SELF-EVOLUTIONARY OPTIMIZATION FOR PARETO FRONT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-task learning (MTL), which aims to improve performance by learning multiple tasks simultaneously, inherently presents an optimization challenge due to multiple objectives. Hence, multi-objective optimization (MOO) approaches have been proposed for multitasking problems. Recent MOO methods approximate multiple optimal solutions (Pareto front) with a single unified model, which is collectively referred to as *Pareto front learning* (PFL). In this paper, we show that PFL can be re-formulated into another MOO problem with multiple objectives, each of which corresponds to different preference weights for the tasks. We leverage an evolutionary algorithm (EA) to propose a method for PFL called *self-evolutionary optimization* (SEO) by directly maximizing the hypervolume. By using SEO, the neural network learns to approximate the Pareto front conditioned on multiple hyper-parameters that drastically affect the hypervolume. Then, by generating a population of approximations simply by inferencing the network, the hyper-parameters of the network can be optimized by EA. Utilizing SEO for PFL, we also introduce *self-evolutionary Pareto networks* (SEPNet), enabling the unified model to approximate the entire Pareto front set that maximizes the hypervolume. Extensive experimental results confirm that SEPNet can find a better Pareto front than the current state-of-the-art methods while minimizing the increase in model size and training cost.

1 INTRODUCTION

Multi-task learning (MTL) is a learning paradigm, which tries to guide a single shared model to simultaneously learn multiple tasks while leveraging the domain information included in related tasks as an inductive bias. The goal of MTL is to improve the performance of each task and reduce inference time for conducting all the necessary works for different tasks. For some applications, MTL achieves state-of-the-arts performance among large scale neural network-based approaches: computer vision (Liu et al., 2019a), natural language processing (Liu et al., 2019b) and recommender systems (Milojkovic et al., 2019). However, it has been observed that since MTL inevitably has multiple objectives corresponding to each task, it is not possible to optimize the performance of all tasks simultaneously due to potential conflicts between these objectives (Kendall et al., 2018). To tackle this issue, some attempts have been made to apply multi-objective optimization (MOO) to MTL problems, which optimizes multiple objectives with potential conflicts (Sener & Koltun, 2018; Lin et al., 2019; Ma et al., 2020). MOO aims to find a Pareto front, which consists of a set of solutions that can no longer be optimized without sacrificing the performance of at least one goal, and an element of it is called a Pareto optimal. When a preferred direction of the Pareto front, called preference vector (a.k.a. preference ray), is known, it is possible to obtain the corresponding solution (Mahapatra & Rajan, 2020).

However, in many cases, it is often difficult to predict the trade-off between the actual objectives according to this preference vector, since this trade-off can only be observed once training is complete. Because the number of optimal points in the Pareto front are theoretically infinite, it is very time-consuming to perform training for each optimal point to find an appropriate preferred vector. Moreover, for some applications, these preference vectors can change over time and context. A possible way to solve this problem is to approximate the entire set of Pareto front at inference time through a single model. Navon et al. (2020) call this problem *Pareto front learning* (PFL).

In PFL, a single model is trained to optimize multiple weighted combinations of the original objectives ($\boldsymbol{\mathcal{L}} = (\mathcal{L}_1(\theta), ..., \mathcal{L}_n(\theta))^T$) determined by the sampled preference vector ($\boldsymbol{r} \in \mathbb{R}^n_+$). Due to the stochastic nature of the preference vector, the model learns to approximate multiple weighted combinations ($\boldsymbol{\mathcal{L}}^T\boldsymbol{r}^1, \boldsymbol{\mathcal{L}}^T\boldsymbol{r}^2, ...$) during training to ultimately achieve a higher hypervolume (HV)[1]. In this respect, PFL can be re-formulated into another MOO with a goal of maximizing HV, posing new challenges compared to conventional MOO. As is often the case in MOO, we demonstrate that the new objectives, which are determined by the preference vectors, are in conflict with each other in certain circumstances, negatively affecting one another. We call this the *preference conflict.*

Due to the characteristics of PFL, carefully crafting a sampling procedure or explicitly optimizing the hypervolume is essential. Earlier works in PFL have implicitly mentioned this issue. Ruchte & Grabocka (2021) and Navon et al. (2020) both need to finetune the parameter of the Dirichlet distribution ($\boldsymbol{\alpha}$). Ruchte & Grabocka additionally add a regularization term that maximizes the cosine similarity between the preference vectors and the original objectives to induce "spreading" of the Pareto Front, where the coefficient (λ) needs to be finetuned. In both cases, the tuning of the hyper-parameters (λ and $\boldsymbol{\alpha}$) drastically influence the final hypervolume, but the computational cost of grid-searching the values on a validation set is extremely expensive. Moreover, $\boldsymbol{\alpha}$ directly affects the sampling of the preference vectors and λ affects the loss term, which are two key components that influence the training trajectory of the model parameters. However, the learning of the model parameters and the grid searching of the hyper-parameters $\boldsymbol{\alpha}$ and λ cannot be jointly performed.

In this paper, we explicitly maximize the hypervolume. First, we discover that what seemed like hyper-parameters can actually be directly optimized to maximize the hypervolume by also conditioning them to the model like the preference vectors. Since using gradient-based optimization for this is non-trivial, we leverage techniques in evolutionary algorithm (EA) to generate, *within the model itself*, population of solutions with different values of λ and $\boldsymbol{\alpha}$ without any additional training. To this end, we propose a self-evolutionary optimization for PFL that outperforms all existing MTL methods in a single unified model.

Our contributions are summarized as follows:

(1) We demonstrate that PFL can be interpreted as another MOO problem comprised of conflicting objectives determined by the preference vectors with a goal of maximizing HV.

