Efficient Continuous Pareto Exploration in Multi-Task Learning

by

Pingchuan Ma

B.Eng., Nankai University (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Author . . . . . .
Department of Electrical Engineering and Computer Science
January 18, 2023

Certified by . . . . . .
Wojciech Matusik
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Efficient Continuous Pareto Exploration in Multi-Task Learning

by

Pingchuan Ma

Submitted to the Department of Electrical Engineering and Computer Science
on January 18, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Tasks in multi-task learning often correlate, conflict, or even compete with each other. As a result, a single solution that is optimal for all tasks rarely exists. Recent papers introduced the concept of Pareto optimality to this field and directly cast multi-task learning as multi-objective optimization problems, but solutions returned by existing methods are typically finite, sparse, and discrete. We present a novel, efficient method that generates locally continuous Pareto sets and Pareto fronts, which opens up the possibility of continuous analysis of Pareto optimal solutions in machine learning problems. We scale up theoretical results in multi-objective optimization to modern machine learning problems by proposing a sample-based sparse linear system, for which standard Hessian-free solvers in machine learning can be applied. We compare our method to the state-of-the-art algorithms and demonstrate its usage of analyzing local Pareto sets on various multi-task classification and regression problems. The experimental results confirm that our algorithm reveals the primary directions in local Pareto sets for trade-off balancing, finds more solutions with different trade-offs efficiently, and scales well to tasks with millions of parameters.

Thesis Supervisor: Wojciech Matusik
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my amazing advisor, Professor Wojciech Matusik.

I would like to thank one of my major collaborators, Professor Tao Du.

I would like to thank all my collaborators, lab mates, families, and friends.

Thank you, mother and father. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conflicting objectives are common in machine learning problems: designing a machine learning model takes into account model complexity and generalizability, training a model minimizes bias and variance errors from datasets, and evaluating a model typically involves multiple metrics that are, more often than not, competing with each other. Such trade-offs among objectives often invalidate the existence of one single solution optimal for all objectives. Instead, they give rise to a set of solutions, known as the Pareto set, with varying preferences on different objectives.

In this paper, we are interested in the topic of recovering Pareto sets in deep multi-task learning (MTL) problems. Despite that MTL is inherently a multi-objective problem and trade-offs are frequently observed in theory and practice, most of prior work focused on obtaining one optimal solution that is universally used for all tasks. To solve this problem, prior approaches proposed new model architectures [Misra et al., 2016] or developed new optimization algorithms [Kendall et al., 2018, Sener and Koltun, 2018]. Work on exploring a diverse set of solutions with trade-offs is surprisingly rare and limited to finite and discrete solutions [Lin et al., 2019]. In this work, we address this challenging problem by proposing an efficient method that reconstructs a first-order accurate continuous approximation to Pareto sets in MTL problems.

The significant leap from finding a discrete Pareto set to discovering a continuous one requires a fundamentally novel algorithm. Typically, generating one solution in a Pareto set is a time-consuming process that requires expensive optimization (e.g., training

a neural network). In order to obtain an efficient algorithm for computing a continuous Pareto set, it is necessary to exploit local information. Our technical method is inspired by second-order methods in multi-objective optimization (MOO) [Hillermeier, 2001, Martín and Schütze, 2018, Schulz et al., 2018] which connect the local tangent plane, the gradient information, and the Hessian matrices at a Pareto optimal solution all in one concise linear equation. This theorem allows us to construct a continuous, first-order approximation of the local Pareto set. However, naively applying this method to deep MTL scales poorly with the number of parameters (e.g., the number of weights in a neural network) due to its need to compute full Hessian matrices. Motivated by other second-order methods in deep learning [Martens, 2010, Vinyals and Povey, 2012], we propose to resolve the scalability issue by using Krylov subspace iteration methods, a family of matrix-free, iterative linear solvers, and present a complete algorithm for generating families of continuous Pareto sets in deep MTL.

We empirically evaluate our method on five datasets with various size and model complexity, ranging from MultiMNIST [Sabour et al., 2017] that consists of 60k images and requires a network classifier with only 20k parameters, to UTKFace [Zhang et al., 2017], an image dataset with 3 objectives and a modern network structure with millions of parameters. The code and data are available online[1]. Experimental results demonstrate that our method generates much denser Pareto sets and Pareto fronts than previous work with small computational overhead compared to the whole MTL training process. We also show in the experiments the continuous Pareto sets can be reparametrized into a low dimensional parameter space, allowing for intuitive manipulation and traversal in the Pareto set. We believe that our efficient and scalable algorithm can open up new possibilities in MTL and foster a deeper understanding of trade-offs between tasks.

[1] https://github.com/mit-gfx/ContinuousParetoMTL

Chapter 2

Related work

Multi-task learning (MTL) is a learning paradigm that jointly optimizes a set of tasks with shared parameters. It is generally assumed that information across different tasks can reinforce the training of shared parameters and improve the overall performance in all tasks. However, since MTL problems share some parameters, performances on different tasks compete with each other. Therefore, trade-offs between performances on different tasks are usually prevalent in MTL. A standard strategy to deal with these trade-offs is to formulate a single-objective optimization problem which assigns weights to each task [Kokkinos, 2017]. Choosing weights for each task is typically empirical, problem-specific, and tedious. To simplify the process of selecting weights, prior work suggests some heuristics on adaptive weights [Chen et al., 2018, Kendall et al., 2018]. However, this family of methods aims to find one optimal solution for all tasks and is not designed for exploring trade-offs.

Instead of solving a weighted sum of tasks as a single objective, some recent papers directly cast MTL as a multi-objective optimization (MOO) problem and introduce multiple gradient-based methods (MGDA) [Fliege and Svaiter, 2000, Désidéri, 2012, Fliege and Vaz, 2016] to MTL. Sener and Koltun [2018] formally formulate MTL as an MOO problem and propose to use MGDA for training a single optimal solution for all objectives. Another recent approach [Lin et al., 2019], which is the most relevant to our setting, pushes the frontier further by pointing out the necessity of exploring Pareto fronts in MTL and presents an MGDA-based method to generate a discrete set of solutions evenly distributed

on the Pareto front. Each solution in their method requires full training from an initial network, which limits its ability to generate a dense set of Pareto optimal solutions.

All the methods discussed so far are based on first-order algorithms in MOO and generate either one solution or a finite set of sparse solutions with trade-offs. A clear distinction between our paper and previous work is that we propose replacing discrete solutions with continuous solution families, allowing for a much denser set of solutions and continuous analysis on them. The advance from discrete to continuous solutions requires a second-order analysis tool in MOO [Hillermeier, 2001, Martín and Schütze, 2018, Schulz et al., 2018], which embeds tangent planes, gradients, and Hessians in one concise linear system. Our work is also related to Hessian-free methods in machine learning [Martens, 2010, Vinyals and Povey, 2012] which rely heavily on Hessian-vector products in neural networks [Pearlmutter, 1994] to solve Hessian systems efficiently.

Chapter 3

Preliminaries

In this work, we consider an unconstrained multi-objective optimization problem described by $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ where each $f_i(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \cdots, m$ represents the objective function of the i-th task to be minimized. For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x}$ dominates $\mathbf{y}$ if and only if $\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{y})$ and $\boldsymbol{f}(\boldsymbol{x}) \neq \boldsymbol{f}(\boldsymbol{y})$. A point $\boldsymbol{x}$ is said to be Pareto optimal if $\boldsymbol{x}$ is not dominated by any points in $\mathbb{R}^n$. Similarly, $\boldsymbol{x}$ is locally Pareto optimal if $\boldsymbol{x}$ is not dominated by any points in a neighborhood of $\boldsymbol{x}$. The Pareto set of this problem consists of all Pareto optimal points, and the Pareto front is the image of the Pareto set. In the context of deep MTL, $\boldsymbol{x}$ represents the parameters of a neural network instance and each $f_i(\boldsymbol{x})$ represents one learning objective, e.g., a certain classification loss.

Similar to single-objective optimization, solving for local Pareto optimality is better established than global Pareto optimality. A standard way is to run gradient-based methods to solve for local Pareto optimality then prune the results. Hillermeier et al. [2001] describes the following necessary condition:

Definition 3.0.1 (Hillermeier et al. 2001)**.** *Assuming each $f_i(\boldsymbol{x})$ is continuously differentiable, a point $\boldsymbol{x}$ is called Pareto stationary if there exists $\boldsymbol{\alpha} \in \mathbb{R}^m$ such that $\alpha_i \geq 0$, $\sum_{i=1}^m \alpha_i = 1$, and $\sum_{i=1}^m \alpha_i \nabla f_i(\boldsymbol{x}) = \mathbf{0}$.*

Proposition 3.0.1 (Hillermeier et al. 2001)**.** *All Pareto optimal points are Pareto stationary.*

Once a Pareto optimal solution $\boldsymbol{x}^*$ is found, previous papers [Hillermeier, 2001, Martín and Schütze, 2018, Schulz et al., 2018] have proven a strong result revealing the first-order

approximation of the local, continuous Pareto set:

Proposition 3.0.2 (Hillermeier 2001)**.** *Assuming that $\boldsymbol{f}(\boldsymbol{x})$ is smooth and $\boldsymbol{x}^*$ is Pareto optimal, consider any smooth curve $\boldsymbol{x}(t) : (-\epsilon, \epsilon) \to \mathbb{R}^n$ in the Pareto set and passing $\boldsymbol{x}^*$ at $t = 0$, i.e., $\boldsymbol{x}(0) = \boldsymbol{x}^*$, then $\exists \boldsymbol{\beta} \in \mathbb{R}^m$ such that:*

$$\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{x}'(0) = \nabla \boldsymbol{f}(\boldsymbol{x}^*)^\top \boldsymbol{\beta} \tag{3.1}$$

where $\boldsymbol{H}(\boldsymbol{x}^*)$ is defined as

$$\boldsymbol{H}(\boldsymbol{x}^*) = \sum_{i=1}^{m} \alpha_i \nabla^2 f_i(\boldsymbol{x}^*) \tag{3.2}$$

and α_i is given by Definition 3.0.1.

In other words, in the Pareto set, for any smooth curve passing $\boldsymbol{x}^*$, $\boldsymbol{H}(\boldsymbol{x}^*)$ transforms its tangent at $\boldsymbol{x}^*$ to a vector in the space spanned by $\{\nabla f_i(\boldsymbol{x}^*)\}$. By gradually changing the curve, its tangent sweeps the tangent plane of the Pareto set at $\boldsymbol{x}^*$. Essentially, the theorem states that $\boldsymbol{H}(\boldsymbol{x}^*)$ connects the whole tangent plane to the column space of $\nabla \boldsymbol{f}(\boldsymbol{x}^*)^\top$. Note that, however, this theorem is not directly applicable to MTL because of its requirement of full Hessians.

Chapter 4

Efficient Pareto Set Exploration

Given an initial $\boldsymbol{x}_0 \in \mathbb{R}^n$, our algorithm is executed in two phases: phase 1 uses gradient-based methods to generate a Pareto stationary solution $\boldsymbol{x}_0^*$ from $\boldsymbol{x}_0$. It then computes a few exploration directions to spawn new $\{\boldsymbol{x}_i\}$. We execute phase 1 recursively by feeding it with a newly generated $\boldsymbol{x}_i$. Phase 2 constructs continuous Pareto sets: we first build a local linear subspace at each Pareto stationary solution by linearly combining its exploration directions. We then check whether two local Pareto fronts collide and stitch them to form a larger continuous set. The major challenge brought by deep MTL is that $\mathbb{R}^n$ is the space of neural network parameters. Therefore, it is computationally prohibitive to explicitly calculate Hessian matrices. We describe phase 1 below and phase 2 will be explained in Chapter 5.

4.1 Gradient-Based Optimization

Our algorithm is compatible with any gradient-based local optimization methods as long as they can return a Pareto stationary solution from any initial $\boldsymbol{x} \in \mathbb{R}^n$. A standard method in MTL is to minimize a weighted sum of objectives with stochastic gradient descent (SGD) [Kokkinos, 2017, Chen et al., 2018, Kendall et al., 2018]. Recent papers [Sener and Koltun, 2018, Lin et al., 2019] also proposed to determine a gradient direction online by solving a small convex problem. Essentially, they minimize a loss by combining gradients with fixed or adaptive weights.

4.2 First-Order Expansion

Once a Pareto stationary point $\boldsymbol{x}_0^*$ is found, we explore its local Pareto set by spawning new points $\{\boldsymbol{x}_i\}$. This is decomposed into two steps: computing $\boldsymbol{\alpha}$ in Definition 3.0.1 at $\boldsymbol{x}_0^*$ and estimating $\{\boldsymbol{v}_i\}$, the basis directions of the tangent plane, from Proposition 3.0.2. The new points $\{\boldsymbol{x}_i\}$ are then computed by $\boldsymbol{x}_i = \boldsymbol{x}_0^* + s\boldsymbol{v}_i$ where s is an empirical step size whose choice will be discussed in our experiments.

We acquire $\boldsymbol{\alpha}$ at $\boldsymbol{x}_0^*$ by solving the following convex problem [Désidéri, 2012], as suggested by Sener and Koltun [2018]:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \| \sum_{i=1}^{m} \alpha_i \nabla f_i(\boldsymbol{x}_0^*) \|_2 \\ s.t. \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \quad \sum_{i=1}^{m} \alpha_i = 1 \end{aligned} \tag{4.1}$$

Note that the objective can be written as a quadratic form of dimension m. Since m is typically very small, solving it takes little time even for large neural networks.

Given $\boldsymbol{\alpha}$, finding $\{\boldsymbol{v}_i\}$ on the tangent plane at $\boldsymbol{x}_0^*$ can be transformed to finding a solution $(\boldsymbol{v}, \boldsymbol{\beta})$ from Equation (3.1):

$$\boldsymbol{H}(\boldsymbol{x}_0^*)\boldsymbol{v} = \nabla \boldsymbol{f}(\boldsymbol{x}_0^*)^\top \boldsymbol{\beta} \tag{4.2}$$

When n is small, we can apply classic $O(n^3)$ methods like Gram-Schmidt process or QR decomposition. However, directly applying them in deep MTL is difficult for two reasons: first, $\boldsymbol{x}_0^*$ is rarely a true Pareto stationary solution because of the early termination in training to avoid overfitting. Second, and more importantly, the large parameter space makes any $O(n^3)$ method prohibitive.

To address the first issue, we propose a variant to Problem (4.1) to find $\boldsymbol{\alpha}$ as well as a correction vector $\boldsymbol{c}$:

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \boldsymbol{c}} \quad & \|\boldsymbol{c}\|_2 \\ s.t. \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \quad \sum_{i=1}^{m} \alpha_i = 1 \\ & \sum_{i=1}^{m} \alpha_i (\nabla f_i(\boldsymbol{x}_0^*) - \boldsymbol{c}) = \mathbf{0} \end{aligned} \tag{4.3}$$

In other words, we seek the minimal modification to the gradients such that if we use $\nabla f_i(\boldsymbol{x}_0^*) - \boldsymbol{c}$ as if they were the true gradients, $\boldsymbol{x}_0^*$ would be Pareto stationary. It is easy to

show that solving this new optimization problem brings little overhead to the original problem (see supplemental material for the proof):

Proposition 4.2.1. *Let $\boldsymbol{\alpha}^*$ be the solution to Problem (4.1), then the solution to Problem 4.3 is $(\boldsymbol{\alpha}, \boldsymbol{c}) = (\boldsymbol{\alpha}^*, \nabla \boldsymbol{f}(\boldsymbol{x}_0^*)^\top \boldsymbol{\alpha}^*)$.*

To address scalability, we consider the following sparse linear system with unknowns $\boldsymbol{v}$:

$$\boldsymbol{H}(\boldsymbol{x}_0^*)\boldsymbol{v} = (\nabla \boldsymbol{f}(\boldsymbol{x}_0^*)^\top - \boldsymbol{c}\mathbf{1}^\top)\boldsymbol{\beta} \tag{4.4}$$

where $\mathbf{1}$ is an m-dimensional column vector with all elements equal to 1 and $\boldsymbol{\beta} \in \mathbb{R}^m$ is randomly sampled. In other words, we solve a linear system with the right-hand side sampled from the space spanned by $\{\nabla f_i(\boldsymbol{x}_0^*) - \boldsymbol{c}\}$. Solving such a large linear system in MTL requires an efficient matrix solver. We propose to use Krylov subspace iteration methods because they are matrix-free and iterative solvers, allowing us to solve the system without complete Hessians and terminate with intermediate results. In our experiment, we choose to use the minimal residual method (MINRES), a classic Krylov subspace method designed for symmetric indefinite matrices [Choi et al., 2011].

