Efficient Pareto Manifold Learning with Low-Rank Structure

Weiyu Chen [1] **James T. Kwok** [1]

Abstract

Multi-task learning, which optimizes performance across multiple tasks, is inherently a multi-objective optimization problem. Various algorithms are developed to provide discrete trade-off solutions on the Pareto front. Recently, continuous Pareto front approximations using a linear combination of base networks have emerged as a compelling strategy. However, it suffers from scalability issues when the number of tasks is large. To address this issue, we propose a novel approach that integrates a main network with several low-rank matrices to efficiently learn the Pareto manifold. It significantly reduces the number of parameters and facilitates the extraction of shared features. We also introduce orthogonal regularization to further bolster performance. Extensive experimental results demonstrate that the proposed approach outperforms state-of-the-art baselines, especially on datasets with a large number of tasks.

1. Introduction

Multi-task learning (MTL) (Caruana, 1997; Vandenhende et al., 2021) endeavors to efficiently and effectively learn multiple tasks, leveraging shared information to enhance overall performance. As tasks may have disparate scales and potentially conflicting objectives, the challenge of balancing the tasks is a critical aspect of MTL.

Building on the seminal work that conceptualizes MTL as a multi-objective optimization (MOO) problem (Sener & Koltun, 2018), a suite of algorithms has been developed to obtain trade-off solutions along the Pareto front (PF). Examples include EPO (Mahapatra & Rajan, 2020), PMTL (Lin et al., 2019), CAGrad (Liu et al., 2021a), and Nash-MTL (Navon et al., 2022). However, these methods are limited to producing a finite set of discrete trade-off solutions, lacking the flexibility to adapt to varying user preferences.

Continuous approximation of the PF aims to provide an arbitrary number of solutions on the PF based on the user-provided preference vector. Pareto hypernetwork-based methods (Navon et al., 2020b; Hoang et al., 2023) utilize a hypernetwork (Ha et al., 2016) to learn the parameter space of a base network. This hypernetwork accepts preference vector as input and outputs the corresponding parameter of base network. Despite its efficacy, the size of the hypernetwork significantly exceeds that of the base network, thus restricting its use to small base networks. COSMOS (Ruchte & Grabocka, 2021) attempts to circumvent this limitation by incorporating preference vectors directly into the base network as an extra input. However, it falls short of achieving satisfactory results due to the limited parameter space.

Pareto Manifold Learning (PaMaL) (Dimitriadis et al., 2023) proposes to find a collection of base networks whose linear combination in the parameter space based on the preference vectors yields the corresponding Pareto-optimal solution. This approach exhibits promising performance. However, as we need to maintain a separate base network for each task, it incurs a corresponding surge in the number of parameters when we have a large number of tasks. Moreover, base networks cannot learn from each other during training. This lack of sharing poses challenges when the ensemble comprises a large number of base networks.

To surmount these challenges, we introduce a novel approach that employs a main network and multiple low-rank matrices, which can significantly reduce the number of parameters when the number of tasks is large. Moreover, this design facilitates the extraction of shared features through the main network while simultaneously capturing task-specific differences via the low-rank matrices. We further integrate orthogonal regularization to promote disentanglement of these matrices. Empirical investigations demonstrate that the proposed algorithm outperforms PaMaL and other baselines, particularly when the number of tasks is large.

Notations. We denote the set $\{1, \ldots, m\}$ as $[m]$, and the m-dimensional simplex $\{\boldsymbol{\alpha} \mid \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0\}$ as Δ^m. The norm $\|\cdot\|$ refers to the Euclidean norm when

[1]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. Correspondence to: Weiyu Chen <wchenbx@cse.ust.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

applied to vectors, and to the Frobenius norm when applied to matrices.

2. Background

2.1. Multi-Objective Optimization

A multi-objective optimization (MOO) (Miettinen, 1999) problem can be formulated as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{d\times k}} \quad \boldsymbol{F}(\boldsymbol{\theta}) = [f_1(\boldsymbol{\theta}), \ldots, f_m(\boldsymbol{\theta})]^\top, \tag{1}$$

where $m \geq 2$ is the number of objectives, and $d \times k$ is the shape of the parameter matrix. In this paper, we represent the parameters in matrix form for convenience when discussing low-rank approximations.

Definition 2.1 (Pareto Dominance and Pareto-Optimal (Miettinen, 1999))**.** A solution $\boldsymbol{\theta}_1$ is *dominated* by another solution $\boldsymbol{\theta}_2$ if and only if $f_i(\boldsymbol{\theta}_1) \geq f_i(\boldsymbol{\theta}_2)$ for $i \in [m]$, and $\exists i \in [m], f_i(\boldsymbol{\theta}_1) > f_i(\boldsymbol{\theta}_2)$. A solution $\boldsymbol{\theta}^*$ is *Pareto-optimal* if and only if it is not dominated by any other $\boldsymbol{\theta}'$.

A *Pareto set* (PS) is the set of all Pareto-optimal solutions. The *Pareto front* (PF) refers to the functional values associated with the solutions in the PS.

2.2. Multi-Task Learning

Multi-task learning (MTL) (Caruana, 1997; Vandenhende et al., 2021) refers to learning multiple tasks simultaneously. In this paper, we focus on the optimization part, adopting the widely utilized shared-bottom architecture (Caruana, 1997) as the base network. More specifically, for a m-task network, we have a shared-bottom $\boldsymbol{\theta}^{sh}$ and m task-specific heads $\boldsymbol{\theta}^{t_1}, \ldots, \boldsymbol{\theta}^{t_m}$. Throughout this paper, our primary focus is on the shared-bottom $\boldsymbol{\theta}^{sh}$. Hence, any mention of $\boldsymbol{\theta}$ hereafter refers specifically to $\boldsymbol{\theta}^{sh}$.

MTL can be viewed as a MOO problem (Sener & Koltun, 2018). To deal with the possible conflicts of different tasks on the shared bottom, various algorithms have been proposed, such as MGDA (Sener & Koltun, 2018; Désidéri, 2012), PCGrad (Yu et al., 2020), IMTL (Liu et al., 2020), Graddrop (Chen et al., 2020), RLW (Lin et al., 2022a), CAGrad (Liu et al., 2021a), Nash-MTL (Navon et al., 2022), and Auto-λ (Liu et al., 2022). These approaches, however, predominantly focus on obtaining a single solution rather than capturing the whole PF.

Discrete Approximation of Pareto Front. Lin et al. (2019) propose to generate a set of solutions to approximate the Pareto front. Subsequent research have introduced innovative strategies to enhance the discrete approximation. Notable among these are EPO (Mahapatra & Rajan, 2020), MOO-SVGD (Liu et al., 2021b), GMOOAR (Chen & Kwok, 2022) and PNG (Ye & Liu, 2022). These methods are inherently limited to producing a fixed set of discrete points on the PF. Ma et al. (2020) propose an algorithm to expand these discrete solutions in their vicinity. However, it can only produce PF segments around each discrete solution instead of the whole continuous PF.

Continuous Approximation of Pareto Front. Continuous approximation of PF enables the derivation of solutions tailored to user preferences, offering an arbitrary resolution of the trade-off solutions. Navon et al. (2020b) introduces the concept of a Pareto hypernetwork (PHN), which learns a hypernetwork (Ha et al., 2016) that takes a preference vector as input and generates the corresponding Pareto-optimal network parameters. PHN-HVI (Hoang et al., 2023) further develops the method using hypervolume maximization (Wang et al., 2017). However, scalability of the approach becomes a concern when applying the hypernetwork to large networks, as the size of the hypernetwork is usually much larger than that of the base network itself.

Conditioned one-shot multi-objective search (COSMOS) (Ruchte & Grabocka, 2021) incorporates the preference vector as an additional input, enabling the generation of outputs with different preferences while minimizing the introduction of extra parameters. Dosovitskiy & Djolonga (2019) propose the use of FiLM layers (Perez et al., 2018) to incorporate preferences through channel-wise multiplication and addition operations on the feature map. However, these methods are sometimes constrained by the relatively limited parameter space, which can lead to suboptimal performance.

Pareto manifold learning (PaMaL) (Dimitriadis et al., 2023) aims to jointly learn multiple base networks $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$, such that for any given preference vector $\boldsymbol{\alpha} \in \Delta^m$, the corresponding weighted combination of base networks leads to the Pareto-optimal model:

$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i \boldsymbol{\theta}_i. \tag{2}$$

PaMaL demonstrates commendable performance. However, it only focuses on two or three tasks. When dealing with a large number of tasks, the number of parameters increases significantly as each task i requires its own network $\boldsymbol{\theta}_i$. Moreover, the base networks cannot benefit from each other during training, which can potentially impair performance, especially when scaling to a larger number of base networks.

Learning a continuous PF is also explored in Bayesian optimization (Lin et al., 2022b) and evolutionary optimization (Lin et al., 2023) for engineering problems with number of parameters up to 7. They fail to converge in deep learning scenarios due to the significantly larger parameter spaces.

2.3. Low-Rank Adaptation

Low-rank adaptation (LoRA) has emerged as a popular approach in the field of finetuning large pre-trained models

(Hu et al., 2021). It only finetunes a low-rank matrix instead of the entire model, resulting in improved efficiency and the prevention of overfitting. Some subsequent studies (Ilharco et al., 2022; Yadav et al., 2023; Ortiz-Jimenez et al., 2023) explore the merging of LoRA modules trained on different tasks. These works focus on direct addition or subtraction without incorporating any training. Audibert et al. (2023) proposes fine-tuning a pretrained model for multiple tasks using a single LoRA module (with some task-specific parameters). Wang et al. (2023a) proposes fine-tuning a pretrained model by adding LoRA modules one by one, where the newly-added LoRA module is orthogonal to the previous ones.

