interval is the most sensitive parameter: setting the minimum interval above 5 epochs degrades performance by up to $\sim$0.5 mIoU, while the maximum interval has negligible effect ($< 0.3$ mIoU); depth and surface normal metrics on NYUv2 degrade proportionally under the same conditions, with no task exhibiting isolated sensitivity, and Cityscapes follows the same pattern. Overall, the maximal deviation across all tested configurations remains within 0.5 mIoU, indicating that LOGIC is highly robust and does not require careful hyperparameter tuning to achieve strong performance.

**Selected hyperparameter values.** The configuration used to produce all main-text and appendix results (Tables 1, 2, 3, 4, 6, 7, 8, and 9) is kept fixed across architectures and benchmarks, with no task, or dataset-specific tuning. We use $M = 8$ Hutchinson probes for trace estimation; a curvature update interval of $\Delta_{\text{curv}} = 100$ training steps; a task-copy learning-rate scale that decays linearly from 1.0 at the start of training to 0.4 at the end; and a synchronization interval that grows linearly from 1 epoch to 15 epochs over the course of training. For the curvature EMA smoothing coefficients we use $\alpha = 0.95$ for the per-task curvature baseline $\bar{\tau}_k$ and $\beta_c = 0.99$ for the centroid weights $\hat{c}_k$. These values are used unchanged in every experiment reported in the paper.

**On the choice of curvature signal.** The Hessian trace is indefinite-sensitive: in a non-convex landscape large positive and negative eigenvalues can cancel (Dauphin et al., 2014; Li et al., 2018), so a small $|\tau_k|$ does not by itself certify a flat region, and quantities that do not cancel, such as $\|H_k\|_F$, the leading eigenvalue, or a Gauss–Newton surrogate (Martens & Grosse, 2015), would measure local conditioning more faithfully. We use the trace because Hutchinson probes (Hutchinson, 1989) make it available at negligible cost at the frequency we require, and because the per-task EMA of Eq. (3) asks a weaker question than flatness: $\rho_k$ compares a task's current curvature against its own recent history, so a systematic cancellation shared with that history is absorbed by the baseline rather than mistaken for stability. We do not claim the trace is the best available signal, only that it is inexpensive and sufficient here; Table 4 bounds what it contributes.

### A.6 The Full Algorithm

Algorithm 1 summarizes the per-iteration logic of LOGIC. The algorithm maintains three groups of parameters: the shared encoder $\theta$, the decoder heads $\phi$, and the task-specific copies $\{\theta_k\}_{k=1}^K$ that evolve along independent loss landscapes. Each iteration proceeds through four logical steps that together implement the geometric-coordination principle of Section 3.

**Step 1** advances every task copy $\theta_k$ along its own task loss using its dedicated optimizer, producing $K$ task-adapted parameter positions. **Step 2** refreshes the curvature signal once every $\Delta_{\text{curv}}$ steps: it re-estimates each per-task Hessian trace via Hutchinson's method, updates the per-task EMA baseline $\bar{\tau}_k$, converts the result into the relative-curvature signal $\rho_k$, and feeds the inverted relative curvature into the outer EMA that defines the smoothed centroid weights $\hat{c}_k$. **Step 3** forms the curvature-weighted centroid $\bar{\theta}$ and the pseudo-gradient $g = \theta - \bar{\theta}$, the parameter-space signal

that drives the shared encoder. **Step 4** computes decoder gradients $\nabla_\phi$ from the joint loss $\mathcal{L}_{\text{joint}}$ and issues two optimizer updates: the decoder heads advance via their backpropagated gradients, while the shared encoder advances along $g$ rather than along any task gradient. This decoupling is the operational instantiation of LOGIC's core idea: decoders learn their tasks via standard backpropagation, while the encoder is moved entirely by geometric coordination of the task-specific copies.

When a synchronization step is scheduled, all task copies are reset in place to the current shared encoder, preserving each task optimizer's internal state (e.g. momentum, second-moment estimates) across the reset so that the post-sync trajectories resume from a coherent starting point rather than from a cold start.