We now discuss MINRES in more detail to better explain why it is the right tool for this problem. The time complexity of MINRES depends on the time spent on each iteration and the number of iterations. The cost of each iteration is dominated by calculating $\boldsymbol{H}\boldsymbol{v}$ for arbitrary $\boldsymbol{v}$, which is in general $O(n^2)$. However, it is well known that Hessian-vector products can be implemented in $O(n)$ time on computational graphs [Pearlmutter, 1994], giving us the first strong reason to use MINRES. Analyzing the number of iterations is hard because it heavily depends on the rarely available eigenvalue distribution. In practice, MINRES is known to converge very fast for systems with fast decay of eigenvalues [Fong and Saunders, 2012]. In our experiments, we specify a maximum number of iterations k. We observed that $k = 50$ was usually sufficient to generate good exploration directions even for networks with millions of parameters. Note that early termination in MINRES still returns meaningful results because the residual error is guaranteed to decrease monotonically with iterations.

To summarize, the efficiency of our exploration algorithm comes from two sources:

Algorithm 1 Efficient Pareto Set Exploration

Input: a random initial neural network $\boldsymbol{x}_0 \in \mathbb{R}^n$
Output: N Pareto stationary networks
$\boldsymbol{x}_0^* \leftarrow$`ParetoOptimize`$(\boldsymbol{x}_0)$
Initialize a queue $q \leftarrow [\boldsymbol{x}_0^*]$
Initialize an empty list to store the output: $output \leftarrow \emptyset$
repeat
  Pop a neural network $\boldsymbol{x}^*$ from q
  for $i = 1$ **to** K **do**
    $\boldsymbol{v}_i \leftarrow$`ParetoExpand`$(\boldsymbol{x}^*)$
    $\boldsymbol{v}_i /= \|\boldsymbol{v}_i\|_2$
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}^* + s\boldsymbol{v}_i$
    $\boldsymbol{x}_i^* \leftarrow$`ParetoOptimize`$(\boldsymbol{x}_i)$
    if No points in $output$ dominates $\boldsymbol{x}_i^*$ **then**
      Append $\boldsymbol{x}_i^*$ to q
      Append $(\boldsymbol{x}_i^*, \boldsymbol{f}(\boldsymbol{x}_i^*), \nabla \boldsymbol{f}(\boldsymbol{x}_i^*), \boldsymbol{x}^*)$ to $output$
    end if
  end for
until The size of $output$ reaches N

exploration on the tangent plane and early termination from a matrix-free, iterative solver. The time cost of getting one tangent direction is $O(kn)$, which scales linearly to the network size.

4.3 The Full Algorithm

We now state the complete algorithm for Pareto set exploration in Algorithm 1. It takes as input a seed network and spawns N Pareto stationary networks in a breadth-first style. Any networks put in the queue are returned by `ParetoOptimize` (Section 4.1) and therefore Pareto stationary by design. When such a network is popped out from the queue, `ParetoExpand` generates K exploration directions (Section 4.2) and spawns K child networks. The algorithm then calls `ParetoOptimize` to refine these networks before appending them to the queue, and terminates after M Pareto stationary networks are collected.

For each output network, we also return the objectives, the gradients, and a reference to its parent. This information is mostly used to construct a continuous linear subspace approximating the local Pareto set, which we will describe in Chapter 5. Another usage is

to remove the sign ambiguity in $\boldsymbol{v}_i$: by definition, both $\boldsymbol{v}_i$ and $-\boldsymbol{v}_i$ are on the tangent plane, and an arbitrary choice can lead to a retraction instead of the desired expansion in the Pareto set. In this case, one can use $\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{f}(\boldsymbol{x}^* + s\boldsymbol{v}_i) - \boldsymbol{f}(\boldsymbol{x}^*) \approx s\nabla\boldsymbol{f}(\boldsymbol{x}^*)\boldsymbol{v}_i$ to predict the changes in the objectives and rule out the undesired direction.

When Algorithm 1 is applied to MTL, it is worth noting that `ParetoOptimize` and `ParetoExpand` rarely return the precise solutions because of stochasticity, early termination, and local minima. As a result, good choices of hyperparameters plays an important role. We discussed in more detail two crucial hyperparameters (k and s) and reported the ablation study in Chapter 6.

Chapter 5

Continuous Parametrization

In this chapter, we describe a post-processing step that builds a continuous approximation to the local Pareto set based on the discrete points $\{\boldsymbol{x}_i^*\}$ returned by Algorithm 1. For each $\boldsymbol{x}_i^*$, we collect its K children $\{\boldsymbol{x}_{i_1}^*, \cdots, \boldsymbol{x}_{i_K}^*\}$ and assign a continuous variable $r_{i\to i_j} \in [0, 1]$ to a vector $\boldsymbol{v}_{i\to i_j} = \boldsymbol{x}_{i_j}^* - \boldsymbol{x}_i^*, j = 1, 2, \cdots, K$. The local Pareto set at $\boldsymbol{x}_i^*$ is then constructed by

$$S(\boldsymbol{x}_i^*) = \{\boldsymbol{x}_i^* + \sum_{i=1}^{K} r_{i\to i_j}\boldsymbol{v}_{i\to i_j} | r_{i\to i_j} \geq 0, \sum_{i=1}^{K} r_{i\to i_j} \leq 1\} \tag{5.1}$$

In other words, $S(\boldsymbol{x}^*)$ is the convex hull of $\boldsymbol{x}_i^*$ and its children $\{\boldsymbol{x}_{i_1}^*, \cdots, \boldsymbol{x}_{i_K}^*\}$. This construction is justified by the fact that a linear combination of tangent vectors is still on the tangent plane. As a special case, when there are only 2 objectives and $K = 1$, $\{\boldsymbol{x}^*\}$ forms a chain, and therefore $S = \cup_i S(\boldsymbol{x}_i^*)$ becomes a piecewise linear set in $\mathbb{R}^n$.

It is possible that two continuous families can collide in the objective space, creating a larger continuous Pareto front. In this case, we create a stitching point in both families and crop solutions dominated by the other family. By repeatedly applying this idea, a single continuous Pareto front covering all families can possibly be created, providing the ultimate solution to continuous traversal in the whole Pareto front. We illustrate this idea on MultiMNIST with our experimental results in Section 6.4.

Since the continuous approximation interpolates different tangent directions, having more directions can enrich the coverage of the continuous set and offer more options to users. It is therefore natural to ask whether the set of tangent directions discovered

in the last chapter could be augmented even further by adding more directions without downgrading the quality of the Pareto front. For the special case of two objectives ($m = 2$), it turns out that we can augment the set of known tangent directions with a *null vector* of the Hessian matrix, as stated in the following proposition:

Proposition 5.0.1. *Assuming $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^2$ is sufficiently smooth. Let $\boldsymbol{x}^*$ be a Pareto optimal point and consider a curve $\boldsymbol{c}_{\boldsymbol{d}}(t) : \mathbb{R} \to \mathbb{R}^2$ defined as $\boldsymbol{c}_{\boldsymbol{d}}(t) = \boldsymbol{f}(\boldsymbol{x}^* + t\boldsymbol{d})$. If $\boldsymbol{x}(t) : (-\epsilon, \epsilon) \to \mathbb{R}^2$ is any smooth curve in Proposition 3.0.2 that satisfies $\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{x}'(0) \neq \boldsymbol{0}$, then for any $\boldsymbol{u} \in \mathbb{R}^n$:*

1) $\boldsymbol{c}_{\boldsymbol{x}'(0)}$ and $\boldsymbol{c}_{\boldsymbol{x}'(0)+\boldsymbol{u}}$ have the same value and tangent direction $(-\alpha_2, \alpha_1)$ at $t = 0$;

2) Furthermore, if $\boldsymbol{u}$ is a null vector of $\boldsymbol{H}(\boldsymbol{x}^*)$, i.e., $\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{u} = \boldsymbol{0}$, then $\boldsymbol{u}$ is not parallel to $\boldsymbol{x}'(0)$, and $\boldsymbol{c}_{\boldsymbol{x}'(0)}(t)$ and $\boldsymbol{c}_{\boldsymbol{x}'(0)+\boldsymbol{u}}(t)$ have the same curvature at $t = 0$.

In this proposition, $\boldsymbol{c}_{\boldsymbol{d}}(t)$ is a parametrized 2D curve: it considers a straight-line trajectory in $\mathbb{R}^n$ that passes $\boldsymbol{x}^*$ in the direction of $\boldsymbol{d}$ and uses $\boldsymbol{f}$ to map this trajectory to the space of $\mathbb{R}^2$, generating a 2D curve. This proposition states that if a tangent direction $\boldsymbol{v}$ is known and if we also have a null vector $\boldsymbol{u}$, then the two curves $\boldsymbol{c}_{\boldsymbol{v}}$ and $\boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}$ are very similar at $\boldsymbol{x}^*$ in the sense that they share the same value, gradients, and curvature. This means that for each tangent direction $\boldsymbol{v}$ found in the previous chapter, $\boldsymbol{v} + \boldsymbol{u}$ can also be used as a backbone direction together with $\boldsymbol{v}$ for continuous parametrization without downgrading the quality of the reconstructed Pareto front.

While this proposition is generally not applicable to real problems due to its need for null vectors, it still has interesting theoretical implications: the fact that $\boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}$ and $\boldsymbol{c}_{\boldsymbol{v}}$ share the same gradients should not be surprising as $\boldsymbol{v} + \boldsymbol{u}$ also satisfies Equation (4.2), but it is less obvious to see that they actually share the same curvature at $\boldsymbol{f}(\boldsymbol{x}^*)$, which we illustrate in Section 6.4 and will prove in our supplemental material. In practice, we observed that neural networks typically have a Hessian matrix with a null space whose dimension is much higher than m. This means a very large set of bases, while not often accessible in real problems, can in theory be used to greatly enrich the Pareto set.

Chapter 6

Experimental Results

6.1 Datasets, Metrics, and Baselines

We applied our method to five datasets in three categories: 1) MultiMNIST [Sabour et al., 2017] and its two variants FashionMNIST [Xiao et al., 2017] and MultiFashionMNIST, which are medium-sized datasets with two classification tasks; 2) UCI Census-Income [Kohavi, 1996], a medium-sized demographic dataset with three binary prediction tasks; 3) UTKFace [Zhang et al., 2017], a large dataset of face images. We used LeNet5 [LeCun et al., 1998] (22,350 parameters) for MultiMNIST and its variants, two-layered multilayer perceptron (158,598 parameters) for UCI Census-Income, and ResNet18 [He et al., 2016] (tens of millions of parameters) for UTKFace. Please refer to our supplemental material for more information about the network architectures, task descriptions, and implementation details in each dataset.

We measure the performance of a method by two metrics: the time cost and the hypervolume [Zitzler and Thiele, 1999]. We measure the time cost by counting the evaluations of objectives, gradients, and Hessian-vector products. The hypervolume metric, explained in Figure 6-1, is a classic MOO metric for measuring the quality of exploration. More concretely, this metric takes as input a set of explored solutions in the objective space and returns a score. Larger hypervolume score indicates a better Pareto front. Using the two metrics, we define that a method is more efficient if, within the same time budget, it generates a Pareto front with a larger hypervolume, or equivalently, if it generates the

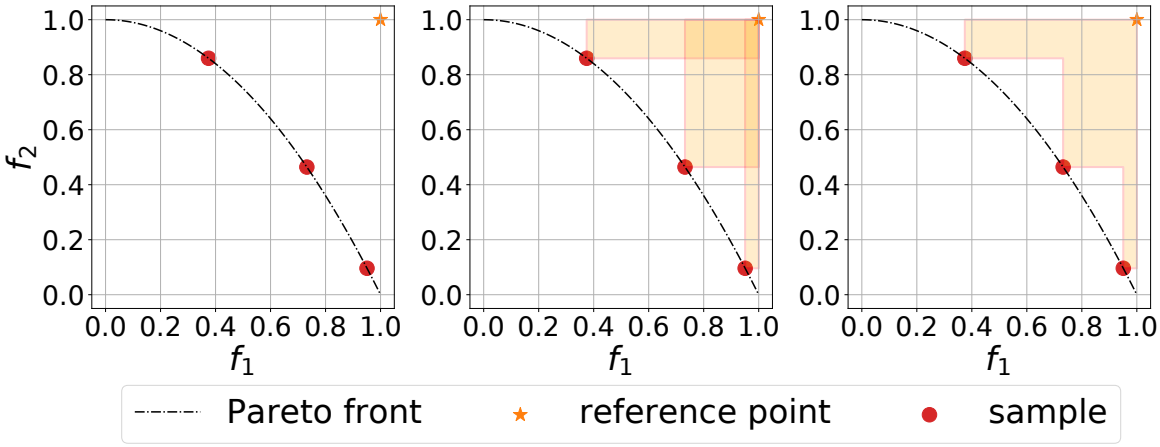

Figure 6-1: Definition of hypervolume. Given a set of sample points (red circles) in $\mathbb{R}^m$, the hypervolume is computed by picking a reference point (orange star), creating axis-aligned rectangles from each point, and calculating the size of their union (orange polygon).

Pareto front with a similar hypervolume but within shorter time. For all figures in this chapter, we use the same random seed whenever possible and report results from more random seeds in the supplemental material.

Our method is not directly comparable to any baselines because no prior work aims to recover a continuous Pareto front in MTL. Instead, we devised two experiments, which we call the sufficiency and necessity tests, to show its effectiveness (Section 6.3). In the sufficiency test, we consider four previous methods: GradNorm [Chen et al., 2018], Uncertainty [Kendall et al., 2018], MGDA [Sener and Koltun, 2018], and ParetoMTL [Lin et al., 2019]. These methods aim at pushing an initial guess to one or a few discrete Pareto optimal solutions. For them, we show that our Pareto expansion procedure is a fast yet powerful complement by comparing the time and hypervolume before and after running it as a post-processing step. We call this experiment the sufficiency test as it demonstrates our method is able to quickly explore Pareto sets and Pareto fronts.