The main difference between these methods and the proposed method is that they learn a single LoRA each time and output a single fine-tuned network, while we train multiple low-rank matrices simultaneously and output a subspace of networks. To mitigate dominance of unitary transforms in weight update as observed in LoRA, Wang et al. (2023b) propose to split the LoRA module to multiple LoRA modules. However, none of the existing research considers joint training of an arbitrary convex combination of low-rank matrices to learn a model subspace.

3. Proposed Method

In this section, we introduce a novel approach that involves learning a main module and m low-rank matrices for each layer (Section 3.1). Orthogonal regularization is further introduced in Section 3.2. Finally, the whole algorithm is presented in Section 3.3.

3.1. Low-Rank Structure

We start by examining the similarities of the base networks obtained by PaMaL (Dimitriadis et al., 2023) on *MultiMNIST* (Sabour et al., 2017), using the same experimental setting as PaMaL. Figure 1 shows the layer-wise cosine similarities between two base networks' parameters throughout training. As can be seen, the similarities are almost zero at the beginning of training. As training progresses, a marked increase in similarity is observed. Furthermore, similarities are higher at the lower-level layers. This motivates us to consider reducing the redundancy of base networks.

Consider the m base networks in (2). Denote the parameters in the lth layer of base network $\boldsymbol{\theta}_i$ by $\boldsymbol{\theta}_i^l$. In the lth layer, (2) can be written as:

$$\boldsymbol{\theta}(\boldsymbol{\alpha})^l = \sum_{i=1}^{m} \alpha_i \boldsymbol{\theta}_i^l. \tag{3}$$

Let $\boldsymbol{\theta}_0^l = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\theta}_i^l$. We can rewrite (3) as:

$$\boldsymbol{\theta}(\boldsymbol{\alpha})^l = \sum_{i=1}^{m} \alpha_i \left(\boldsymbol{\theta}_0^l + \frac{1}{m}\sum_{j=1}^{m} (\boldsymbol{\theta}_i^l - \boldsymbol{\theta}_j^l) \right). \tag{4}$$

Given the similarities observed from Figure 1, we approximate the difference between any two modules $\boldsymbol{\theta}_i^l - \boldsymbol{\theta}_j^l$ by a low-rank matrix. Then, we use the low-rank matrix $\boldsymbol{B}_i^l \boldsymbol{A}_i^l$, to replace $\frac{1}{m}\sum_{j=1}^{m}(\boldsymbol{\theta}_i^l - \boldsymbol{\theta}_j^l)$ in (4), leading to:

$$\boldsymbol{\theta}(\boldsymbol{\alpha})^l = \sum_{i=1}^{m} \alpha_i (\boldsymbol{\theta}_0^l + \boldsymbol{B}_i^l \boldsymbol{A}_i^l). \tag{5}$$

We further add a scaling factor s to regulate the significance of the low-rank component, as:

$$\boldsymbol{\theta}(\boldsymbol{\alpha})^l = \boldsymbol{\theta}_0^l + s\sum_{i=1}^{m} \alpha_i \boldsymbol{B}_i^l \boldsymbol{A}_i^l. \tag{6}$$

Through the above transformation, instead of learning m base module in (3), we learn a main module $\boldsymbol{\theta}_0^l$ and m low-rank matrices $\boldsymbol{B}_1^l \boldsymbol{A}_1^l, \ldots, \boldsymbol{B}_m^l \boldsymbol{A}_m^l$. The main modules are expected to capture the common features across multiple tasks, thereby providing a shared foundation that each low-rank adaptation can leverage. An illustrative example of the proposed method is shown in Figure 2.

3.1.1. Computational Efficiency

In the lth layer, let $\boldsymbol{B}_i^l \in \mathbb{R}^{d^l \times r^l}$ and $\boldsymbol{A}_i^l \in \mathbb{R}^{r^l \times k^l}$. The proposed approach reduces the number of parameters to only $d^l k^l + m(d^l r^l + r^l k^l)$. Notably, the rank r^l is usually set to be much smaller compared to k^l and d^l. As a result, the proposed method achieves higher parameter efficiency compared to PaMaL, which has $md^l k^l$ parameters. This difference becomes more significant when dealing with larger networks and a larger number of tasks.

3.1.2. Approximation Power

Similar to PaMaL (Dimitriadis et al., 2023), we show in this section the approximation power of the proposed structure. While PaMaL only considers two tasks, we consider the more general case with m tasks. Denote the optimal mapping from network input $\boldsymbol{x} \in X \subset \mathbb{R}^u$ and preference vector $\boldsymbol{\alpha} \in \Delta^m$ to the corresponding point on the PF as $t(\boldsymbol{x}, \boldsymbol{\alpha}) : X \times \Delta^m \to \mathbb{R}^m$. We have the following Theorem.

Theorem 3.1. *Assume that $X \times \Delta^m$ is compact and $t(\boldsymbol{x}, \boldsymbol{\alpha})$ is continuous. For any $\epsilon > 0$, there exists a ReLU MLP h with main network $\boldsymbol{\theta}_0$ and m low-rank matrices $\boldsymbol{B}_1\boldsymbol{A}_1, \ldots, \boldsymbol{B}_m\boldsymbol{A}_m$, such that $\forall \boldsymbol{x} \in X, \forall \boldsymbol{\alpha} \in \Delta^m$,*

$$\left\| t(\boldsymbol{x}, \boldsymbol{\alpha}) - h\left(\boldsymbol{x}; \boldsymbol{\theta}_0 + \sum_{i=1}^{m} \alpha_i \boldsymbol{B}_i \boldsymbol{A}_i\right) \right\| \leq \epsilon.$$

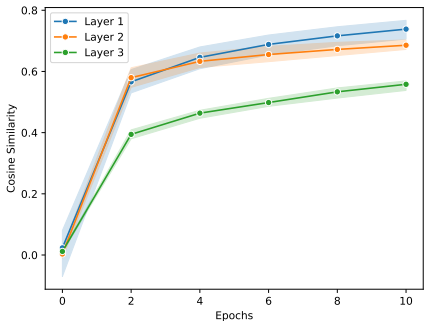


Figure 1. Layer-wise similarities between base networks obtained by PaMaL on *MultiMNIST* over three random seeds. Shaded areas represent the 95% confidence interval.

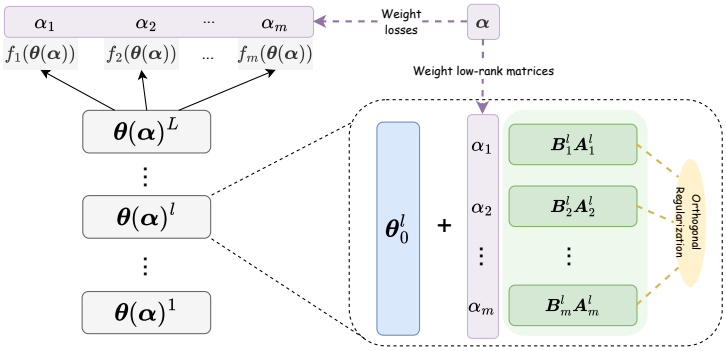


Figure 2. Illustration of the proposed LORPMAN on a L-layer base network with m tasks. For each layer, we aim to learn m low-rank matrices which are orthogonal to each other.

The proof is in Appendix A. This Theorem shows that given any preference vector $\boldsymbol{\alpha}$, the proposed main network with low-rank matrices can approximately output the corresponding point on the PF within any given error margin ϵ.

3.2. Orthogonal Regularization

Orthogonal regularization, which encourages rows or columns of weight matrices to be approximately orthogonal, has been employed in various scenarios to regulate the parameters in a neural network (Xie et al., 2017; Bansal et al., 2018; Wang et al., 2020). Here, we propose to extend it for orthogonality among the low-rank matrices.

First, we flatten each low-rank matrix into a 1-dimensional vector and normalize: $\boldsymbol{w}_i^l = \frac{\texttt{flatten}(\boldsymbol{B}_i^l \boldsymbol{A}_i^l)}{\left\|\texttt{flatten}(\boldsymbol{B}_i^l \boldsymbol{A}_i^l)\right\|}$. Next, we concatenate them to construct a $(d^l k^l) \times m$ matrix $\boldsymbol{W}^l = \texttt{concatenate}(\boldsymbol{w}_1^l, \ldots, \boldsymbol{w}_m^l)$. Our objective is to encourage orthogonality among the matrix columns. This orthogonality loss can be computed as: $\mathcal{R}_o^l = \left\|(\boldsymbol{W}^l)^\top \boldsymbol{W}^l - \boldsymbol{I}\right\|_2^2$, where $\boldsymbol{I}$ is the $m \times m$ identity matrix. Finally, we compute the loss for the entire network as:

$$\mathcal{R}_o = \frac{1}{L} \sum_{l=1}^{L} \mathcal{R}_o^l, \tag{7}$$

where L is the number of layers.

The objective of the orthogonal regularization is to reduce redundancy and conflicts between different low-rank matrices. By reducing redundancy, the model encourages learning common features in the main network and task-specific features in the low-rank matrices, which can lead to more efficient parameter usage and better performance.