**Algorithm 1:** LOGIC, one training step

**Input:** Shared encoder $\theta$; decoder heads $\phi$; task copies $\{\theta_k\}_{k=1}^K$; task optimizers $\{\mathrm{Opt}_k\}$; shared optimizer $\mathrm{Opt}$

// Initialization (once, before first iteration)
$\hat{c}_k \leftarrow 1/K \;\; \forall k$ // uniform centroid weights
$\bar{\tau}_k \leftarrow |\tau_k|$ on first curvature update // per-task EMA baseline
// Step 1: Independent per-task update
**for** $k = 1, \ldots, K$ **do**
  $\theta_k \leftarrow \mathrm{Opt}_k(\theta_k,\ \nabla_{\theta_k} \mathcal{L}_k(\theta_k))$
**end**
// Step 2: Curvature estimation (every $\Delta_{\text{curv}}$ steps)
**if** *curvature update is due* **then**
  **for** $k = 1, \ldots, K$ **do**
    $\tau_k \leftarrow \mathrm{tr}\big(\nabla^2_{\theta_k} \mathcal{L}_k(\theta_k)\big)$ // Hutchinson estimator
    $\bar{\tau}_k \leftarrow \alpha\, \bar{\tau}_k + (1-\alpha)\, |\tau_k|$ // per-task EMA baseline
    $\rho_k \leftarrow |\tau_k| / (\bar{\tau}_k + \varepsilon)$ // relative curvature
  **end**
  $\hat{c}_k \leftarrow \beta_c\, \hat{c}_k + (1-\beta_c)\, \dfrac{1/(\rho_k+\varepsilon)}{\sum_j 1/(\rho_j+\varepsilon)}$ // invert and smooth
**end**
// Step 3: Curvature-weighted centroid and pseudo-gradient
$\bar{\theta} \leftarrow \sum_{k=1}^K \hat{c}_k\, \theta_k$
$g \leftarrow \theta - \bar{\theta}$
// Step 4: Decoder update and pseudo-gradient injection
$\mathcal{L}_{\text{joint}} \leftarrow \sum_k \mathcal{L}_k(\theta, \phi)$
$\nabla_\phi \leftarrow \partial \mathcal{L}_{\text{joint}} / \partial \phi$ // decoder gradients via backprop
$\phi \leftarrow \mathrm{Opt}(\phi,\ \nabla_\phi)$ // decoder heads step
$\theta \leftarrow \mathrm{Opt}(\theta,\ g)$ // encoder steps via pseudo-gradient
**if** *synchronization is due* **then**
  $\theta_k \leftarrow \theta \;\; \forall k$ // in-place reset; preserves optimizer state
**end**

### A.7 PASCAL-CONTEXT BENCHMARK

We further evaluate LOGIC on **PASCAL-Context** (Mottaghi et al., 2014; Everingham et al., 2010), a five-task dense-prediction benchmark of $4{,}998$ training and $5{,}105$ testing images covering semantic segmentation, human-part segmentation, saliency, surface normals, and edge detection; the saliency and normal labels are distilled following Maninis et al. (2019). With five heterogeneous objectives spanning semantic, structural, and boundary predictions, this benchmark introduces stronger and more diverse task interactions than the three-task NYUv2 and two-task Cityscapes settings of the main text, making it a demanding test of cross-task coordination. Following standard practice on this benchmark (Maninis et al., 2019; Vandenhende et al., 2022), we report a single metric per task: mIoU for segmentation, human parts, and saliency, mean angular error (mErr) for surface normals,

and IoU for edge detection. As before, overall performance is summarized with MR and $\Delta m\%$ (Liu et al., 2021a) averaged over three shared random seeds.

Table 6: **PASCAL-Context: five-task scene understanding under HPS.** Results are averaged over three random seeds, each trained for 300 epochs. The backbone was the widely-used ResNet-50 (He et al., 2016), pretrained on ImageNet (Deng et al., 2009). $\uparrow$ / $\downarrow$ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all metrics (lower is better). **Bold** denotes the best result per column; underline the second best.