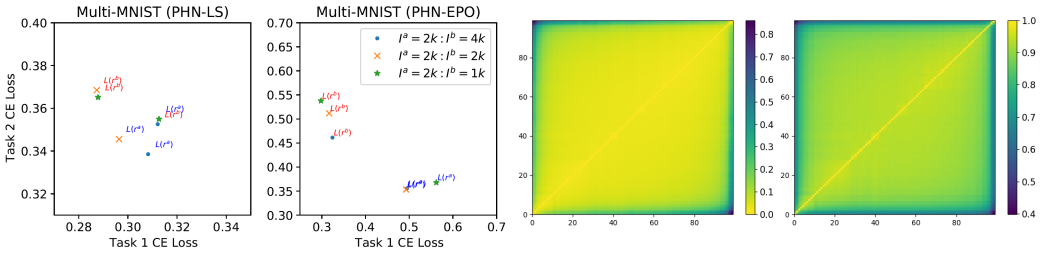
(2) We propose self-evolutionary optimization (SEO) for jointly optimizing the hyper-parameters that affect HV along with the model parameters. SEO explores the interval of hyper-parameters by conditioning the model with it during training. Once training is done, this allows the model to generate population of approximated solutions using certain value of hyper-parameters, which can be used for optimization using EA without further training.

(3) We introduce the application of SEO on PFL called Self-evolutionary Pareto Networks (SEPNet). SEPNet is an unified model that can not only approximate the entire set of Pareto front, but can estimate reward of hyper-parameters for various sampling and regularization, allowing direct optimization of HV. Extensive experimental results empirically confirm that the proposed method shows state-of-the-art performance on various MTL tasks and works effectively on large models and large-scale datasets.

2 Related Works

Multi-task learning (MTL) aims to learn a more general representation by jointly learning on a set of related multiple tasks. By avoiding overfitting on a particular task and taking advantage of the increased data efficiency, a model learned with multiple tasks can potentially achieve better generalization and a faster convergence toward an optimal solution. Crawshaw (2020) categorized MTL into multi-task architecture and multi-task optimization. On the architecture side, Liu et al. (2019a) proposed a task-shared feature extractor and task-specific attention modules to produce task-specific features. On the optimization side, Chen et al. (2018) minimized the difference in the per-task gradients while controlling the weights of per-task loss depending on their learning speed.

[1]Hypervolume can be considered as a measure of distance from a set of optimal points in a Pareto front to the worst possible loss vector called the reference point. For exact definition see (Navon et al., 2020).



(a) Preference conflict (b) Correlation matrix of preferences

Figure 1: Illustration of preference conflict in PFL (left) and correlation of preferences (right). I^j denotes the number of iterations in which $\boldsymbol{r}^j$ is selected.

Multi-objective optimization is used to find a Pareto front. To populate the Pareto frontier, earlier works (Sener & Koltun, 2018; Lin et al., 2019) relied on separately learning each point, which is not scalable for large deep networks. Quickly, many efforts have tackled this issue by conditioning on the objective preference during training (Navon et al., 2020; Ruchte & Grabocka, 2021, PHN, COSMOS) or through transfer-learning on the existing solutions (Ma et al., 2020). Linear scalarization (LS), which is a linear weighted sum of the objectives and preference vectors, has been used as a common approach to model a surrogate loss. Since LS often yields solution that does not lie on the preference vector, Mahapatra & Rajan (2020, EPO) propose a gradient-based algorithm to align the loss and the preference vector. All MOO methods are evaluated on the hypervolume (HV) metric, which is the volume created by the set of pareto optimal points and a reference point. HV considers the quality of the individual Pareto optimal points as well as the diversity of them. In this work, we re-formulate PFL as another MOO problem and firstly point out a crucial problem of trade-off between the objective preferences due to preference conflict. Moreover, we propose a method to optimize the HV metric directly using evolutionary algorithm.

Multi-objective evolutionary alogrithms (MOEA) have been one of the most active research areas in EA for recent decades (Coello, 2006; Deb, 2011). EAs have been applied to MOO due to their ability to generate sets of solutions called population in a single run. The seminal work of Deb et al. (2002) improves upon one of the early MOEA method (Srinivas & Deb, 1994) by reducing computational complexity and enhancing aspects of genetic algorithm. With the advent of neural networks in MOO, large scale problems have rendered these methods computationally too expensive as they scale poorly to training neural networks. To the best of our knowledge, our proposed method is the first work to use a single neural network through input conditioning to approximate population of multiple parameters to integrate EA into large scale MOO, enabling self-generation of population.

3 Revisiting Pareto Front Learning

A general multi-objective optimization (MOO) problem consisting of n different loss functions can be described by

$$\boldsymbol{\mathcal{L}}(\theta) = (\mathcal{L}_1(\theta), \mathcal{L}_2(\theta), ..., \mathcal{L}_n(\theta))^T, \tag{1}$$

where $\mathcal{L}_i(\theta)$ is the loss for the i-th[2] objective with parameter θ. MOO exploits the shared structures and information among them to optimize all objectives simultaneously. However, no single solution can optimize all objectives at the same time. Instead, we expect to obtain a solution which cannot improve the corresponding objective without degradation in another. Following the definition of Navon et al. (2020), θ^1 is said to dominate θ^2 if $\mathcal{L}(\theta^1) \preceq \mathcal{L}(\theta^2)$ and $\mathcal{L}_i(\theta^1) < \mathcal{L}_i(\theta^2)$ for some $i \in [n]$. An optimal solution that is not dominated by any other point is called *Pareto Optimal*.

Pareto Front Learning (PFL) is an approach to approximate the *Pareto front*, the set of all Pareto optimal, at inference time through a single model (Navon et al., 2020). Existing works minimize the empirical risk of the following loss

$$\bar{\mathcal{L}}_{PFL}(\theta) = \mathbb{E}_{\boldsymbol{r}\sim Dir(\boldsymbol{\alpha})} \sum_i g(r_i, \mathcal{L}_i(\theta)) = \mathbb{E}_{\boldsymbol{r}\sim Dir(\boldsymbol{\alpha})} \sum_i r_i \mathcal{L}_i(\theta) \tag{2}$$

[2]In this paper, we use a subscript to represent an element of the corresponding vector, while a superscript denotes its instantiation or a sample.