Note that the computational complexity of the loss calculation is $O(m^2)$, which becomes expensive for large m. To address this, we use stochastic approximation when m exceeds 3. Specifically, in each iteration, a subset of tasks $\mathcal{T}$, with cardinality 3, is randomly selected from the set of all m tasks. We then construct $\hat{\boldsymbol{W}}^l$ by concatenating $\{\boldsymbol{w}_i^l\}_{i \in \mathcal{T}}$. Then, $\mathcal{R}_o^l$ is estimated as:

$$\mathcal{R}_o^l = \left\|(\hat{\boldsymbol{W}}^l)^\top \hat{\boldsymbol{W}}^l - \boldsymbol{I}\right\|_2^2, \tag{8}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix.

This stochastic approximation ensures the complexity is independent of m. Empirical evaluations in Section 4.3 demonstrate that this still maintains satisfactory performance.

3.3. Optimization

To learn the Pareto manifold, we minimize the expectation of loss given an $\boldsymbol{\alpha}$ over the Dirichlet distribution $Dir(\boldsymbol{p})$. The optimization objective (without regularization) can be written as:

$$\min_{\boldsymbol{\theta}_0, \mathbf{B}_1\mathbf{A}_1, \ldots, \mathbf{B}_m\mathbf{A}_m} \mathbb{E}_{\boldsymbol{\alpha}} \mathbb{E}_{\boldsymbol{\xi}} \left[\sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{\theta}(\boldsymbol{\alpha}); \boldsymbol{\xi}) \right], \tag{9}$$

where $\boldsymbol{\xi}$ is a mini-batch of multi-task training data $\{(x_i^1, x_i^2, \ldots, x_i^m, y_i^1, y_i^2, \ldots, y_i^m)\}_{i=1}^q$ and q is the batch size. For any given $\boldsymbol{\alpha}$, we aim to minimize $\mathbb{E}_{\boldsymbol{\xi}}[\sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta}(\boldsymbol{\alpha}); \boldsymbol{\xi})]$, which is the linear scalarization of the m objective functions. Note that the solution obtained by linear scalarization is Pareto-optimal, which can be formally stated as follows:

Proposition 3.2 ((Boyd & Vandenberghe, 2004))**.** *Given any $\boldsymbol{\alpha} \in \{\boldsymbol{\alpha} \mid \alpha_i > 0\}$, the optimal solution of the scalarized problem $\min_{\boldsymbol{\theta}} \sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta})$ is a Pareto-optimal solution of the original multi-objective optimization problem* (1).

For the reader's convenience, the proof is reproduced in Appendix B. Thus, when (9) is minimized, the obtained

Algorithm 1 LORPMAN.

Input: learnable main model parameters $\boldsymbol{\theta}_0$, learnable matrix $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{i=1}^m$, distribution parameters $\boldsymbol{p}$, window size b, bath size q, regularization coefficients λ_p, λ_o, scaling parameter s.
while not converged **do**
  sample a minibatch of multi-task training data $\boldsymbol{\xi} = \{(x_i^1, x_i^2, \ldots, x_i^m, y_i^1, y_i^2, \ldots, y_i^m)\}_{i=1}^q$;
  independently sample $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^b$ from $\text{Dir}(\boldsymbol{p})$;
  compute corresponding model parameter for each $\boldsymbol{\alpha}^j$ $\boldsymbol{\theta}(\boldsymbol{\alpha}^j)^l = \boldsymbol{\theta}_0^l + s\sum_{i=1}^m \alpha_i^j \boldsymbol{B}_i^l \boldsymbol{A}_i^l$;
  compute multi-forward regularization loss $\mathcal{R}_p$;
  compute orthogonal loss $\mathcal{R}_o$ in Eq. (7);
  compute loss $\mathcal{L} = \sum_{j=1}^b \sum_{i=1}^m \alpha_i^j f_i(\boldsymbol{\theta}(\boldsymbol{\alpha}^j); \boldsymbol{\xi}) + \lambda_p \mathcal{R}_p + \lambda_o \mathcal{R}_o$;
  if current epoch $<$ freeze epoch **then**
    take a gradient descent step on $\boldsymbol{\theta}_0$ and $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{i=1}^m$;
  else
    take a gradient descent step on $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{i=1}^m$;
  end if
end while

$\boldsymbol{\theta}(\boldsymbol{\alpha})$'s are Pareto-optimal solutions. Here, we only consider linear scalarization. More other scalarization methods (Bowman Jr, 1975; Sener & Koltun, 2018; Navon et al., 2022) can be considered.

The proposed algorithm, which is called LOw-Rank Pareto MANifold Learning (LORPMAN), is shown in Algorithm 1. The training process is divided into two phases: First, we adapt both the main model $\boldsymbol{\theta}_0$ and low-rank matrices. After a certain number of epochs (which is referred to as *freeze epoch*), we fix the main model and only adapt the low-rank matrices. This encourages the low-rank matrices to learn task-specific representations instead of always relying on the main model to improve performance.

In each iteration, we sample b preference vectors $\{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^b\}$ from the Dirichlet distribution. For each $\boldsymbol{\alpha}^i$, we compute the corresponding network parameters according to (6). Then, following (Dimitriadis et al., 2023), we calculate the multi-forward regularization loss $\mathcal{R}_p$ which penalizes incorrect solution ordering on the PF (details in Appendix C). Subsequently, we incorporate the orthogonal loss in (7) to encourage orthogonality among the different low-rank matrices, as discussed in the previous section.

4. Experiments

In this section, we first demonstrate the training process of the proposed algorithm on a toy problem (Section 4.1). Then, we perform experiments on datasets with two or three tasks (Section 4.2). Next, we scale up the number of tasks to up to 40 (Section 4.3). We then compare the proposed method with algorithms that generate discrete PFs (Section 4.4) and algorithms that can generate a single solution (Section 4.5). Finally, ablation studies are presented in Section 4.6.

We adopt the Hypervolume (HV) (Zitzler & Thiele, 1998; Knowles et al., 2003) as performance indicator, which is widely used and can evaluate both the convergence and diversity of the obtained PF. Details of the HV indicator are in Appendix D.1. To evaluate the performance of the obtained PFs, following (Dimitriadis et al., 2023), we sample 11 solutions on the obtained PF in the two-objective cases, 66 solutions in the three-objective cases, and 100 solutions when there are more than three objectives. We tune the hyperparameters according to the HV value on the validation datasets.

We compare the proposed method with state-of-the-art continuous PF approximation algorithms, namely PHN (Navon et al., 2020b), PHN-HVI (Hoang et al., 2023), COSMOS (Ruchte & Grabocka, 2021), and PaMaL (Dimitriadis et al., 2023). Since PHN and PHN-HVI require training a hypernetwork with significantly more parameters than the base network, they can only be run on *MultiMNIST* and *Census*.

4.1. Illustrative Example

To demonstrate the training process of the proposed algorithm, we employ a widely used two-objective toy problem (Yu et al., 2020; Liu et al., 2021a; Navon et al., 2022), with parameter $\boldsymbol{\theta} = [\theta^1, \theta^2] \in \mathbb{R}^2$. The detailed definition of the toy problem is in Appendix D.2.

We initialize $\boldsymbol{\theta}_0$ to $[4.5, 4.5]$. Since the parameters of the toy problem do not have a matrix structure, instead of using $\boldsymbol{B}_i\boldsymbol{A}_i$s for low-rank approximation, we use $\Delta\boldsymbol{\theta}_i \in \mathbb{R}^2$, where the second component is fixed to mimic the low-rank approximation constraint. We initialize $\Delta\boldsymbol{\theta}_1 = [-4.5, 0.0]$ and $\Delta\boldsymbol{\theta}_2 = [4.5, 0.0]$. Given an $\boldsymbol{\alpha}$, the corresponding Pareto optimal parameter is $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \boldsymbol{\theta}_0 + \alpha_1\Delta\boldsymbol{\theta}_1 + \alpha_2\Delta\boldsymbol{\theta}_2$.

Figure 3 shows the trajectories of $\boldsymbol{\theta}$ obtained by LORPMAN with $\boldsymbol{\alpha} = [0.5, 0.5]$, $[1, 0]$, and $[0, 1]$. We can see that $\boldsymbol{\theta}$ reaches the middle of the PF for $\boldsymbol{\alpha} = [0.5, 0.5]$, and to the two extreme points of the PF when $\boldsymbol{\alpha} = [1, 0]$ and $[0, 1]$. From Figure 3b, we can see that these three solutions are on the same line in parameter space and we can obtain the whole Pareto set by varying $\boldsymbol{\alpha}$.

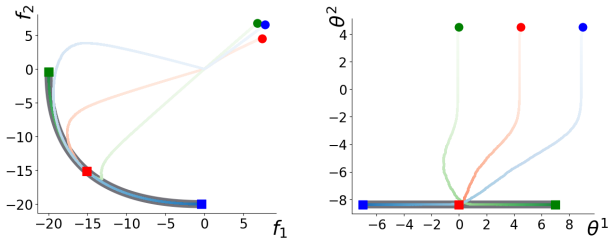
4.2. Datasets with Two or Three Tasks

In this section, we examine the performance on datasets with two or three tasks as used in (Dimitriadis et al., 2023), namely, *MultiMNIST*, *Census*, and *UTKFace*.

MultiMNIST and Census. *MultiMNIST* (Sabour et al., 2017) is a digit classification dataset with two tasks: classi-

Table 1. HV values obtained on *MultiMNIST* and *Census* (averaged over three random seeds). The standard deviation is shown in parentheses. For PHN and PHN-HVI, we report the number of parameters in the hypernetwork.