| Method | SemSeg (mIoU)↑ | Parts (mIoU)↑ | Sal (mIoU)↑ | Normals (mErr)↓ | Edge (IoU)↑ | MR ↓ | $\Delta m\%$ ↓ |
|---|---|---|---|---|---|---|---|
| STL | 61.20 | 58.06 | 66.19 | 12.95 | 14.32 | – | – |
| EW | 58.10 | 57.11 | 65.79 | 14.91 | 13.35 | 10.80 | 5.83 $\pm$1.838 |
| UW (Kendall et al., 2018) | 65.56 | 60.58 | 66.48 | 13.87 | 14.80 | 5.00 | $-$1.64 $\pm$1.505 |
| GradNorm (Chen et al., 2018) | 59.39 | 56.06 | 66.93 | 13.46 | 15.07 | 6.20 | 0.79 $\pm$0.989 |
| DWA (Liu et al., 2019) | 62.32 | 57.24 | 65.92 | 14.27 | 13.26 | 9.80 | 3.52 $\pm$2.074 |
| MGDA (Sener & Koltun, 2018) | 45.11 | 48.82 | 62.61 | 13.74 | 13.91 | 10.20 | 11.31 $\pm$0.749 |
| MoCo (Fernando et al., 2023) | 38.83 | 42.86 | 59.77 | 16.24 | 11.53 | 14.00 | 23.46 $\pm$0.625 |
| PCGrad (Yu et al., 2020) | 63.05 | 58.57 | 66.90 | 14.03 | 13.32 | 7.00 | 2.06 $\pm$1.779 |
| CAGrad (Liu et al., 2021a) | 62.16 | 57.91 | 66.51 | 13.88 | 14.07 | 7.40 | 1.43 $\pm$1.986 |
| IMTL-G (Liu et al., 2021b) | 62.72 | 60.72 | 66.78 | **13.25** | 15.31 | 3.60 | $-$2.51 $\pm$0.383 |
| Nash-MTL (Navon et al., 2022) | 62.38 | 58.44 | 65.34 | 13.75 | 13.82 | 7.80 | 1.67 $\pm$0.683 |
| RLW (Lin et al., 2022) | 63.26 | 57.66 | 67.16 | 14.28 | 12.96 | 7.80 | 3.11 $\pm$1.267 |
| FAMO (Liu et al., 2023) | 48.58 | 49.12 | 65.44 | 13.91 | 13.80 | 10.60 | 9.63 $\pm$0.891 |
| FairGrad (Ban & Ji, 2024) | 65.09 | 60.46 | 66.82 | 13.38 | 16.05 | 3.20 | $-$4.05 $\pm$0.320 |
| **LOGIC (Ours)** | **66.43** | **60.73** | **67.32** | 13.49 | **16.39** | **1.60** | **$-$5.03 $\pm$ 0.317** |

As shown in Table 6, LOGIC achieves the best overall performance, obtaining the leading MR (1.60) and $\Delta m\%$ ($-5.03$) and the top score on four of the five tasks (segmentation, human parts, saliency, and edge). It trails on surface normals, where the single-task reference, IMTL-G (Liu et al., 2021b), FairGrad (Ban & Ji, 2024) and GradNorm (Chen et al., 2018) lead, consistent with the task-specific nature of normal estimation already observed on NYUv2. Among the baselines, FairGrad (Ban & Ji, 2024) and IMTL-G are again the most competitive, mirroring their behavior on NYUv2, yet LOGIC improves the overall $\Delta m\%$ over the strongest of them by a clear margin. LOGIC additionally attains the lowest cross-seed standard deviation ($\pm 0.317$), narrowly ahead of FairGrad ($\pm 0.320$) and well below the remaining methods. These results confirm that the stability and positive transfer of LOGIC extend to a distinct and more heterogeneous five-task benchmark, beyond the settings considered in the main text.

### A.8 Task-Scaling of Runtime and Memory on PASCAL-Context

The cost–accuracy analysis of Appendix A.3 fixes the number of tasks; here we instead vary it, measuring how the per-epoch training time and peak GPU memory of each method scale as objectives are added. We use the five-task PASCAL-Context benchmark of Appendix A.7, incrementally activating its objectives from $K{=}1$ (segmentation only) to $K{=}5$ (segmentation, human parts, saliency, edge, and surface normals), and profile every method under an identical HPS / ResNet-50 protocol at a fixed batch size of 8 on a single **NVIDIA RTX 4090**. Both quantities are normalized to the equal-weighting (EW) baseline *at the same* $K$, so a value of $c\times$ denotes a $c$-fold overhead relative to plain joint training at that task count. Two baselines are defined only for $K{\geq}2$: MGDA (Sener & Koltun, 2018) seeks a minimum-norm vector in the convex hull of task gradients, and Nash-MTL (Navon et al., 2022) frames aggregation as a multi-player bargaining game; neither admits a well-posed single-task instance, so both curves begin at $K{=}2$.

