CURVATURE-MOTIVATED SPARSE TRANSFORMATION ENHANCED LORA REPRESENTATION AGGREGATION FOR PARETO FRONT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Pareto front learning (PFL) is able to capture diverse tradeoffs among conflicting tasks in multi-task learning (MTL). Recent approaches adopt linear interpolation of low-rank adapters (LoRA) to offer a scalable parameterization of the Pareto front. However, these methods typically assume that tradeoff solutions lie within the convex hull of task-specific parameters, thereby potentially overlooking the nonlinear geometry of the Pareto manifold shaped by the local curvatures of the task objectives. Consequently, the limited expressiveness of linear interpolation may cause these methods to fall short of the performance gains expected from increasing the LoRA rank. To address these issues, we propose a curvature-motivated PFL method, termed Sparse Transformation enhanced Representation Aggregation (STeRA) from LoRA Modules. Specifically, given the high cost of curvature modeling in parameter space, we instead derive a sample-wise merging strategy in representation space by minimizing the tradeoff-weighted sum of local quadratic objective approximations, yielding a curvature-weighted aggregation of task-specific representations. To avoid materializing multiple task-specific curvature matrices, we establish, under mild conditions, the existence of a transformation from the linearly interpolated representations to the derived curvature-aware representation. Rather than explicitly constructing this dense transformation, we approximate its effect with a learnable sparse transformation that performs independent planar rotations over adjacent representation dimensions, parameterized by sample and tradeoff conditioned rotation angles. By allocating a portion of the LoRA rank budget to modeling these angles, STeRA provides the flexibility to model the underlying nonlinear geometry beyond linear LoRA interpolation. Extensive evaluations across diverse MTL datasets demonstrate the superiority of our method in both performance and parameter efficiency.

1 INTRODUCTION

Multi-task learning (MTL) jointly solves multiple related tasks within a unified model by exploiting shared knowledge, with the aim of reducing memory and computational costs while potentially improving generalization over separately trained task-specific models (Caruana, 1997; Ruder, 2017; Crawshaw, 2020; Vandenhende et al., 2021). In practical settings, conflicting task objectives often require tradeoffs, as improving one task may come at the expense of others, while mismatched objective scales can lead to imbalanced training dynamics (Crawshaw, 2020). For instance, the capacitated vehicle routing problem may involve balancing the total length of all routes against the length of the longest route (Lin et al., 2022b), while large language model (LLM) alignment commonly considers the tradeoff between helpfulness and harmlessness (Zhong et al., 2024). It is therefore important to explicitly account for the tradeoffs among competing tasks in MTL.

MTL can be formulated within a multi-objective optimization (MOO) framework (Sener & Koltun, 2018). Several methods seek a single tradeoff solution through gradient manipulation or loss reweighting (Sener & Koltun, 2018; Liu et al., 2021a; 2022), while some methods approximate the Pareto front using a discrete set of tradeoff solutions (Lin et al., 2019; Mahapatra & Rajan, 2020; Ma et al., 2020), limiting adaptation to unseen or varying tradeoffs. Pareto front learning (PFL) instead offers a continuous, preference-conditioned approximation of the Pareto front (Navon et al.,

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.

2021; Ruchte & Grabocka, 2021). Pareto hypernetwork-based methods map preferences to model parameters but incur large memory overhead (Navon et al., 2021), whereas COSMOS reduces this cost by augmenting the model input with the preference, at the expense of limited parameterization capacity and potentially suboptimal performance (Ruchte & Grabocka, 2021).

To circumvent these limitations in PFL, Pareto Manifold Learning (PaMaL) parameterizes the Pareto front by linearly interpolating an ensemble of task-specific model parameters according to a preference vector, while weighting the task objectives using the same interpolation coefficients during training (Dimitriadis et al., 2023). To alleviate the memory overhead of maintaining separate task-specific models and promote knowledge sharing across tasks, subsequent approaches adopt a preference-weighted linear combination of low-rank adapters (LoRA) (Hu et al., 2022) built upon a shared base model (Chen & Kwok, 2024; Dimitriadis et al., 2025). However, these methods typically assume that tradeoff solutions lie within the convex hull of the task-specific parameters, thereby constraining the attainable solution space and potentially preventing faithful modeling of the true Pareto manifold.

To illustrate the limitation of restricting tradeoff solutions to linear interpolations of task-specific parameters, we consider optimizing a preference-weighted sum of task objectives over the full parameter space. Its optimum depends not only on the task-specific optima, but also on the directional penalties for deviating from them. For a given objective, a direction with large local curvature incurs a greater objective increase for a fixed displacement, causing the objective to more strongly resist movement along that direction. Conversely, along directions with smaller curvature, the objective can yield more readily to competing objectives. *We therefore argue that linear aggregation of task-specific parameters should be augmented with curvature-aware transformations to better account for the nonlinear geometry of the underlying Pareto manifold.* However, modeling curvature with respect to neural network parameters is computationally expensive because curvature matrices scale quadratically with parameter dimensionality. Layer representations are substantially lower-dimensional, making curvature modeling more tractable. Moreover, explicitly constructing task-specific curvature matrices further multiplies the computational burden. *We thus advocate curvature-aware combination in the representation space while avoiding explicit construction of task-specific curvature matrices.*

In this paper, we propose a curvature-motivated PFL method, termed Sparse Transformation enhanced Representation Aggregation (STeRA) from LoRA modules. Specifically, to account for the nonlinear geometry of the underlying Pareto manifold shaped by the local curvatures of the task objectives, we characterize each objective using a local quadratic approximation, in which curvature captures the directional penalty for deviations from the task-specific solution. Since explicitly modeling curvature with respect to model parameters is computationally prohibitive, we instead formulate the local objective over lower-dimensional sample-wise representations. We then derive a representation merging strategy by minimizing the tradeoff-weighted sum of these local quadratic approximations. The resulting solution takes the form of a preference and curvature weighted aggregation of task-specific locally optimal representations, normalized by the total curvature. To avoid materializing multiple task-specific curvature matrices, we establish, under mild conditions, the existence of a transformation that maps the linearly interpolated representations to the derived curvature-aware representation. By transforming the linearly aggregated representations, our method allows the resulting solutions to deviate from the convex interpolation path, thereby providing flexibility to model the curvature-motivated nonlinear geometry of the Pareto manifold.

In practice, rather than explicitly constructing this dense transformation, we approximate its effect using a learnable sparse transformation that performs independent planar rotations over adjacent representation dimensions, parameterized by sample and tradeoff conditioned rotation angles. Following the preference-weighted aggregation used to construct the representation for a given sample, we introduce a parallel aggregation branch with an analogous structure to generate the rotation angles. Specifically, an additional set of task-specific LoRA modules produces angle embeddings from the same sample, which are aggregated according to the preference vector and then passed through a π-scaled $\tanh(\cdot)$ function to obtain the angle vector. Although these LoRA modules do not explicitly estimate task-specific curvature, their task-specific and preference-weighted structure is consistent with the derived curvature-aware formulation, in which each task contributes according to both its preference and local geometry information. Moreover, to accommodate odd-dimensional representations, including scalar outputs, we similarly parameterize a scaling transformation. We further allow a lightweight adjustment of the preference weights used for representation aggregation

to enlarge the attainable solution space. Extensive experimental results demonstrate the superiority of our method in terms of both performance and parameter efficiency.

2 Related Work

This Section briefly reviews three related topics: MTL with a single tradeoff solution, Pareto front approximation, and LoRA merging. A more detailed discussion is provided in Appendix C.

MTL with a Single Tradeoff Solution. We first review representative optimization methods for MTL that seek a single tradeoff solution among competing tasks. Existing approaches address conflicting objectives and mismatched loss scales through loss reweighting (Chen et al., 2018; Kendall et al., 2018; Liu et al., 2022), gradient manipulation (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2023), or their combination (Liu et al., 2021b; Lin et al., 2022a). In addition, Zhou et al. (2025b) leverages LoRA to identify an alternative loss-equivalent solution with reduced task conflicts. However, these methods are not designed to accommodate arbitrary user-specified preferences.

Pareto Front Approximation. Several methods approximate the Pareto front using discrete tradeoff solutions (Lin et al., 2019; Mahapatra & Rajan, 2020; Chen & Kwok, 2022), while Ma et al. (2020); Zhou et al. (2025c) further explore the vicinity of solutions or solution segment. Yet, these methods provide only partial or discrete approximations, whereas Pareto front learning enables a continuous, tradeoff-conditioned parameterization of the Pareto front. Pareto hypernetwork-based methods (Lin et al., 2020; Navon et al., 2021) map preferences to model parameters but incur high overhead, whereas Ruchte & Grabocka (2021); Dosovitskiy & Djolonga (2019) condition inputs or features on preferences with limited parameterization capacity. PaMaL (Dimitriadis et al., 2023) and its LoRA-based variants (Chen & Kwok, 2024; Dimitriadis et al., 2025) improve efficiency through preference-weighted parameter interpolation. However, these methods constrain tradeoff solutions to the convex hull of model parameters, potentially limiting faithful modeling of the Pareto manifold.

LoRA Merging. LoRA (Hu et al., 2022) enables parameter-efficient adaptation through low-rank updates to a pretrained model, while LoRA merging combines separately trained task-specific experts (Wu et al., 2023; Muqeeth et al., 2024), and some jointly trained multi-LoRA methods lack explicit expert-to-task correspondence (Dou et al., 2024; Liu & Luo, 2024). In contrast, our method adopts the paradigm of jointly training task-specific LoRA experts by aligning their interpolation weights with task-objective weights. Matena & Raffel (2022); Daheim et al. (2024); Nguyen et al. (2025) explicitly approximate Fisher or curvature matrices, or introduce learnable modules to model them in parameter space, incurring costly task-specific operations. Representation Surgery (Yang et al., 2024a;b; Wei et al., 2025; Wu et al., 2026) manipulates representations for expert merging but relies on pretrained task-specific experts for supervision, differing from our setting. In addition, Guo et al. (2025) introduces a sparse transformation of a similar form over representations, but each task-specific transformation repeatedly depends on other task representations, increasing computational cost. In contrast, our method performs sparse transformation for LoRA merging in the representation space without explicit task-specific transformation modeling. Instead, the angle embedding is generated through the aggregation of task-specific LoRA outputs, avoiding repeated task-specific operations for each experts while implicitly incorporating task-dependent transformation effects.

3 Preliminary

A multi-objective optimization (MOO) problem can be characterized by a 3-tuple $(\mathcal{D}, \boldsymbol{\ell}, T)$, where $T \geq 2$ is the number of tasks, $\mathcal{D}$ denotes the data space, and $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_T)$ collects the task-specific objectives. Each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ consists of an observation $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ and a label vector $\boldsymbol{y} = (y_1, \ldots, y_T) \in \boldsymbol{\mathcal{Y}} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_T$, where $y_i \in \mathcal{Y}_i$ and $\ell_i : \mathcal{Y}_i \times \mathcal{Y}_i \to \mathbb{R}$ correspond to task i. The aim is to learn a universal function $f_{\boldsymbol{\theta}} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$, parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, that jointly predicts $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_T) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$. For task i, the expected objective over the data distribution $p_{\mathcal{D}}$ is $\mathcal{L}_i(\boldsymbol{\theta}) = \mathbf{E}_{(\boldsymbol{x}, y_i) \sim p_{\mathcal{D}}}[\ell_i(y_i, \hat{y}_i)]$. The objective of the resulting MOO problem is defined as $\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \boldsymbol{\mathcal{L}}(\boldsymbol{\theta}) = (\mathcal{L}_1(\boldsymbol{\theta}), \ldots, \mathcal{L}_T(\boldsymbol{\theta}))$. When $f_{\boldsymbol{\theta}}$ is materialized as an N-layer neural network, its parameters are expressed as $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)})$, where $\boldsymbol{\theta}^{(n)} \in \boldsymbol{\Theta}^{(n)}$ denotes the parameters of the n-th layer. To characterize tradeoffs among conflicting task objectives, we introduce the notion of **Pareto optimality** as follows. For two solutions $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}$, $\boldsymbol{\theta}$ dominates $\boldsymbol{\theta}'$, denoted by $\boldsymbol{\theta} \succ \boldsymbol{\theta}'$, if $\mathcal{L}_i(\boldsymbol{\theta}) \leq \mathcal{L}_i(\boldsymbol{\theta}')$ for all $i \in \{1, \ldots, T\}$ and $\exists i$ such that $\mathcal{L}_i(\boldsymbol{\theta}) < \mathcal{L}_i(\boldsymbol{\theta}')$. A solution $\boldsymbol{\theta}$ is Pareto

optimal if $\nexists \boldsymbol{\theta}' \in \boldsymbol{\Theta}$ such that $\boldsymbol{\theta}' \succ \boldsymbol{\theta}$. The set of all Pareto-optimal solutions forms the **Pareto set**, whose image in the objective space is the **Pareto front**.