	MultiMNIST		*Census*	
	HV	# Parameters	HV	# Parameters
PHN	0.900 (0.0068)	2.793M	0.649 (0.0007)	12.251M
PHN-HVI	0.906 (0.0020)	2.793M	0.650 (0.0008)	12.251M
COSMOS	0.888 (0.0077)	0.028M	0.652 (0.0014)	0.122M
PaMaL	0.905 (0.0010)	0.055M	0.654 (0.0006)	0.242M
LORPMAN	**0.918 (0.0018)**	0.046M	**0.656 (0.0004)**	0.133M



(a) Objective space. (b) Parameter space.

Figure 3. Trajectory of $\boldsymbol{\theta}$ obtained by LORPMAN with $\boldsymbol{\alpha} = [0.5, 0.5]$ (red), $\boldsymbol{\alpha} = [1, 0]$ (green), and $\boldsymbol{\alpha} = [0, 1]$ (blue) in objective space (a) and parameter space (b). Circles denote the initial points and squares denote the final points. The gray lines in (a) and (b) denote the PF and PS, respectively.

fication of the top-left digit and classification of the bottom-right digit in each image. *Census* (Kohavi, 1996) is a tabular dataset with two tasks: age prediction and education level classification.

Following (Dimitriadis et al., 2023), we use the LeNet[1] (LeCun et al., 1998) and Multilayer Perceptron (MLP) as shared-bottom of the base network for *MultiMNIST* and *Census*, respectively. For all algorithms, the number of training epochs is set to 10. For LORPMAN, we choose the scaling factor $s \in \{1, 2, 4, 6\}$ and freeze epoch $\in \{4, 6, 8\}$ based on the validation set. For both datasets, the rank r for all layers is set to 8 and the orthogonal regularization coefficient λ_o is set to 1. We do not tune r and λ_o. More experimental details on the two datasets are in Appendices D.3 and D.4, respectively.

The resulting PFs are shown in Figure 4, and the corresponding HV values in Table 1. As can be seen, LORPMAN can obtain the PF closer to the top-right region (i.e., with better accuracies on both objectives). COSMOS exhibits limited accuracies due to the constraints imposed by the small number of parameters. Despite PaMaL, PHN, and PHN-HVI having a larger number of parameters, they show inferior HVs when compared to LORPMAN.

UTKFace. *UTKFace* (Zhang et al., 2017) is a dataset with three tasks: age prediction, gender classification and race classification. Following (Dimitriadis et al., 2023), we use the ResNet-18 (He et al., 2016) as the shared bottom of the base network. The number of training epochs is 100. We tune $\lambda_o \in \{0.1, 0.5, 1\}$, freeze epoch $\in \{60, 80\}$, and $r \in \{16, 32, 64\}$. The scaling factor s is set to 1 without tuning. More experimental details are in Appendix D.5. We do not compare with PHN and PHN-HVI because using the same hypernetwork structure as in *MultiMNIST* and *Census* will result in a hypernetwork with approximately 1 billion parameters.

Table 2 shows the HVs obtained by COSMOS, PaMaL, and LORPMAN and the number of parameters of each algorithm. As can be seen, LORPMAN outperforms PaMaL and COSMOS, despite LORPMAN has fewer parameters than PaMaL. Figure 5 shows the PFs obtained by PaMaL and LORPMAN.[2] We can see the solutions obtained by LORPMAN outperform PaMaL in all three objectives.

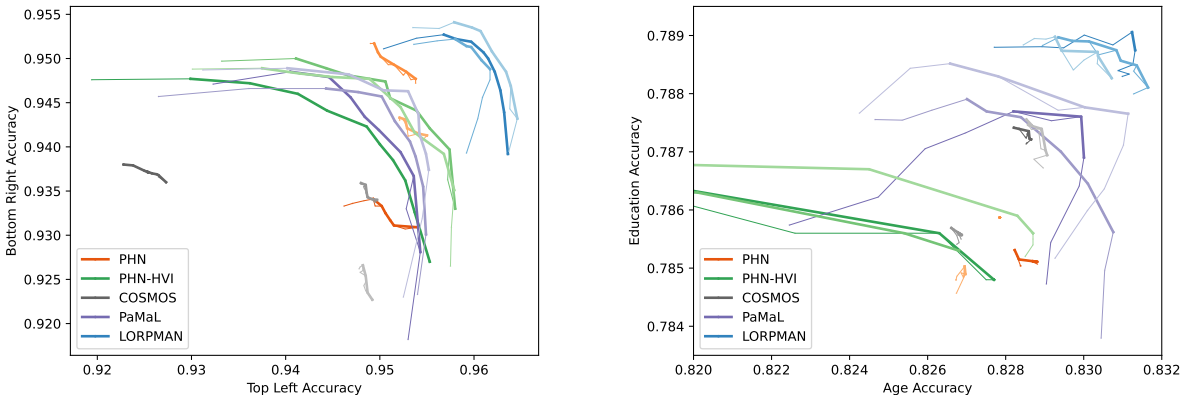
4.3. Scale to Large Number of Tasks

CIFAR-100. CIFAR-100 (Krizhevsky et al., 2009) is an image classification dataset with 100 classes. These classes are further organized into 20 superclasses. As in (Rosenbaum et al., 2017; Yu et al., 2020; Liu et al., 2022), we consider each superclass as a task. The objective of each task is to accurately classify the image into one of the corresponding 5 more-specific classes.

Following (Liu et al., 2022), we use VGG-16 (Simonyan & Zisserman, 2014) as the base network. The number of training epochs is set to 300 and we freeze the main model after 250 epochs, which is similar to the proportion in *UTKFace*. The scaling factor s is set to 1 without tuning. We search the orthogonal regularization coefficient $\lambda_o \in \{0.001, 0.005, 0.01, 0.1\}$ and rank $r \in \{8, 16\}$. More

[1]The parameters of a convolution layer is a four-dimensional tensor instead of a matrix. We adopt the approach in (Hu et al., 2021) to transform it into a matrix and perform low-rank approximation.

[2]Since the PF obtained by COSMOS is inferior, it is omitted from the figure to maintain visual clarity.



(a) *MultiMNIST*. (b) *Census*.

Figure 4. Test performance on *MultiMNIST* and *Census*. The PF is shown in bold. We show the results obtained by three different random seeds.

experimental details are in Appendix D.6.

Table 2. HV values obtained on *UTKFace* (averaged over three random seeds). The standard deviation is shown in parentheses.

	HV	# Parameters
COSMOS	0.281 (0.003)	11.2M
PaMaL	0.304 (0.003)	33.6M
LORPMAN	**0.314 (0.001)**	24.0M

The obtained HVs are shown in Table 3. The obtained PFs are in Appendix E. The results highlight the challenges encountered by PaMaL when dealing with a large number of objectives. PaMaL needs to jointly train 20 base networks, which leads to a large number of parameters and a small hypervolume. In contrast, LORPMAN achieves significantly better hypervolume while utilizing only 8.9% of the parameters compared to PaMaL. This underscores the efficiency and effectiveness of LORPMAN in addressing problems with a large number of tasks.

Table 3. HV values obtained on *CIFAR-100* (averaged over three random seeds). The standard deviation is shown in parentheses.

	HV ($\times 10^{-2}$)	# Parameters
COSMOS	0.344 (0.018)	15.0M
PaMaL	0.0583 (0.010)	296.5M
LORPMAN	**0.887 (0.047)**	26.4M

Figure 6 shows the convergence of the training losses of PaMaL with different learning rates. The training loss of LORPMAN (with learning rate 0.01) is also shown for comparison. We can observe that the poor performance of PaMaL on *CIFAR-100* is due to its slow convergence, since it has to jointly train 20 base networks. Simply increasing the learning rate does not help the convergence of PaMaL.

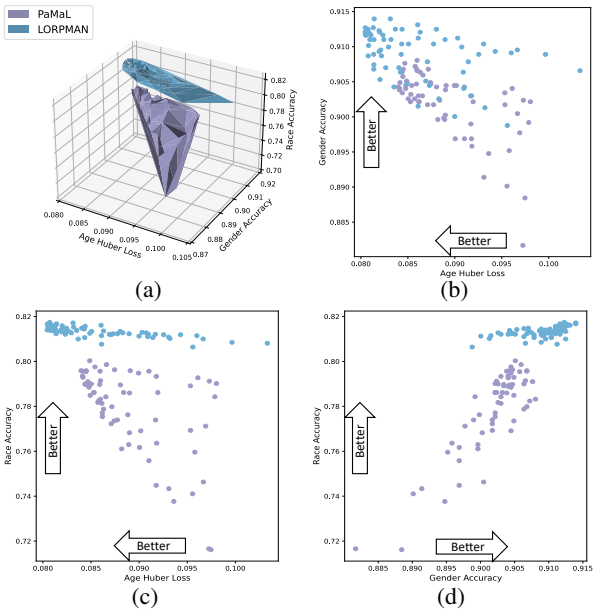


Figure 5. Test performance of PaMaL and LORPMAN on *UTKFace*. Figures (b), (c), (d) are 2D projections of (a) for better illustration of the 3D surface.