**Runtime.** Figure 10(a) reports the result. LOGIC's time overhead grows sublinearly and, notably, more slowly than that of the gradient-manipulation baselines: it rises from $1.90\times$ at $K{=}1$ to only

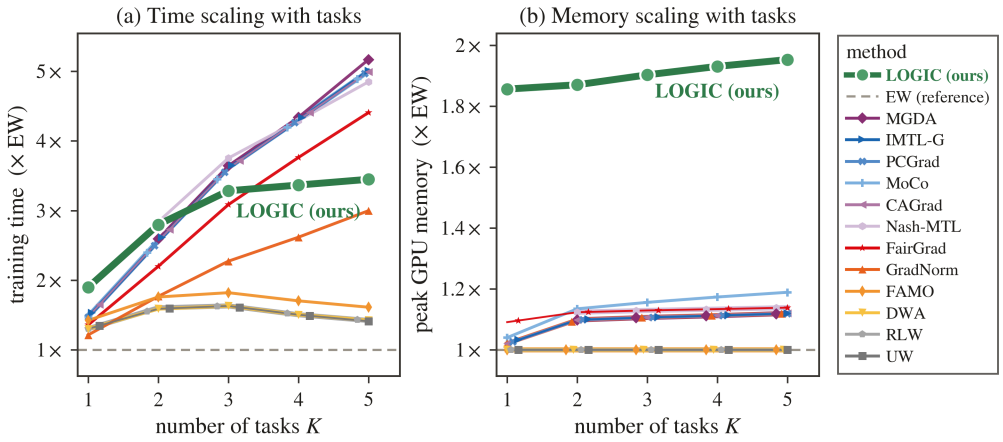



Figure 10: **Task-scaling of training cost on PASCAL-Context** (HPS, ResNet-50, batch size 8, single NVIDIA RTX 4090). **(a)** per-epoch training time and **(b)** peak GPU memory as a function of the number of tasks $K$, each normalized to the EW baseline ($1\times$, dashed) at the same $K$. LOGIC (green) carries a larger but nearly constant memory overhead, while its time-scaling slope is *gentler* than that of the gradient-manipulation family. MGDA and Nash-MTL are undefined at $K{=}1$ and their curves begin at $K{=}2$.

$3.45\times$ at $K{=}5$, where its increments have all but plateaued. Over the same range MGDA (Sener & Koltun, 2018), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), IMTL-G (Liu et al., 2021b), and MoCo (Fernando et al., 2023) all reach $\approx 5\times$, and Nash-MTL (Navon et al., 2022) $4.85\times$, at $K{=}5$. These methods require $K$ separate backward passes through the shared parameters to recover per-task gradients, a cost that compounds as tasks accumulate; empirically their wall-clock overhead therefore grows faster with $K$ than LOGIC's, whose per-copy passes are comparatively light. Only the pure loss-weighting methods (UW, DWA, RLW, FAMO; $1.4$–$1.6\times$) and GradNorm ($3.00\times$, albeit with large run-to-run variance) remain below LOGIC, none of which actively resolves directional conflict. This extends the fixed-$K$ finding of Appendix A.3, that LOGIC is the fastest conflict-resolving method, to the multi-task-scaling regime.

**Memory.** LOGIC carries the largest memory footprint, as expected from the $(1{+}K)\times$ encoder copies it maintains during training, yet this overhead stays essentially flat as tasks are added (Figure 10(b)): it varies only from $1.86\times$ at $K{=}1$ to $1.95\times$ at $K{=}5$, since the copy footprint and the EW baseline grow with $K$ in proportion. All gradient-manipulation baselines sit between $1.12\times$ and $1.19\times$, and the loss-weighting methods remain at $1\times$. Crucially, this overhead is confined to training: at inference LOGIC reduces to a single shared encoder with no additional memory or latency, so its scaling cost is paid only while learning and vanishes entirely at deployment.