PaMaL parameterizes the Pareto front through a preference-weighted interpolation of task-specific model parameters, denoted as $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i \boldsymbol{\theta}_i$, where $\boldsymbol{\theta}_i$ denotes the task-specific parameters for task i, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$ denotes the preference vector satisfying $\alpha_i \geq 0$ and $||\boldsymbol{\alpha}||_1 = 1$. PaMaL minimizes $\mathbf{E}_{\boldsymbol{\alpha} \sim p_{\boldsymbol{\alpha}}}[L(\boldsymbol{\theta}(\boldsymbol{\alpha}), \boldsymbol{\alpha})]$ over the distribution $p_{\boldsymbol{\alpha}}$, where $L(\boldsymbol{\theta}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i \mathcal{L}_i(\boldsymbol{\theta}(\boldsymbol{\alpha}))$. This alignment between objective weights and task-specific model contributions implicitly assumes that each task-specific model corresponds to the Pareto-optimal solution for its one-hot preference vector. Equivalently, $\boldsymbol{\theta}(\boldsymbol{\alpha})$ can be expressed as $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \boldsymbol{\theta}_0 + \sum_{i=1}^{T} \alpha_i (\boldsymbol{\theta}_i - \boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_0 = (1/T) \sum_{i=1}^{T} \boldsymbol{\theta}_i$ denotes the shared base model that captures knowledge common across tasks, while $\boldsymbol{\theta}_i - \boldsymbol{\theta}_0$ represents the task-specific parameter difference for task i. This difference can be approximated by a LoRA expert $\mathbf{B}_i^{\text{exp}} \mathbf{A}_i^{\text{exp}}$, where $\mathbf{B}_i^{\text{exp}}$ and $\mathbf{A}_i^{\text{exp}}$ are low-rank matrices, as adopted in Low-Rank Pareto Manifold Learning (LORP-MAN) (Chen & Kwok, 2024) and Pareto Low-Rank Adapters (PaLoRA) (Dimitriadis et al., 2025). As shown in Figure 1, PaLoRA often yields limited hypervolume gains as the LoRA rank increases. This also motivates reallocating part of the LoRA rank budget to alternative components for performance improvement. Please refer to Appendix A for PaLoRA results across more datasets.

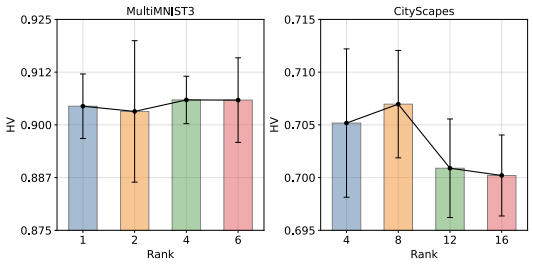


Figure 1: Hypervolume of PaLoRA as the LoRA rank increases across different datasets.

4 METHOD

In this Section, we address the limitations of PaMaL and its LoRA-based variants by augmenting linear LoRA aggregation with curvature-motivated transformations. We first derive curvature-aware aggregation in parameter space and then reformulate it in representation space for efficiency. We further show that, under mild conditions, the curvature-aware representation can be obtained by transforming the linearly interpolated representation, motivating an efficient sparse transformation parameterized within the LoRA rank budget.

4.1 CURVATURE-AWARE AGGREGATION IN PARAMETER SPACE

To illustrate that the optimum of a tradeoff-weighted sum of task objectives depends not only on the task-specific local optima but also on the directional penalties induced by the local objective curvatures, we characterize each task objective $\mathcal{L}_i(\boldsymbol{\theta})$ using a local quadratic approximation around its local optimum $\boldsymbol{\theta}_i^*$. Accordingly, its first-order derivative vanishes at $\boldsymbol{\theta}_i^*$, i.e., $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_i^*) = 0$. Let $\mathbf{H}_i(\boldsymbol{\theta}_i^*)$ denote the Hessian matrix of the task objective $\mathcal{L}_i(\boldsymbol{\theta})$ evaluated at the local optimum $\boldsymbol{\theta}_i^*$. The tradeoff-weighted sum of the local quadratic approximations, denoted by $\tilde{L}(\boldsymbol{\theta})$, and its optimum $\boldsymbol{\theta}^*$ can therefore be written as:

$$\tilde{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{T} \alpha_i [\boldsymbol{\theta} - \boldsymbol{\theta}_i^*]^{\mathrm{T}} \mathbf{H}_i(\boldsymbol{\theta}_i^*) [\boldsymbol{\theta} - \boldsymbol{\theta}_i^*]; \quad \boldsymbol{\theta}^* = \left[\sum_{i=1}^{T} \alpha_i \mathbf{H}_i(\boldsymbol{\theta}_i^*) \right]^{-1} \sum_{i=1}^{T} \alpha_i \mathbf{H}_i(\boldsymbol{\theta}_i^*) \boldsymbol{\theta}_i^*, \quad (1)$$

where the constant term $\sum_{i=1}^{T} \alpha_i \mathcal{L}_i(\boldsymbol{\theta}_i^*)$ is omitted in $\tilde{L}(\boldsymbol{\theta})$. The derivation of Equation 1 is provided in Appendix B.1. This strategy assumes that the merged solution $\boldsymbol{\theta}^*$ remains within the local validity region of the quadratic approximations around the task-specific optima $\boldsymbol{\theta}_i^*$. Each task-specific parameter $\boldsymbol{\theta}_i^*$ is weighted not only by its preference α_i but also by its local curvature $\mathbf{H}_i(\boldsymbol{\theta}_i^*)$. The preference weight α_i reflects the importance of task i, while the Hessian matrix $\mathbf{H}_i(\boldsymbol{\theta}_i^*)$ weights the components of $\boldsymbol{\theta}_i^*$ along its curvature directions. It places greater emphasis on directions with large curvature, where deviations can cause larger increases in the task objective, while directions with small curvature can more readily yield to other objectives. $\sum_{i=1}^{T} \alpha_i \mathbf{H}_i(\boldsymbol{\theta}_i^*)$ serves as the normalization term. However, even with a block-diagonal approximation of the Hessian that enables a layer-wise merging strategy in the same form as the solution in Equation 1, constructing a separate layer-wise Hessian in parameter space for each task remains computationally prohibitive.

4.2 Curvature-Aware Aggregation in Representation Space

Because representation space is substantially lower-dimensional than parameter space, we view each task objective as a function of the representations given a sample $\boldsymbol{x}$. Specifically, we consider the sample-wise task loss $\ell_i(y_i, \hat{y}_i)$. Let $\boldsymbol{z} = (\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)})$ denote the collection of layer-wise representations across the N layers. We can then express the sample-wise objective for task i as $\ell_i(\boldsymbol{z}; \boldsymbol{x})$. Given a task-specific local optimum $\boldsymbol{\theta}_i^*$, let $\boldsymbol{z}_i^*$ denote the corresponding representations for the sample $\boldsymbol{x}$. $\boldsymbol{z}_i^*$ is not necessarily a local optimum of the sample-wise objective, since $\boldsymbol{\theta}_i^*$ is obtained by minimizing the expected objective $\mathcal{L}_i(\boldsymbol{\theta})$, rather than the loss $\ell_i(\boldsymbol{z}; \boldsymbol{x})$. $\ell_i(\boldsymbol{z}; \boldsymbol{x})$ can be locally approximated around $\boldsymbol{z}_i^*$. Let $\mathbf{G}_i(\boldsymbol{z}_i^*)$ and $\boldsymbol{g}_i(\boldsymbol{z}_i^*)$ denote the Hessian matrix and gradient vector of the task objective evaluated at $\boldsymbol{z}_i^*$. The tradeoff-weighted sum of these local quadratic approximations, denoted by $\tilde{\ell}(\boldsymbol{z}; \boldsymbol{x})$, and its closed-form solution $\boldsymbol{z}^*$ can be written as:

$$\begin{aligned} \tilde{\ell}(\boldsymbol{z}; \boldsymbol{x}) &= \frac{1}{2} \sum_{i=1}^{T} \alpha_i [\boldsymbol{z} - \boldsymbol{z}_i^*]^{\mathrm{T}} \mathbf{G}_i(\boldsymbol{z}_i^*) [\boldsymbol{z} - \boldsymbol{z}_i^*] + \sum_{i=1}^{T} \alpha_i \boldsymbol{g}_i(\boldsymbol{z}_i^*)^{\mathrm{T}} [\boldsymbol{z} - \boldsymbol{z}_i^*]; \\ \boldsymbol{z}^* &= \left[\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*) \right]^{-1} \sum_{i=1}^{T} \alpha_i [\mathbf{G}_i(\boldsymbol{z}_i^*) \boldsymbol{z}_i^* - \boldsymbol{g}_i(\boldsymbol{z}_i^*)]. \end{aligned} \tag{2}$$

The derivation of Equation 2 is provided in Appendix B.2. Please note that $\mathbf{G}_i(\boldsymbol{z}_i^*)\boldsymbol{z}_i^* - \boldsymbol{g}_i(\boldsymbol{z}_i^*) = \mathbf{G}_i(\boldsymbol{z}_i^*)[\boldsymbol{z}_i^* - \mathbf{G}_i(\boldsymbol{z}_i^*)^{-1}\boldsymbol{g}_i(\boldsymbol{z}_i^*)]$. Here, $-\mathbf{G}_i(\boldsymbol{z}_i^*)^{-1}\boldsymbol{g}_i(\boldsymbol{z}_i^*)$ corresponds to a Newton step from $\boldsymbol{z}_i^*$ for sample $\boldsymbol{x}$. Therefore, $\boldsymbol{z}_i^* - \mathbf{G}_i(\boldsymbol{z}_i^*)^{-1}\boldsymbol{g}_i(\boldsymbol{z}_i^*)$ can be viewed as the task-specific locally optimal representation for sample $\boldsymbol{x}$, which we denote by $\boldsymbol{z}_i^\dagger$. The solution $\boldsymbol{z}^*$ in Equation 2 thus aggregates the task-specific $\boldsymbol{z}_i^\dagger$ according to both the preference weights and the task-specific curvatures. We can decompose $\boldsymbol{z}_i^\dagger$ in the eigenbasis of $\mathbf{G}_i(\boldsymbol{z}_i^*)$. Accordingly, $\mathbf{G}_i(\boldsymbol{z}_i^*)\boldsymbol{z}_i^\dagger$ can be expressed as a weighted combination of the eigenvectors, with each weight involving the corresponding eigenvalue. Since larger eigenvalues indicate stronger curvature, this merging strategy balances the task-specific $\boldsymbol{z}_i^\dagger$ according to the preferences while weighting different directions according to their curvatures.

To make the merging strategy in Equation 2 tractable, we approximate each full Hessian with a block-diagonal structure, enabling layer-wise aggregation. *For brevity, we omit the superscript $^{(n)}$ from the vectors and matrices when referring to their layer-specific counterparts.* To further simplify this merging strategy, we assume that the layer-specific gradient vector $\boldsymbol{g}_i(\boldsymbol{z}_i^*)$ can be expressed by applying a linear representation-conditioned operator $\mathbf{T}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)$ to the representation $\boldsymbol{z}_i^*$. Let $\mathbf{G}_i(\boldsymbol{z}_i^*)$ denote the layer-specific Hessian. The resulting layer-wise merging strategy is given by:

$$\boldsymbol{z}^* = \left[\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*) \right]^{-1} \sum_{i=1}^{T} \alpha_i \underbrace{[\mathbf{G}_i(\boldsymbol{z}_i^*) - \mathbf{T}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)]}_{\mathbf{G}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)} \boldsymbol{z}_i^* \tag{3}$$

where $\mathbf{G}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*) = \mathbf{G}_i(\boldsymbol{z}_i^*) - \mathbf{T}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)$ that contains both curvature and gradient information. Given the layer-specific representation $\boldsymbol{z}_i^*$ induced by the task-specific local optimum $\boldsymbol{\theta}_i^*$ for a sample $\boldsymbol{x}$, the layer-wise merging strategy can be implemented by introducing task-specific learnable modules to approximate the effect of $\mathbf{G}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)$, together with an additional module for normalization. However, constructing task-specific modules at every layer still incurs substantial computational overhead.

4.3 Sparse Transformation enhanced Representation Aggregation

PaMaL merges representations through $\boldsymbol{z}' = \sum_{i=1}^{T} \alpha_i \boldsymbol{z}_i^* \in \mathbb{R}^{d^{\text{out}}}$. *In contrast, the curvature-aware solution in Equation 3 depends on both the task-specific representations and their local objective geometries, and therefore need not coincide with this linear interpolation.* This motivates augmenting $\boldsymbol{z}'$ with additional transformation capacity. The following theorem establishes that, under a mild condition, the curvature-aware representation can always be related to $\boldsymbol{z}'$ through a transformation.