CelebA. *CelebA* (Liu et al., 2015) is a face attribute classification dataset with 40 tasks, where each task is a binary classification of a face attribute. Following (Sener & Koltun, 2018), we use ResNet-18 (He et al., 2016) as the base network. The number of training epochs is 50, and we freeze the main model after 40 epochs, which is similar to the proportion in *UTKFace*. The scaling factor s is set to 1 without tuning. We search the orthogonal regularization coefficient $\lambda_o \in \{0.1, 0.5, 1\}$ and rank $r \in \{8, 16, 32\}$. Here, we consider two more baselines: (i) PHN with chunking (Navon et al., 2020a; Lin et al., 2020), both with the original setting in (Navon et al., 2020a) (i.e, the hidden dimension is set to

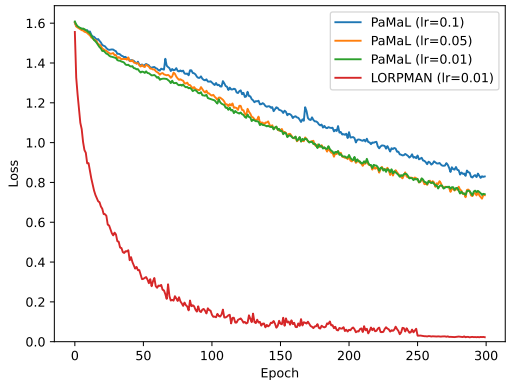


Figure 6. Training loss of PaMaL with number of epochs, using different learning rates.

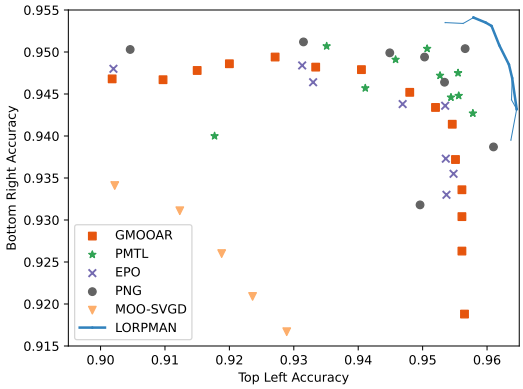


Figure 7. Test performance of LORPMAN and various discrete PF algorithms on *MultiMNIST*.

100) and a scaled setting with comparable number of parameters as LORPMAN (i.e., the hidden dimension is set to 500); and (ii) FiLM condition, with a FiLM condition layer added after each block of ResNet-18. More experimental details are described in Appendix D.7.

The HV values are shown in Table 4. We can see that LORPMAN achieves much better performance while using only 21% of parameters compared to PaMaL. The FiLM condition suffers similar problems as COSMOS due to the limited number of parameters. PHN with chunking also shows worse performance than LORPMAN. In comparison, the proposed LORPMAN is a more straightforward approach that achieves good performance and parameter efficiency.

Table 4. HV values obtained on *CelebA* (averaged over three random seeds). The standard deviation is shown in parentheses.

Method	HV ($\times 10^{-2}$)	# Parameters
PHN-Chunking (original)	0.663 (0.027)	36.6M
PHN-Chunking (scaled)	0.681 (0.048)	92.3M
FiLM	0.803 (0.022)	11.4M
COSMOS	0.783 (0.013)	11.4M
PaMaL	0.472 (0.018)	453.3M
LORPMAN	**1.167 (0.008)**	96.8M

4.4. Comparison with Discrete PF Baselines

In this experiment, we compare with five algorithms that obtain discrete PFs: EPO (Mahapatra & Rajan, 2020), PMTL (Lin et al., 2019), MOO-SVGD (Liu et al., 2021b), PNG (Ye & Liu, 2022), and GMOOAR (Chen & Kwok, 2022). Since most of these algorithms have performed experiments on *MultiMNIST*, we use *MultiMNIST* for comparison. The settings are the same as in Section 4.2.

The obtained solutions are shown in Figure 7. As can be seen, the proposed algorithm has better performance than existing discrete PF algorithms, even though they can only generate a predetermined number of solutions.

4.5. Comparison with Single-Solution Baselines

In this experiment, we compare with multi-task learning baselines that can only generate a single solution. These include Linear Scalarization (LS), UW (Kendall et al., 2018), MGDA (Sener & Koltun, 2018), DWA (Liu et al., 2019), PCGrad (Yu et al., 2020), IMTL (Liu et al., 2020), Graddrop (Chen et al., 2020), CAGrad (Liu et al., 2021a), RLW (Lin et al., 2022a), Nash-MTL (Navon et al., 2022), RotoGrad (Javaloy & Valera, 2021), and Auto-λ (Liu et al., 2022). For a more complete comparison, we also include the results obtained by Single-Task Learning (STL), where each task is trained independently, and the baselines of COSMOS and PaMaL which provide continuous approximations of the PF.

Following (Dimitriadis et al., 2023), we use a widely-used dataset *CityScapes* (Cordts et al., 2016). It is a scene-understanding dataset with two tasks: semantic segmentation and depth regression. We adopt the SegNet (Badrinarayanan et al., 2017) as the base network and use the same parameter configuration as in (Dimitriadis et al., 2023). The number of training epochs is 100. We tune freeze epoch $\in \{80, 90\}$, $s \in \{0.5, 1\}$, and $r \in \{32, 64\}$. More experimental details are in Appendix D.8.

LORPMAN/COSMOS/PaMaL can provide the whole PF. We only select one solution from their obtained PFs for comparison with the single-solution algorithms. Table 5 shows the performance on semantic segmentation (mIOU and pixel accuracy) and depth prediction (absolute and relative error) on the test set. We can see that LORPMAN not only surpasses COSMOS and PaMaL but also outperforms the single-solution algorithms. While MGDA achieves slightly better relative error in depth estimation, it has much worse performance in the semantic segmentation task. It is important to note that our objective is to learn the entire PF rather than focusing on a single solution.

Table 5. Test performance on *CityScapes*. We show the average over three random seeds. Results, excluding LORPMAN, are taken from (Dimitriadis et al., 2023)

	Segmentation		Depth	
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓
STL	70.96	92.12	0.0141	**38.644**
LS	70.12	91.90	0.0192	124.061
UW	70.20	91.93	0.0189	125.943
MGDA	66.45	90.79	0.0141	53.138
DWA	70.10	91.89	0.0192	127.659
PCGrad	70.02	91.84	0.0188	126.255
IMTL	70.77	92.12	0.0151	74.230
Graddrop	70.07	91.93	0.0189	127.146
CAGrad	69.23	91.61	0.0168	110.139
RLW	68.79	91.52	0.0213	126.942
Nash-MTL	71.13	92.23	0.0157	78.499
RotoGrad	69.92	91.85	0.0193	127.281
Auto-λ	70.47	92.01	0.0177	116.959
COSMOS	69.78	91.79	0.0539	136.614
PaMaL	70.35	91.99	0.0141	54.520
LORPMAN	**72.13**	**92.57**	**0.0135**	54.942

4.6. Ablation Studies

In this section, we investigate the effects of (i) orthogonal regularization, (ii) rank r, (iii) freeze epoch, and (iv) scaling factor s on the performance of LORPMAN. Experiment is performed on the *UTKFace* dataset.

Orthogonal Regularization. Table 6 shows the impact of orthogonal regularization on HV and the average correlation[3] over all pairs of low-rank matrices. We can see that with orthogonal regularization, the correlation between low-rank matrices is significantly reduced. Similar observations can also be found in traditional orthogonal regularization for parameters within a single neural network (Xie et al., 2017; Bansal et al., 2018). Such reduction in correlation encourages learning common features in the main network and differences in the low-rank matrices, thus leading to better HV value.

Table 6. Effects of orthogonal regularization on *UTKFace* (averaged over six random seeds).

	HV	Correlation
w/o orthogonal reg	0.309 (0.002)	0.466 (0.045)
w/ orthogonal reg	**0.314 (0.001)**	0.067 (0.012)

Rank r. Table 7 shows the effects of the rank r on the HV and number of parameters. We can observe that LORPMAN with a rank 8 can already outperform COSMOS and PaMaL (see Table 2). By increasing the rank to 64, even better performance can be achieved. However, further increasing the rank does not yield significant change.

[3]The correlation between a pair of low-rank matrices $\mathbf{B}_i\mathbf{A}_i$ and $\mathbf{B}_j\mathbf{A}_j$ is computed as $\texttt{flatten}(\mathbf{B}_i\mathbf{A}_i)^\top \texttt{flatten}(\mathbf{B}_j\mathbf{A}_j)/(\|\mathbf{B}_i\mathbf{A}_i\| \, \|\mathbf{B}_j\mathbf{A}_j\|)$.

Table 7. Effects of rank r on *UTKFace* (averaged over six random seeds).

r	HV	# parameters
4	0.306 (0.002)	12.1M
8	0.310 (0.001)	12.8M
16	0.310 (0.002)	14.5M
32	0.311 (0.001)	17.6M
64	**0.314 (0.001)**	24.0M
128	0.313 (0.002)	35.5M

Freeze Epoch. Table 8 shows the effect of freeze epoch on HV. Compared with not freezing the main module (i.e., freeze epoch = 100), freezing during the latter half of the training process encourages the low-rank matrices to learn task-specific representations instead of always relying on the main model, thus leading to better performance.

Scaling Factor s. Table 9 shows the effect of scaling factor s on HV. As can be seen, setting s within a reasonable range (such as $[0.1, 1]$) leads to stable performance.

Table 8. Effects of freeze epoch on *UTKFace* (averaged over six random seeds).

freeze	HV
20	0.305 (0.001)
40	0.311 (0.001)
60	0.312 (0.002)
80	**0.314 (0.001)**
100	0.307 (0.003)

Table 9. Effects of scaling factor s on *UTKFace* (averaged over six random seeds).

s	HV
0.1	0.313 (0.001)
0.5	0.313 (0.001)
1	**0.314 (0.001)**
1.5	0.312 (0.002)
2	0.310 (0.002)

5. Conclusion

In this paper, we introduce a novel method for continuous approximation of the PF. We use a main network with a collection of low-rank matrices. Our approach leverages the inherent structure of Pareto manifold to effectively learn trade-off solutions between tasks. Extensive empirical evaluation on various datasets demonstrates the superior performance of the proposed algorithm, especially when the number of tasks is large.