### A.9 PAWS-X Benchmark

All benchmarks considered so far are dense-prediction vision problems in which tasks differ in output structure and loss family. To examine whether the advantages of LOGIC persist in a fundamentally different regime, we further evaluate on **PAWS-X** (Yang et al., 2019), the cross-lingual extension of the adversarial paraphrase-identification dataset PAWS (Zhang et al., 2019b), as distributed within the XTREME benchmark (Hu et al., 2020). We follow the multilingual multi-task protocol introduced by Lin et al. (2022) and implemented in LibMTL (Lin & Zhang, 2023): each language is treated as a separate task, and the four languages with the most training data (English, German, Spanish, and Chinese) form a four-task problem over a shared multilingual BERT (mBERT) encoder (Devlin et al., 2019) with per-language classification heads. Unlike NYUv2, Cityscapes, and PASCAL-Context, this is a *multi-input* setting: every task receives its own data stream, so the shared encoder must accommodate four distinct input distributions rather than four distinct output structures. Performance is measured by binary classification accuracy on the test set of each language and summarized, as before, with MR and $\Delta m\%$ (Liu et al., 2021a) averaged over three shared

random seeds. All methods, including LOGIC, follow an identical training protocol of 100 epochs, using the same backbone initialization and data splits, without any method-specific tuning.

This benchmark also isolates a distinct aspect of the mechanism. All four tasks optimize the same binary cross-entropy objective, so their absolute Hessian traces are commensurable and the loss-family offset analyzed in Appendix A.2 is absent by construction. The per-task EMA normalization of Eq. (3) therefore acts purely as a temporal-stability signal, weighting each language according to how well-conditioned its trajectory currently is relative to its own optimization history, rather than as a corrector for incommensurable loss scales.

Table 7: **PAWS-X: four-language paraphrase identification under HPS.** Each language is treated as a separate task in the multi-input setting of LibMTL (Lin & Zhang, 2023). Results are averaged over three random seeds, each trained for 100 epochs, using a shared multilingual BERT encoder (Devlin et al., 2019) with per-language classification heads. ↑ / ↓ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all four languages (lower is better). **Bold** denotes the best result per column; underline the second best.

| Method | en (Acc)↑ | de (Acc)↑ | es (Acc)↑ | zh (Acc)↑ | MR ↓ | $\Delta m\%$ ↓ |
|---|---|---|---|---|---|---|
| STL | 90.26 | 82.35 | 85.50 | 81.17 | – | – |
| EW | <u>92.85</u> | <u>87.57</u> | 88.80 | <u>82.60</u> | <u>3.00</u> | <u>−3.71 ±0.467</u> |
| UW (Kendall et al., 2018) | 91.68 | 86.68 | <u>89.00</u> | 81.50 | 5.75 | −2.83 ±0.927 |
| GradNorm (Chen et al., 2018) | 90.90 | 86.75 | 88.92 | 81.82 | 5.50 | −2.71 ±0.431 |
| DWA (Liu et al., 2019) | 92.03 | 87.27 | 88.82 | **82.82** | 3.25 | −3.46 ±0.616 |
| MGDA (Sener & Koltun, 2018) | 90.57 | 85.25 | 87.15 | 79.80 | 11.50 | −1.03 ±9.622 |
| MoCo (Fernando et al., 2023) | 72.84 | 69.71 | 71.18 | 68.37 | 14.00 | 16.79 ±0.548 |
| PCGrad (Yu et al., 2020) | 92.60 | 87.05 | 88.83 | 81.67 | 4.38 | −3.20 ±0.382 |
| CAGrad (Liu et al., 2021a) | 89.85 | 83.92 | 86.67 | 79.58 | 12.75 | −0.22 ±1.077 |
| IMTL-G (Liu et al., 2021b) | 90.60 | 85.95 | 87.63 | 79.30 | 10.50 | −1.23 ±1.586 |
| Nash-MTL (Navon et al., 2022) | 91.82 | 85.78 | 88.50 | 81.62 | 7.00 | −2.49 ±1.329 |
| RLW (Lin et al., 2022) | 91.60 | 85.68 | 88.48 | 81.80 | 7.25 | −2.45 ±0.606 |
| FAMO (Liu et al., 2023) | 91.32 | 85.38 | 88.02 | 81.32 | 9.75 | −2.00 ±0.503 |
| FairGrad (Ban & Ji, 2024) | 91.58 | 84.67 | 88.28 | 81.67 | 8.88 | −2.04 ±1.243 |
| **LOGIC (Ours)** | **93.33** | **87.70** | **91.38** | 82.15 | **1.50** | **−4.50 ± 0.332** |