Theorem 1. *Given $\boldsymbol{z}' = \sum_{i=1}^{T} \alpha_i \boldsymbol{z}_i^*$ and $\boldsymbol{z}^* = [\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)]^{-1} \sum_{i=1}^{T} \alpha_i \mathbf{G}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*) \boldsymbol{z}_i^*$, we assume $\boldsymbol{z}' \neq \mathbf{0}$. Then, there always exists a transformation conditioned on the sample $\boldsymbol{x}$ and preference vector $\boldsymbol{\alpha}$, denoted by $\mathbf{T}^{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{\alpha})$, such that $\mathbf{T}^{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{\alpha})\boldsymbol{z}' = \boldsymbol{z}^*$. All such transformations can be expressed as $\frac{\boldsymbol{z}^* \boldsymbol{z}'^{\mathrm{T}}}{\|\boldsymbol{z}'\|_2^2} + \mathbf{Q}(\mathbf{I} - \Pi_{\boldsymbol{z}'})$, where $\mathbf{Q}$ is an arbitrary matrix, $\mathbf{I}$ is the identity matrix, and $\Pi_{\boldsymbol{z}'}$ denotes the orthogonal projection onto $\operatorname{span}(\boldsymbol{z}')$.*

Remark. *Importantly, Theorem 1 is intended to establish the existence of a transformation beyond linear interpolation rather than to prescribe its exact parameterization. Explicitly recovering the transformation associated with Equation 3 would still require the task-specific curvature and gradient that we seek to avoid modeling. We thus use the theorem as a motivation for introducing a learnable transformation family that enlarges the solution space of linear LoRA aggregation.*

The proof of Theorem 1 is included in Appendix B.3. The condition $\boldsymbol{z}' \neq \mathbf{0}$ is mild in practice, as an exactly zero aggregated representation is uncommon. However, $\mathbf{T}^{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{\alpha})$ remains dense and costly at each layer. Since a linear transformation can involve geometric operations such as rotation, reflection, scaling, and shearing, we approximate its effect using a limited set of operations with a sparse structure. *Following the "No Free Lunch" intuition, the efficiency gained from the sparse transformation comes at the expense of reduced transformation expressiveness.* Inspired by Rotary Position Embedding (RoPE) (Su et al., 2024), we propose a learnable sparse transformation, denoted by $\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}(\boldsymbol{x}, \boldsymbol{\alpha}) \in \mathbb{R}^{d^{\text{out}} \times d^{\text{out}}}$, that applies independent planar rotations to adjacent pairs of dimensions in the linearly interpolated representation $\boldsymbol{z}'$. $\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}(\boldsymbol{x}, \boldsymbol{\alpha})$ is parameterized by a sample and preference conditioned angle vector $\boldsymbol{\phi} \in \mathbb{R}^{d^{\text{out}}/2}$. Since both $\boldsymbol{z}'$ and $\boldsymbol{\phi}$ depend on the sample and preference, we introduce a parallel aggregation branch with an analogous structure to generate the rotation angles. At each layer, the input representation $\boldsymbol{z}^{\text{in}} \in \mathbb{R}^{d^{\text{in}}}$ carries sample-specific information and is therefore used in place of $\boldsymbol{x}$ for conditioning. Specifically, an additional set of task-specific LoRA modules, denoted by $\mathbf{B}^{\text{rot}}_i \mathbf{A}^{\text{rot}}_i$ for task i, generates angle embeddings from $\boldsymbol{z}^{\text{in}}$, which are aggregated according to $\boldsymbol{\alpha}$ and passed through a π-scaled $\tanh(\cdot)$ function to obtain $\boldsymbol{\phi}$. The resulting representation $\hat{\boldsymbol{z}}^* \in \mathbb{R}^{d^{\text{out}}}$ is then obtained by applying $\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}(\boldsymbol{z}^{\text{in}}, \boldsymbol{\alpha})$ to $\boldsymbol{z}'$, as follows:

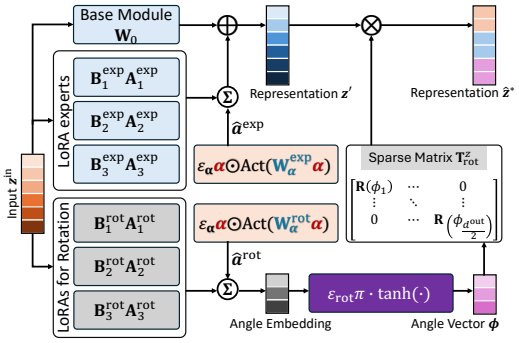


Figure 2: Sparse Transformation Enhanced Representation Aggregation for a Single Layer.

$$\hat{\boldsymbol{z}}^* = \underbrace{\text{BlockDiag}(\mathbf{R}(\phi_1), \ldots, \mathbf{R}(\phi_{d^{\text{out}}/2}))}_{\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}(\boldsymbol{z}^{\text{in}}, \boldsymbol{\alpha})} \underbrace{\left[\mathbf{W}_0 \boldsymbol{z}^{\text{in}} + \frac{\beta}{r^{\text{exp}}} \sum_{i=1}^{T} \alpha_i \mathbf{B}_i^{\text{exp}} \mathbf{A}_i^{\text{exp}} \boldsymbol{z}^{\text{in}}\right]}_{\boldsymbol{z}'};$$
$$\mathbf{R}(\phi_j) = \begin{pmatrix} \cos\phi_j & -\sin\phi_j \\ \sin\phi_j & \cos\phi_j \end{pmatrix}; \quad \boldsymbol{\phi} = \epsilon_{\text{rot}}\pi \cdot \tanh\left(\sum_{i=1}^{T} \alpha_i \underbrace{\mathbf{B}_i^{\text{rot}} \mathbf{A}_i^{\text{rot}} \boldsymbol{z}^{\text{in}}}_{\text{angle embedding}}\right), \tag{4}$$

where ϵ_{rot} and β are hyperparameters, while r^{exp} and r^{trans} are the LoRA ranks of $\mathbf{B}_i^{\text{exp}}\mathbf{A}_i^{\text{exp}}$ and $\mathbf{B}_i^{\text{trans}}\mathbf{A}_i^{\text{trans}}$, respectively. $\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}(\boldsymbol{z}^{\text{in}}, \boldsymbol{\alpha})$ is block diagonal, with each block $\mathbf{R}(\phi_j)$ parameterized by the corresponding angle ϕ_j in $\boldsymbol{\phi}$. $\boldsymbol{z}'$ is constructed from task-specific LoRA experts $\mathbf{B}_i^{\text{exp}}\mathbf{A}_i^{\text{exp}}$ built upon a shared base module $\mathbf{W}_0$, following the PaLoRA formulation. Given $\mathbf{B}_i^{\text{exp}} \in \mathbb{R}^{d^{\text{out}} \times r^{\text{exp}}}$ and $\mathbf{B}_i^{\text{rot}} \in \mathbb{R}^{(d^{\text{out}}/2) \times r^{\text{trans}}}$, the transformation modules require fewer parameters than the task-specific LoRA experts when r^{exp} and r^{trans} are comparable. Thus, our method provides an alternative route for performance improvement: when increasing the expert LoRA rank r^{exp} yields limited gains, part of the rank budget can instead be allocated to r^{trans} for parameterizing the transformation $\mathbf{T}^{\boldsymbol{z}}_{\text{rot}}$, improving model flexibility without increasing the overall parameter budget.

Our practical implementation builds upon PaLoRA (Dimitriadis et al., 2025), which employs linearly interpolated LoRA experts for both the shared backbone and task-specific heads. To handle layers with odd-dimensional outputs, i.e., $d^{\text{out}} \bmod 2 \neq 0$, which commonly arise in output layers and may include scalar outputs, we introduce a sparse scaling transformation $\mathbf{T}^{\boldsymbol{z}}_{\text{scal}}(\boldsymbol{z}^{\text{in}}, \boldsymbol{\alpha})$ parameterized as a diagonal matrix. Similar to the rotation branch, an additional set of task-specific LoRA modules, denoted by $\mathbf{B}_i^{\text{scal}}\mathbf{A}_i^{\text{scal}}$, generates scaling embeddings from $\boldsymbol{z}^{\text{in}}$, where $\mathbf{B}_i^{\text{scal}} \in \mathbb{R}^{d^{\text{out}} \times r^{\text{trans}}}$ and $\mathbf{A}_i^{\text{scal}} \in \mathbb{R}^{r^{\text{trans}} \times d^{\text{in}}}$. These embeddings are aggregated according to $\boldsymbol{\alpha}$ and passed through a ϵ_{scal}-scaled $\text{sigmoid}(\cdot)$ function to obtain the diagonal entries in $\mathbf{T}^{\boldsymbol{z}}_{\text{scal}}$, where ϵ_{scal} is a hyperparameter. To further enlarge the attainable solution space, we introduce a

Table 1: Comparison of PFL methods across different datasets. HV ↑ and Std. report the mean hypervolume and standard deviation over three random seeds, respectively.

	MultiMNIST			MultiMNIST3			CityScapes			NYUv2		
Method	HV ↑	Std.	# Params.	HV ↑	Std.	# Params.	HV ↑	Std.	# Params.	HV ↑	Std.	# Params.
COSMOS	0.8746	0.0160	0.0282M	0.8509	0.0173	0.0317M	0.6576	0.0230	25.0190M	0.0495	0.0033	25.0574M
PHN	0.9148	0.0041	2.7930M	0.9004	0.0035	3.1021M	-	-	-	-	-	-
PaMaL	0.9106	0.0023	0.0549M	0.8906	0.0054	0.0915M	0.6929	0.0052	50.0353M	0.0527	0.0030	75.1656M
PaLoRA	0.9219	0.0050	0.0309M	0.9044	0.0077	0.0361M	0.7052	0.0070	26.0650M	0.0646	0.0048	26.6409M
PaLoRA*	0.9245	0.0032	0.0343M	0.9032	0.0168	0.0418M	0.7070	0.0051	27.1123M	0.0628	0.0028	28.2265M
STeRA	**0.9302**	0.0002	0.0334M	**0.9179**	0.0029	0.0405M	**0.7112**	0.0019	26.5881M	**0.0714**	0.0007	27.8297M

separate lightweight preference-adjustment module for each set of LoRA modules, including the LoRA experts and the LoRA modules associated with $\mathbf{T}^{\boldsymbol{z}}_{\mathrm{rot}}$ and $\mathbf{T}^{\boldsymbol{z}}_{\mathrm{scal}}$. Specifically, for each branch $\mathbf{s} \in \{\mathrm{exp}, \mathrm{rot}, \mathrm{scal}\}$, we define its adjusted preference vector as $\hat{\boldsymbol{\alpha}}^{\mathbf{s}} = \epsilon_{\boldsymbol{\alpha}} \boldsymbol{\alpha} \odot \mathrm{Act}(\mathbf{W}^{\mathbf{s}}_{\boldsymbol{\alpha}} \boldsymbol{\alpha})$, where $\mathbf{W}^{\mathbf{s}}_{\boldsymbol{\alpha}} \in \mathbb{R}^{T \times T}$ is a branch-specific trainable matrix, $\epsilon_{\boldsymbol{\alpha}}$ is a scaling hyperparameter, $\odot$ denotes element-wise multiplication, and $\mathrm{Act}(\cdot)$ is an activation function. Accordingly, the LoRA experts, rotation modules, and scaling modules are weighted by $\hat{\boldsymbol{\alpha}}^{\mathrm{exp}}$, $\hat{\boldsymbol{\alpha}}^{\mathrm{rot}}$, and $\hat{\boldsymbol{\alpha}}^{\mathrm{scal}}$, respectively. This branch-specific adjustment provides additional flexibility around the original preference $\boldsymbol{\alpha}$ while preserving the correspondence between tasks and their associated LoRA modules. The overall workflow of our STeRA method for a single layer is illustrated in Figure 2. We emphasize that the proposed sparse transformations can be naturally extended to convolutional layers. Suppose the linearly interpolated convolutional LoRA experts built upon a shared convolutional base module produce a feature map $\mathbf{Z}' \in \mathbb{R}^{C \times H \times W}$, where C, H, and W denote the number of channels, height, and width, respectively. For $\mathbf{T}^{\boldsymbol{z}}_{\mathrm{rot}}$, an additional set of convolutional LoRA modules generates an angle feature map $\boldsymbol{\Phi} \in \mathbb{R}^{(C/2) \times H \times W}$. At each spatial location (h, w), we regard $\mathbf{Z}[:, h, w] \in \mathbb{R}^{C}$ as a representation and apply a block-diagonal rotation matrix parameterized by the corresponding angle vector $\boldsymbol{\Phi}[:, h, w]$. An analogous formulation can be applied to $\mathbf{T}^{\boldsymbol{z}}_{\mathrm{scal}}$.