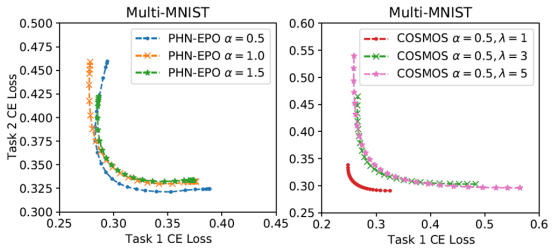


Figure 2: Comparison of the Pareto fronts on different hyper-parameters. For both methods, the Pareto front and hence the HV are sensitive to the hyper-parameters.

Table 1: Hypervolume on Multi-MNIST depending on hyper-parameter α and λ

Method	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 1.5$
PHN-LS	2.84	2.84	2.86
PHN-EPO	2.88	2.87	2.86
COSMOS ($\lambda = 1$)	2.99	2.95	2.97
COSMOS ($\lambda = 3$)	2.94	2.95	2.95
COSMOS ($\lambda = 5$)	2.96	2.97	2.95

where $g()$ is a function that weights each objective $\mathcal{L}_i$ by the corresponding weight r_i which is sampled from an n-dimensional Dirichlet distribution. The last equality comes from using LS as $g()$.

3.1 Preference Conflict

In this secsion, we describe the additional challenges of PFL that arise when a single model approximates multiple optimal points. We can interpret PFL as approximating distinct m points of the pareto front denoted as $\mathcal{L}(\theta, \boldsymbol{r}^j), j \in [m]$, with a single network:

$$\boldsymbol{\mathcal{L}}_{PFL}(\theta) = (\mathcal{L}(\theta, \boldsymbol{r}^1), \mathcal{L}(\theta, \boldsymbol{r}^2), ..., \mathcal{L}(\theta, \boldsymbol{r}^m))^T. \tag{3}$$

This give rises to another MOO problem for PFL. Here, we analyze the characteristics of PFL with distinct preference vectors $\boldsymbol{r}^j$.

Figure 1(a) shows the results of using two preference vectors $\boldsymbol{r}^a$ and $\boldsymbol{r}^b$ and varying their sampling frequency for PHN (Navon et al., 2020) and COSMOS (Ruchte & Grabocka, 2021). To show how each affects one another, we fix the sampling frequency of $\boldsymbol{r}^a$ and increase or decrease that of $\boldsymbol{r}^b$. The results indicate that this affects not only the results of $\boldsymbol{r}^b$, but also influences that of $\boldsymbol{r}^a$ as well, indicating the existence of conflictual interaction between the two. We define this as the *preference conflict*. Figure 1 shows that the preference conflict also exists when EPO, instead of LS, is used for optimization. Due to this conflict, one may not be able to improve the performance of one preference objective without sacrificing another preference objective. In Figure 1(b), we show the correlation between individually trained models for 100 linearly spaced preference vectors from 0.01 to 0.99. This is computed by the average of the JS divergence (left) and cosine similarity (right) of the outputs of each models. The diagonal elements indicate perfect correlation and all the preference vectors have some degree of correlation with its adjacent elements as shown by the high value of cosine similarity and low JS divergence. Note the gradation of the correlation between adjacent vectors slowly decreases (higher JS divergence, lower cosine similarity) as we move towards both ends of the preference vectors. Thus, solving the new MOO problem involves finding the optimal point considering the trade-off between the highly correlated preference vectors. We offer how to mitigate this problem in Section 4 by introducing the sparse sampling strategy.

3.2 Hypervolume Maximization

Here we motivate SEO from our new formulation (Eq. 3). Linear scalarization (LS) is the most straightforward approximation of multiple objectives in MOO. LS defines a single surrogate loss $\mathcal{L}_{\boldsymbol{r}}(\theta) = \sum_i r_i \mathcal{L}_i(\theta)$ given a preference vector $\boldsymbol{r} \in \mathbb{R}^n_+$ with $\sum_i r_i = 1$ for multiple objectives. In this paper, we define a new surrogate loss

$$\mathcal{L}'_{\boldsymbol{r}}(\theta) = \frac{1}{m} \sum_i^m \mathcal{L}(\theta, \boldsymbol{r}^i) \tag{4}$$

and show that $\mathcal{L}'_{\boldsymbol{r}}$ is exactly the same as $\mathcal{L}_{\boldsymbol{r}}$ in expectation. The proof is quite straightforward as follows: As in previous works, if we sample preference vectors according to the Dirichlet distribution

Algorithm 1 Self-evolutionary optimization (SEO)

Input: noise standard deviation σ, population size n
for *training epochs*: $k = 0, 1, 2, ...$ **do**
  for *training iterations*: $t = 0, 1, 2, ...$ **do**
    Sample $\epsilon \sim U(l, h)$
    Sample $(x, y) \sim p_D$
    $g_\theta \leftarrow \nabla_\theta \mathcal{L}(y, f(x, \phi + \epsilon; \theta), \phi + \epsilon)$
    $\theta \leftarrow \theta - \gamma_\theta g_\theta$
  Sample $\epsilon_1, ..., \epsilon_n \sim \mathcal{N}(0, I)$
  Compute $\mathcal{S}_i = \mathcal{S}(f(\phi + \sigma\epsilon_i; \theta))$ for $i \in [n]$ on the entire validation dataset
  $\phi \leftarrow \phi + \gamma \frac{1}{n\sigma} \sum_{i=1}^{n} \mathcal{S}_i \epsilon_i$

with parameter $\boldsymbol{\alpha}$, the expected loss becomes

$$\begin{aligned} \bar{\mathcal{L}}'_{PFL}(\theta) &= \mathbb{E}_{\{\boldsymbol{r}^1, \cdots, \boldsymbol{r}^m\} \sim Dir(\boldsymbol{\alpha})} \frac{1}{m} \sum_{i}^{m} \mathcal{L}(\theta, \boldsymbol{r}^i) \\ &= \mathbb{E}_{\boldsymbol{r} \sim Dir(\boldsymbol{\alpha})} \mathcal{L}(\theta, \boldsymbol{r}) = \mathbb{E}_{\boldsymbol{r} \sim Dir(\boldsymbol{\alpha})} \sum_{j}^{n} r_j \mathcal{L}_j(\theta), \end{aligned} \tag{5}$$

which is identical to Eq. 2. Note that the expected loss $\bar{\mathcal{L}}_{PFL}$ depends on hyper-parameter $\boldsymbol{\alpha}$.