As shown in Table 7, LOGIC attains the best overall performance, obtaining the leading MR (1.50) and $\Delta m\%$ ($-4.50$) and the highest accuracy on English, German, and Spanish. The improvements over single-task learning are particularly pronounced on the non-English languages, $+5.88$ points on Spanish and $+5.35$ on German, indicating that the shared encoder reaches a representation that serves the individual languages substantially better than any gradient-aggregation compromise. On Chinese, the typologically most distant language from the remaining three, LOGIC (82.15) ranks third behind DWA (82.82) and EW (82.60) while still exceeding STL by roughly one point; the balance achieved across all four languages nevertheless remains the strongest of any method, as reflected in both summary measures.

The relative ordering of the baselines differs markedly from the dense-prediction benchmarks. Simple joint training and loss-weighting methods are comparatively strong here (EW $-3.71$, DWA $-3.46$, PCGrad $-3.20$), whereas several gradient-manipulation methods that lead on NYUv2 and PASCAL-Context become noticeably weaker (FairGrad $-2.04$, IMTL-G $-1.23$, MGDA $-1.03$, CAGrad $-0.22$). This is consistent with the structure of the multi-input regime: the tasks share an identical objective and differ primarily in input distribution, so their gradients conflict far less in *direction* than in dense prediction, and machinery designed to resolve directional conflict yields limited benefit while still constraining the shared update. MoCo again fails to reach a competitive solution ($+16.79$), mirroring its behavior on the vision benchmarks. LOGIC, by contrast, leads on both families of benchmark: because its shared direction is not formed by aggregating task gradients evaluated at a single point, its advantage does not rest on the presence of strong directional conflict,

but on coordinating fully task-adapted solutions in parameter space. These results confirm that the benefits of LOGIC extend beyond dense prediction to multilingual language understanding, across a different modality, backbone family, and task-interaction regime.

### A.10 NYUv2 Training From Scratch

Table 8: **NYUv2: three-task scene understanding under HPS, trained from scratch**. Results are averaged over three random seeds, each trained for 400 epochs. The backbone is a ResNet-50 (He et al., 2016) trained from scratch (no ImageNet pre-training). ↑ / ↓ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all NYUv2 metrics (lower is better). **Bold** denotes the best result per column; underline denotes the second best.

| Method | Segmentation | | Depth | | Surface Normal | | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | mIoU↑ | PixAcc↑ | AbsErr↓ | RelErr↓ | Mean↓ | Median↓ | $< 11.25^\circ$ ↑ | $< 22.5^\circ$ ↑ | $< 30^\circ$ ↑ | MR↓ | $\Delta m\%$ ↓ |
| STL | 38.44 | 63.66 | 0.5101 | 0.2015 | 25.6206 | 17.9720 | 33.34 | 58.47 | 68.88 | – | – |
| EW | 42.02 | 66.65 | 0.4649 | 0.1804 | 26.6316 | 19.7532 | 30.08 | 55.05 | 66.34 | 7.44 | $-0.02 \pm 2.083$ |
| GradNorm (Chen et al., 2018) | 41.56 | 66.24 | 0.4682 | 0.1806 | 25.9560 | 18.7391 | 31.67 | 57.00 | 67.92 | 6.56 | $-1.81 \pm 2.857$ |
| DWA (Liu et al., 2019) | 41.63 | 66.23 | 0.4630 | 0.1803 | 26.7423 | 19.8695 | 29.90 | 54.83 | 66.15 | 8.33 | $0.37 \pm 1.824$ |
| PCGrad (Yu et al., 2020) | 42.09 | 66.88 | 0.4605 | 0.1808 | 26.5665 | 19.6036 | 30.36 | 55.32 | 66.53 | 6.67 | $-0.45 \pm 1.912$ |
| CAGrad (Liu et al., 2021a) | 42.65 | 67.18 | 0.4549 | 0.1760 | 25.3663 | 18.0690 | 32.82 | 58.37 | 69.11 | 4.00 | $-4.33 \pm 2.213$ |
| IMTL-G (Liu et al., 2021b) | 43.24 | 67.72 | 0.4490 | 0.1739 | 25.0933 | 17.8514 | 33.25 | 58.83 | 69.56 | 3.00 | $-5.40 \pm 1.023$ |
| Nash-MTL (Navon et al., 2022) | 41.78 | 66.56 | 0.4600 | 0.1776 | 26.3830 | 19.4523 | 30.58 | 55.64 | 66.87 | 6.00 | $-0.85 \pm 0.987$ |
| FairGrad (Ban & Ji, 2024) | **43.87** | 68.32 | 0.4440 | 0.1713 | 24.8555 | 17.4843 | 34.01 | 59.54 | 70.08 | 1.89 | $-6.74 \pm 0.823$ |
| **LOGIC (Ours)** | 43.48 | **68.35** | **0.4249** | **0.1652** | **24.2702** | **17.0779** | **34.80** | **60.41** | **70.94** | **1.11** | $\mathbf{-8.46 \pm 0.723}$ |