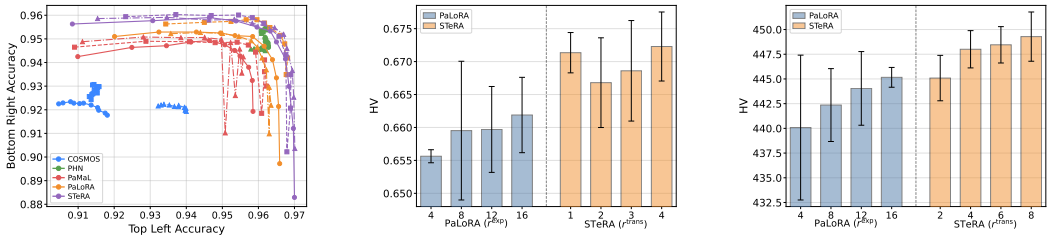
5 Experiments

In this section, we compare STeRA with single-solution MTL and PFL methods across datasets with two to seven tasks, covering multi-label classification (`MultiMNIST` (Sabour et al., 2017), `MultiMNIST3` (Dimitriadis et al., 2023)), regression (`SARCOS` (Vijayakumar & Schaal, 2000)), mixed classification and regression (`UTKFace` (Zhang et al., 2017)), and scene understanding (`CityScapes` (Cordts et al., 2016), `NYUv2` (Silberman et al., 2012)), followed by ablation and hyperparameter studies. For PFL methods, we use hypervolume (HV) (Zitzler & Thiele, 1999) to evaluate the convergence and diversity of the approximated Pareto front. To construct the Pareto front, we evaluate different numbers of evenly spaced preference vectors over the corresponding T-dimensional simplex for different datasets, following (Dimitriadis et al., 2023; 2025).

For comparison, we include single-task learning (STL), single-solution MTL methods, including LS (Caruana, 1997), UW (Kendall et al., 2018), MGDA (Sener & Koltun, 2018), DWA (Liu et al., 2019), PCGrad (Yu et al., 2020), IMTL (Liu et al., 2021b), CAGrad (Liu et al., 2021a), NashMTL (Navon et al., 2022), RLW (Lin et al., 2022a), GradDrop (Chen et al., 2020), RotoGrad (Javaloy & Valera, 2022), and Auto-λ (Liu et al., 2022), as well as PFL methods, including PHN (Navon et al., 2021), COSMOS (Ruchte & Grabocka, 2021), PaMaL (Dimitriadis et al., 2023), and PaLoRA (Dimitriadis et al., 2025). For a fair parameter-budget comparison, we additionally evaluate PaLoRA with doubled LoRA ranks, denoted as PaLoRA*, whose parameter count is at least comparable to that of our method. Although Chen & Kwok (2024) also adopts LoRA combinations for PFL, its implementation is not publicly available. We therefore do not reproduce it to avoid potentially unfair reimplementation and instead build our method solely upon PaLoRA.

5.1 Multi-Label Classification

In this section, we evaluate STeRA on `MultiMNIST` (2 objectives) and `MultiMNIST3` (3 objectives), which classify two and three digits per image, respectively. Following PaLoRA, all meth-



(a) MultiMNIST (b) UTKFace (c) SARCOS

Figure 3: (a) STeRA achieves improved tradeoff coverage. (b) On UTKFace, STeRA outperforms the baselines with fewer parameters. On SARCOS with seven tasks, STeRA consistently achieves better performance under comparable parameter budgets.

ods use LeNet (Lecun et al., 1998) and are trained for 10 and 20 epochs on MultiMNIST and MultiMNIST3, respectively. We set $r^{\text{exp}} = r^{\text{trans}} = 1$ for STeRA, $r^{\text{exp}} = 1$ for PaLoRA (the default setting), and $r^{\text{exp}} = 2$ for PaLoRA*. As shown in Table 1, STeRA achieves the highest HVs of 0.9302 and 0.9179, outperforming all PFL baselines. Compared with PaLoRA*, STeRA attains higher HV with fewer parameters on both datasets, supporting the effectiveness of allocating part of the LoRA budget to sparse transformations. STeRA also yields lower standard deviations than PaLoRA and PaLoRA*. Figure 3a further shows improved tradeoff coverage. Additional experimental details are provided in Appendix A.2.1A.2.2.

5.2 Mixed Task Types and Scaling to More Tasks

We evaluate scalability across task types and with an increasing number of tasks using UTKFace and SARCOS. UTKFace contains gender and ethnicity classification and age regression. Following PaLoRA, all methods use a ResNet backbone (He et al., 2016) and are trained for 100 epochs. In Figure 3b, PaLoRA increases r^{exp} from 4 to 16, while STeRA fixes $r^{\text{exp}} = 4$ and varies r^{trans} from 1 to 4. Even $r^{\text{trans}} = 1$ yields a clear improvement. SARCOS is a robotics dataset with seven regression tasks. Following PaLoRA, all methods use a three-layer MLP and are trained for 100 epochs. In Figure 3c, PaLoRA increases r^{exp} from 4 to 16, while STeRA fixes $r^{\text{exp}} = 8$ and varies r^{trans} from 2 to 8. Under comparable parameter budgets, STeRA consistently outperforms PaLoRA. Additional experimental details are provided in Appendix A.2.3A.2.4.

5.3 Scene Understanding

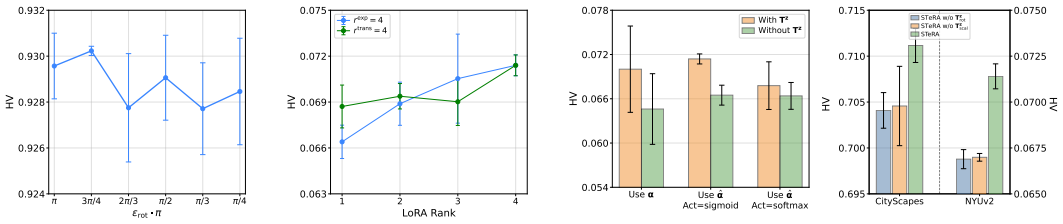
We benchmark STeRA on two large-scale scene-understanding datasets, CityScapes and NYUv2. CityScapes includes semantic segmentation and depth regression, while NYUv2 additionally includes surface-normal prediction. Following PaMaL and PaLoRA, we report the best-of-sampled-subspace results for comparison with single-solution MTL methods, and all methods use a SegNet backbone (Badrinarayanan et al., 2017) trained for 100 and 200 epochs on CityScapes and NYUv2, respectively. For STeRA, we set $(r^{\text{exp}}, r^{\text{trans}}) = (3, 4)$ on CityScapes and $(4, 4)$ on NYUv2. PaLoRA uses the default configuration on both datasets, i.e., $r^{\text{exp}} = 4$, while PaLoRA* uses $r^{\text{exp}} = 8$. On CityScapes, STeRA achieves the highest HV of 0.7112 in Table 1, surpassing PaLoRA and PaLoRA* while using fewer parameters than PaLoRA*. It also achieves the best task-specific results across all four metrics in Table 2, showing consistent gains in both segmentation and depth estimation. On NYUv2, STeRA achieves an HV of 0.0714 in Table 1, improving over PaLoRA by about 10.5% and outperforming PaLoRA* with fewer parameters. In Table 2, STeRA obtains the best segmentation results and second-best performance across all surface-normal and depth metrics, while outperforming the other PFL methods across all reported metrics. More experimental details are provided in Appendix A.2.5A.2.6. Since PHN does not scale well to large networks and its chunking-based variant performs poorly on these datasets, we omit its results.

5.4 Hyperparameter And Ablation Studies

In this section, we conduct hyperparameter and ablation studies of STeRA. Figure 4a studies the rotation range controlled by ϵ_{rot}, showing that an appropriate range can improve performance. For preference adjustment, we initialize $\mathbf{W}^{\text{s}}_{\boldsymbol{\alpha}} = \mathbf{0}$ so that $\hat{\boldsymbol{\alpha}}$ starts from $\boldsymbol{\alpha}$. Accordingly, we set $\epsilon_{\boldsymbol{\alpha}} = 2$

Table 2: Test performance averaged over three seeds on CityScapes and NYUv2. **Best** and second-best results are highlighted.

		CityScapes				NYUv2								
		Segmentation		Depth		Segmentation		Surface Normal					Depth	
	Method	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	mIoU ↑	Pix Acc ↑	30° ↑	22.5° ↑	11.25° ↑	Med. ↓	Mean ↓	Abs Err ↓	Rel Err ↓
	STL	70.35	91.94	0.0138	33.4324	37.95	63.33	70.10	59.05	32.36	18.26	25.10	0.7094	0.2802
MTL	LS	70.18	91.95	0.0188	119.7280	36.37	62.92	63.31	51.06	26.19	22.15	27.82	0.5493	0.2233
MTL	UW	70.15	91.94	0.0176	116.8410	32.00	61.25	62.82	50.78	26.42	22.31	27.99	0.5436	0.2215
MTL	MGDA	66.45	90.79	0.0141	53.1376	31.19	60.22	**70.28**	**59.04**	**32.45**	**18.30**	**24.84**	0.5745	0.2267
MTL	DWA	70.20	91.92	0.0193	118.5660	37.01	63.27	63.81	51.50	26.43	21.94	27.61	0.5485	0.2224
MTL	PCGrad	69.66	91.79	0.0201	114.4590	36.53	63.39	63.88	51.65	26.58	21.84	27.56	0.5489	0.2224
MTL	IMTL	70.77	92.12	0.0151	74.2300	36.03	64.17	68.08	56.32	30.04	19.57	25.82	0.5364	0.2132
MTL	CAGrad	70.05	91.91	0.0154	70.7687	36.41	63.94	68.16	56.38	30.02	19.55	25.71	0.5411	0.2168
MTL	Nash-MTL	71.13	92.23	0.0157	78.4993	37.66	64.75	68.55	56.90	30.59	19.30	25.56	**0.5279**	**0.2109**
MTL	RLW	69.17	91.55	0.0206	121.7290	34.95	62.13	61.69	49.40	25.11	23.05	28.63	0.5830	0.2234
MTL	GradDrop	70.39	92.04	0.0191	118.6720	36.61	63.41	63.78	51.48	26.32	21.93	27.61	0.5435	0.2239
MTL	RotoGrad	70.30	91.87	0.0137	52.2582	38.97	65.11	66.37	54.37	28.47	20.49	26.61	0.5408	0.2143
MTL	Auto-λ	70.22	91.88	0.0179	118.5670	36.86	62.82	61.14	48.73	24.55	23.42	28.93	0.5746	0.2372
PFL	COSMOS	70.06	91.86	0.0615	114.1940	38.10	64.04	64.57	52.39	26.94	21.49	27.36	0.5974	0.2258
PFL	PaMaL	70.31	91.94	0.0145	55.7526	35.78	62.62	66.13	54.19	28.51	20.54	26.74	0.5643	0.2231
PFL	PaLoRA	71.55	92.15	0.0144	50.4111	37.85	63.76	68.92	57.36	30.77	19.12	25.55	0.5316	0.2141
PFL	PaLoRA*	71.72	92.30	0.0142	49.1489	37.32	63.82	68.72	57.08	30.52	19.24	25.63	0.5363	0.2136
PFL	STeRA	**72.10**	**92.50**	**0.0137**	**39.2291**	**40.69**	**65.65**	70.00	58.66	31.93	18.54	25.05	0.5282	0.2130



(a) MultiMNIST (b) NYUv2 (c) NYUv2 (d) Cityscapes and NYUv2

Figure 4: Hyperparameter and Ablation Studies of STeRA.

for sigmoid and $\epsilon_{\boldsymbol{\alpha}} = T$ for softmax, since their outputs at zero are $1/2$ and $1/T$, respectively. Similarly, since $\mathbf{B}_i^{\text{scal}}\mathbf{A}_i^{\text{scal}}$ is initialized to zero, we set $\epsilon_{\text{scal}} = 2$ so that the sigmoid-based scaling starts from the identity transformation. Figure 4b studies r^{exp} and r^{trans} by varying one while fixing the other to 4. Reducing either rank generally degrades performance, while the larger drop from decreasing r^{trans} highlights the importance of transformation capacity on NYUv2. Figure 4c further shows that $\mathbf{T}^{\boldsymbol{z}}$ improves performance with either the original $\boldsymbol{\alpha}$ or adjusted $\hat{\boldsymbol{\alpha}}$, while sigmoid-based adjustment provides additional gains. Finally, Figure 4d shows that removing either $\mathbf{T}^z_{\text{rot}}$ or $\mathbf{T}^z_{\text{scal}}$ degrades performance on both datasets. Although $\mathbf{T}^z_{\text{scal}}$ is applied only to the final layer, it directly affects the output representation, while $\mathbf{T}^z_{\text{rot}}$ improves intermediate representations. Thus, both transformations contribute jointly, supporting transformation across layers for better representations. Additional experimental results and details are provided in Appendix A.1.