Our necessity test, which focuses on the value of the tangent directions in exploring Pareto fronts, deserves some discussions on its baselines. There is a trivial baseline for Pareto expansion: rerunning an SGD-based method from scratch to optimize a perturbed weight combination of objectives. Since each new run requires full training, our method clearly dominates this baseline (30 times faster on MultiMNIST). Another trivial baseline

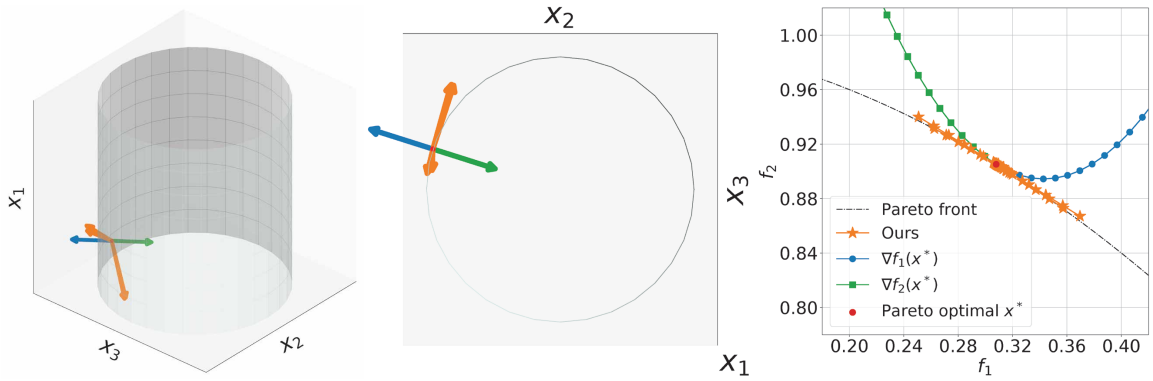

Figure 6-2: Comparisons of different exploration directions at a Pareto optimal solution $\boldsymbol{x}^*$ (red circle). Left: the analytic Pareto set (the cylindrical surface) of ZDT2-variant, the gradients $\nabla f_1(\boldsymbol{x}^*)$ (blue) and $\nabla f_2(\boldsymbol{x}^*)$ (green), and our exploration directions $\{\boldsymbol{v}_i\}$ (orange) predicted by MINRES. Middle: a top-down view to show ours are almost tangent to the Pareto set. Right: plots of $\boldsymbol{f}(\boldsymbol{x}^* + s\boldsymbol{d})$ where $s \in [-0.1, 0.1]$ and $\boldsymbol{d}$ is $\nabla f_1(\boldsymbol{x}^*)$ (blue circles), $\nabla f_2(\boldsymbol{x}^*)$ (green squares), and our directions (orange stars).

is to use a random direction instead of the tangent direction for Pareto expansion. We tested this idea but do not include it in our experiments as its performance is significantly worse than any other methods, which is understandable due to the high dimensionality of neural network parameters: with the increase of dimensionality, the chance of a random guess still staying on the tangent plane decays exponentially. The baseline we considered in this experiment is WeightedSum, which runs SGD from the last Pareto optimal solution but with weights on objectives different from the weights used in training. Specifically, we choose weights from one-hot vectors for each task as well as a vector assigning equal weights to every task. We call this experiment the necessity test as we use this experiment to establish that the choice of expansion strategies is not arbitrary, and tangent directions are indeed the source of efficiency in our method.

6.2 Synthetic Examples

6.2.1 ZDT2-variant

Our first example, ZDT2-variant, was originated from ZDT2 [Zitzler et al., 2000], a classic benchmark problem in multi-objective optimization with $n = 3$ and $m = 2$. Both the

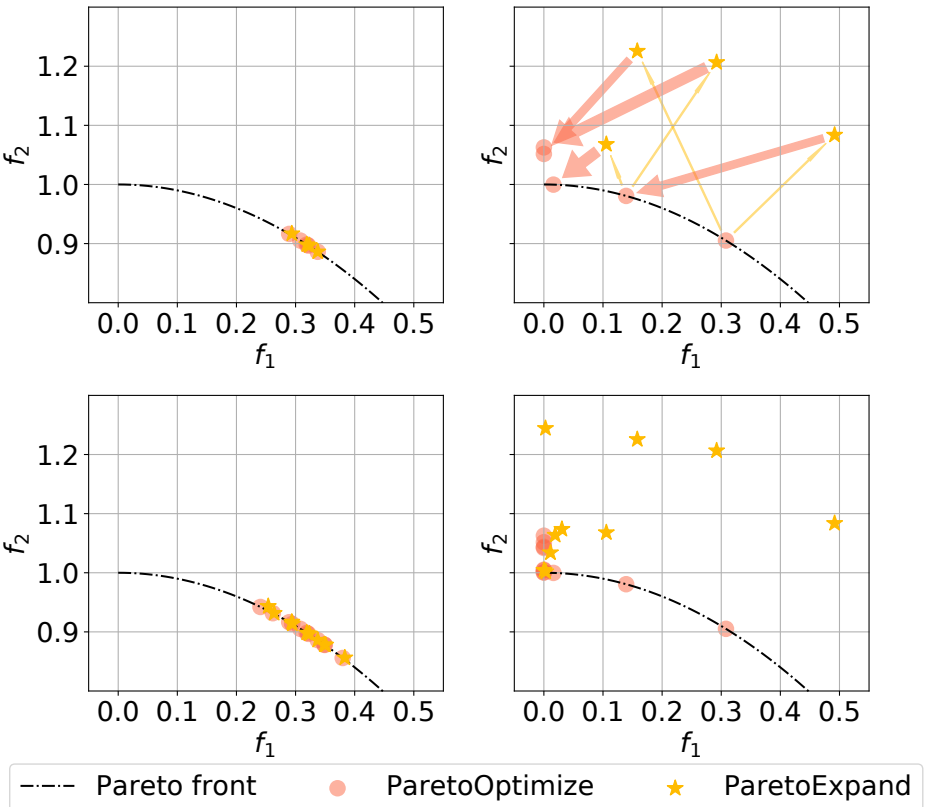

Figure 6-3: Comparisons of two expansion strategies in Algorithm 1. Starting with a given Pareto optimal point (red circle), the algorithm iteratively calls `ParetoExpand` (orange arrows from circles to stars) and `ParetoOptimize` (red arrows from stars to circles), generating a series of explored points (orange stars) and Pareto optimal solutions (red circles). Arrow thickness indicates the time cost of each function. Top row: expansion using our predicted tangent directions (top left) versus using gradients (top right). Bottom row: running both strategies until 10 Pareto optimal points were collected.

Pareto set and the Pareto front of this example can be computed analytically. This makes ZDT2-variant an ideal example for visualizing Proposition 3.0.2 and Algorithm 1. Figure 6-2 compares the gradients to our tangent directions when used to explore the Pareto front. We used MINRES with $k = 1$ to solve 5 tangent directions. It can be seen that our directions are much closer to the Pareto set and tracked the true Pareto front much better than the gradients. We further compare their performances in Algorithm 1 with MGDA [Désidéri, 2012, Sener and Koltun, 2018] as the optimizer in Figure 6-3. This figure shows that the gradients expanded the neighborhood not on the Pareto set but to the dominated interior, resulting in a much more expensive correction step to follow. On the other hand,

expanding with our predicted tangents steadily grew the solution set along the Pareto front.

6.2.2 MultiMNIST Subset

To understand the behavior of our algorithm when neural networks are involved, we picked a subset of 2048 images from MultiMNIST and trained a simplified LeNet [LeCun et al., 1998] with 1500 parameters to minimize two classification errors. We generated an empirical Pareto front by optimizing the weighted sum of the two objectives with varying weights. We then picked a Pareto optimal $\boldsymbol{x}^*$ and visualized trajectories generated by traversing along gradients and the approximated tangents after 10, 20, and 50 iterations of MINRES (Figure 6-4 left). Just as in ZDT2-variant, our approximated tangents tracked the Pareto front much more closely. We then compared using approximated tangents after 50 iterations of MINRES (MINRES-50) to the WeightedSum baseline (Section 6.1) after 50 iterations of SGD. The two methods had roughly the same time budgets, and MINRES-50 outperformed the WeightedSum baseline in that it explored a much wider Pareto front (Figure 6-4 middle). Specifically, its advantage comes from a much larger step size enabled by the approximated tangents (Figure 6-4 right).

6.3 Pareto Expansion

We first conducted the sufficiency test described in Section 6.1 to analyze Pareto expansion, the core of our algorithm. We ran ParetoMTL, the state of the art, on all datasets to generate discrete seeds for Pareto expansion. Moreover, for smaller datasets (MultiMNIST and its variants), we also ran the other baselines for a more thorough analysis. Compared to the time cost of generating discrete solutions (Table 6.1 column 2), our Pareto expansion only used a small fraction of the training time (Table 6.1 column 4) but generated much denser Pareto fronts (Figure 6-5 and Table 6.1 column 5). This experiment, as a natural extension to the synthetic experiments, confirms the efficacy of Pareto expansion on large neural networks and datasets.

The sufficiency test has established that our expansion method has a positive effect on

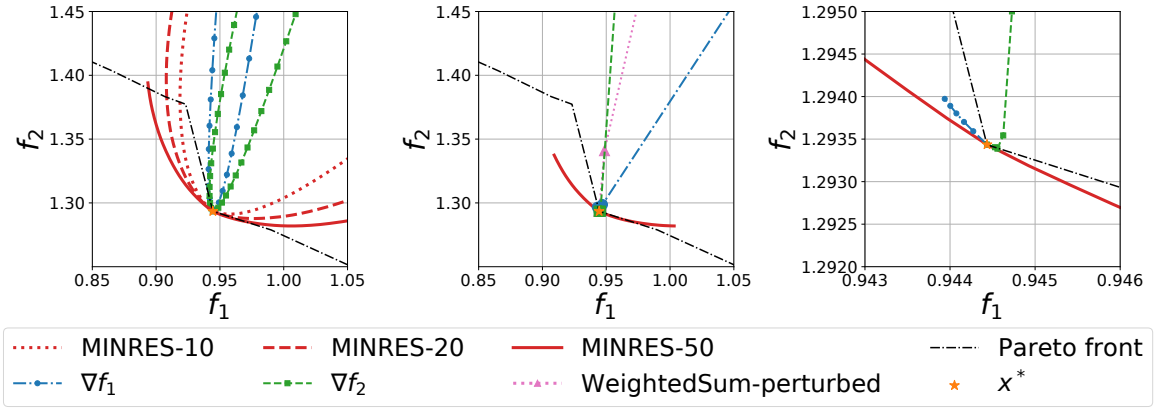

Figure 6-4: Comparisons of expansion strategies on MultiMNIST subset. Left: trajectories of different expansion strategies in the objective space. Curves closer to the Pareto front mean better expansion. Middle: trajectories generated by running SGD to minimize f_1 (blue circles), f_2 (green squares), or a weighted combination (pink triangles) within the same time budget as MINRES-50 (red). Right: a zoom-in version showing that tiny step sizes have to be used by SGD to avoid deviating from the Pareto front too much.

discovering more solutions. However, one can still argue there could be simpler expansion strategies that are as good as ours. It remains to show that the benefit indeed comes from approximated tangent directions. We verified this with the necessity test described in Section 6.1, which directly compared our Pareto expansion to the WeightedSum expansion strategy. Starting with the same seed solution, we gave both methods the same time budget, so the area of their expansions directly reflected their performances. We display the results on MultiMNIST, UCI Census-Income, and UTKFace in Figure 6-6. New solutions were generated after each run of MINRES in our method and after each epoch in WeightedSum. We provide more results in the supplementary material. We see from these experiments that our method discovered solutions that clearly dominated what WeightedSum returned on 4 out of the 5 datasets except UCI Census-Income. From this experiment, we conclude the tangent directions in Pareto expansion are indeed the core reason for the good performance of our algorithm.

The effectiveness of our Pareto expansion method can also be understood by noticing it uses higher-order derivatives than previous work for determining the optimal expansion directions. Consider the three possible methods for the task of expanding the local Pareto

set from a known Pareto optimal solution $\boldsymbol{x}^*$: simply retraining the neural network from scratch with a different initial guess reuses nothing from $\boldsymbol{x}^*$; rerunning SGD from $\boldsymbol{x}^*$ leverages the first-order gradient information at $\boldsymbol{x}^*$; our method exploits both the first-order and the second-order information at $\boldsymbol{x}^*$ and therefore is the most effective among the three.

It is worth mentioning that our Pareto expansion strategy is still a local optimization method, meaning that it inevitably suffers from being trapped in local minima. As a result, there is no theoretical guarantee on the resulting Pareto fronts being globally Pareto optimal. We alleviate this issue by exploring from multiple Pareto optimal solutions returned by previous methods and stitching them together, which we will explain shortly in the next section.

Table 6.1: A summary of the improvement brought by calling Pareto expansion from solutions generated by baselines. TRAIN: the training time used by each baseline, measured by the aggregated number of evaluations of objectives, gradients, and Hessian-vector products; HV: the hypervolume of the solution at the end of training; EXPAND: the time cost of our Pareto expansion; NEW HV: the hypervolume after expansion. Larger hypervolume is better.

MultiMNIST	train	hv	expand	new hv
GradNorm	21150	7.463	4520	7.628
Uncertainty	21150	7.615	4520	7.756
MGDA	21150	7.831	4520	7.896
WeightedSum	70500	8.019	22600	8.034
ParetoMTL	106281	8.025	22600	8.046
UCI	train	hv	expand	new hv
WeightedSum	467400	5.685	165600	5.725
ParetoMTL	934888	5.642	165600	5.675
Face	train	hv	expand	new hv
ParetoMTL	35568	2.257	9920	5.030

6.4 Continuous Parametrization

From discrete solutions returned by Algorithm 1, our continuous parametrization creates low-dimensional, locally smooth Pareto sets. Moreover, we stitch them together when their Pareto fronts collide, forming a larger continuous approximation. We illustrate this idea in Figure 6-7: we ran Algorithm 1 on MultiMNIST with $K = 2$ and $N = 10$ for each Pareto stationary solution $\boldsymbol{x}^*$, generating two chains of solutions favoring small f_1 and small f_2 respectively. As described in Chapter 5, we then constructed a piecewise linear curve parametrized by $t \in [-1, 1]$. By continuously varying t, we explore a diverse set of solutions from favoring small f_1 to small f_2. We highlight this mapping from a single control variable to a wider-range Pareto front because it demonstrates the real advantage of a continuous reconstruction over discrete solutions. As a straightforward application, one can analyze this mapping by running single-variable gradient-descent to pick an optimal solution, which would be impossible if only discrete solutions were provided. We give more results in the supplemental material.

We conclude our discussion on continuous parametrization by demonstrating Proposition 5.0.1 on MultiMNIST subset in Figure 6-7. We precomputed its full null space and revealed over 600 bases. We then expanded the Pareto set at a Pareto optimal $\boldsymbol{x}^*$ with three directions: a tangent direction $\boldsymbol{v}$, $\boldsymbol{v}$ plus a null vector $\boldsymbol{u}$, and $\boldsymbol{v}$ plus a random direction. As expected, expanding with the first two directions led to trajectories sharing the same gradient and curvature at $\boldsymbol{x}^*$, showing that we can enrich the Pareto set by adding null space bases without degrading its quality.

6.5 Ablation Study

Finally, we conducted ablations tests on two crucial hyperparameters in our algorithm: the maximum number of iterations k in MINRES and the step size s that controls the expansion speed. We started with a random Pareto stationary point $\boldsymbol{x}^*$ returned by ParetoMTL, followed by running Algorithm 1 with fixed parameters $K = 1$ and $N = 5$ on MultiMNIST and its two variants. The results are summarized in Figure 6-8, Table 6.2, and

the supplemental material.

To see the influence of k, we fixed $s = 0.1$ and ran experiments with $k \in \{20, 30, 50, 100, 500\}$, whose trajectories are in Figure $6-8$. Between $k = 20$ and 50, the trajectories were pushed towards its lower left, indicating a better approximated Pareto front. This is as expected since more iterations in MINRES were consumed. This trend plateaued between $k = 50$ and 100. Moreover, the tail of the trajectory drifted away after $k = 500$ iterations. We hypothesized that the tangent after 500 iterations explored a new region in $\mathbb{R}^n$ where the constant step size $s = 0.1$ was not proper. Based on these observations, we used $k = 50$ in all experiments.