One limitation of the proposed work is that we consider the same rank for all layers. Using different ranks for different layers to achieve better parameter efficiency and exploring automatic rank setting strategies could be interesting future directions.

Acknowledgements

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 16202523).

Impact Statement

The proposed algorithm can effectively discover trade-off solutions among multiple tasks, leading to improved performance and cost efficiency in real-world applications involving multiple objectives. However, we should be aware the potential biases and privacy concerns when processing data from different tasks.

References

Audibert, A., Amini, M. R., Usevich, K., and Clausel, M. Low-rank updates of pre-trained weights for multi-task learning. In *Findings of the Association for Computational Linguistics*, pp. 7544–7554, 2023.

Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

Bansal, N., Chen, X., and Wang, Z. Can we gain more from orthogonality regularizations in training deep networks? In *Neural Information Processing Systems*, 2018.

Bowman Jr, V. J. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In *Multiple Criteria Decision Making: Proceedings of a Conference Jouy-en-Josas*, pp. 76–86. Springer, 1975.

Boyd, S. P. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Caruana, R. Multitask learning. *Machine Learning*, 28: 41–75, 1997.

Chen, W. and Kwok, J. Multi-objective deep learning with adaptive reference vectors. In *Neural Information Processing Systems*, pp. 32723–32735, 2022.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Neural Information Processing Systems*, pp. 2039–2050, 2020.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

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

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

Dosovitskiy, A. and Djolonga, J. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, 2019.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. Preprint arXiv:1609.09106, 2016.

Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hoang, L. P., Le, D. D., Tuan, T. A., and Thang, T. N. Improving Pareto front learning via multi-sample hypernetworks. In *AAAI Conference on Artificial Intelligence*, pp. 7875–7883, 2023.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. Preprint arXiv:2106.09685, 2021.

Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in Statistics: Methodology and Distribution*, pp. 492–518. Springer, 1992.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2022.

Javaloy, A. and Valera, I. Rotograd: Gradient homogenization in multitask learning. Preprint arXiv:2103.02631, 2021.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

Knowles, J. D., Corne, D. W., and Fleischer, M. Bounded archiving using the Lebesgue measure. In *Congress on Evolutionary Computation*, pp. 2490–2497, 2003.

Kohavi, R. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *International Conference on Knowledge Discovery and Data Mining*, pp. 202–207, 1996.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kurin, V., De Palma, A., Kostrikov, I., Whiteson, S., and Mudigonda, P. K. In defense of the unitary scalarization for deep multi-task learning. In *Neural Information Processing Systems*, pp. 12169–12183, 2022.

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

Lin, B., Feiyang, Y., Zhang, Y., and Tsang, I. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022a.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. Pareto multi-task learning. In *Neural Information Processing Systems*, 2019.

Lin, X., Yang, Z., Zhang, Q., and Kwong, S. Controllable Pareto multi-task learning. Preprint arXiv preprint arXiv:2010.06313, 2020.

Lin, X., Yang, Z., Zhang, X., and Zhang, Q. Pareto set learning for expensive multi-objective optimization. pp. 19231–19247, 2022b.

Lin, X., Zhang, X., Yang, Z., and Zhang, Q. Evolutionary Pareto set learning with structure constraints. Preprint arXiv:2310.20426, 2023.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. In *Neural Information Processing Systems*, pp. 18878–18890, 2021a.

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2020.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019.

Liu, S., James, S., Davison, A., and Johns, E. Auto-Lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022.

Liu, X., Tong, X., and Liu, Q. Profiling Pareto front with multi-objective stein variational gradient descent. In *Neural Information Processing Systems*, pp. 14721–14733, 2021b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, 2015.

Ma, P., Du, T., and Matusik, W. Efficient continuous Pareto exploration in multi-task learning. In *International Conference on Machine Learning*, pp. 6522–6531, 2020.

Mahapatra, D. and Rajan, V. Multi-task learning with user preferences: Gradient descent with controlled ascent in Pareto optimization. In *International Conference on Machine Learning*, pp. 6597–6607, 2020.

Miettinen, K. *Nonlinear Multiobjective Optimization*, volume 12. Springer Science & Business Media, 1999.

Navon, A., Shamsian, A., Chechik, G., and Fetaya, E. Learning the Pareto front with hypernetworks. Preprint arXiv:2010.04104, 2020a.

Navon, A., Shamsian, A., Fetaya, E., and Chechik, G. Learning the Pareto front with hypernetworks. In *International Conference on Learning Representations*, 2020b.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, pp. 16428–16446, 2022.

Ortiz-Jimenez, G., Favero, A., and Frossard, P. Task arithmetic in the tangent space: Improved editing of pre-trained models. Preprint arXiv:2305.12827, 2023.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Rosenbaum, C., Klinger, T., and Riemer, M. Routing networks: Adaptive selection of non-linear functions for multi-task learning. Preprint arXiv:1711.01239, 2017.

Ruchte, M. and Grabocka, J. Scalable Pareto front approximation for deep multi-objective learning. In *IEEE International Conference on Data Mining*, pp. 1306–1311, 2021.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Neural Information Processing Systems*, 2017.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Neural Information Processing Systems*, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. Preprint arXiv:1409.1556, 2014.

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (7):3614–3633, 2021.

Wang, H., Deutz, A., Bäck, T., and Emmerich, M. Hypervolume indicator gradient ascent multi-objective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 654–669, 2017.

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. Orthogonal convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11505–11515, 2020.

Wang, X., Chen, T., Ge, Q., Xia, H., Bao, R., Zheng, R., Zhang, Q., Gui, T., and Huang, X.-J. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics*, pp. 10658–10671, 2023a.

Wang, Y., Lin, Y., Zeng, X., and Zhang, G. MultiLoRA: Democratizing LoRA for better multi-task learning. Preprint arXiv:2311.11501, 2023b.

Xie, D., Xiong, J., and Pu, S. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6176–6185, 2017.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-merging: Resolving interference when merging models. In *Neural Information Processing Systems*, 2023.

Ye, M. and Liu, Q. Pareto navigation gradient descent: A first-order algorithm for optimization in Pareto set. In *Uncertainty in Artificial Intelligence*, pp. 2246–2255, 2022.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. In *Neural Information Processing Systems*, pp. 5824–5836, 2020.

Zhang, Z., Song, Y., and Qi, H. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5810–5818, 2017.

Zitzler, E. and Thiele, L. Multiobjective optimization using evolutionary algorithms — A comparative case study. In *International Conference on Parallel Problem Solving from Nature*, pp. 292–301, 1998.

A. Proof of Theorem 3.1

Denote the optimal mapping from network input $\boldsymbol{x} \in X \subset \mathbb{R}^u$ and preference vector $\boldsymbol{\alpha} \in \Delta^m$ to the corresponding point on the PF as $t(\boldsymbol{x}, \boldsymbol{\alpha}) : X \times \Delta^m \to \mathbb{R}^m$. We have the following Theorem.

Theorem A.1. *Assume that $X \times \Delta^m$ is compact and $t(\boldsymbol{x}, \boldsymbol{\alpha})$ is continuous. For any $\epsilon > 0$, there exists a ReLU MLP h with main network $\boldsymbol{\theta}_0$ and m low-rank matrices $\boldsymbol{B}_1\boldsymbol{A}_1, \ldots, \boldsymbol{B}_m\boldsymbol{A}_m$, such that $\forall \boldsymbol{x} \in X, \forall \boldsymbol{\alpha} \in \Delta^m$,*

$$\left\| t(\boldsymbol{x}, \boldsymbol{\alpha}) - h\left(\boldsymbol{x}; \boldsymbol{\theta}_0 + \sum_{i=1}^{m} \alpha_i \boldsymbol{B}_i \boldsymbol{A}_i\right) \right\| \leq \epsilon.$$

Proof. Denote $\sigma(\boldsymbol{x}) \equiv \max(\boldsymbol{0}, \boldsymbol{x})$. From the universal approximation theorem (Haykin, 1998), for any $\epsilon > 0$, there exists $v \in \mathbb{N}, \boldsymbol{M} \in \mathbb{R}^{v \times (u+m)}, \boldsymbol{N} \in \mathbb{R}^v, \boldsymbol{C} \in \mathbb{R}^{m \times v}$ such that

$$\sup_{\boldsymbol{x} \in X, \alpha \in \Delta^m} \|t(\boldsymbol{x}, \boldsymbol{\alpha}) - g(\boldsymbol{x}, \boldsymbol{\alpha})\| \leq \epsilon,$$

where $g(\boldsymbol{x}, \boldsymbol{\alpha}) = \boldsymbol{C}\sigma(\boldsymbol{M}[\boldsymbol{x}, \boldsymbol{\alpha}]^\top + \boldsymbol{N})$.