Since the final goal of PFL is to maximize the hypervolume (HV), we seek to solve $\arg\max_{\theta, \boldsymbol{\alpha}} HV(\theta, \boldsymbol{\alpha})$. For COSMOS, an additional hyper-parameter λ exists for maximizing the cosine similarity of the preference vector $\boldsymbol{r}$ and objective vector $\boldsymbol{\mathcal{L}}$, which leads to finding the optimal model parameters by $\theta^* = \arg\min_\theta \mathbb{E}_{\boldsymbol{r} \sim Dir(\boldsymbol{\alpha})} \sum_i r_i \mathcal{L}_i(\theta) - \lambda C(\boldsymbol{r}, \boldsymbol{\mathcal{L}})$ where C denotes the cosine similarity between two vectors. Note that to reduce the search space, we set all the elements of $\boldsymbol{\alpha}$ identically as α. By combining all the hyper-parameters as $\phi = [\alpha, \lambda]^T$, our PFL problem ultimately solves

$$(\theta^*, \phi^*) = \arg\max_{\theta, \phi} HV(\theta, \phi). \tag{6}$$

We show in Table 1 and Figure 2 that α and λ drastically influences HV and the final Pareto front depending on the value, which necessitates grid-search.

4 Proposed Method

4.1 Self-evolutionary Optimization

As mentioned in Section 3, PFL tries to find a network with optimal model parameters θ and hyper-parameters ϕ. However, training a neural network from scratch to find ϕ is computationally very expensive. Instead, we design our network to approximate the HV when certain ϕ values are used in inference by conditioning them while training. Then, the approximated population of solutions can be optimized by an algorithm based on evolutionary strategy (ES) to find the optimal ϕ. We call our method self-evolutionary optimization (SEO).

Given a fitness function $\mathcal{S}$[3], hyper-parameters ϕ, a noise standard deviation σ, the ES algorithm that uses Gaussian search makes use of an estimator for the gradient $\nabla_\phi \mathbb{E}_{\epsilon \sim N(0,I)} \mathcal{S}(\phi + \sigma\epsilon)$ (Nesterov & Spokoiny, 2017; Salimans et al., 2017). Our SEO that utilizes this ES algorithm consists of two phases. First, we redefine our original network $f(x; \theta)$ as a conditional function of ϕ as well by $f(x, \phi; \theta)$. In the training phase, $f(x, \phi; \theta)$ is trained through gradient descent with the following gradient:

$$\nabla_\theta \mathbb{E}_{\epsilon \sim U(l,h)(x,y) \sim P_D} \mathcal{L}(y, f(x, \phi + \epsilon; \theta), \phi + \epsilon). \tag{7}$$

Here, $U(l, h)$ indicates uniform sampling between l and h[4].

[3]Here, the hypervolume is directly used as our fitness function.

[4]A uniform distribution is used in the training phase as we find the Gaussian distribution deteriorates the performance by excessively sampling values near the mode.

Table 2: Quantitative Evaluation on Image Classification and Fairness.

Method	Multi-MNIST	Multi-Fashion	Multi-F+MNIST	Adult	Compass	Default
PMTL	2.90	2.27	2.74	2.92	2.15	3.10
PHN-EPO	2.86	2.20	2.80	3.34	3.71	3.11
PHN-LS	2.85	2.19	2.77	3.34	3.71	3.12
COSMOS	2.94	2.32	2.84	3.34	**3.72**	3.12
SEPNet (ours)	**3.04**	**2.39**	**2.93**	**3.36**	**3.72**	**3.13**

Then, in the validation phase, we search optimal ϕ using the ES algorithm. Using n population of ϕ, which is obtained using additive Gaussian noise, the score according to each ϕ is estimated through $f(\phi;\theta)$ and the fitness function $\mathcal{S}$. We directly optimize ϕ with stochastic gradient ascent using the following gradient estimator on the right (Salimans et al., 2017):

$$\nabla_{\phi}\mathbb{E}_{\epsilon\sim N(0,I)}\mathcal{S}(f(\phi+\sigma\epsilon;\theta)) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim N(0,I)}\{\mathcal{S}(f(\phi+\sigma\epsilon;\theta))\epsilon\}. \tag{8}$$

Algorithm 1 illustrates this process of alternating between the training phase and the validation phase.

4.2 Sparse Sampling

We showed in Section 3.1 that PFL learns a single network to solve MOO of highly correlated preference vectors. These vectors are sampled from a continuous distribution and used as additional input for the network, meaning the model has to theoretically approximate infinite objectives. Moreover, due to the high correlation between the adjacent samples, continuous sampling may interfere with the weights of adjacent samples. To reduce the overhead caused by additional infinite inputs and the influence of adjacent preference vectors, we propose a simple trick called "*sparse sampling*". Sparse sampling quantizes the samples from a continuous distribution to a smaller set of discrete finite values. We define a quantization function $Q(x;\Gamma,\tau)$ that quantizes x to $\overline{x}$, which corresponds to the center of a bin among uniformly divided τ number of bins for range $\Gamma=(l,h)$. Applying the quantization function to $\boldsymbol{r}$ for $\mathbb{E}_{\boldsymbol{r}\sim Dir(\alpha)}\sum_i r_i\mathcal{L}_i(\theta)$ yields $\mathbb{E}_{\boldsymbol{r}\sim Dir(\alpha)}\sum_i \overline{r_i}\mathcal{L}_i(\theta)$.