Table 8 shows that even when trained from scratch, LOGIC substantially improves the most challenging tasks while preserving performance on segmentation mIoU. This translates into a substantial improvement in global performance, achieving the lowest MR and $\Delta m\%$, and outperforming all baselines in overall Pareto efficiency. LOGIC also maintains the lowest cross-seed standard deviation ($\pm 0.723$) in this more challenging setting, where several gradient-based baselines exhibit standard deviations exceeding $1.0$ and up to $4\times$ higher, further confirming its optimization stability even without pretrained initialization.

### A.11 NYUv2 with a ViT-B/16 Backbone

Table 9 shows that even under a ViT-B/16 backbone, LOGIC preserved the same superiority observed under the convolutional ResNet-50 encoder. LOGIC achieved the best overall performance on NYUv2, obtaining the leading $\Delta m\%$ ($-17.74$) and Mean Rank (1.56), while the strongest gradient-based baseline (FairGrad (Ban & Ji, 2024)) reached only $-14.13$. The largest gains were observed in segmentation, where LOGIC raised mIoU to 59.66, substantially exceeding both STL (44.54) and the best competing method (GradNorm (Chen et al., 2018), 53.05), while simultaneously achieving the best results on both depth metrics. These observations indicate that the advantages of LOGIC do not depend on a specific backbone family, but instead stem from its ability to resolve cross-task conflicts at the optimization level rather than at the level of shared features. Even under an attention-based backbone whose optimization geometry differs substantially from convolutional networks, the core principles of LOGIC continue to yield consistent and well-balanced positive transfer. Cross-seed standard deviation remains the lowest ($\pm 0.191$), further confirming the optimization stability of LOGIC under a transformer-based backbone.

**On the single-task references.** The single-task runs in Table 9 use the same protocol as the multi-task runs, with no per-setting tuning, as elsewhere in the paper. The gap between them is nonetheless larger than under ResNet-50, and we read this as a property of the backbone rather than an artifact of the protocol. ViT-B/16 lacks the convolutional inductive biases that make ResNets sample-efficient, and NYUv2 provides only 795 training images; a single-task run therefore extracts supervision from one set of labels per image, while a multi-task run extracts it from three. Under a backbone

Table 9: **NYUv2: three-task scene understanding under HPS with a ViT-B/16 backbone.** Results are averaged over three random seeds, each trained for 300 epochs. The backbone was the widely-used ViT-B/16 (Dosovitskiy et al., 2021), pretrained on ImageNet (Deng et al., 2009). $\uparrow$ / $\downarrow$ indicates higher/lower is better. MR and $\Delta m\%$ summarize overall performance across all NYUv2 metrics (lower is better). **Bold** denotes the best result per column; underline denotes the second best.