Table 6.2: Hypervolumes (HV) from the ablation study on hyperparameters k and s. The time cost of experiments is proportional to k and inverse proportional to s. Best results are in bold.

k	20	30	50	100	500
HV	7.731	**7.739**	7.734	7.727	7.669
s	0.05	0.10	0.25	0.50	
HV	**7.741**	7.733	7.728	7.712	

To understand how s affects expansion, we reran the same experiments with a fixed $k = 50$ and chose s from $\{0.05, 0.1, 0.25, 0.5\}$. For each s, we set the number of points to be generated to $1/s$, i.e., the product of the step size and the step number is constant. From Figure 6-8 right, we noticed a conservative s was likely to follow the Pareto front more closely while an aggressive step size quickly led the search to the dominated interior. This is consistent with the fact that our tangents are a first-order approximation to the true Pareto set.

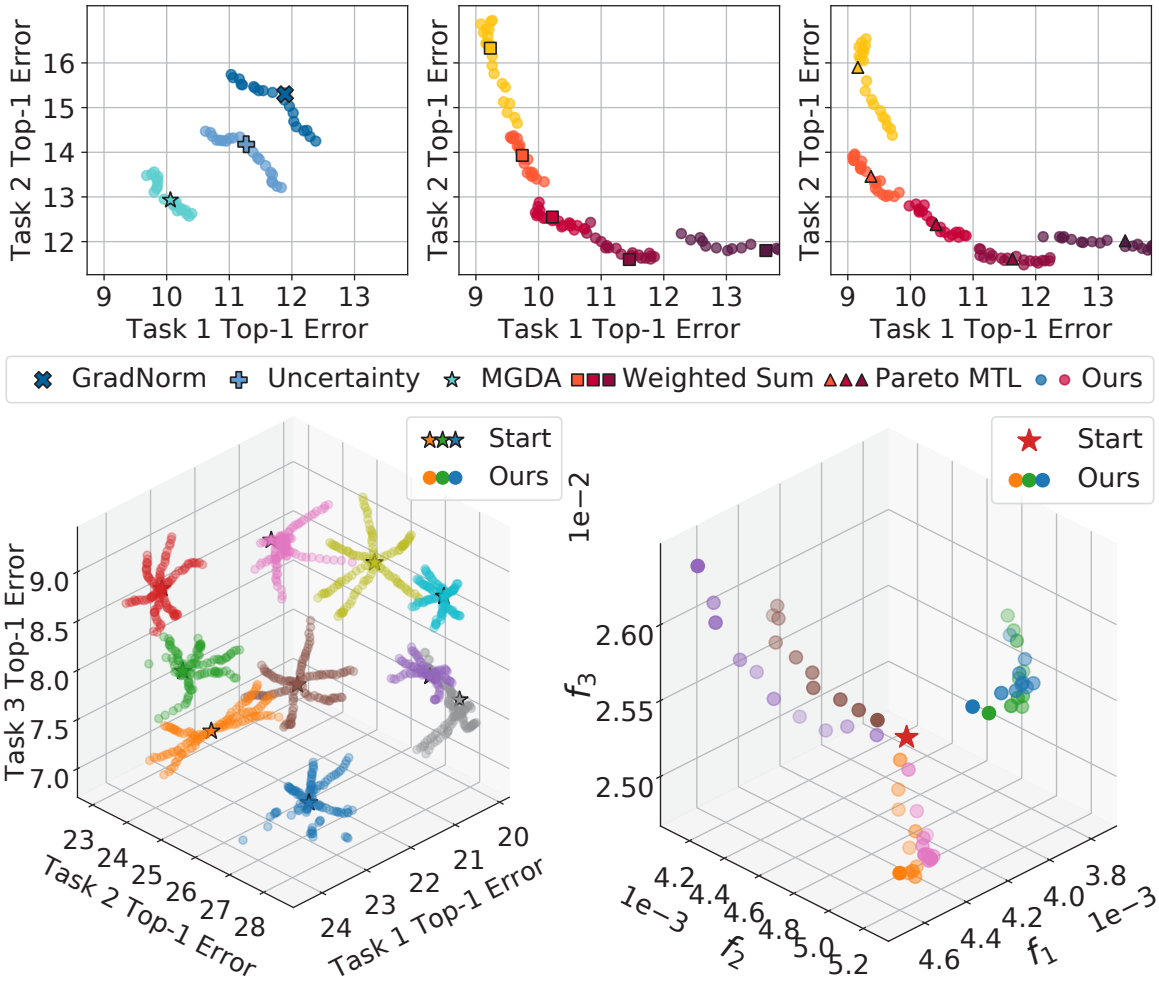

Figure 6-5: Using Pareto expansion to grow dense Pareto fronts (colorful circles) from discrete solutions generated by baselines on MultiMNIST and its variants (top row), UCI Census-Income (bottom left), and UTKFace (bottom right). Points expanded from the same discrete solution have the same color.

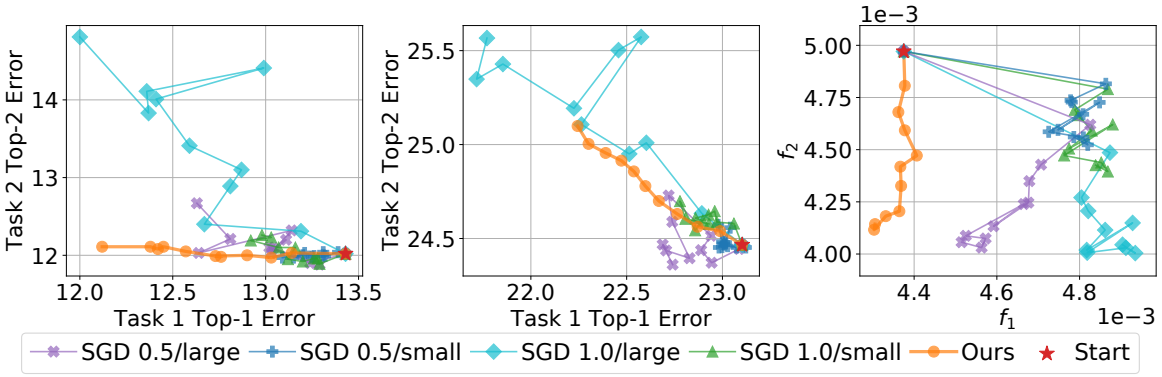

Figure 6-6: Comparisons of two expansion methods (ours and running SGD with a weighted sum) from a given Pareto optimal network. We display results on MultiMNIST (left), UCI Census-Income (middle), and UTKFace (right). In all figures, lower left regions mean better solutions. All SGD methods are labeled with preference on task 1/the type of learning rates (large or small). UCI Census-Income and UTKFace have three objectives and we show results from considering f_1 and f_2 only. Results on FashionMNIST and MultiFashionMNIST and other combinations of objectives in UCI Census-Income and UTKFace can be found in our supplemental material.

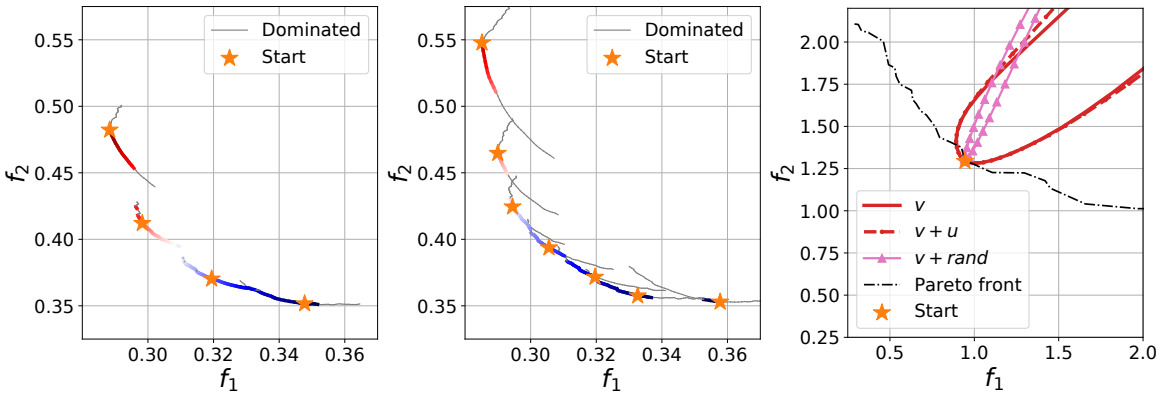

Figure 6-7: Illustrations of the continuous parametrization. Left: Continuous Pareto fronts grown from 4 Pareto optimal solutions (orange stars) on MultiMNIST. Curve colors indicate the value of t from -1 (dark blue) to 1 (dark red) and the thin gray lines indicated dominated solutions. Middle: larger approximations were formed by stitching 10 Pareto fronts. Right: comparisons between expansions with three directions: a tangent $\boldsymbol{v}$ (red and solid), $\boldsymbol{v}$ plus a null vector $\boldsymbol{u}$ (red and dash), and $\boldsymbol{v}$ plus a random direction (pink triangles).

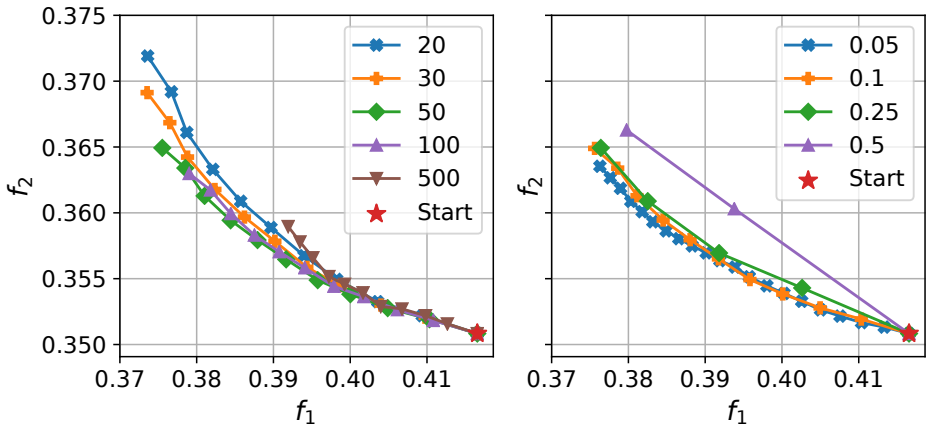

Figure 6-8: Pareto expansion from a Pareto optimal solution (red star) on MultiMNIST by various k and s. Left: expansion with fixed s and $k \in \{20, 30, 50, 100, 500\}$. Lower curves are more dominant. Right: expansion with fixed k and $s \in \{0.05, 0.1, 0.25, 0.5\}$. The number of runs was chosen such that its product with s equals 1.

Chapter 7

Conclusions

We presented a novel, efficient method to construct continuous Pareto sets and fronts in MTL. Our method is originated from second-order analytical results in MOO, and we combined it with matrix-free iterative linear solvers to make it a practical tool for large-scale problems in MTL. We analyzed thoroughly the source of efficiency with demonstrations on synthetic examples. Moreover, experiments showed our method is scalable to modern machine learning datasets and networks with millions of parameters.

While the majority of work in MTL aims to find one near-optimal solution, we believe conflicting objectives in MTL are common and the full answer should be a wide range of candidates with varying trade-offs. Although we are not the first to explore Pareto fronts in MTL or apply second-order techniques to neural networks, we are, to our best knowledge, the first to introduce second-order analysis to Pareto exploration in MTL and the first to propose a continuous reconstruction. We believe our work enables lots of opportunities that would otherwise be impossible if only finite, sparse, and discrete solutions were given, for example, revealing the dimensionality and underlying structure of local Pareto sets, developing interpretable analysis tools for deep MTL networks, and encoding dense Pareto sets and fronts with limited storage.

Appendix A

Proofs

A.1 Proof of Proposition 4.2.1

Proof. The last constraint establishes the connection between $\boldsymbol{c}$ and $\boldsymbol{\alpha}$:

$$\sum_{i=1}^{m} \alpha_i \nabla f_i(\boldsymbol{x}_0^*) = (\sum_{i=1}^{m} \alpha_i)\boldsymbol{c} = \boldsymbol{c} \tag{A.1}$$

The second equality comes from the sum of α_i being 1. Therefore, the optimal solution $\boldsymbol{\alpha}^*$ and $\boldsymbol{c}^*$ must satisfy $\boldsymbol{c}^* = \nabla \boldsymbol{f}(\boldsymbol{x}_0^*)^\top \boldsymbol{\alpha}^*$. Plugging $\boldsymbol{c}^*$ back to Problem (5) reduces it to Problem (3), showing that both problems share the same optimal $\boldsymbol{\alpha}^*$. □

A.2 Proof of Proposition 5.0.1

Proof. For simplicity, we let $\boldsymbol{v} = \boldsymbol{x}'(0)$. We first prove $\boldsymbol{c}_{\boldsymbol{v}}(t) = \boldsymbol{f}(\boldsymbol{x}^* + t\boldsymbol{v})$ and $\boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}(t) = \boldsymbol{f}(\boldsymbol{x}^* + t(\boldsymbol{v}+\boldsymbol{u}))$ have the same value and tangent direction at $t = 0$, i.e., $\boldsymbol{c}_{\boldsymbol{v}}(0) = \boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}(0)$ and $\boldsymbol{c}'_{\boldsymbol{v}}(0) = \boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0)$. The first equality is trivial because both equals $\boldsymbol{f}(\boldsymbol{x}^*)$. To show they have the same tangent direction, note that

$$\begin{aligned} \boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(t) =& (f_1'(\boldsymbol{x}^* + t(\boldsymbol{v}+\boldsymbol{u})), f_2'(\boldsymbol{x}^* + t(\boldsymbol{v}+\boldsymbol{u}))) \\ =& \nabla \boldsymbol{f}(\boldsymbol{x}^* + t(\boldsymbol{v}+\boldsymbol{u}))(\boldsymbol{v}+\boldsymbol{u}) \end{aligned} \tag{A.2}$$

Taking $t = 0$ gives

$$\boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0) = \nabla \boldsymbol{f}(\boldsymbol{x}^*)(\boldsymbol{v} + \boldsymbol{u}) \tag{A.3}$$

Since $\boldsymbol{x}^*$ is Pareto optimal, we have $\boldsymbol{\alpha}^\top \nabla \boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$ (Proposition 3.1). Therefore, the dot product between $\boldsymbol{\alpha}$ and $\boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0)$ is

$$\boldsymbol{\alpha}^\top \boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0) = \boldsymbol{\alpha}^\top \nabla \boldsymbol{f}(\boldsymbol{x}^*)(\boldsymbol{v} + \boldsymbol{u}) = 0 \tag{A.4}$$

This indicates that $\boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0)$ is orthogonal to $\boldsymbol{\alpha}$. Since $m = 2$, we conclude $\boldsymbol{c}'_{\boldsymbol{v}+\boldsymbol{u}}(0)$ is parallel to $(-\alpha_2, \alpha_1)$ no matter how $\boldsymbol{u}$ is chosen.

We now prove the second part of the proposition. First, $\boldsymbol{v}$ and $\boldsymbol{u}$ are not parallel because $\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{u} = \boldsymbol{0}$ and $\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{v} \neq \boldsymbol{0}$. The implication is that adding $\boldsymbol{u}$ to the Pareto set spanned by $\nabla f_1(\boldsymbol{x}^*)$ and $\nabla f_2(\boldsymbol{x}^*)$ indeed augments it by bringing a new dimension.