4.3 Self-evolutionary Pareto Network

Here, we introduce our method by applying SEO to Pareto front learning called *self-evolutionary Pareto networks* (SEPNet). SEPNet defines the fitness function $\mathcal{S}$ as HV (Eq. 6) and optimizes it via SEO. SEPNet first optimizes the model parameter θ while being conditioned on ϕ, which consists of α and λ, i.e, $\phi=[\alpha,\lambda]^T$. And we apply sparse sampling for preference rays r and ϕ. In this case, the gradient in Eq. 7 becomes

$$\nabla_{\theta}\mathbb{E}_{\epsilon\sim U(l,h),\boldsymbol{r}\sim Dir(\phi_\alpha+\epsilon_\alpha),(x,y)\sim P_D}\mathcal{L}(y, SEPNet(x,\overline{\boldsymbol{r}},\overline{\phi+\epsilon};\theta),\overline{\phi+\epsilon}). \tag{9}$$

Then in the validation phase, it optimizes ϕ using Eq. 8. To condition $\boldsymbol{r}$ and ϕ to our network, we utilize a condition injection module that takes in the input condition $C=[r,\phi]^T$ (i.e, instantiation of $\boldsymbol{r}$ and ϕ). Using conditioning techniques from Huang & Belongie (2017); Perez et al. (2018), we define the conditioning module G as an affine transformation function on the feature space x as $\tilde{x}=G_{scale}(C)x+G_{shift}(C)$, where G is a MLP with two fully connected layers. And two or three conditioning modules are inserted after different layers of the baseline networks. Therefore, the increase in parameter is negligible.

5 Experiments

In this section, we verify our SEPNet advances the state-of-the-art in PFL on various datasets, significantly outperforming existing works. Additionally, we conduct an ablation study to test the effectiveness of self-evolutionary optimization and sparse sampling.

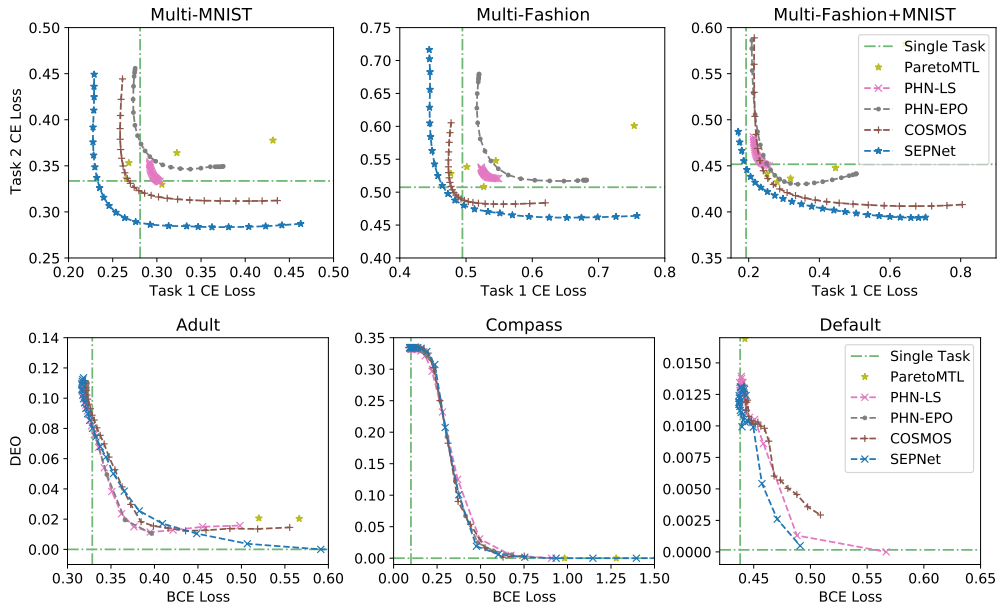


Figure 3: Comparison of the Pareto fronts on Image classification (top) and Fairness (bottom) tasks.

5.1 Experimental Settings

Our experiments are conducted following the general settings of recent MOO works (Lin et al., 2019; Navon et al., 2020). We compare with Navon et al. (2020, PHN-LS, PHN-EPO), Lin et al. (2019, ParetoMTL) and Ruchte & Grabocka (2021, COSMOS) as baselines which generate Pareto fronts with gradient-based method. The baselines are trained with officially implemented code, and early stopping is applied using the HV on the validation set. Unless otherwise stated, we use Adam (Kingma & Ba, 2014), a batch size of 256 and a learning rate 1e-3. The reported scores and Pareto fronts are the average of learning 5 times each, and we take (2,2) as a reference point of HV.

For our SEO, we use $\sigma = 0.1$ and population size $n = 10$ for the ES algorithm, and fix the learning rate to 0.01. We alternate between α and λ every 5 epochs for optimization. We evaluate the HV of every population using 5 preference rays on the entire validation dataset. We reduce the number of preference rays from 25 to 5 in the validation phase, but the overall validation time is approximately doubled by using $n = 10$. The overhead in training time was within 10% compared to a single objective baseline. For sparse sampling, we quantize the samples of Dirichlet distribution by a grid of 0.1. For SEO, we choose the offset of α to be between -0.2 and +0.2, which is sampled within a discrete bin of size 0.1. Similarly, the offset of λ is between -1.0 and 1.0 with bin size 0.5. SEO optimizes α and λ in the validation set and the found values are used for evaluation in the test set.

Image Classification We evaluate our method on multi-MNIST (Sabour et al., 2017) and two variants of the FashionMNIST dataset (Xiao et al., 2017) called multi-Fashion and Fashion+MNIST (Lin et al., 2019). For each data set, two images are randomly sampled from the corresponding dataset (e.g. MNIST + MNIST, Fashion + MNSIT) , then the images are slightly overlapped by placing them on top-left (TL) and bottom-right (BR). In MTL, the model classifies both instances at the same time. Each dataset consists of 120,000 training examples and 20,000 test examples, we allocate 10% of training examples for validation. We use LeNet (LeCun et al., 1999) and learning rate are decayed by 0.1 at 40, 80, 90 epoch.