6 Conclusion and Limitations

We introduce STeRA, a curvature-motivated approach for LoRA-based Pareto front learning beyond preference-weighted linear interpolation. To avoid costly parameter-space curvature modeling, we derive sample-wise curvature-aware aggregation in representation space and approximate it with sample and preference conditioned learnable sparse operators. STeRA allocates part of the LoRA rank budget to transformation modeling, achieving consistently higher hypervolume with comparable or fewer parameters across classification, regression, and scene-understanding benchmarks. Our method also has limitations. The formulation relies on local second-order and layer-wise approximations, neglecting cross-layer curvature interactions. The sparse rotation and scaling operators improve efficiency but restrict transformation expressiveness, and they learn curvature-related geometry implicitly rather than explicitly estimating task-specific curvature. Future work could explore richer transformations, cross-layer interactions, and adaptive rank allocation.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript polishing, grammar checking, and improving readability and organization. All sections were initially drafted by the authors, after which AI tools were used to refine the language and presentation. We also used AI tools for coding assistance when implementing our proposed method. For mathematical proofs, all derivations and arguments were completed by the authors, and AI tools were used only to help transcribe them into LaTeX. We did not use generative AI tools for dataset generation, research ideation or execution, mathematical reasoning or proof development, literature discovery or retrieval, initial manuscript drafting, or other substantive aspects of the research. All AI-assisted content was carefully reviewed by the authors to ensure that it accurately reflected our intended meaning and did not introduce ethical or plagiarism concerns. We also verified AI-assisted code to ensure that the implementation was correct. We take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

Since our work focuses on machine learning methodology, particularly Pareto front learning, and all experiments are conducted on publicly available datasets, we do not identify any specific ethical concerns associated with this work.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide the source code as part of the submitted supplementary material. In addition, we report detailed experimental settings in the paper, including the computational environment, dataset configurations, training settings, evaluation protocols, and hyperparameter configurations.

REFERENCES

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615.

Ashmita Bhattacharya and Malyaban Bal. Hyperpalora: Parameter-efficient pareto hypernetworks via preference-based diverse low-rank adaptations. In *OPT 2025: Optimization for Machine Learning*, 2025.

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

Weiyu Chen and James Kwok. Multi-objective deep learning with adaptive reference vectors. *Advances in Neural Information Processing Systems*, 35:32723–32735, 2022.

Weiyu Chen and James Kwok. Pareto merging: Multi-objective optimization for preference-aware model merging. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=D7qRwx6BOS.

Weiyu Chen and James T. Kwok. Efficient pareto manifold learning with low-rank structure. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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

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

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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016. doi: 10.1109/CVPR.2016.350.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Nico Daheim, Thomas Möllenhoff, Edoardo M Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *International Conference on Learning Representations*, volume 2024, pp. 11836–11854, 2024.

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

Nikolaos Dimitriadis, Pascal Frossard, and François Fleuret. Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models. In *International Conference on Machine Learning*, pp. 8015–8052. PMLR, 2023.

Nikos Dimitriadis, Pascal Frossard, and Francois Fleuret. Pareto low-rank adapters: Efficient multi-task learning with preferences. In *International Conference on Learning Representations*, volume 2025, pp. 22323–22346, 2025.

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International conference on learning representations*, 2019.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

Heshan Devaka Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *The eleventh international conference on learning representations*, 2023.

Hongcan Guo, Guoshun Nan, Yuan Yang, Diyang Zhang, Haotian Li, Zhican Chen, Qinchuan Zhou, Yuhan Ran, Xinye Cao, Sicong Leng, et al. Two is better than one: Rotations scale loras. *arXiv preprint arXiv:2505.23184*, 2025.

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 (CVPR)*, June 2016.

Long P Hoang, Dung D Le, Tran Anh Tuan, and Tran Ngoc Thang. Improving pareto front learning via multi-sample hypernetworks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 7875–7883, 2023.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=TrloAXEJ2B.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=T8wHz4rnuGL.

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

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856. URL https://openreview.net/forum?id=jjtFD8A1Wx.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313*, 2020.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=QuObT9BTWo.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021a.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36:57226–57243, 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 *International conference on learning representations*, 2021b.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 1871–1880. IEEE, 2019.

Shikun Liu, Stephen James, Andrew Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=KKeCMim5VN.

Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14721–14733. Curran Associates, Inc., 2021c.

Zefang Liu and Jiahua Luo. AdamoLE: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=ndY9qFf9Sa.

Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient continuous pareto exploration in multi-task learning. In *International Conference on Machine Learning*, pp. 6522–6531. PMLR, 2020.

Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6597–6607. PMLR, 13–18 Jul 2020.

Michael S Matena and Colin Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16428–16446. PMLR, 17–23 Jul 2022.

Dung Viet Nguyen, Minh Nguyen, Luc Nguyen, Rachel Teo, Tan Nguyen, and Duy Linh Tran. Camex: Curvature-aware merging of experts. In *International Conference on Learning Representations*, volume 2025, pp. 20335–20360, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. In *2021 IEEE international conference on data mining (ICDM)*, pp. 1306–1311. IEEE, 2021.

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

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3859–3869, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. ECCV'12, pp. 746–760, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642337147. doi: 10.1007/978-3-642-33715-4_54. URL https://doi.org/10.1007/978-3-642-33715-4_54.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Anke Tang, Li Shen, Yong Luo, Shiwei Liu, Han Hu, Bo Du, and Dacheng Tao. Towards efficient pareto set approximation via mixture of experts based model fusion. *arXiv preprint arXiv:2406.09770*, 2024.

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 transactions on pattern analysis and machine intelligence*, 44(7):3614–3633, 2021.

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 1079–1086, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

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

Qi Wei, Shuo He, Enneng Yang, Tingcong Liu, Haobo Wang, Lei Feng, and Bo An. Representation surgery in model merging with probabilistic modeling. In *Forty-second International Conference on Machine Learning*, 2025.

Jialin Wu, Jian Yang, Handing Wang, Jiajun Wen, and Zhiyong Yu. From parameter to representation: A closed-form approach for controllable model merging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26903–26912, 2026.

Xun Wu, Shaohan Huang, and Furu Wei. Mole: Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xingwei Wang, Xiaocun Cao, Jie Zhang, and Dacheng Tao. Surgeryv2: Bridging the gap between model merging and multi-task learning with deep representation surgery. *arXiv preprint arXiv:2410.14389*, 2024b.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, volume 2024, pp. 22743–22763, 2024c.

Mao Ye and Qiang Liu. Pareto navigation gradient descent: a first-order algorithm for optimization in pareto set. In *Uncertainty in artificial intelligence*, pp. 2246–2255. PMLR, 2022.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Jinghan Zhang, Junteng Liu, Junxian He, et al. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610, 2023.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Haojun Chen, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. Panacea: Pareto alignment via preference adaptation for LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gL5nT4y8fn.

Zhipeng Zhou, Liu Liu, Peilin Zhao, and Wei Gong. Injecting imbalance sensitivity for multi-task learning. IJCAI '25, 2025a. ISBN 978-1-956792-06-5. doi: 10.24963/ijcai.2025/805. URL https://doi.org/10.24963/ijcai.2025/805.

Zhipeng Zhou, Ziqiao Meng, Pengcheng Wu, Peilin Zhao, and Chunyan Miao. Continual optimization with symmetry teleportation for multi-task learning. *Advances in Neural Information Processing Systems*, 38:171209–171230, 2025b.

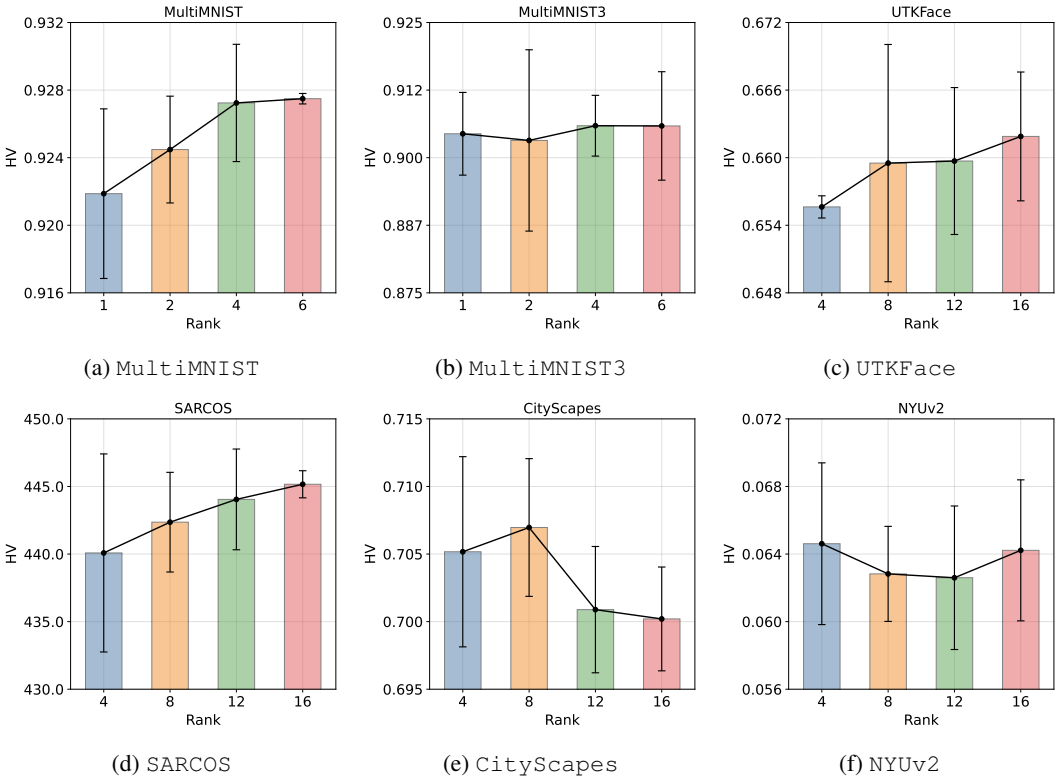
Zhipeng Zhou, Ziqiao Meng, Pengcheng Wu, Peilin Zhao, and Chunyan Miao. Exploring tradeoffs through mode connectivity for multi-task learning. *Advances in Neural Information Processing Systems*, 38:81268–81285, 2025c.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. doi: 10.1109/4235.797969.

A Additional Experimental Results and Details

In this section, we present additional experimental results, including baseline comparisons, hyperparameter studies, and ablation studies. We also provide further experimental details on the benchmarks, baseline implementations, and configurations of our method.

A.1 Additional Experimental Results



(a) MultiMNIST (b) MultiMNIST3 (c) UTKFace

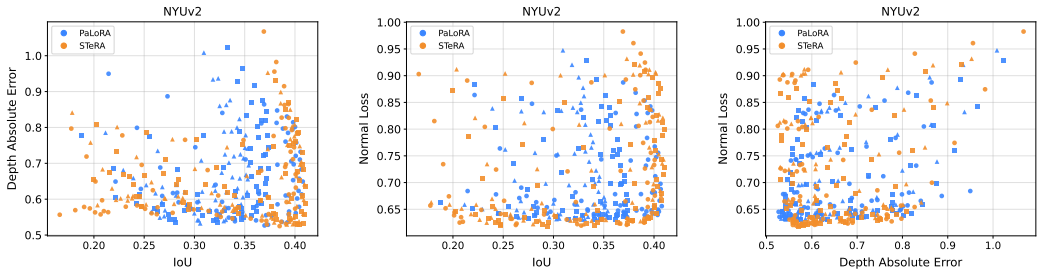
(d) SARCOS (e) CityScapes (f) NYUv2

Figure 5: Hypervolume of PaLoRA as the LoRA rank increases across different datasets.

As shown in Figure 5, increasing the LoRA rank of PaLoRA does not consistently translate into proportional HV gains. On MultiMNIST, the improvement gradually saturates as the rank increases, while MultiMNIST3 remains nearly unchanged across different ranks. Similar limited gains are observed on UTKFace and SARCOS, where increasing the rank yields only modest improvements. More notably, the HV on CityScapes peaks at an intermediate rank and then decreases, while NYUv2 exhibits little overall improvement as the rank increases. These observations suggest that PaLoRA may fall short of the performance gains expected from simply increasing model capacity through larger LoRA ranks. This motivates reallocating part of the LoRA parameter budget to alternative modules that expand the attainable solution space beyond preference-weighted linear interpolation, as adopted by STeRA.