Second, we show $\boldsymbol{c}_{\boldsymbol{v}}(t)$ and $\boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}(t)$ have the same curvature at $t = 0$. To see this, note that the curvature of $\boldsymbol{c}_{\boldsymbol{d}}(t)$ at $t = 0$ is defined as:

$$\kappa = \frac{f_1' f_2'' - f_2' f_1''}{(f_1'^2 + f_2'^2)^{3/2}} \tag{A.5}$$

where $f_i' = f_i'(\boldsymbol{x}^* + t\boldsymbol{d})|_{t=0}$ and $f_i'' = f_i''(\boldsymbol{x}^* + t\boldsymbol{d})|_{t=0}$, $i = 1, 2$. It is now sufficient to show the denominators and numerators are the same for $\boldsymbol{d} = \boldsymbol{v}$ and $\boldsymbol{d} = \boldsymbol{v} + \boldsymbol{u}$. We prove the following equality to establish the denominators are the same:

$$f_i'(\boldsymbol{x}^* + t\boldsymbol{v})|_{t=0} = f_i'(\boldsymbol{x}^* + t(\boldsymbol{v} + \boldsymbol{u}))|_{t=0}, \quad i = 1, 2 \tag{A.6}$$

To see this, we expand the right-hand side:

$$\begin{aligned} f_i'(\boldsymbol{x}^* + t(\boldsymbol{v} + \boldsymbol{u}))|_{t=0} =& (\boldsymbol{v} + \boldsymbol{u})^\top \nabla f_i(\boldsymbol{x}^*) \\ =& \boldsymbol{v}^\top \nabla f_i(\boldsymbol{x}^*) + \boldsymbol{u}^\top \nabla f_i(\boldsymbol{x}^*) \\ =& f_i'(\boldsymbol{x}^* + t\boldsymbol{v})|_{t=0} + \boldsymbol{u}^\top \nabla f_i(\boldsymbol{x}^*) \end{aligned} \tag{A.7}$$

It remains to show that $\boldsymbol{u}^\top \nabla f_i(\boldsymbol{x}^*) = 0$, or these two vectors are orthogonal. Recall that

$$\alpha_1 \nabla f_1(\boldsymbol{x}^*) + \alpha_2 \nabla f_2(\boldsymbol{x}^*) = \boldsymbol{0} \tag{A.8}$$

where α_i comes from Proposition 3.1. Since $\alpha_1 + \alpha_2 = 1$, at least one of them is nonzero. Without loss of generality, we assume $\alpha_1 \neq 0$, which gives us

$$\nabla f_1(\boldsymbol{x}^*) = -\frac{\alpha_2}{\alpha_1} \nabla f_2(\boldsymbol{x}^*) \tag{A.9}$$

If $\nabla f_2(\boldsymbol{x}^*) = \boldsymbol{0}$, $\nabla f_1(\boldsymbol{x}^*)$ has to be $\boldsymbol{0}$ as well, and $\boldsymbol{u}^\top \nabla f_i(\boldsymbol{x}_i) = 0$ is trivial. Below we assume $\nabla f_2(\boldsymbol{x}^*) \neq \boldsymbol{0}$. Therefore, the space spanned by $\{\nabla f_1(\boldsymbol{x}^*), \nabla f_2(\boldsymbol{x}^*)\}$ is effectively a one-dimensional line in the direction of $\nabla f_2(\boldsymbol{x}^*)$. Now consider applying Proposition 3.2 to $\boldsymbol{v}$:

$$\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{v} = \nabla f_2(\boldsymbol{x}^*)\beta \tag{A.10}$$

where β is some scalar whose exact value is determined by Proposition 3.2. Note that the right-hand side has been simplified due to the fact that $\nabla f_1(\boldsymbol{x}^*)$ and $\nabla f_2(\boldsymbol{x}^*)$ are parallel. Using the fact that $\boldsymbol{u}$ is a null vector of $\boldsymbol{H}(\boldsymbol{x}^*)$ and $\boldsymbol{H}(\boldsymbol{x}^*)$ is a symmetric matrix, we establish the orthogonality between $\boldsymbol{u}$ and $\nabla f_2(\boldsymbol{x}^*)\beta$ as follows:

$$\boldsymbol{u}^\top \nabla f_2(\boldsymbol{x}^*)\beta = \boldsymbol{u}^\top \boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{v} = (\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{u})^\top \boldsymbol{v} = 0 \tag{A.11}$$

Since $\boldsymbol{H}(\boldsymbol{x}^*)\boldsymbol{v} \neq \boldsymbol{0}$, β is nonzero. We then conclude $\boldsymbol{u}^\top \nabla f_2(\boldsymbol{x}^*) = 0$. It follows that $\boldsymbol{u}^\top \nabla f_1(\boldsymbol{x}^*) = -\alpha_2 \boldsymbol{u}^\top \nabla f_2(\boldsymbol{x}^*)/\alpha_1 = 0$.

To show the numerators are the same, we first calculate the second-order derivatives for $\boldsymbol{d} = \boldsymbol{v}$ as follows:

$$\begin{aligned} f_i'(\boldsymbol{x}^* + t\boldsymbol{v}) &= \boldsymbol{v}^\top \nabla f_i(\boldsymbol{x}^* + t\boldsymbol{v}) \\ f_i''(\boldsymbol{x}^* + t\boldsymbol{v}) &= \boldsymbol{v}^\top \nabla^2 f_i(\boldsymbol{x}^* + t\boldsymbol{v})\boldsymbol{v} \\ f_i''(\boldsymbol{x}^* + t\boldsymbol{v})|_{t=0} &= \boldsymbol{v}^\top \nabla^2 f_i(\boldsymbol{x}^*)\boldsymbol{v} \end{aligned} \tag{A.12}$$

As a result, when $\boldsymbol{d} = \boldsymbol{v}$, the numerator is (we simplified the notation by ignoring $\boldsymbol{x}^*$ in ∇f_i

and $\nabla^2 f_i$)

$$
\begin{aligned}
& f_1' f_2'' - f_2' f_1'' \\
=& \boldsymbol{v}^\top \nabla f_1 \boldsymbol{v}^\top \nabla^2 f_2 \boldsymbol{v} - \boldsymbol{v}^\top \nabla f_2 \boldsymbol{v}^\top \nabla^2 f_1 \boldsymbol{v} \\
=& \boldsymbol{v}^\top (\nabla f_1 \boldsymbol{v}^\top \nabla^2 f_2 - \nabla f_2 \boldsymbol{v}^\top \nabla^2 f_1) \boldsymbol{v} \\
=& \boldsymbol{v}^\top (-\frac{\alpha_2}{\alpha_1} \nabla f_2 \boldsymbol{v}^\top \nabla^2 f_2 - \nabla f_2 \boldsymbol{v}^\top \nabla^2 f_1) \boldsymbol{v} \\
=& -\frac{1}{\alpha_1} \boldsymbol{v}^\top \nabla f_2 \boldsymbol{v}^\top (\alpha_2 \nabla^2 f_2 + \alpha_1 \nabla^2 f_1) \boldsymbol{v} \\
=& -\frac{1}{\alpha_1} \boldsymbol{v}^\top \nabla f_2 \boldsymbol{v}^\top \boldsymbol{H} \boldsymbol{v}
\end{aligned}
\tag{A.13}
$$

Replacing $\boldsymbol{v}$ with $\boldsymbol{v} + \boldsymbol{u}$ in the last equation gives us the numerator when $\boldsymbol{d} = \boldsymbol{v} + \boldsymbol{u}$:

$$
\begin{aligned}
& f_1' f_2'' - f_2' f_1'' \\
=& -\frac{1}{\alpha_1} (\boldsymbol{v} + \boldsymbol{u})^\top \nabla f_2 (\boldsymbol{v} + \boldsymbol{u})^\top \boldsymbol{H} (\boldsymbol{v} + \boldsymbol{u}) \\
=& -\frac{1}{\alpha_1} \boldsymbol{v}^\top \nabla f_2 (\boldsymbol{v} + \boldsymbol{u})^\top \boldsymbol{H} (\boldsymbol{v} + \boldsymbol{u}) \\
=& -\frac{1}{\alpha_1} \boldsymbol{v}^\top \nabla f_2 \boldsymbol{v}^\top \boldsymbol{H} (\boldsymbol{v} + \boldsymbol{u}) \\
=& -\frac{1}{\alpha_1} \boldsymbol{v}^\top \nabla f_2 \boldsymbol{v}^\top \boldsymbol{H} \boldsymbol{v}
\end{aligned}
\tag{A.14}
$$

where the second equality was derived with the fact $\boldsymbol{u}^\top \nabla f_2 = 0$ and the last two equalities used $\boldsymbol{H}\boldsymbol{u} = \boldsymbol{0}$. This shows that the two curves have the same numerators at $t = 0$. Putting it together, we have proven $\boldsymbol{c}_{\boldsymbol{v}}(t)$ and $\boldsymbol{c}_{\boldsymbol{v}+\boldsymbol{u}}(t)$ have the same curvature at $t = 0$ when $\boldsymbol{u}$ is a null vector of $\boldsymbol{H}$. □

A.3 Experimental Setup

A.3.1 ZDT2-variant

This example has an analytic $\boldsymbol{f}(\boldsymbol{x}) : (x_1, x_2, x_3) \in \mathbb{R}^3 \to (f_1, f_2) \in \mathbb{R}^2$, defined as follows:

$$\begin{aligned} y_1 =& \frac{\sin(x_1 + x_2^2 + x_3^2) + 1}{2} \\ y_2 =& \frac{\cos(x_2^2 + x_3^2) + 1}{2} \\ y_3 =& y_2 \\ g =& 1 + \frac{9}{2}(y_2 + y_3) \\ f_1(x_1, x_2, x_3) =& y_1 \\ f_2(x_1, x_2, x_3) =& g - \frac{y_1^2}{g} \end{aligned} \tag{A.15}$$

The Pareto front is given by

$$f_2 = 1 - f_1^2 \quad f_1 \in [0, 1] \tag{A.16}$$

and the analytic Pareto set is

$$x_2^2 + x_3^2 = (2k + 1)\pi, \quad k = 0, 1, 2, \cdots \tag{A.17}$$

which is a family of concentric cylinders. In the paper, we analyzed the innermost Pareto set $x_2^2 + x_3^2 = \pi$. The rightmost figure in Figure 2 in the paper was generated by plotting $\boldsymbol{f}(\boldsymbol{x}^* + s\boldsymbol{d}), s \in [-0.1, 0.1]$ with $\boldsymbol{d}$ being a unit vector of $\nabla f_1(\boldsymbol{x}^*)$, $\nabla f_2(\boldsymbol{x}^*)$, and the approximated tangent directions after 2 iteration of MINRES respectively.

The experiments in Figure 3 of the main paper were set up as follows: starting with a randomly chosen Pareto optimal $\boldsymbol{x}^*$, we spawned a new $\boldsymbol{x}$ by computing $\boldsymbol{x} = \boldsymbol{x}^* + 0.1\boldsymbol{d}$ where $\boldsymbol{d}$ is a unit vector calculated from two methods: 1) running MINRES for 2 iteration to get the approximated tangent direction; 2) perturbing α at $\boldsymbol{x}^*$ to get α' and letting $\boldsymbol{d} = \alpha_1'(\boldsymbol{x}^*) + \alpha_2'\nabla f_2(\boldsymbol{x}^*)$. The second method is the WeightedSum baseline introduced in the main paper and can be interpreted as exploring by running one iteration of gradient-descent to minimize $\alpha_1' f_1 + \alpha_2' f_2$. We then used MGDA [Désidéri, 2012] plus line search to

push new $\boldsymbol{x}$ back to the Pareto front. The step size in our line search was initially 1 and decayed by 0.9 exponentially.

A.3.2 MultiMNIST Subset

We first generated the full MultiMNIST dataset (see Section A.3.3) and picked a subset of 2048 images, downsampled from 28×28 to 14×14, as our MultiMNIST Subset example. The two objectives are the cross entropy losses of classifying the top-left and bottom-right digits evaluated on all 2048 images. Regarding the classifier, we used a modified LeNet5 [LeCun et al., 1998] network, which has 1500 parameters. Our modified network starts with a convolutional layer with 10 channels, a 5×5 kernel, and a stride of 2 pixels, followed by a 2×2 max pooling layer. Next, the results are fed into a fully connected layer of size 20×10 and then sent to two fully connected layers, one for each task. We use ReLU as the nonlinear function in the network. Essentially, this synthetic example attempts to use a small network to overfit 2048 images. To generate the Pareto front, we ran BFGS [Nocedal and Wright, 2006] to optimize $w_1 f_1 + w_2 f_2$ with $w_1 = 0, 0.01, 0.02, \cdots, 1$ from the same random initial guess, which generated a list of 101 solutions $\boldsymbol{x}_0^*, \boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_{101}^*$. We then linearly interpolated $\boldsymbol{f}(\boldsymbol{x}_i^*), i = 0, 1, 2, \cdots, 100$ and treat the resulting piecewise linear spline as the (empirical) Pareto front.

The experiment in Figure 4 of the main paper was conducted as follows: starting with a randomly chosen $\boldsymbol{x}_i^*$, we plotted $\boldsymbol{f}(\boldsymbol{x}_i^* + s\boldsymbol{d}), s \in [-0.5, 0.5]$ where $\boldsymbol{d}$ is a unit vector of the approximated tangent direction. We got the tangent direction by running 50 MINRES iterations to solve Equation (6) of the main paper with $\boldsymbol{\beta}$ sampled from a standard normal distribution. In particular, we found gradient correction (Equation (5) of the main paper) useful in this example. We then ran 50 iterations of gradient-descent (GD) to minimize f_1, f_2, and $w_1 f_1 + w_2 f_2$ respectively. Here w_1 and w_2 are perturbed from the corresponding $\boldsymbol{\alpha}$ vector at $\boldsymbol{x}_i^*$. This shows how well gradient-descent can explore the Pareto front within the time budget of 50 times of back-propagation. We used $1/\sqrt{t+1}$ where t is the iteration index to decay the learning rate in GD from 0.005.

A.3.3 MultiMNIST and Its Variants

Dataset and Task Description We followed Sabour et al. [2017] to generate MultiMNIST, FashionMNIST, and MultiFashionMNIST. We first created 36×36 images by placing two 28×28 images from MNIST or FashionMNIST [Xiao et al., 2017] in the upper-left and lower-right corner with a random shift of up to 2 pixels in each direction. The synthesized images were then resized to 28×28 and normalized with a mean of 0.1307 and a standard deviation of 0.3081. No data augmentation was used for training or testing. Following ParetoMTL [Lin et al., 2019], we built MultiMNIST from MNIST, MultiFashion from FashionMNIST, and MultiFashionMNIST from both (Figure A-1). Each dataset has 60,000 training images and 10,000 test images. The objectives are the cross entropy losses of classifying the upper-left and lower right items in the image.

Network Architecture The backbone network is a modified LeNet [LeCun et al., 1998]. Our network starts from two convolutional layers with a 5×5 kernel and a stride of 1 pixel. The two layers have 10 and 20 channels respectively. A fully connected layer of 50 channels appends the convolutional layers, which is then followed by two 10-channel fully connected layers, one for each task. We add a 2×2 max pooling layer right after each convolutional layer and use ReLU as the nonlinear function. The network contains 22,350 trainable parameters.

Training We trained all baselines for 30 epochs of SGD. We used 256 as our mini-batch size and set the momentum to 0.9. The learning rate started from 0.01 and decayed with a cosine annealing scheduler.

For our method, we used 50 iterations of MINRES to solve Equation (4) of the main paper with the right-hand side sampled between $\nabla f_1(\boldsymbol{x}_0^*)$ and $\nabla f_2(\boldsymbol{x}_0^*)$. We did not correct the gradients (Equation (5) of the main paper) in this example as we found using the original gradients were more effective.