Fairness We conduct experiments for incorporating fairness into the model. We use the Adult (Dua & Graff, 2017), Compass (Angwin et al., 2016) and Default (Yeh & Lien, 2009) dataset. We choose 'sex' as a binary sensible attribute and optimize the fairness objective which is a hyperbolic tangent relaxation of *difference of equality of opportunity* ($\widehat{DEO}$) defined in Padh et al. (2020). We train two hidden layers (60 and 25 dimensions) of MLP with ReLU activation for 50 epochs. Each dataset is divided into train/validation/test sets of 70%/10%/20%, respectively.

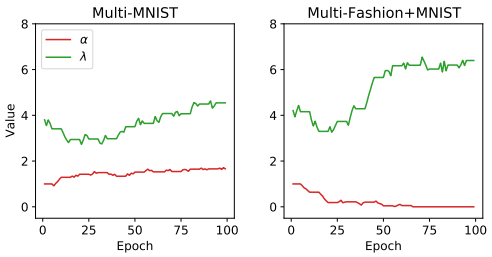


Figure 4: Trends in α and λ during training for the two Image Classification datasets.

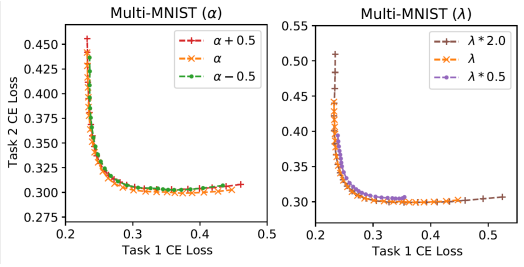


Figure 5: Various generated Pareto fronts by conditioning the network inputs α and λ. We see that by SEO, the network is able to generate various Pareto fronts dependent on the hyper-parameters.

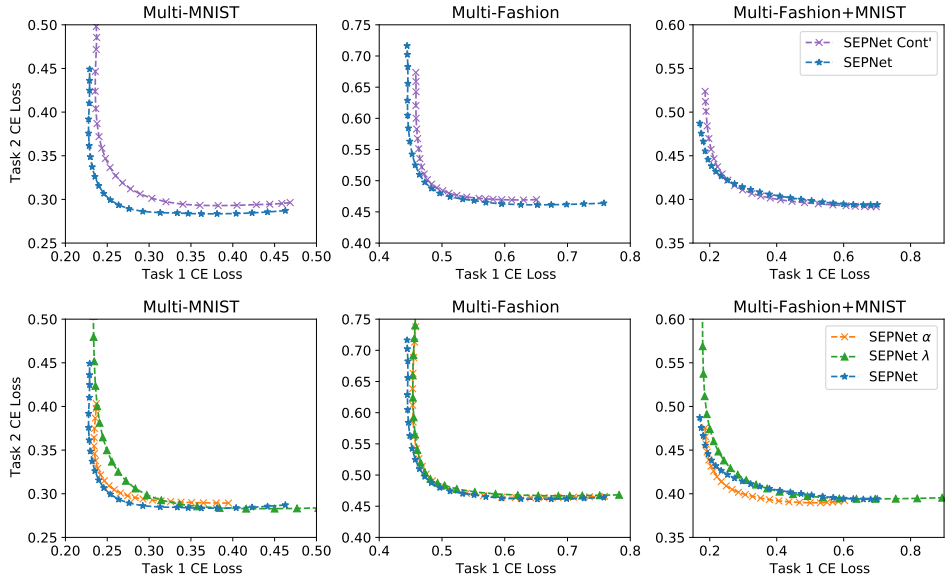


Figure 6: Qualitative results of ablation study. The first row illustrates the effect of sparse sampling, while the second row shows that of SEO.

CelebA CelebA (Liu et al., 2015) is a large-scale face attributes dataset with more than 200K celebrity images. We adopt the experimental setup of COSMOS. We rescale the images of CelebA to 64x64, and adopt ImageNet-pretrained EfficientNet-B4 (Tan & Le, 2019) as a backbone. SEPNet is trained by setting the learning rate to 5e-4 and the batch size to 32. For MTL, we optimize binary cross-entropy for two easy and two hard tasks as done by Ruchte & Grabocka (2021). Facial attributes such as Oval Face and Pointy Nose are combined to make the hard task, and Goatee and Mustache are used as the easy task.

5.2 Results and Discussion

Table 2 shows that SEPNet outperforms both methods using a seperate model for each preference vector (PMTL) and methods using an unified model (PHN, COSOMOS). Qualitatively, Figure 3 shows that SEPNet creates a wider Pareto front with superior solutions. In particular, for the multi-FASION+MNIST dataset, SEPNet attains solutions that perform better than that of single task for Task 1, whereas all the other methods are unable to reach this point.

On the Fairness dataset, we quantitatively attain higher HV than the baseline, but the improvement is less noticeable. We believe this is due to the use of small neural network (e.g. 2-layer MLP) that has a limited capacity for learning the whole Pareto frontier.

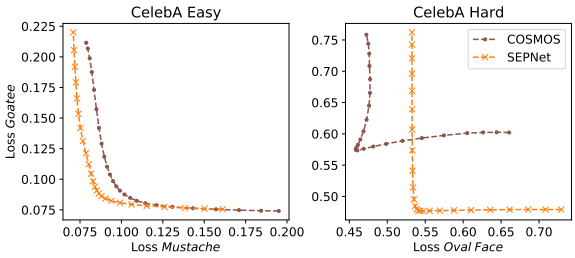


Figure 7: The Pareto fronts on CelebA

Table 3: Hypervolume on CelebA Easy and Hard tasks

Method	CelebA Easy	CelebA Hard
Single Task	3.719	2.222
COSMOS	3.706	2.221
SEPNet	**3.713**	**2.235**

Figure 4 shows the trend of α and λ while training for Image Classification. Note that the different optimal values are found in each dataset, which implies that to find the optimal values for other methods, these should be tuned as hyper-parameters for each dataset by grid-search. After training, the two parameters can be conditioned to different values for inference to generate multiple Pareto fronts as shown by Figure 5. Hence by following the protocol explained in Section 4.1, we can find the optimal parameters in the validation set through self-evolutionary optimization.