| Method | Segmentation | | Depth | | Surface Normal | | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | mIoU↑ | PixAcc↑ | AbsErr↓ | RelErr↓ | Mean↓ | Median↓ | $< 11.25^\circ \uparrow$ | $< 22.5^\circ \uparrow$ | $< 30^\circ \uparrow$ | MR↓ | $\Delta m\% \downarrow$ |
| STL | 44.54 | 68.21 | 0.6227 | 0.2544 | 23.5476 | 15.7590 | 37.88 | 62.78 | 72.44 | – | – |
| EW | 49.79 | 72.52 | 0.4008 | 0.1637 | 23.9278 | 16.3443 | 36.48 | 61.66 | 71.66 | 10.33 | $-8.61 \pm 1.935$ |
| UW (Kendall et al., 2018) | 50.72 | 73.13 | 0.3953 | 0.1599 | 23.7408 | 16.1654 | 37.02 | 62.01 | 71.97 | 8.00 | $-9.69 \pm 2.135$ |
| GradNorm (Chen et al., 2018) | 53.05 | 74.73 | 0.3854 | 0.1566 | 22.5513 | 15.3073 | 38.57 | 64.19 | 74.05 | 3.11 | $-13.18 \pm 6.263$ |
| DWA (Liu et al., 2019) | 49.20 | 72.28 | 0.4040 | 0.1678 | 23.9323 | 16.1629 | 37.08 | 61.85 | 71.67 | 9.89 | $-8.53 \pm 1.708$ |
| MGDA (Sener & Koltun, 2018) | 40.74 | 65.15 | 0.4922 | 0.1951 | 23.3141 | 15.6821 | 38.07 | 62.99 | 72.73 | 9.11 | $-3.77 \pm 5.392$ |
| MoCo (Fernando et al., 2023) | 16.12 | 42.57 | 0.8476 | 0.3276 | 37.7848 | 35.6437 | 11.63 | 29.18 | 41.14 | 14.00 | $57.66 \pm 5.369$ |
| PCGrad (Yu et al., 2020) | 51.51 | 73.45 | 0.3957 | 0.1599 | 23.3483 | 15.9309 | 37.46 | 62.52 | 72.46 | 6.50 | $-10.58 \pm 2.022$ |
| CAGrad (Liu et al., 2021a) | 51.69 | 73.74 | 0.3912 | 0.1587 | 22.5162 | 15.0560 | 39.26 | 64.60 | 74.21 | 3.33 | $-12.98 \pm 1.917$ |
| IMTL-G (Liu et al., 2021b) | 50.40 | 73.24 | 0.3961 | 0.1614 | 23.1026 | 15.4148 | 38.41 | 63.80 | 73.51 | 6.00 | $-11.34 \pm 0.366$ |
| Nash-MTL (Navon et al., 2022) | 50.29 | 72.91 | 0.3953 | 0.1583 | 23.5610 | 15.9537 | 37.40 | 62.42 | 72.29 | 7.28 | $-10.08 \pm 0.555$ |
| RLW (Lin et al., 2022) | 45.46 | 69.06 | 0.4246 | 0.1753 | 24.9617 | 17.3401 | 34.81 | 59.50 | 69.66 | 12.11 | $-3.67 \pm 2.493$ |
| FAMO (Liu et al., 2023) | 43.29 | 67.63 | 0.4534 | 0.1841 | 24.2010 | 16.1730 | 36.91 | 61.82 | 71.52 | 11.67 | $-4.49 \pm 2.596$ |
| FairGrad (Ban & Ji, 2024) | 52.54 | 74.68 | 0.3870 | 0.1572 | 22.4301 | **14.7598** | **39.88** | **65.32** | 74.80 | 2.11 | $-14.13 \pm 0.398$ |
| **LOGIC (Ours)** | **59.66** | **79.54** | **0.3423** | **0.1395** | **22.3020** | 15.1108 | 38.89 | 64.95 | **74.91** | **1.56** | $\mathbf{-17.74 \pm 0.191}$ |

this data-hungry the additional supervision is worth more, which is consistent with every multi-task method in the table improving on the single-task reference by a wide margin, not only LOGIC. The consequence for reading the table is that $\Delta m\%$ is measured against a reference that the transformer setting disadvantages, so the absolute values are larger here than under ResNet-50 and the informative comparison is the ordering among the multi-task methods.