We define two matrices $\boldsymbol{R} \in \mathbb{R}^{u \times (2u+m)}$ and $\boldsymbol{S} \in \mathbb{R}^{(2u+m) \times u}$ as follows:

$$\boldsymbol{R}_{i,j} = \begin{cases} 1, & \text{if } j = 2i - 1 \\ -1, & \text{if } j = 2i \\ 0, & \text{otherwise} \end{cases},$$

and

$$\boldsymbol{S}_{i,j} = \begin{cases} 1, & \text{if } i = 2j - 1 \text{ or } (i > 2u \text{ and } j = u) \\ -1, & \text{if } i = 2j \\ 0, & \text{otherwise} \end{cases}.$$

Define m vectors $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_m \in \{0, 1\}^{2u+m}$ as:

$$(\boldsymbol{U}_i)_j = \begin{cases} 1, & j = 2m + i \\ 0, & \text{otherwise} \end{cases}.$$

Then, we have

$$\boldsymbol{S}\sigma\left(\boldsymbol{R}\boldsymbol{x} + \sum_{i=1}^{m} \alpha_i \boldsymbol{U}_i\right) = [\boldsymbol{x}, \boldsymbol{\alpha}].$$

We can construct a MLP $h(\boldsymbol{x}; \boldsymbol{M}, \boldsymbol{N}, \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{S}, \boldsymbol{U}) = \boldsymbol{C}\sigma(\boldsymbol{M}\boldsymbol{S}\sigma(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{U}) + \boldsymbol{N})$.

Let $\boldsymbol{\theta}_0 = (\boldsymbol{M}, \boldsymbol{N}, \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{S}, \boldsymbol{0})$ and $\boldsymbol{\theta}_i = (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{U}_i), i = 1, \ldots, m$. We have

$$\begin{aligned} h(x; \boldsymbol{\theta}_0 + \sum_{i=1}^{m} \alpha_i \boldsymbol{\theta}_i) &= h(\boldsymbol{x}; \boldsymbol{M}, \boldsymbol{N}, \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{S}, \sum_{i=1}^{m} \alpha_i \boldsymbol{U}_i) \\ &= \boldsymbol{C}\sigma(\boldsymbol{M}\boldsymbol{S}\sigma(\boldsymbol{R}\boldsymbol{x} + \sum_{i=1}^{m} \alpha_i \boldsymbol{U}_i) + \boldsymbol{N}) \\ &= g(\boldsymbol{S}\sigma(\boldsymbol{R}\boldsymbol{x} + \sum_{i=1}^{m} \alpha_i \boldsymbol{U}_i)) \\ &= g(\boldsymbol{x}, \boldsymbol{\alpha}). \end{aligned}$$

Since $\boldsymbol{\theta}_i$ consists of a single non-zero entry (equal to 1) with all other elements being zero, it can be reshaped into a matrix $\boldsymbol{B}_i\boldsymbol{A}_i$ with rank 1. □

B. Proof of Proposition 3.2

Proposition B.1 ((Boyd & Vandenberghe, 2004)). *Given any $\boldsymbol{\alpha} \in \{\boldsymbol{\alpha} \mid \alpha_i > 0\}$, the optimal solution of the scalarized problem $\min_{\boldsymbol{\theta}} \sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta})$ is a Pareto-optimal solution of the original multi-objective optimization problem (1).*

Proof. Suppose $\boldsymbol{\theta}$ is the optimal solution of $\sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta})$ but it is not Pareto optimal. Based on the definition of Pareto optimal, there exists $\boldsymbol{\theta}'$, such that for $i \in [m]$, $f_i(\boldsymbol{\theta}) \geq f_i(\boldsymbol{\theta}')$, and $\exists i \in [m]$, $f_i(\boldsymbol{\theta}) > f_i(\boldsymbol{\theta}')$. Since $\alpha_i > 0$, we have

$$\sum_{i=1}^m \alpha_i (f_i(\boldsymbol{\theta}) - f_i(\boldsymbol{\theta}')) > 0.$$

It can be rewritten as:

$$\sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta}) > \sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta}'),$$

which contradicts the assumption that $\boldsymbol{\theta}$ is the optimal solution of $\sum_{i=1}^m \alpha_i f_i(\boldsymbol{\theta})$. □

C. Multi-Forward Regularization

Dimitriadis et al. (2023) propose the multi-forward regularization to penalize the wrong ordering of solutions. Denote the set of current sampled reference vectors as $\mathcal{V} \equiv \{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^b\}$. Then, a directed graph $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}_i)$ is constructed for each task where $\mathcal{E}_i = \{(\boldsymbol{\alpha}, \boldsymbol{\alpha}') \in \mathcal{V} \times \mathcal{V} : \alpha_i < \alpha_i'\}$. The multi-forward regularization loss is defined as:

$$\mathcal{R}_p = \sum_{i=1}^m \log \left(\frac{1}{|\mathcal{E}_i|} \sum_{(\boldsymbol{\alpha}, \boldsymbol{\alpha}') \in \mathcal{E}_i} e^{[f(\boldsymbol{\theta}(\boldsymbol{\alpha})) - f(\boldsymbol{\theta}(\boldsymbol{\alpha}'))]_+} \right).$$

D. Experimental Details

D.1. Hypervolume

Hypervolume (HV) (Zitzler & Thiele, 1998; Knowles et al., 2003) is a popular indicator in multi-objective optimization (MOO), offering a measure of the performance of the obtained solution set. Formally, for a given solution set $\mathcal{P}$ and a reference point $\boldsymbol{r}$, the HV is defined as:

$$HV(\mathcal{P}; \boldsymbol{r}) = \Lambda(\{\boldsymbol{q} \in \mathbb{R}^m \mid \exists \boldsymbol{p} \in \mathcal{P} : \boldsymbol{p} \geq \boldsymbol{q} \geq \boldsymbol{r}\}), \tag{10}$$

where Λ denotes the Lebesgue measure. Figure 8 provides an illustration of the HV indicator in a two-objective maximization problem. The shaded area encapsulated by the solution set and the reference point represents the HV.

For all datasets except *UTKFace* (Zhang et al., 2017), the reference point for HV evaluation is set to $[0, 0, \ldots, 0]$, since 0 is the smallest-possible accuracy. For *UTKFace*, the first objective is the Huber loss (Huber, 1992), not accuracy. Since 0.5 is close to the observed largest Huber loss in the experiment, we set the reference point to $[0.5, 0, 0]$.

D.2. Toy Problem

We use the two-objective toy problem adopted by (Liu et al., 2021a; Navon et al., 2022; Dimitriadis et al., 2023), with parameters $\boldsymbol{\theta} = [\theta^1, \theta^2] \in \mathbb{R}^2$. The problem is formulated as follows:

$$\begin{aligned}
\text{Minimize } & f_1(\boldsymbol{\theta}) = c_1(\theta) h_1(\theta) + c_2(\theta) g_1(\theta) \;\text{ and }\; f_2(\boldsymbol{\theta}) = c_1(\theta) h_2(\theta) + c_2(\theta) g_2(\theta), \\
\text{where } & h_1(\boldsymbol{\theta}) = \log\big(\max(|0.5(-\theta^1 - 7) - \tanh(-\theta^2)|,\; 0.000005)\big) + 6, \\
& h_2(\boldsymbol{\theta}) = \log\big(\max(|0.5(-\theta^1 + 3) - \tanh(-\theta^2) + 2|,\; 0.000005)\big) + 6, \\
& g_1(\boldsymbol{\theta}) = \big((-\theta^1 + 7)^2 + 0.1 * (-\theta^2 - 8)^2\big)/10 - 20, \\
& g_2(\boldsymbol{\theta}) = \big((-\theta^1 - 7)^2 + 0.1 * (-\theta^2 - 8)^2\big)/10 - 20, \\
& c_1(\boldsymbol{\theta}) = \max(\tanh(0.5 * \theta^2),\; 0) \;\text{ and }\; c_2(\boldsymbol{\theta}) = \max(\tanh(-0.5 * \theta^2),\; 0).
\end{aligned}$$

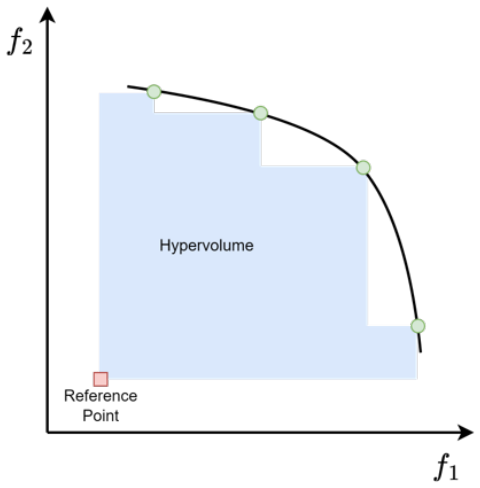


Figure 8. Illustration of the hypervolume indicator.

D.3. MultiMNIST

Network Structure. Following (Dimitriadis et al., 2023), the base network consists of a shared bottom and two task-specific heads. The shared bottom is the LeNet (LeCun et al., 1998) with two convolution layers and a fully-connected layer. Each task-specific head consists of two full-connect layers. For PHN and PHN-HVI, we use the same hypernetwork structure as (Navon et al., 2020b). The input preference vector $\boldsymbol{\alpha}$ first goes through a three-layer MLP to get the ray embedding. Then, for each layer of the base network, a linear layer is used to generate the base network parameters based on the ray embedding.

Parameter Settings. Following (Dimitriadis et al., 2023), for LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 0, and window size b to 4. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 1 for all algorithms. For PHN and PHN-HVI, we use the same parameter setting as in the original paper. We use the Adam optimizer. For COSMOS, the cosine similarity regularization coefficient λ_c is set to 1. The learning rate is set to $1e-3$ and the batch size is set to 256.

D.4. Census

Network Structure. Following (Dimitriadis et al., 2023), the base network consists of a shared bottom and two task-specific heads. The shared bottom is an one-layer MLP. Each task-specific head is a fully-connected layer. The hypernetwork is constructed in the same way as for *MultiMNIST*.