5.3 Ablation

Sparse Sampling Figure 6 indicates the Pareto front of SEPNet Cont', which uses continuous sampling instead of sparse sampling. For all three datasets, SEPNet creates superior optimal points overall. Also note that while sparse training only samples 10 points from the segment $[0, 1]$, it is able to interpolate all the 25 preference vectors, which were not seen during training.

Optimization Parameters using SEO In the second row of Figure 6, "SEPNet α" only optimizes α while fixing λ, and vice-versa. We find that the optimizing the two parameters has a distinct effect. For instance, optimizing for λ leads to a wider Pareto front, which offers more diverse solutions and increases HV. On the other hand, optimizing for α usually advances the preference vectors near the center ray (0.5, 0.5). This phenomenon is most noticeable for Multi-MNIST. Overall, though training for only single parameter may have individual preference vectors that perform better than that of SEPNet that optimizes both, SEPNet achieves the highest HV. Quantitative analysis can be found in Appendix.

5.4 Scaling Up SEPNet

We experiment with CelebA to verify our SEPNet can also be applied to large dataset, using a modern architecture model (i.e., EfficientNet-B4). Because PHN and ParetoMTL have poor scalability, we compare our method with COSMOS and Single Task baseline, trained using a single task objective. Table 3 shows the quantitative comparison of SEPNet and other baselines. Since a Single Task does not approximate the Pareto front for multiple reference rays, the HV is calculated at the cross point of the two task losses following convention. COSMOS did not show better results than Single Task baseline in both Easy and Hard tasks. In contrast, SEPNet has higher HV than all baselines for Hard Task and obtains similar results to Single Task baseline on Easy Task. And as shown in Figure 7, our method approximates a well-spread smooth Pareto front even in CelebA. These results indicate that SEPNet can approximate the entire Pareto fronts on a large dataset using a modern architecture.

6 Conclusion

We propose a new perspective to PFL as another MOO problem. We tackle conflicts between the preference vectors and propose to optimize hyper-parameters jointly with model parameters by leveraging an evolutionary strategy. In doing so, we set the fitness function for hyper-parameter optimization as the hypervolume and could achieve SOTA performances on various MTL tasks. Our work can be improved upon by utilizing a more effective evolutionary algorithm such as genetic algorithms. In addition, additional methods can be devised to optimize the model parameters also for the hypervolume metric.

REFERENCES

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica, May*, 23 (2016):139–159, 2016.

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

CA Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine*, 1(1):28–36, 2006.

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

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pp. 3–34. Springer, 2011.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pp. 319–345. Springer, 1999.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32:12060–12070, 2019.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019a.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, 2019b.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

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

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

Nikola Milojkovic, Diego Antognini, Giancarlo Bergamin, Boi Faltings, and Claudiu Musat. Multi-gradient descent for multi-objective recommender systems. *arXiv preprint arXiv:2001.00846*, 2019.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*, 2020.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Kirtan Padh, Diego Antognini, Emma Lejal Glaude, Boi Faltings, and Claudiu Musat. Addressing fairness in classification with a model-agnostic multi-objective algorithm. *arXiv preprint arXiv:2009.04441*, 2020.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Michael Ruchte and Josif Grabocka. Efficient multi-objective optimization for deep learning. *arXiv preprint arXiv:2103.13392*, 2021.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

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

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480, 2009.

A APPENDIX

A.1 EXPERIMENT DETAILS

A.1.1 EXPERIMENTAL DETAILS OF PREFERENCE CONFLICT

For each training, preference ray $\boldsymbol{r}^a$ is used for I^a iterations, and $\boldsymbol{r}^b$ is used for I^b iterations. $\boldsymbol{r}^a$ or $\boldsymbol{r}^b$ is chosen randomly according to each number of iterations.

$\boldsymbol{r}^a$ = [0.2, 0.8]
$\boldsymbol{r}^b$ = [0.8, 0.2]

A.1.2 POSITION OF CONDITIONING MODULES

We use an MLP with two fully connected (FC) layers as a conditioning module. The first FC layer has the same number of nodes as the target feature layer and the second FC layer is twice the first. We utilize the half of the output as a scaling parameter and the other half as a shifting parameter.

Figure A1: LeNet (Method & Multi-MNIST & Multi-Fashion & Multi-F+MNIST)

Input x
Convolution
Conditioning Module
Convolution
Conditioning Module
Shared Fully Connected
Head Fully Connected

Figure A2: MLP (Adult & Compass & Default)

Input x
Fully Connected
Conditioning Module
Fully Connected
Conditioning Module
Head Fully Connected

Table A1: EfficientNet-B4 (CelebA)

Input x
Stem Convolution
MBConvBlocks (0-10)
Conditioning Module
MBConvBlocks (11-20)
Conditioning Module
MBConvBlocks (21-30)
Conditioning Module
MBConvBlocks (31)
Head Convolution
Head Fully Connected

A.2 ADDITIONAL EXPERIMENTS

Table A2: Hypervolume with or without Sparse Sampling *SEPNet Cont'* denotes the SEPNet without Sparse Sampling.