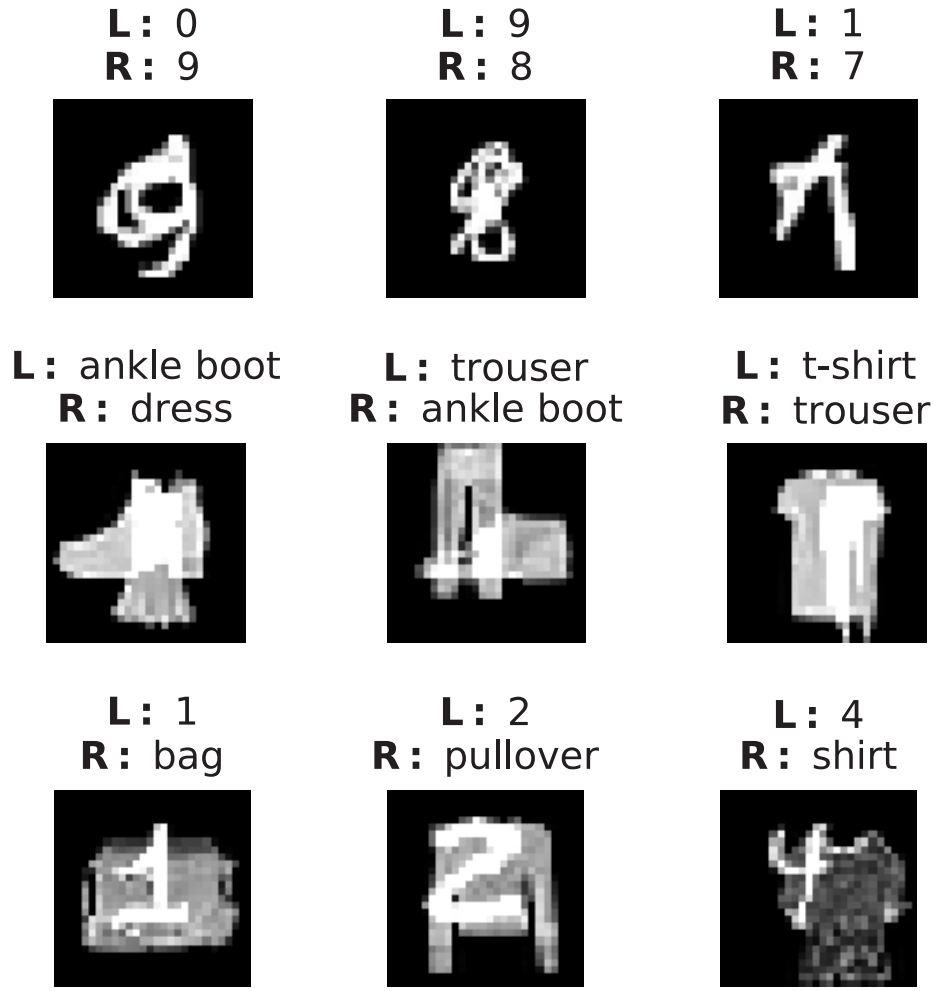

Figure A-1: Sample images from MultiMNIST (top), MultiFashion (middle), and MultiFashionMNIST (bottom). Above each image are the labels of the upper-left (L) and lower-right (R) items.

A.3.4 UCI Census-Income

Dataset and Task Description UCI Census-Income [Kohavi, 1996] is a demographic dataset consisting of information about around 300,000 adults in the United States. Lin et al. [2019], one of the state-of-the-art baselines, proposed three tasks on this dataset: 1) whether the person's income exceeds \$50K/year, 2) whether the person's education level is at least college, and 3) whether the person is never married. We did not use their first task because the results are highly imbalanced (93.8% of the dataset would have the same label). Instead, our first task is whether the person's age is greater than or equal to 40. The tasks

were evaluated by cross-entropy losses. We converted all categorical data into one-hot vectors and concatenated them along with continuous data into a 487 dimensional feature vector. After removing invalid data, training and test sets have 199,523 and 99,762 samples respectively.

Network Architecture We used a multilayer perceptron (MLP) with two hidden layers of 256 and 128 channels as the shared feature extractor and a fully connected layer as the classifier for each task. We chose ReLU as the nonlinear activation function. This network contains 158,598 trainable parameters in total.

Training We trained all baselines with 30 epochs of SGD and used a mini-batch of size 256 and a momentum of 0.9. The learning rate started from 0.001 and decayed with a cosine annealing scheduler.

For our method, we used 100 iterations of MINRES to solve the tangent direction and gradient correction was not used. The right-hand side of Equation (4) was sampled as follows: for each task f_i, we flipped a coin to determine a binary label $l_i \in \{0, 1\}$. The right-hand side was then the sum of all $\nabla f_i(\boldsymbol{x}_0^*)$ with $l_i = 1$. We skipped a sample if $l_1 = l_2 = l_3$.

A.3.5 UTKFace

Dataset and Task Description UTKFace [Zhang et al., 2017] is a dataset of over 20,000 face images. Each image has 200×200 pixels and 3 color channels. We considered three tasks on this dataset: 1) predicting the age of each face, 2) classifying the gender, and 3) classifying the race. We used the Huber loss with $\delta = 1$ for task 1 and cross entropy losses for task 2 and 3. We preprocessed the age information by normalizing it to the standard normal distribution. Moreover, each image was resized to 64×64 and each pixel was further normalized with mean values and standard deviations from ImageNet [Deng et al., 2009]. We created the training and test set with an 80/20 split of UTKFace. After data cleaning, our training and test sets have 18,964 and 4,741 images respectively.

Network Architecture Our network was built upon a standard ResNet18 [He et al., 2016] by appending a fully connected layer to it for each task. Batch normalization [Ioffe and Szegedy, 2015] was used with a momentum of 0.1. The network contains 11,180,616 trainable parameters.

Training We ran all baseline experiments with 30 epochs of SGD and a mini-batch size 256. We used a weight decay of 1e−5 and a momentum of 0.9. The learning rate started from 0.01 and decayed with a cosine annealing scheduler. Batch normalization was frozen when we were expanding the Pareto front from a Pareto optimal network.

For our method, the training process was the same as in UCI Census-Income except that we used 50 MINRES iterations instead of 100.

A.4 Synthetic Examples

A.4.1 ZDT2-variant

Here we present more experimental results on ZDT2-variant from multiple random seeds. Figure A-2 left shows 40 random Pareto optimal solutions and expansions from them with tangent directions and gradients. This essentially repeated Figure 2 in the main paper 40 times. It can be seen that tangent directions behave consistently better than gradients in terms of exploring the Pareto front across all random samples. Figure A-2 middle and right implemented Algorithm 1 from 10 random seeds and collected 10 Pareto optimal solutions each time. This experiments duplicated Figure 3 in the main paper with 10 different random seeds, and they show that using tangent directions allows us to slide on the Pareto front closely as expected. It is worth noting that part of the solutions optimized by MGDA clustered along the line segment $f_1 = 0, f_2 \geq 1$. Due to the design of ZDT2-variant, solutions like these are not Pareto optimal but Pareto stationary, and thus MGDA could not make further progress from them. Moreover, we report the time cost in Table A.1, which confirms that the time saving mostly comes from the near-optimal inputs to `ParetoOptimize`.

Table A.1: The number of evaluations of objectives ($\boldsymbol{f}$), gradients ($\nabla \boldsymbol{f}$), and Hessian-vector products ($\nabla^2 \boldsymbol{f}$) in Figure A-2 middle (ours) and right (WeightedSum). The abbreviation EXP and OPT means the time cost from `ParetoExpand` and `ParetoOptimize` respectively.

Method	$\#\boldsymbol{f}$	$\#\nabla \boldsymbol{f}$	$\#\nabla^2 \boldsymbol{f}$
Ours (Exp)	0	50	300
Ours (Opt)	100	100	0
WeightedSum (Exp)	0	50	0
WeightedSum (Opt)	17931	1818	0

A.4.2 MultiMNIST Subset

We now present more results on MultiMNIST Subset in Figure A-3 as we extended the experiments in Figure 4 of the main paper. We sampled 26 Pareto optimal points $\{\boldsymbol{x}_i^*\}$ evenly

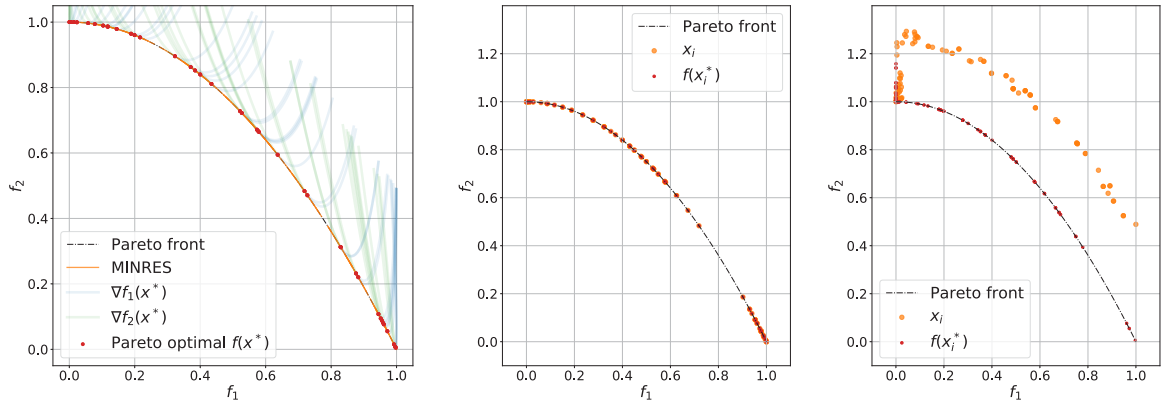

Figure A-2: More experimental results on ZDT2-variant. Left: plotting $\boldsymbol{f}(\boldsymbol{x}^* + s\boldsymbol{d}), s \in [-0.1, 0.1]$ with 40 random $\boldsymbol{x}^*$ (red) and $\boldsymbol{d}$ being tangent directions (orange) and gradients (blue and green); Middle and right: running Algorithm 1 with MGDA as the optimizer and comparing two expansion strategies: moving along the tangent directions from MINRES after 2 iterations (middle) and walking along the perturbed weighted sum of gradients (right). The experiments were repeated on 10 random seeds, and all explored points on the Pareto front are colored in red. Results returned by `ParetoExpand` are colored in orange.

distributed on the empirical Pareto front. For each of them, we depicted $\boldsymbol{f}(\boldsymbol{x}_i^* + s\boldsymbol{v}), s \in [-0.5, 0.5]$ (red) where $\boldsymbol{v}$ is returned by running MINRES after 50 iterations. Furthermore, we minimized f_1 and f_2 from $\boldsymbol{x}_i^*$ with 50 iterations of GD and BFGS and plotted the trajectory of intermediate solutions at each iteration (blue and green). It can be seen from Figure A-3 that the tangent directions expanded the empirical Pareto front more accurately and clearly dominated the region explored by GD or BFGS.

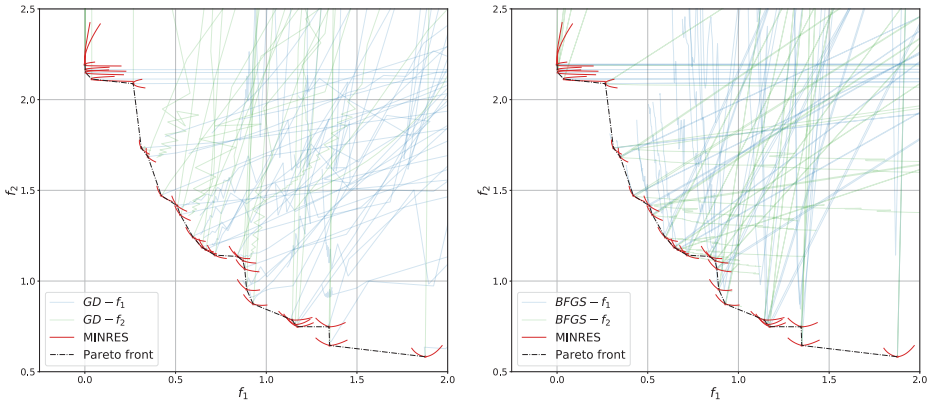

Figure A-3: Comparisons of three Pareto expansion strategies on MultiMNIST Subset. The empirical Pareto front is plotted in black with Pareto optimal solutions $\boldsymbol{x}^*$ drawn as 26 red dots. The red curves in both figures show $\boldsymbol{f}(\boldsymbol{x}^* + s\boldsymbol{v}), s \in [-0.5, 0.5]$ where $\boldsymbol{v}$ is the tangent from 50 iterations of MINRES. We used 50 iterations of GD (left) and BFGS (right) to minimize f_1 and f_2 from $\boldsymbol{x}^*$, with intermediate solutions shown in blue and green respectively.

Appendix B

Pareto Expansion

In this section, we repeated the two experiments described in Section 6.3 of the main paper on all five datasets with more random seeds. Essentially, the results in this sections extend Figure 5 and Figure 6 of the main paper. To recap, the first experiment uses our Pareto expansion method to grow dense Pareto fronts from known Pareto optimal solutions, and the second experiment compares our method to the WeightedSum baseline to establish the necessity of using tangent directions. For simplicity, we will call them sufficiency and necessity experiments respectively.

B.1 MultiMNIST and Its Variants

Figure B-1 displays the results of our sufficiency experiment on MultiMNIST and its two variants. We grew Pareto fronts from 5 seeds optimized by two baselines: WeightedSum and ParetoMTL. This figure is an extension to Figure 5 in the main paper. We stress again that growing such dense Pareto fronts only took a fraction of the training time spent on getting one Pareto optimal solution from baselines.

Similarly, we reran the necessity experiment and summarized the results in Figure B-2. For each dataset, we repeated the experiment on 5 different Pareto optimal solutions found by ParetoMTL (squares and triangles in Figure B-1). We second that in all figures, lower left indicates better performances, and the region expanded by our method (orange lines) dominates SGD with various learning rates and weight combinations.

B.2 UCI Census-Income

Figure B-3 displays the result of the sufficiency experiment. Note that this dataset has three objectives. We repeated this experiment with 5 random seeds. For each random seed, we ran SGD 10 times with different weight combinations to generate 10 Pareto optimal solutions that are evenly distributed on the Pareto front, which is roughly a concave surface viewed from the camera position. Points with smaller values (farther away from the camera in the figure) are preferred.

Furthermore, Figure B-4 summarizes the necessity experiment on this dataset. We first ran SGD to minimize a combination of three objectives with a preference vector $(1/3, 1/3, 1/3)$ to obtain a Pareto optimal solution $\boldsymbol{x}^*$. We then considered three pairs of losses (f_i, f_j) where $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$. For each (f_i, f_j) pair, we ran MINRES from $\boldsymbol{x}^*$ and compared it to SGD baselines with different weight combinations and learning rates. For all figures, lower left region is Pareto optimal. In most cases, Pareto fronts revealed by our method dominate SGD results.

B.3 UTKFace

The sufficiency experiment is reported in Figure B-5. We randomly picked 5 initial networks and ran SGD to minimize a combination of three objectives with a weight vector $(1/3, 1/3, 1/3)$. We then expanded the local Pareto front with our method by running 50 MINRES iterations 6 times, generating 6 trajectories from the Pareto optimal solution. The choice of 6 comes from the fact that three objectives have 8 possible combinations of binary labels (see Section A.3.4) and we skipped combinations of all-zero or all-one labels in the sufficiency experiment.

We present the necessity experiment in Figure B-6. Since both UTKFace and UCI Census-Income have three objectives, we inherited the same experiment setup from UCI Census-Income. Methods that can explore towards the lower left region are preferred. Among the 15 experiments and 4 SGD baselines reported in Figure B-6, we summarize that our method almost dominated all SGD baselines in 5 experiments (row 1 column 4, row 2

column 3, and the rightmost column), was clearly outperformed by one SGD baseline in our experiment (purple in row 2 column 4), and performed comparably in the remaining experiments.