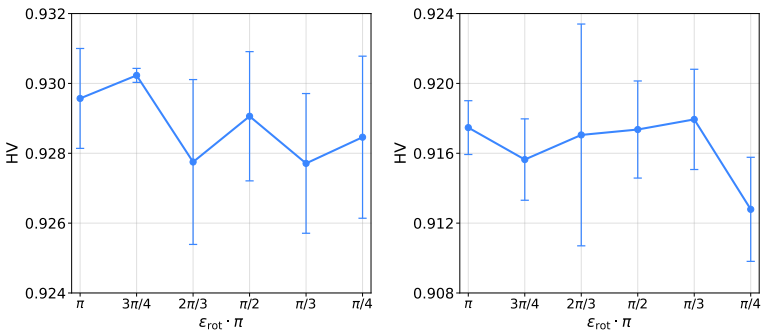
Since directly inspecting the three-dimensional Pareto front on NYUv2 is difficult, we further visualize its pairwise projections in Figure 6. Across the three objective pairs, STeRA generally extends the attainable solutions toward more favorable regions than PaLoRA. In particular, for IoU versus depth absolute error, STeRA provides more solutions with higher IoU while maintaining low depth error, indicating improved tradeoffs between segmentation and depth estimation. A similar tendency is observed for IoU versus normal loss, where STeRA reaches high-IoU solutions with comparatively low normal loss. The depth-error versus normal-loss projection further shows that STeRA maintains competitive solutions toward the low-error region for both objectives. These pairwise visualizations complement the HV results and provide a more detailed view of the improved tradeoff coverage achieved by STeRA on NYUv2.

Figure 7 studies the effect of ϵ_{rot}, which controls the range of rotation angles. On MultiMNIST, the HV varies across different settings and reaches its highest value around $3\pi/4$. On MultiMNIST3,



(a) IoU vs. Depth Absolute Error. (b) IoU vs. Normal Loss. (c) Depth Absolute Error vs. Normal Loss.

Figure 6: Pairwise visualization of the Pareto solutions obtained by PaLoRA and STeRA on NYUv2.



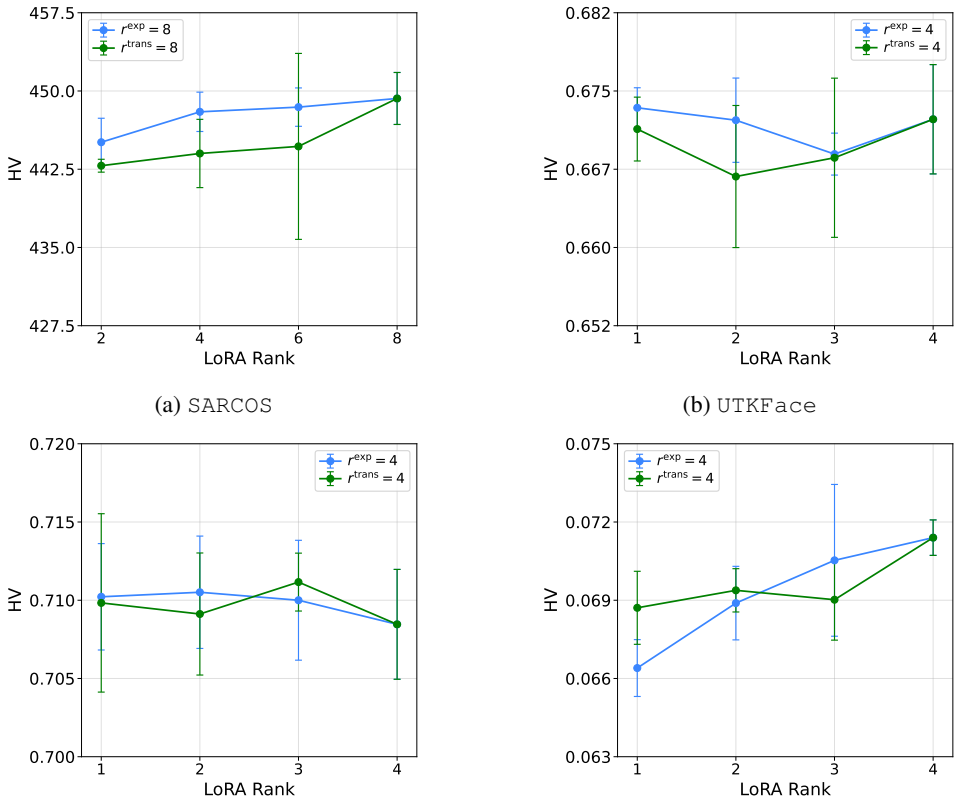
(a) MultiMNIST (b) MultiMNIST3

Figure 7: Effect of the rotation range controlled by ϵ_{rot}.

the performance also varies with the rotation range, with relatively better performance observed around intermediate settings. Overall, these results suggest that an appropriate choice of ϵ_{rot} may contribute to improved performance, while the optimal range can vary across datasets.

Figure 8 studies the effects of r^{exp} and r^{trans} by varying one rank while fixing the other. On SARCOS, reducing either rank generally decreases HV, indicating that both the LoRA experts and sparse transformation modules contribute to performance. The larger degradation when reducing r^{exp} suggests that expert capacity is particularly important for this seven-task regression benchmark. On UTKFace, HV remains relatively stable across different rank configurations without a clear monotonic trend. This suggests that lightweight transformation and expert ranks can already provide competitive performance, while simply increasing either rank does not necessarily yield consistent gains. On CityScapes, the variations are also relatively small and non-monotonic, with substantial overlap among the error bars. This indicates that performance is relatively robust to the allocation between r^{exp} and r^{trans} within the evaluated range. On NYUv2, reducing either rank generally degrades performance, but the decrease is more pronounced when reducing r^{trans}. This suggests that transformation capacity plays a particularly important role on NYUv2, supporting the allocation of LoRA capacity to the sparse transformation modules rather than solely increasing the expert rank.

Figure 9 compares STeRA with and without the proposed sparse transformation under the original preference $\boldsymbol{\alpha}$ and two adjusted-preference strategies. On MultiMNIST, the sparse transformation improves HV under all three weighting strategies, with the sigmoid-based adjustment achieving the highest performance, suggesting that both representation transformation and lightweight preference adjustment can contribute to the improvement. On MultiMNIST3, adding the sparse transformation also consistently improves performance across all weighting strategies. The original preference $\boldsymbol{\alpha}$ achieves the highest HV in this case, indicating that the transformation itself remains effective without requiring preference adjustment. On UTKFace, the transformation provides clear gains when using the original $\boldsymbol{\alpha}$ and the softmax-adjusted preference, while the sigmoid setting shows



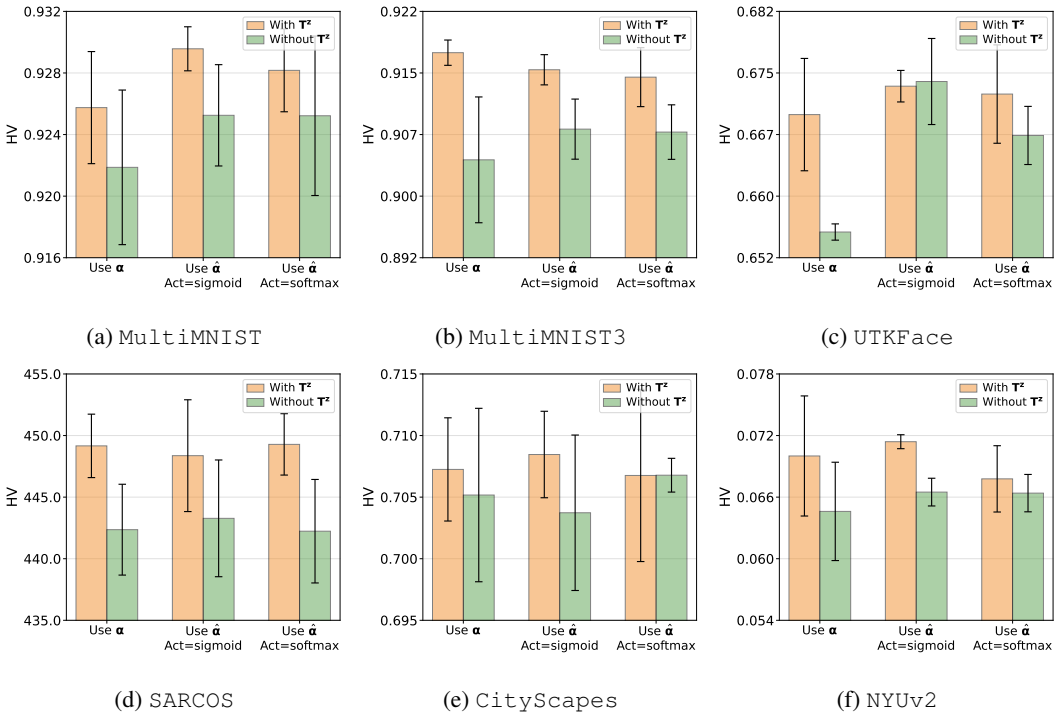
(a) SARCOS (b) UTKFace

(c) CityScapes (d) NYUv2

Figure 8: Effect of the expert and transformation ranks on STeRA. For each curve, one rank is varied while the other is fixed.

comparable performance with and without the transformation. Among the transformed variants, the sigmoid-based adjustment achieves the highest HV, indicating that the preferred weighting strategy can depend on the dataset. On SARCOS, the sparse transformation consistently improves HV under all three weighting strategies, while the differences among the transformed variants are relatively small. This suggests that the performance gain mainly comes from the proposed representation transformation rather than a particular preference-adjustment strategy. On CityScapes, the transformation improves performance with the original preference and most clearly with the sigmoid-adjusted preference, whereas the softmax setting shows little difference. The sigmoid-based adjustment achieves the highest HV, supporting the benefit of a slight preference modulation for this dataset. On NYUv2, the sparse transformation yields higher HV under all three weighting strategies, with the sigmoid-based preference adjustment giving the strongest performance. This further indicates that the sparse transformation is effective independently of preference adjustment, while an appropriate adjustment strategy can provide additional gains.

Table 3 reports the task-specific classification accuracy on MultiMNIST and MultiMNIST3. On MultiMNIST, STeRA achieves the highest accuracy on both the top-left and bottom-right tasks, reaching 96.97% and 95.95%, respectively, while also exhibiting low variance across random seeds. Compared with PaLoRA, these correspond to improvements of 0.39 and 0.49 percentage points. On MultiMNIST3, STeRA achieves the highest accuracy on Tasks 2 and 3, with 96.77% and 97.22%, respectively. On Task 1, STeRA reaches 97.58%, which is nearly identical to the 97.59% achieved by PaLoRA*. Compared with PaLoRA, STeRA improves all three tasks by 0.55, 0.53, and 0.33 percentage points, respectively. Overall, the task-specific results show that the improved HV of STeRA is accompanied by consistent accuracy gains across individual objectives rather than being driven by only a subset of tasks.



(a) MultiMNIST (b) MultiMNIST3 (c) UTKFace

(d) SARCOS (e) CityScapes (f) NYUv2

Figure 9: Effect of the proposed sparse transformation under different preference-weighting strategies.

Table 3: Task-specific accuracy on MultiMNIST and MultiMNIST3, averaged over three random seeds.