Method	Multi-MNIST	Multi-Fashion	Multi-F+MNIST
SEPNet	**3.04**	**2.39**	**2.93**
SEPNet Cont'	3.01	2.36	2.91

Table A3: Hypervolume according to a selection of hyper-parameters for SEO

Method	Multi-MNIST	Multi-Fashion	Multi-F+MNIST
SEPNet $\alpha\&\lambda$	**3.04**	**2.39**	**2.93**
SEPNet α	3.02	2.37	2.92
SEPNet λ	3.03	2.37	2.92

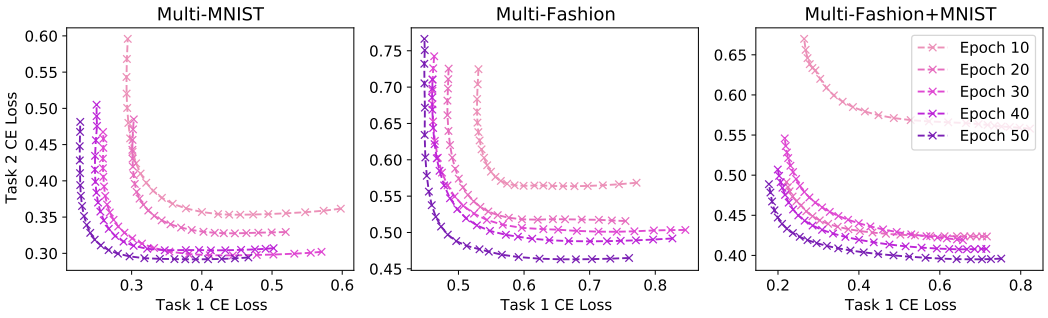


Figure A3: SEPNet's convergence graph of the Pareto fronts according to model updates SEPNet approximates well-spread Pareto fronts from the beginning of training and converges very fast. And, even though only discrete preference rays are used in training due to Sparse Sampling, it can approximate the smooth Pareto front for all preference rays that are not used for training.

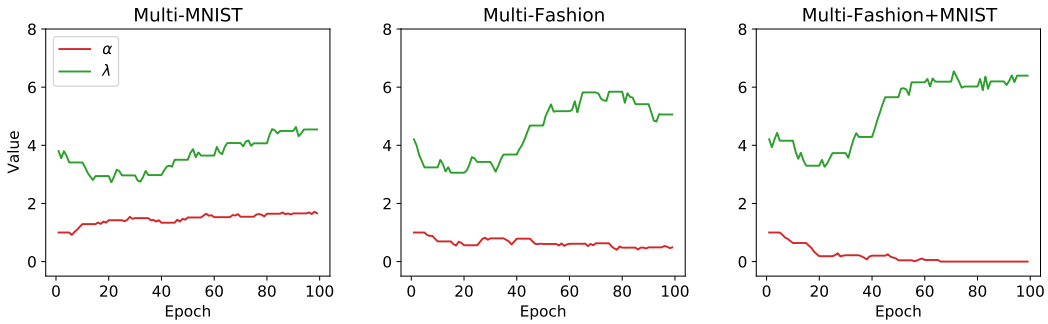


Figure A4: Trends in α and λ during training for the three Image Classfication datasets Each parameter is optimized with different optimal points for different datasets.

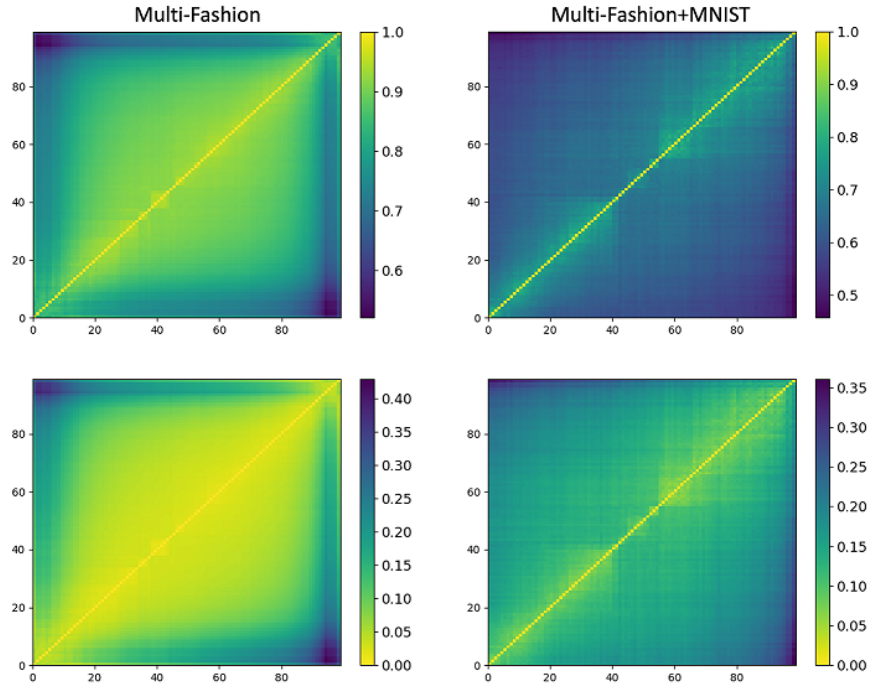


Figure A5: Illustration of correlation of preferences. First row uses cosine similarity while the second row uses JS divergence.

Table A4: Model size of each method The separate model requires n models for n preference rays.

Method	Number of Parameters
	Multi-MNIST & Multi-Fashion & Multi-F+MNIST
Single Task	n×42k
ParetoMTL	n×42k
PHN-LS	3,243k
PHN-EPO	3,243k
COSMOS	43k
SEPNet	44k
	Adult
Single Task	n×6k
ParetoMTL	n×6k
PHN-LS	716k
PHN-EPO	716k
COSMOS	7k
SEPNet	16k
	Compass
Single Task	n×2k
ParetoMTL	n×2k
PHN-LS	304k
PHN-EPO	304k
COSMOS	2k
SEPNet	12k
	Default
Single Task	n×7k
ParetoMTL	n×7k
PHN-LS	728k
PHN-EPO	728k
COSMOS	7k
SEPNet	16k