Parameter Settings. Following (Dimitriadis et al., 2023), for LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 5, and window size b to 2. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 1 for all algorithms. For PHN-HVI, we use the same parameter setting as in the original paper. For COSMOS, the cosine similarity regularization coefficient λ_c is set to 1. We use the Adam optimizer. The learning rate is set to $1e-3$ and the batch size is set to 256.

D.5. UTKFace

Network Structure. Following (Dimitriadis et al., 2023), the base network consists of a shared bottom and three task-specific heads. The shared bottom is a ResNet-18 (He et al., 2016). Each task-specific head is a fully-connected layer.

Parameter Settings. Following (Dimitriadis et al., 2023), for LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 1, and window size b to 3. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 2 for all algorithms. For COSMOS, the cosine similarity regularization coefficient λ_c is set to 1. We use the Adam optimizer. The learning rate is set to $1e-3$ and the batch size is set to 256.

D.6. CIFAR-100

CIFAR-100 has 20 5-way classification tasks:
1. Aquatic Mammals: beaver, dolphin, otter, seal, whale
2. Fish: aquarium fish, flatfish, ray, shark, trout
3. Flowers: orchids, poppies, roses, sunflowers, tulips
4. Food Containers: bottles, bowls, cans, cups, plates
5. Fruits and Vegetables: apples, mushrooms, oranges, pears, sweet peppers
6. Household Electrical Devices: clock, computer keyboard, lamp, telephone, television
7. Household Furniture: bed, chair, couch, table, wardrobe
8. Insects: bee, beetle, butterfly, caterpillar, cockroach
9. Large Carnivores: bear, leopard, lion, tiger, wolf
10. Large Man-made Outdoor Things: bridge, castle, house, road, skyscraper
11. Large Natural Outdoor Scenes: cloud, forest, mountain, plain, sea
12. Large Omnivores and Herbivores: camel, cattle, chimpanzee, elephant, kangaroo
13. Medium-sized Mammals: fox, porcupine, possum, raccoon, skunk
14. Non-insect Invertebrates: crab, lobster, snail, spider, worm
15. People: baby, boy, girl, man, woman
16. Reptiles: crocodile, dinosaur, lizard, snake, turtle
17. Small Mammals: hamster, mouse, rabbit, shrew, squirrel
18. Trees: maple, oak, palm, pine, willow
19. Vehicles 1: bicycle, bus, motorcycle, pickup truck, train
20. Vehicles 2: lawn-mower, rocket, streetcar, tank, tractor

Network Structure. Following (Liu et al., 2022), the base network consists of a shared bottom and 20 task-specific heads. The shared bottom is a VGG-16 (Simonyan & Zisserman, 2014). Each task-specific head is a fully-connected layer.

Parameter Settings. For LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 1, and window size b to 2. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 2 for all algorithms. For COSMOS, the cosine similarity regularization coefficient λ_c is set to 1. Following (Liu et al., 2022), we use the SGD optimizer. The learning rate is set to $1e-2$ and the batch size is set to 64.

D.7. CelebA

CelebA (Liu et al., 2015) has 40 binary classification tasks: 5 o'Clock Shadow, Arched Eyebrows, Attractive, Bags Under Eyes, Bald, Bangs, Big Lips, Big Nose, Black Hair, Blond Hair, Blurry, Brown Hair, Bushy Eyebrows, Chubby, Double Chin, Eyeglasses, Goatee, Gray Hair, Heavy Makeup, High Cheekbones, Male, Mouth Slightly Open, Mustache, Narrow Eyes, No Beard, Oval Face, Pale Skin, Pointy Nose, Receding Hairline, Rosy Cheeks, Sideburns, Smiling, Straight Hair, Wavy Hair, Wearing Earrings, Wearing Hat, Wearing Lipstick, Wearing Necklace, Wearing Necktie, Young.

Network Structure. Following (Sener & Koltun, 2018), the base network consists of a shared bottom and 40 task-specific heads. The shared bottom is a ResNet-18 (He et al., 2016). Each task-specific head is a fully-connected layer. Since CelebA has the overfitting problem (Kurin et al., 2022), we adopt dropout regularization as in (Kurin et al., 2022) for all algorithms (i.e., we add a dropout layer in each block as well as after the first convolution layer and after average pooling layer).

Parameter Settings. For LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 1, and window size b to 3. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 2 for all algorithms. For COSMOS, the cosine similarity regularization coefficient λ_c is set to 1. We use the Adam optimizer. The learning rate is set to $1e-3$ and the batch size is set to 128.

D.8. Cityscapes

Network Structure. Following (Liu et al., 2021a; Navon et al., 2022; Dimitriadis et al., 2023), the base network consists of a shared bottom and 2 task-specific heads. The shared bottom is a SegNet (Badrinarayanan et al., 2017). Each task-specific head consists of two convolution layers.

Parameter Settings. Following (Dimitriadis et al., 2023), for LORPMAN and PaMaL, we set the multi-forward regularization coefficient λ_p to 1, and window size b to 3. The distribution parameters p for sampling $\boldsymbol{\alpha}$ is set to 7 for all algorithms. We use the Adam optimizer. The initial learning rate is set to $1e-4$ and is halved after 75 epochs. The batch size is set to 8.

E. Pareto Front Obtained on CIFAR-100

Figure 9 shows the models obtained by COSMOS, PaMaL and LORPMAN. We can observe that the solutions obtained by PaMaL have much lower accuracies while the solutions obtained by COSMOS concentrate in a small region. The proposed LORPMAN achieves good accuracy and diversity.

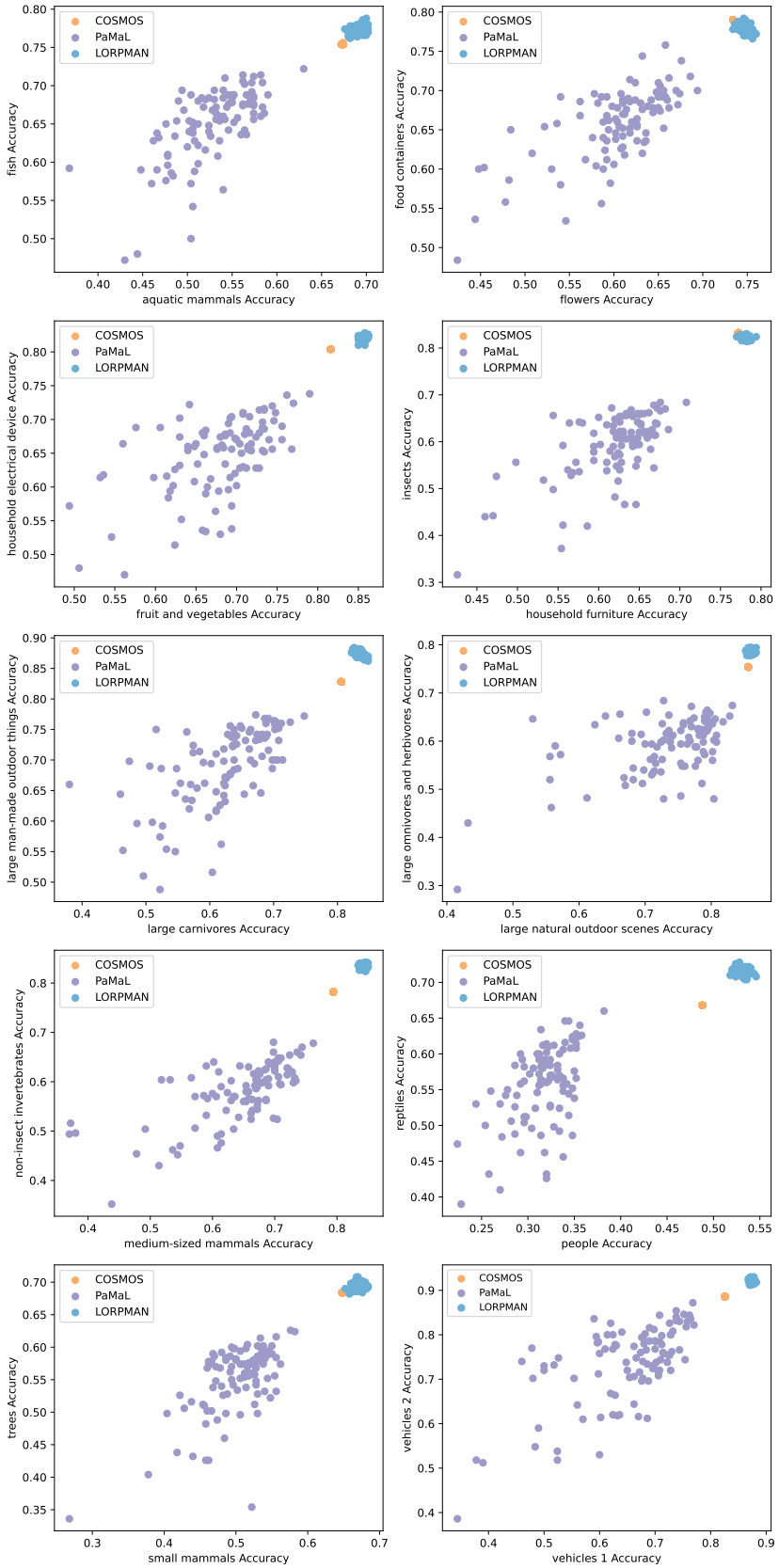


Figure 9. Test performance of COSMOS, PaMaL, and LORPMAN on *CIFAR-100*.