		MultiMNIST				MultiMNIST3					
		Top-left		Bottom-right		Task 1		Task 2		Task 3	
	Method	Mean (%)	Std.	Mean (%)	Std.	Mean (%)	Std.	Mean (%)	Std.	Mean (%)	Std.
	STL	95.55	0.0033	94.28	0.0018	97.32	0.0030	95.96	0.0025	96.41	0.0008
MTL	LS	94.77	0.0031	93.37	0.0091	96.44	0.0070	95.10	0.0033	95.80	0.0033
MTL	UW	95.16	0.0044	93.57	0.0077	96.86	0.0038	95.60	0.0010	96.27	0.0018
MTL	MGDA	94.65	0.0011	93.33	0.0052	96.84	0.0037	94.74	0.0062	95.51	0.0016
MTL	DWA	94.77	0.0031	93.43	0.0081	96.43	0.0065	95.19	0.0044	95.94	0.0025
MTL	PCGrad	94.61	0.0035	93.28	0.0070	96.58	0.0050	95.15	0.0049	95.90	0.0027
MTL	IMTL	94.78	0.0032	93.22	0.0077	96.77	0.0048	95.05	0.0030	95.86	0.0038
MTL	CAGrad	94.78	0.0026	93.19	0.0077	96.57	0.0071	95.19	0.0016	95.91	0.0045
MTL	Nash-MTL	94.71	0.0041	93.18	0.0057	96.53	0.0056	95.19	0.0028	95.75	0.0048
MTL	RLW	94.42	0.0014	93.07	0.0040	96.43	0.0044	94.64	0.0085	95.42	0.0007
MTL	GradDrop	94.60	0.0033	93.26	0.0076	96.33	0.0051	94.85	0.0025	95.52	0.0049
MTL	RotoGrad	94.89	0.0020	93.40	0.0019	94.56	0.0150	93.10	0.0069	94.32	0.0068
MTL	Auto-λ	95.01	0.0039	93.89	0.0017	96.84	0.0042	94.83	0.0082	95.98	0.0005
PFL	COSMOS	94.03	0.0083	93.02	0.0088	95.70	0.0026	93.81	0.0106	94.79	0.0090
PFL	PHN	96.24	0.0004	95.05	0.0042	97.22	0.0011	95.98	0.0003	96.49	0.0028
PFL	PaMaL	95.88	0.0032	94.97	0.0010	97.00	0.0024	95.76	0.0014	95.90	0.0066
PFL	PaLoRA	96.58	0.0023	95.46	0.0031	97.03	0.0073	96.24	0.0013	96.89	0.0026
PFL	PaLoRA*	96.62	0.0006	95.69	0.0038	**97.59**	0.0024	95.55	0.0141	96.92	0.0028
PFL	STeRA	**96.97**	0.0006	**95.95**	0.0007	97.58	0.0039	**96.77**	0.0010	**97.22**	0.0005

A.2 Experimental Details

All experiments are implemented in PyTorch (Paszke et al., 2019) and conducted on an AMD EPYC 7532 32-Core CPU and an NVIDIA A100-SXM4-80GB GPU. We report results over three random seeds for each method. Our method is built upon the implementations of PaLoRA (Dimitriadis et al.,

2025) and PaMaL (Dimitriadis et al., 2023). Accordingly, the reproduction of both single-solution MTL and PFL baselines also follows their repositories.

A.2.1 MULTIMNIST

We use the `MultiMNIST` dataset constructed in the repository of (Dimitriadis et al., 2023; 2025), a synthetic dataset derived from MNIST samples. For each `MultiMNIST` image, two 28×28 MNIST images are sampled with replacement and placed on a 36×36 canvas, with one at the top-left and the other at the bottom-right, after which the resulting image is rescaled to 28×28. The training, validation, and test sets contain 50,000, 10,000, and 10,000 samples, respectively. Following PaMaL and PaLoRA, we use LeNet as the shared backbone, consisting of two convolutional layers and one fully connected layer, followed by task-specific heads with two fully connected layers. During training, we adopt the same training configuration as PaLoRA. The model is trained for 10 epochs using the Adam optimizer with a learning rate of 0.001. We also follow PaLoRA in using the same annealed sampling strategy for preference vectors and sample three preference vectors per batch. We set $r^{\text{exp}} = r^{\text{trans}} = 1$, $\text{Act}(\cdot) = \text{sigmoid}(\cdot)$, and $\epsilon_{\text{rot}} = 3/4$. During evaluation, we compute hypervolume using the top-left and bottom-right accuracies and evenly sample 11 tradeoffs from the corresponding preference simplex. The reference point is set to $(0, 0)$. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories

A.2.2 MULTIMNIST3

We use the `MultiMNIST3` dataset introduced in (Dimitriadis et al., 2023; 2025), which requires classifying three digits in each image. For each sample, three 28×28 MNIST images are sampled and placed on a 42×42 canvas at the top-left, bottom-left, and middle-right positions, after which the resulting image is resized to 28×28. The training, validation, and test sets contain 50,000, 10,000, and 10,000 samples, respectively. Likewise, LeNet is used as the shared backbone, together with three task-specific heads. During training, we adopt the same configurations as PaLoRA. The model is trained for 20 epochs using the Adam optimizer with a learning rate of 0.001. We also follow PaLoRA in using an annealed Dirichlet sampling strategy for the preference vector and sample three preference vectors per batch. We set $r^{\text{exp}} = r^{\text{trans}} = 1$, $\epsilon_{\text{rot}} = 1/3$, and use the original preference vector $\boldsymbol{\alpha}$. During evaluation, we compute hypervolume using the accuracies of the three classification tasks and evenly sample 66 tradeoffs from the corresponding preference simplex. The reference point is set to $(0, 0, 0)$. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories.

A.2.3 UTKFACE

We use the `UTKFace` dataset (Zhang et al., 2017), which contains over 20,000 face images and includes three tasks: age regression, gender classification, and ethnicity classification. The age regression task uses the Huber loss, while the two classification tasks use the cross-entropy loss. Each image is resized to 64×64. The dataset is split into 70%, 10%, and 20% for training, validation, and testing, respectively. We use ResNet-18 (He et al., 2016) as the shared backbone, followed by three task-specific fully connected heads with output dimensions of 1, 2, and 5 for age, gender, and ethnicity, respectively. During training, we adopt the same configurations as PaLoRA. The model is trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. We also follow PaLoRA in using an annealed Dirichlet sampling strategy for the preference vector and sample three preference vectors per batch. We set $r^{\text{exp}} = 4$, $r^{\text{trans}} = 1$, $\epsilon_{\text{rot}} = 1$, and $\text{Act}(\cdot) = \text{sigmoid}(\cdot)$. During evaluation, we compute hypervolume using the classification accuracies for gender and ethnicity and the Huber loss for age, and evenly sample 66 tradeoffs from the corresponding preference simplex. The reference point is set to $(0, 0, 1)$. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories.

A.2.4 SARCOS

We use the SARCOS robotics dataset, which contains seven regression tasks for predicting joint torques from joint positions, velocities, and accelerations. Each task is optimized using the mean squared error (MSE) loss. From the original training set, the first 4,449 samples are used for validation, while the remaining samples are used for training. The shared backbone is a three-layer MLP, followed by seven task-specific single-layer fully connected regression heads. During training, we adopt the same configurations as PaLoRA. The model is trained for 100 epochs using the Adam optimizer with a learning rate of 0.0001. We also follow PaLoRA in using an annealed Dirichlet sampling strategy for the preference vector and sample ten preference vectors per batch. We set $r^{\text{exp}} = 8$, $r^{\text{trans}} = 8$, $\epsilon_{\text{rot}} = 1$, and $\text{Act}(\cdot) = \text{softmax}(\cdot)$. During evaluation, we compute hypervolume using the MSE losses and set the reference point to $(1, 5, 1, 1, 10, 10, 1)$, following PaLoRA. Since SARCOS contains seven tasks, we adopt the same preference sampling strategy as PaLoRA, selecting the seven one-hot preference vectors together with one central point in the preference simplex. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories.

A.2.5 CITYSCAPES

We use the same configuration as (Navon et al., 2022; Dimitriadis et al., 2023; 2025) for the CityScapes (Cordts et al., 2016) dataset, which includes two tasks: semantic segmentation and depth regression. The segmentation task considers seven classes, rather than the original 19 classes. From the training split, 2,500 samples are used for training and 475 samples are used for validation, while the original validation split is used for testing. We use SegNet (Badrinarayanan et al., 2017) as the shared backbone. The model is trained for 100 epochs using the Adam optimizer. The learning rate is set to 0.0001 for the first 75 epochs and decayed by a factor of 0.5 for the remaining epochs. We also follow PaLoRA in using an annealed sampling strategy for the preference vector and sample five preference vectors per batch. We set $r^{\text{exp}} = 3$, $r^{\text{trans}} = 4$, $\epsilon_{\text{rot}} = 1$, and $\text{Act}(\cdot) = \text{sigmoid}(\cdot)$. During evaluation, we compute hypervolume using semantic IoU and depth absolute error, and set the reference point to $(0, 1)$. We evenly sample 11 tradeoffs from the corresponding preference simplex. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories.

A.2.6 NYUV2

We use the NYUv2 dataset (Silberman et al., 2012), which includes three tasks: semantic segmentation, depth estimation, and surface-normal prediction. The semantic segmentation task contains 13 classes. From the training split, 700 samples are used for training and 95 samples are reserved for validation, while the original validation split is used for testing. Following PaLoRA, we use SegNet (Badrinarayanan et al., 2017) as the shared backbone. The model is trained for 200 epochs using the Adam optimizer. The learning rate is initialized to 0.0001 for the first 100 epochs and decayed by a factor of 0.5 for the remaining epochs. We also follow PaLoRA in using an annealed sampling strategy for the preference vector and sample six preference vectors per batch. We set $r^{\text{exp}} = 4$, $r^{\text{trans}} = 4$, $\epsilon_{\text{rot}} = 1$, and $\text{Act}(\cdot) = \text{sigmoid}(\cdot)$. During evaluation, we compute hypervolume using semantic IoU, depth absolute error, and the L1 loss for surface-normal prediction, and set the reference point to $(0, 1, 1)$. We evenly sample 66 tradeoffs from the corresponding preference simplex. For all remaining settings and baseline reproductions, we follow the configurations reported in the original PaMaL and PaLoRA papers or their corresponding repositories.

B PROOFS

B.1 PROOF OF EQUATION 1

For each task i, we consider a local second-order Taylor approximation of the task-specific objective $\mathcal{L}_i(\boldsymbol{\theta})$ around its local optimum $\boldsymbol{\theta}_i^*$:

$$\mathcal{L}_i(\boldsymbol{\theta}) \approx \mathcal{L}_i(\boldsymbol{\theta}_i^*) + \nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta}_i^*)^{\mathrm{T}}(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*)^{\mathrm{T}}\mathbf{H}_i(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*), \tag{5}$$

where $\mathbf{H}_i(\boldsymbol{\theta}_i^*)$ denotes the Hessian of $\mathcal{L}_i$ evaluated at $\boldsymbol{\theta}_i^*$. Since $\boldsymbol{\theta}_i^*$ is a local optimum of task i, $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta}_i^*) = \mathbf{0}$. Therefore, after omitting the constant $\mathcal{L}_i(\boldsymbol{\theta}_i^*)$, the local approximation becomes

$$\tilde{\mathcal{L}}_i(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*)^{\mathrm{T}}\mathbf{H}_i(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*). \tag{6}$$

Given a preference vector $\boldsymbol{\alpha}$, the preference-weighted local objective is

$$\tilde{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{T}\alpha_i(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*)^{\mathrm{T}}\mathbf{H}_i(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*). \tag{7}$$

Taking the gradient with respect to $\boldsymbol{\theta}$ gives

$$\nabla_{\boldsymbol{\theta}}\tilde{L}(\boldsymbol{\theta}) = \sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta} - \boldsymbol{\theta}_i^*), \tag{8}$$

where the Hessian matrices are symmetric. Setting the gradient to zero yields

$$\sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta}^* - \boldsymbol{\theta}_i^*) = \mathbf{0}. \tag{9}$$

Rearranging gives

$$\left[\sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)\right]\boldsymbol{\theta}^* = \sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)\boldsymbol{\theta}_i^*. \tag{10}$$

Assuming that $\sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)$ is invertible, we obtain

$$\boldsymbol{\theta}^* = \left[\sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)\right]^{-1}\sum_{i=1}^{T}\alpha_i\mathbf{H}_i(\boldsymbol{\theta}_i^*)\boldsymbol{\theta}_i^*. \tag{11}$$

B.2 PROOF OF EQUATION 2

For each task i and sample $\boldsymbol{x}$, we locally approximate the task-specific loss $\ell_i(\boldsymbol{z};\boldsymbol{x})$ around the task-specific representation $\boldsymbol{z}_i^*$. Its second-order Taylor approximation is

$$\ell_i(\boldsymbol{z};\boldsymbol{x}) \approx \ell_i(\boldsymbol{z}_i^*;\boldsymbol{x}) + \boldsymbol{g}_i(\boldsymbol{z}_i^*)^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{z}_i^*) + \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_i^*)^{\mathrm{T}}\mathbf{G}_i(\boldsymbol{z}_i^*)(\boldsymbol{z} - \boldsymbol{z}_i^*), \tag{12}$$

where $\boldsymbol{g}_i(\boldsymbol{z}_i^*) = \nabla_{\boldsymbol{z}}\ell_i(\boldsymbol{z};\boldsymbol{x})\big|_{\boldsymbol{z}=\boldsymbol{z}_i^*}$ and $\mathbf{G}_i(\boldsymbol{z}_i^*) = \nabla_{\boldsymbol{z}}^2\ell_i(\boldsymbol{z};\boldsymbol{x})\big|_{\boldsymbol{z}=\boldsymbol{z}_i^*}$. Unlike the parameter-space case, $\boldsymbol{z}_i^*$ is not necessarily a sample-wise optimum of $\ell_i(\boldsymbol{z};\boldsymbol{x})$, and hence $\boldsymbol{g}_i(\boldsymbol{z}_i^*)$ is generally nonzero. Given the preference vector $\boldsymbol{\alpha}$, we consider the preference-weighted local objective. After omitting terms independent of $\boldsymbol{z}$, we obtain

$$\tilde{\ell}(\boldsymbol{z};\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{T}\alpha_i(\boldsymbol{z} - \boldsymbol{z}_i^*)^{\mathrm{T}}\mathbf{G}_i(\boldsymbol{z}_i^*)(\boldsymbol{z} - \boldsymbol{z}_i^*) + \sum_{i=1}^{T}\alpha_i\boldsymbol{g}_i(\boldsymbol{z}_i^*)^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{z}_i^*). \tag{13}$$