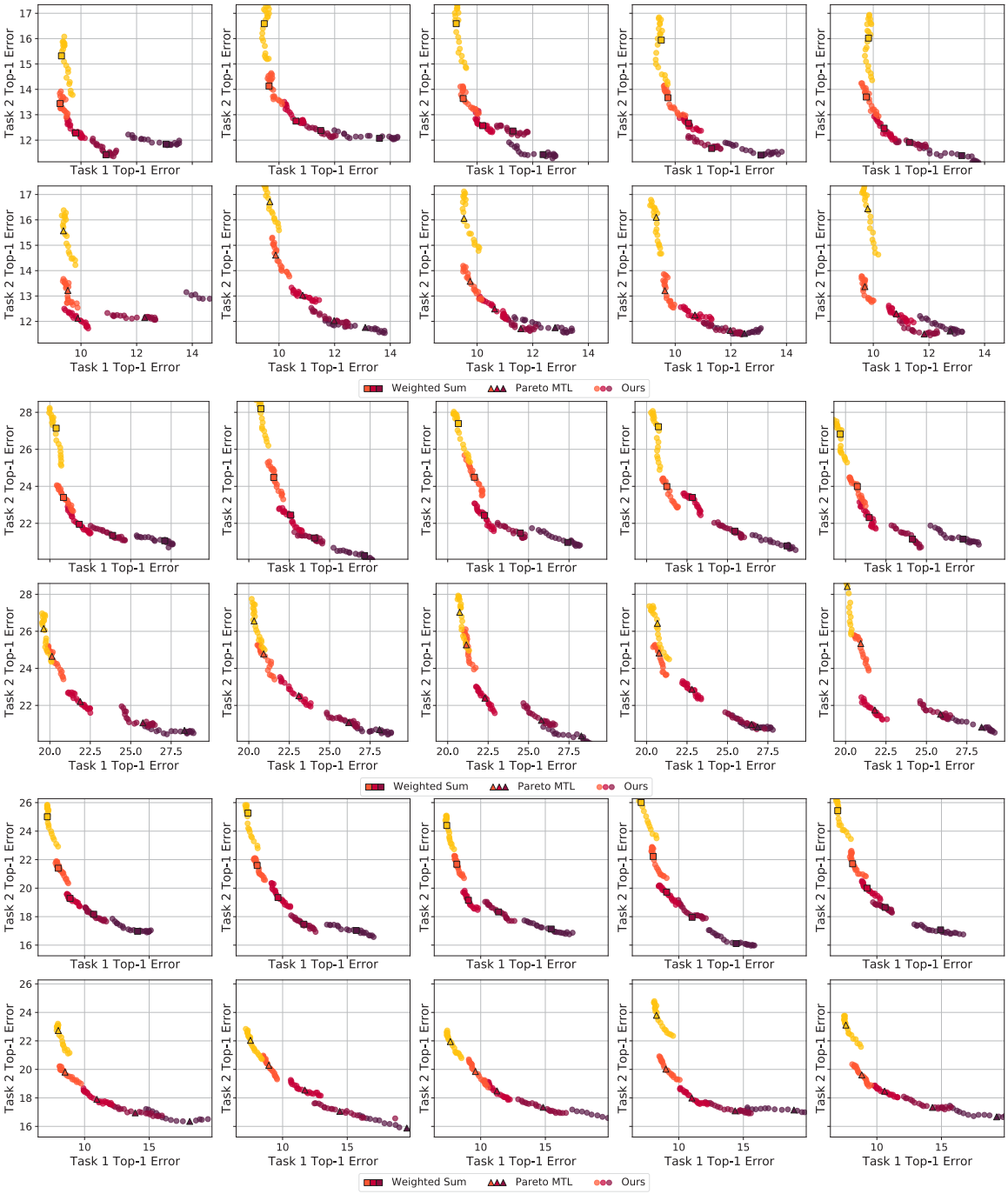

Figure B-1: Expanding the Pareto front with our method on MultiMNIST (top two rows), MultiFashion (middle two rows), and MultiFashionMNIST (bottom two rows) from 5 Pareto optimal seeds generated by the WeightedSum baseline (squares) and ParetoMTL (triangles) with different initial random guesses (left to right). Our method grew dense Pareto fronts (colorful circles) from these 5 seeds.

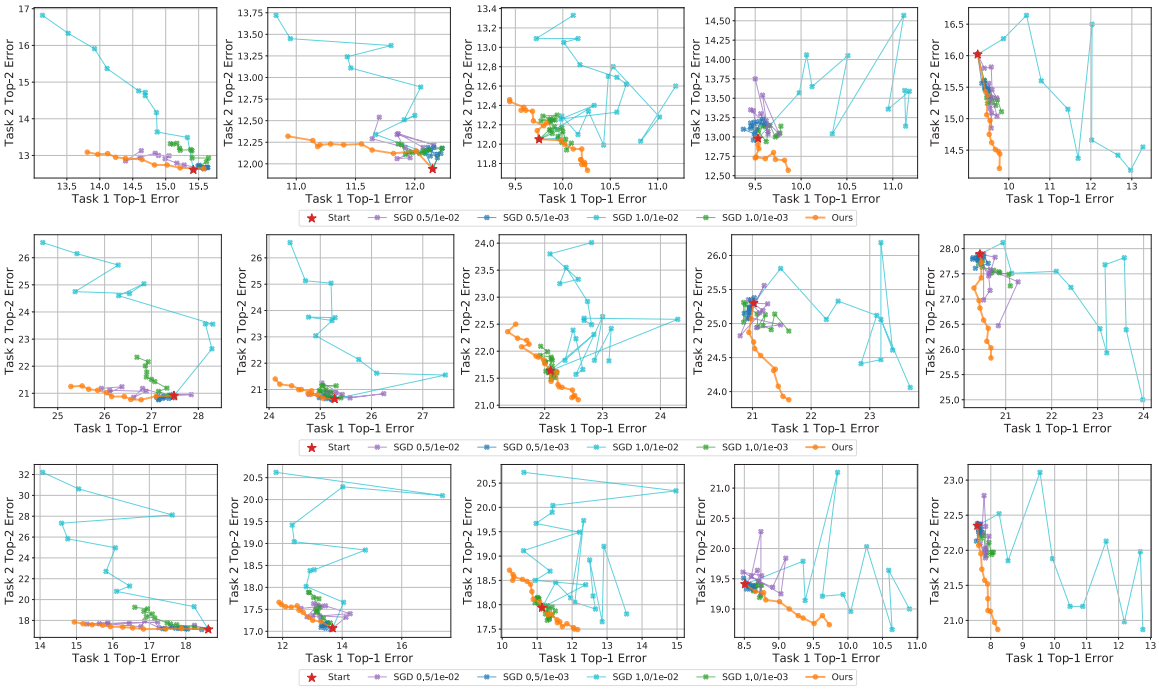

Figure B-2: Comparisons of two expansion methods (ours and running SGD with a weighted sum) from a given Pareto optimal solution (red star). Top to bottom: results on MultiMNIST, MultiFashion, and MultiFashionMNIST. Left to right: we started the experiments from five different Pareto optimal solutions found by ParetoMTL. In all figures, lower left means better solutions. All SGD methods are labeled with preference on task 1/learning rate.

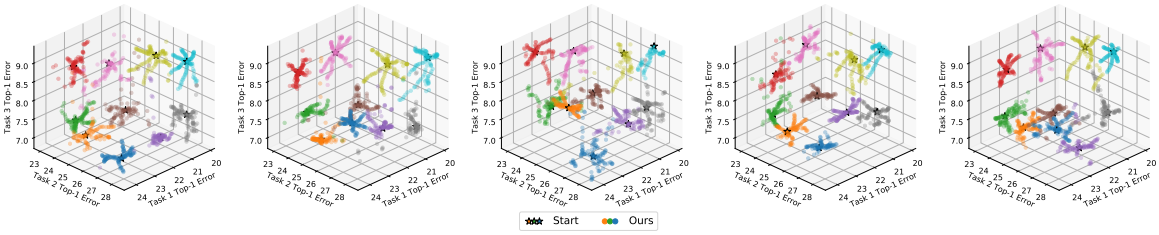

Figure B-3: Expanding the Pareto front with our method on UCI Census-Income from 10 Pareto optimal seeds generated by the WeightedSum baseline. Five random initial guesses (left to right) were used to generate these results.

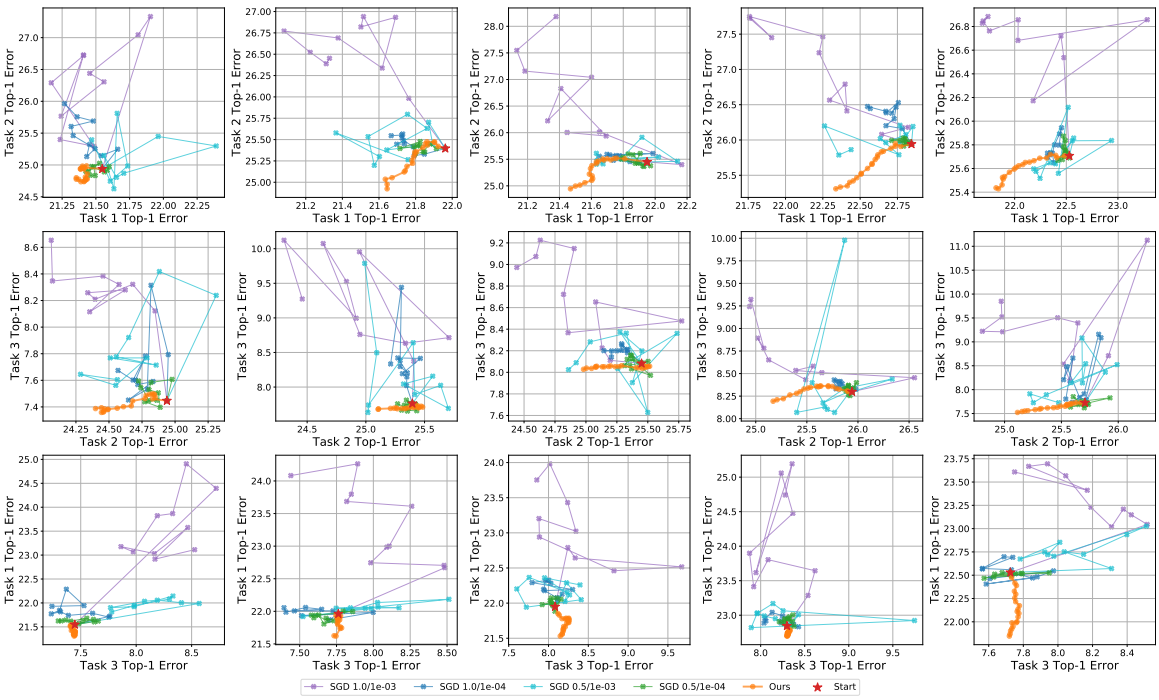

Figure B-4: Comparisons of two expansion methods (ours and running SGD with a weighted sum) from a given Pareto optimal solution (red star) on UCI Census-Income. Left to right: we started the experiments from five different Pareto optimal solutions found by SGD with weights $(1/3, 1/3, 1/3)$. In all figures, lower left means better solutions. All SGD methods are labeled with preference on task of the horizontal axis/learning rate.

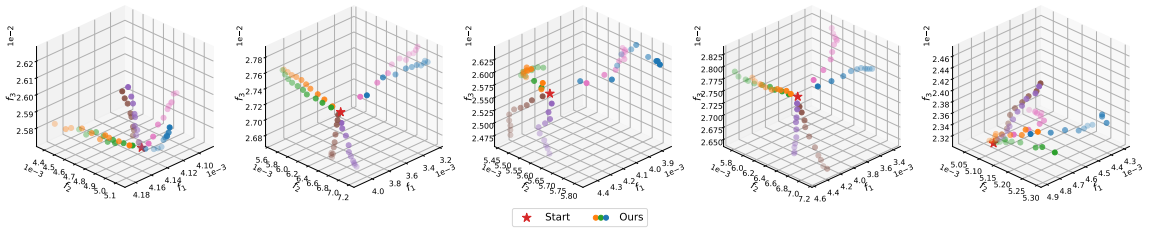

Figure B-5: Expanding the Pareto front with our method on UTKFace from five random initializations. We grew our solutions from a seed (red star) to 6 directions (colorful circles) computed by 50 MINRES iterations.

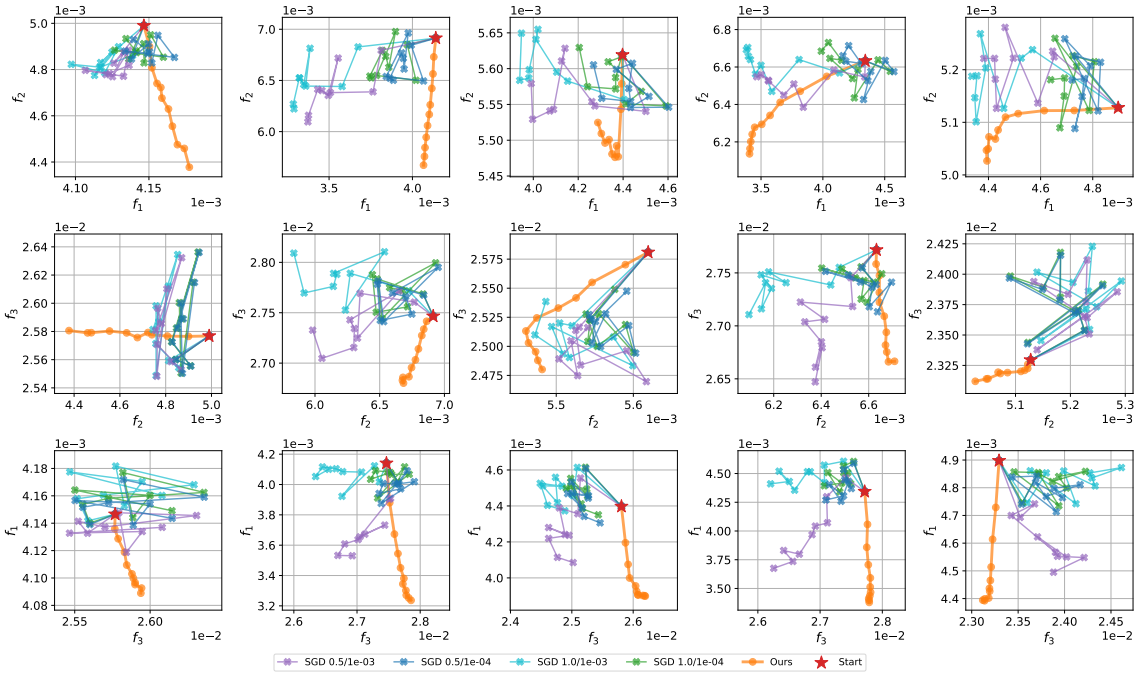

Figure B-6: Comparisons of two expansion methods (ours and running SGD with a weighted sum) from a given Pareto optimal solution (red star) on UTKFace. Left to right: we started the experiments from five different Pareto optimal solutions found by SGD with weights $(1/3, 1/3, 1/3)$. In all figures, lower left means better solutions. All SGD methods are labeled with preference on task of the horizontal axis/learning rate.

Appendix C

Continuous Parametrization

In this section, we present results that extend Figure 8 of the main paper. The main idea we want to demonstrate is twofold: locally, we show that Pareto optimal solutions found by our method can be used as backbones to grow a continuous, approximated Pareto front; Globally, such Pareto fronts can be stitched together to cover a wide range of solutions with varying trade-offs.