Taking the gradient with respect to $\boldsymbol{z}$ gives

$$\nabla_{\boldsymbol{z}}\tilde{\ell}(\boldsymbol{z};\boldsymbol{x}) = \sum_{i=1}^{T}\alpha_i\left[\mathbf{G}_i(\boldsymbol{z}_i^*)(\boldsymbol{z} - \boldsymbol{z}_i^*) + \boldsymbol{g}_i(\boldsymbol{z}_i^*)\right], \tag{14}$$

where $\mathbf{G}_i(\boldsymbol{z}_i^*)$ is symmetric as a Hessian matrix. Setting the gradient to zero at $\boldsymbol{z}^*$ yields

$$\sum_{i=1}^{T} \alpha_i \left[\mathbf{G}_i(\boldsymbol{z}_i^*)(\boldsymbol{z}^* - \boldsymbol{z}_i^*) + \boldsymbol{g}_i(\boldsymbol{z}_i^*)\right] = \mathbf{0}. \tag{15}$$

Rearranging the terms gives

$$\left[\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)\right] \boldsymbol{z}^* = \sum_{i=1}^{T} \alpha_i \left[\mathbf{G}_i(\boldsymbol{z}_i^*)\boldsymbol{z}_i^* - \boldsymbol{g}_i(\boldsymbol{z}_i^*)\right]. \tag{16}$$

Assuming that $\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)$ is invertible, the stationary solution is

$$\boldsymbol{z}^* = \left[\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)\right]^{-1} \sum_{i=1}^{T} \alpha_i \left[\mathbf{G}_i(\boldsymbol{z}_i^*)\boldsymbol{z}_i^* - \boldsymbol{g}_i(\boldsymbol{z}_i^*)\right]. \tag{17}$$

If the weighted curvature $\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)$ is positive definite, this stationary point is the unique minimizer of the local quadratic objective.

B.3 Proof of Theorem 1

Proof. We first establish the necessary and sufficient condition for the existence of a linear transformation $\mathbf{T}^{\boldsymbol{z}}$ satisfying $\mathbf{T}^{\boldsymbol{z}}\boldsymbol{z}' = \boldsymbol{z}^*$.

Necessary condition. Suppose such a transformation exists. If $\boldsymbol{z}' = \mathbf{0}$, then $\boldsymbol{z}^* = \mathbf{T}^{\boldsymbol{z}}\boldsymbol{z}' = \mathbf{T}^{\boldsymbol{z}}\mathbf{0} = \mathbf{0}$. Therefore, a necessary condition is $\boldsymbol{z}' = \mathbf{0} \Rightarrow \boldsymbol{z}^* = \mathbf{0}$.

Sufficient condition. We next show that this condition is also sufficient. If $\boldsymbol{z}' = \mathbf{0}$, the condition implies $\boldsymbol{z}^* = \mathbf{0}$, and any matrix $\mathbf{T}^{\boldsymbol{z}}$ satisfies $\mathbf{T}^{\boldsymbol{z}}\boldsymbol{z}' = \boldsymbol{z}^*$. If $\boldsymbol{z}' \neq \mathbf{0}$, consider

$$\mathbf{T}_0^{\boldsymbol{z}} = \frac{\boldsymbol{z}^* \boldsymbol{z}'^{\mathrm{T}}}{\|\boldsymbol{z}'\|_2^2}. \tag{18}$$

Then

$$\mathbf{T}_0^{\boldsymbol{z}}\boldsymbol{z}' = \frac{\boldsymbol{z}^* \boldsymbol{z}'^{\mathrm{T}}\boldsymbol{z}'}{\|\boldsymbol{z}'\|_2^2} = \frac{\boldsymbol{z}^* \|\boldsymbol{z}'\|_2^2}{\|\boldsymbol{z}'\|_2^2} = \boldsymbol{z}^*. \tag{19}$$

Hence, a transformation exists whenever $\boldsymbol{z}' \neq \mathbf{0}$. Therefore,

$$\exists\, \mathbf{T}^{\boldsymbol{z}} \text{ such that } \mathbf{T}^{\boldsymbol{z}}\boldsymbol{z}' = \boldsymbol{z}^* \quad \Longleftrightarrow \quad \boldsymbol{z}' = \mathbf{0} \Rightarrow \boldsymbol{z}^* = \mathbf{0}. \tag{20}$$

Under the assumption in Theorem 1 that $\boldsymbol{z}' \neq \mathbf{0}$, this condition is automatically satisfied. Thus, for

$$\boldsymbol{z}' = \sum_{i=1}^{T} \alpha_i \boldsymbol{z}_i^*; \quad \boldsymbol{z}^* = \left[\sum_{i=1}^{T} \alpha_i \mathbf{G}_i(\boldsymbol{z}_i^*)\right]^{-1} \sum_{i=1}^{T} \alpha_i \mathbf{G}_i^{\boldsymbol{g}}(\boldsymbol{z}_i^*)\boldsymbol{z}_i^*, \tag{21}$$

there always exists a transformation $\mathbf{T}^{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{\alpha})$ satisfying $\mathbf{T}^{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{\alpha})\boldsymbol{z}' = \boldsymbol{z}^*$. Here, the dependence on $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ follows from the sample- and preference-dependent representations and the resulting $\boldsymbol{z}'$ and $\boldsymbol{z}^*$. We finally characterize all such transformations. Define the orthogonal projection onto $\mathrm{span}(\boldsymbol{z}')$ as $\Pi_{\boldsymbol{z}'} = \frac{\boldsymbol{z}'\boldsymbol{z}'^{\mathrm{T}}}{\|\boldsymbol{z}'\|_2^2}$. Since $(\mathbf{I} - \Pi_{\boldsymbol{z}'})\boldsymbol{z}' = \mathbf{0}$ for any matrix $\mathbf{Q}$,

$$\mathbf{T}^{\boldsymbol{z}} = \frac{\boldsymbol{z}^* \boldsymbol{z}'^{\mathrm{T}}}{\|\boldsymbol{z}'\|_2^2} + \mathbf{Q}(\mathbf{I} - \Pi_{\boldsymbol{z}'}) \tag{22}$$

satisfies

$$\mathbf{T}^{\boldsymbol{z}}\boldsymbol{z}' = \frac{\boldsymbol{z}^* \boldsymbol{z}'^{\mathrm{T}}\boldsymbol{z}'}{\|\boldsymbol{z}'\|_2^2} + \mathbf{Q}(\mathbf{I} - \Pi_{\boldsymbol{z}'})\boldsymbol{z}' = \boldsymbol{z}^*. \tag{23}$$

□

C Related Work

MTL with a Single Tradeoff Solution. We first review some representative works on optimization for MTL that seek a single tradeoff solution among competing tasks, rather than architectural innovations in neural network design for MTL. These methods typically adopt a shared backbone with task-specific heads (Caruana, 1997), while addressing conflicting objectives and mismatched loss scales that can degrade overall performance and lead to imbalanced training. Several methods adjust task contributions through loss reweighting, including GradNorm (Chen et al., 2018), UW (Kendall et al., 2018), DWA (Liu et al., 2019), and Auto-λ (Liu et al., 2022), to balance task learning and promote positive transfer. Meanwhile, methods such as MGDA (Désidéri, 2012; Sener & Koltun, 2018), GradDrop (Chen et al., 2020), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), Gradient Vaccine (Wang et al., 2021), RotoGrad (Javaloy & Valera, 2022), NashMTL (Navon et al., 2022), FAMO (Liu et al., 2023), MoCo (Fernando et al., 2023), and IMGrad (Zhou et al., 2025a) manipulate task gradients to mitigate optimization conflicts and improve joint learning. There are also methods that jointly consider loss reweighting and gradient manipulation (Liu et al., 2021b; Lin et al., 2022a). Zhou et al. (2025b) uses LoRA to identify an alternative loss-equivalent solution with reduced task conflicts. However, these methods are not designed to accommodate arbitrary user-specified preferences.

Pareto Front Approximation. To approximate the Pareto front, several methods identify a discrete set of tradeoff solutions, including Pareto MTL (Lin et al., 2019), EPO (Mahapatra & Rajan, 2020), MOO-SVGD (Liu et al., 2021c), GMOOAR (Chen & Kwok, 2022), and PNG (Ye & Liu, 2022). In addition, Ma et al. (2020) searches for more tradeoff solutions in the vicinity of the obtained solutions, while Zhou et al. (2025c) starts from task-specific models and leverages mode connectivity to construct a continuous segment of tradeoff solutions. However, these methods provide only partial or discrete approximations rather than a continuous parameterization of the entire Pareto front.

Pareto front learning, by contrast, provides a continuous, tradeoff-conditioned approximation of the Pareto front. Pareto hypernetwork-based methods (Lin et al., 2020; Navon et al., 2021; Hoang et al., 2023; Bhattacharya & Bal, 2025) directly map a tradeoff to the parameters of a target model, but incur substantial overhead for large target models. To improve scalability, COSMOS (Ruchte & Grabocka, 2021) augments the target-model input with the tradeoff, while Dosovitskiy & Djolonga (2019) conditions feature transformations on the tradeoff using FiLM (Perez et al., 2018). However, their limited parameterization capacity can lead to suboptimal performance. PaMaL (Dimitriadis et al., 2023) addresses these limitations by constructing preference-weighted combinations of task-specific models while weighting the task objectives using the same preferences. To improve parameter efficiency, subsequent approaches leverage linearly interpolated LoRA modules to capture task-specific knowledge built upon a base model (Chen & Kwok, 2024; Dimitriadis et al., 2025). The use of preferences as interpolation weights has also been extended to LLM alignment (Zhong et al., 2024) and model merging (Tang et al., 2024; Chen & Kwok, 2025) to obtain diverse tradeoff solutions. However, these methods restrict tradeoff solutions to the convex hull of the model parameters, potentially hindering faithful modeling of the underlying Pareto manifold.

LoRA Merging. LoRA (Hu et al., 2022) enables parameter-efficient fine-tuning by learning low-rank updates that capture task-specific adaptations from a pretrained model. LoRA merging extends the model-merging paradigm to low-rank adapters, with both approaches combining multiple separately trained task-specific experts using strategies such as task arithmetic (Ilharco et al., 2023; Zhang et al., 2023; Yadav et al., 2023), weighted interpolation (Yang et al., 2024c; Huang et al., 2024), or learned gating (Wu et al., 2023; Muqeeth et al., 2024). In contrast, our method adopts the paradigm of jointly training task-specific LoRA components, with task specificity established by aligning their interpolation weights with the corresponding task-objective weights. Some approaches also jointly train multiple LoRA experts, but typically rely on input-dependent routing or combination without explicit expert-to-task correspondence (Dou et al., 2024; Liu & Luo, 2024).

Furthermore, Matena & Raffel (2022), Daheim et al. (2024), and Nguyen et al. (2025) explicitly approximate Fisher or curvature matrices, or introduce learnable modules to model curvature, in parameter space, thereby incurring costly task-specific arithmetic operations. Representation Surgery (Yang et al., 2024a) and its variants (Yang et al., 2024b; Wei et al., 2025; Wu et al., 2026) manipulate representations for expert merging but rely on pretrained task-specific experts for supervision, which differs from our setting. In addition, Guo et al. (2025) introduces a sparse transforma-

tion of a similar form over representations, but each task-specific transformation repeatedly depends on representations from other tasks, further increasing the computational burden. In contrast, our method performs curvature-motivated sparse transformation for LoRA merging in the representation space without explicitly modeling a separate transformation for each task. Instead, the angle embedding is generated through preference-weighted aggregation of task-specific LoRA outputs, avoiding repeated task-specific operations for individual experts while implicitly incorporating task-dependent transformation effects.