C.1 MultiMNIST and Its Variants

Figure C-1 depicts the continuous parametrization on MultiMNIST and its two variants. For each dataset, we gradually increased the number of Pareto optimal seeds from 3 to 25 and reconstructed a continuous approximation of the Pareto front (a curve in this 2D case) from each seed. It can be seen that as we added more seeds, the continuous Pareto fronts became more connected. By stitching them together, we have created a union of continuous Pareto fronts that offers very diverse choices of trade-offs. We further reparametrized it with a single scalar for easy manipulation and intuitive visualization.

C.2 UCI Census-Income

We display the continuous parametrization results on UCI Census-Income in Figure C-2. We started with 36 Pareto optimal seeds, densely sampled the continuous Pareto set

grown from each seed to evaluate their performances, and labeled samples from the same patch with a unique color. We gradually increased the number of samples in order to show how our continuous Pareto fronts were constructed progressively. Additionally, we reconstructed a 3D surface mesh from the Pareto fronts for better visualization.

C.3 UTKFace

Figure C-3 shows the continuous parametrization results on UTKFace. The setup and visualization is the same as in UCI Census-Income except that the continuous Pareto front was reconstructed from only 1 Pareto optimal seed. Therefore, a single color was used for all samples.

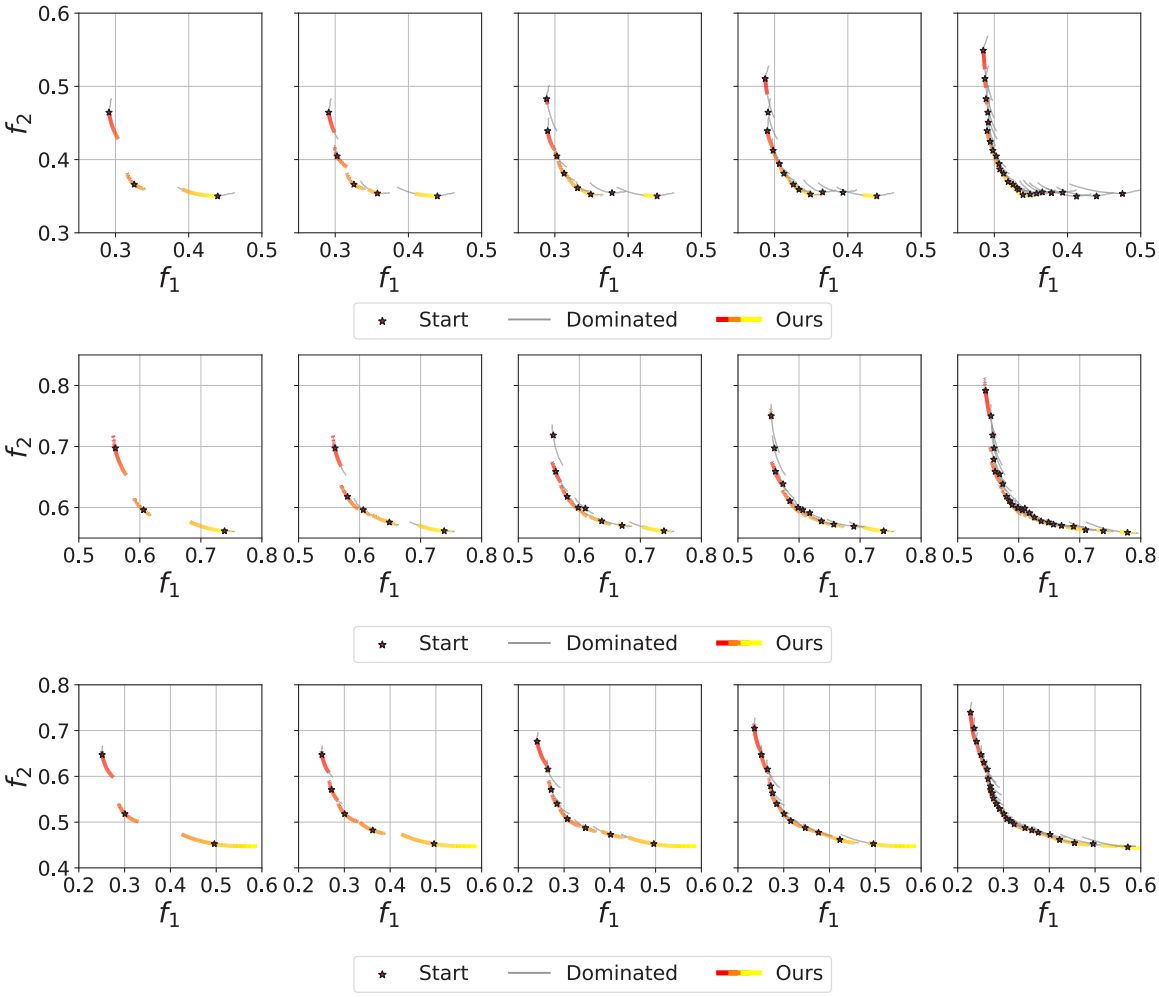

Figure C-1: Continuous parametrization on MultiMNIST (top), MultiFashion (middle), and MultiFashionMNIST (bottom). From left to right: we gradually increased the number of Pareto optimal solutions (red stars) obtained from running SGD with different weights. We then ran Algorithm 1 to grow a continuous Pareto front from each solution (colorful circles) and filtered out dominated solutions (gray). The red-to-yellow color indicates the value of the scalar parameter that traverses the whole final Pareto front.

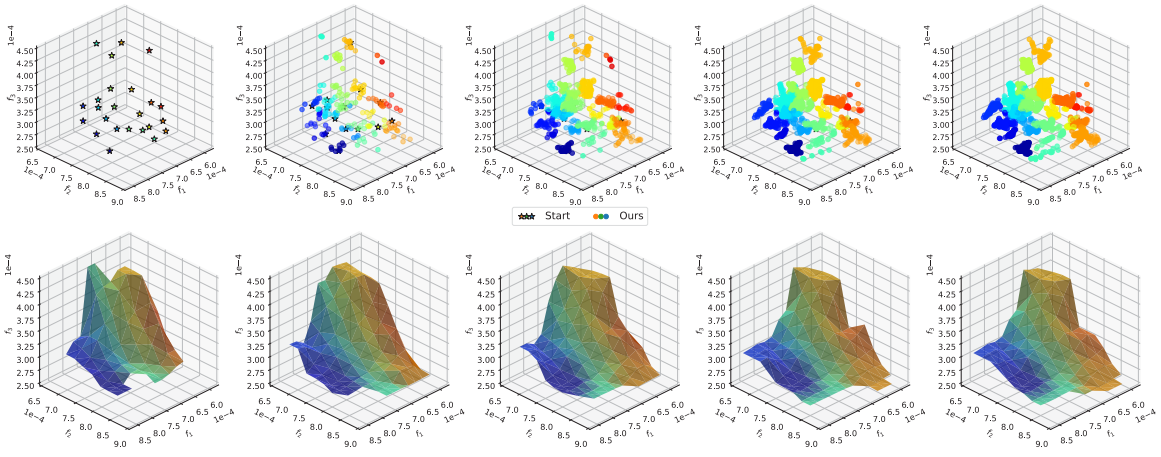

Figure C-2: Continuous parametrization on UCI Census-Income. Top: starting with discrete Pareto optimal seeds returned by running SGD with various weights on objectives (colorful stars), we constructed the continuous Pareto sets (not shown) with our method, densely sampled new solutions from these Pareto sets, and plotted their performances (colorful circles). Samples from the same seed share the same color. Bottom: we reconstructed a continuous surface mesh to approximate the Pareto front revealed by these samples above. The color on the surface mesh has a one-to-one correspondence to the color of Pareto optimal seeds; Left to right: We gradually increased the number of samples to show the progress of our reconstruction.

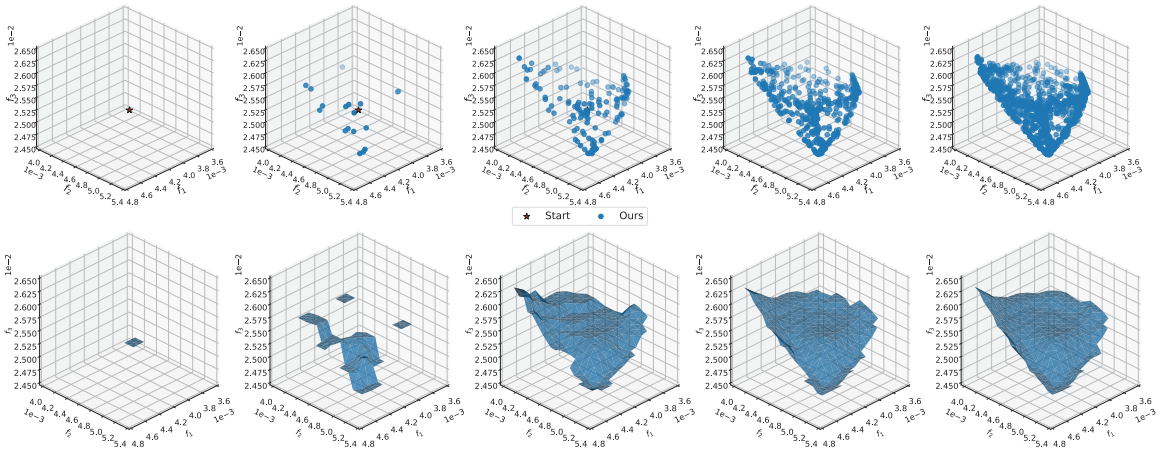

Figure C-3: Continuous parametrization on UTKFace. The setup is identical to Figure C-2 except that one Pareto seed obtained from ParetoMTL was used.

Appendix D

Ablation Study

Finally, we present more results of ablation study on MultiMNIST and its two variants in Figure D-1. For each dataset, we ran ParetoMTL to generate 5 Pareto optimal solutions that are evenly distributed on the Pareto front. From each solution, we conducted the ablation study on hyperparameters k and s as described in the main paper and produced one column of Figure D-1. It can be seen that our claims in the main paper on the influence of k and s are consistently observed across these 5 solutions with various trade-offs on these datasets.

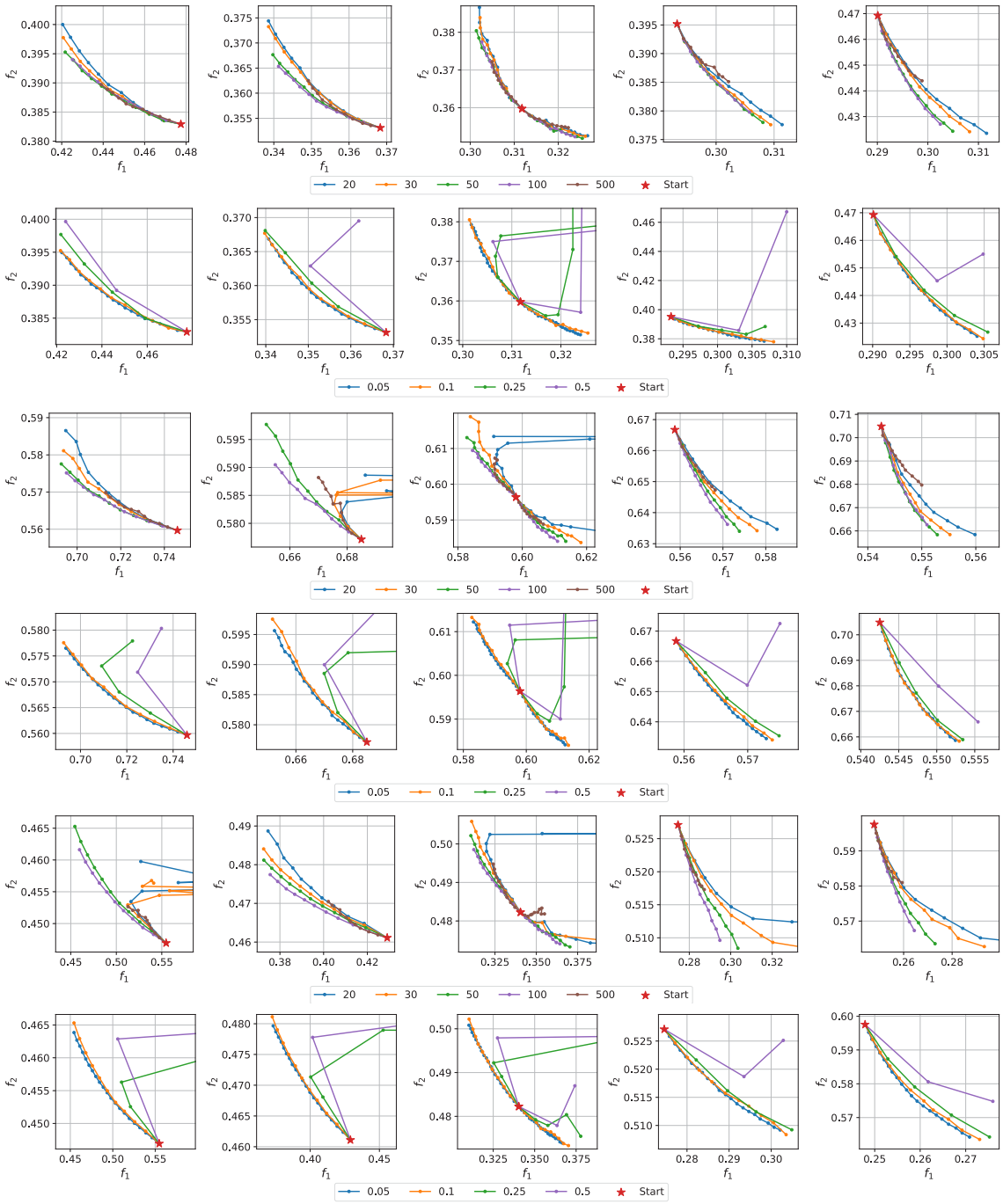

Figure D-1: Ablation study on the maximum number of MINRES iterations k and the step size s on MultiMNIST (top two rows), MultiFashion (middle two rows), and MultiFashion-MNIST (bottom two rows). We repeated the experiments from different Pareto optimal solutions returned by ParetoMTL (left to right).

Bibliography

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*, pages 794–803, 2018.

Sou-Cheng T Choi, Christopher C Paige, and Michael A Saunders. MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems. *SIAM Journal on Scientific Computing*, 33(4):1810–1836, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

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

Jorg Fliege and A Ismael F Vaz. A method for constrained multiobjective optimization based on SQP techniques. *SIAM Journal on Optimization*, 26(4):2091–2119, 2016.

David Chin-Lung Fong and Michael Saunders. CG versus MINRES: An empirical comparison. *Sultan Qaboos University Journal for Science [SQUJS]*, 17(1):44–62, 2012.

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*, pages 770–778, 2016.

Claus Hillermeier. Generalized homotopy approach to multiobjective optimization. *Journal of Optimization Theory and Applications*, 110(3):557–583, 2001.

Claus Hillermeier et al. *Nonlinear multiobjective optimization: a generalized homotopy approach*, volume 135. Springer Science & Business Media, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

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*, pages 7482–7491, 2018.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

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

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, pages 12037–12047, 2019.

James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

Adanay Martín and Oliver Schütze. Pareto tracer: a predictor–corrector method for multi-objective optimization problems. *Engineering Optimization*, 50(3):516–536, 2018.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1): 147–160, 1994.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

Adriana Schulz, Harrison Wang, Eitan Grinspun, Justin Solomon, and Wojciech Matusik. Interactive exploration of design trade-offs. *ACM Transactions on Graphics (TOG)*, 37(4): 1–14, 2018.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. In *Artificial Intelligence and Statistics*, pages 1261–1268, 2012.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.