Multi-Objective Deep Learning with Adaptive Reference Vectors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many deep learning models involve optimizing multiple objectives. Since objectives are often conflicting, we aim to get diverse and representative trade-off solutions among these objectives. Gradient-based multi-objective optimization (MOO) algorithms using reference vectors have shown promising performance. However, they may still produce undesirable solutions due to mismatch between the pre-specified reference vectors and the problem's underlying Pareto front. In this paper, we propose a novel gradient-based MOO algorithm with adaptive reference vectors. We formulate reference vector adaption as a bilevel optimization problem, and solve it with an efficient solver. Theoretical convergence analysis is also provided. Experiments on an extensive set of learning scenarios demonstrate the superiority of the proposed algorithm over the state-of-the-art.

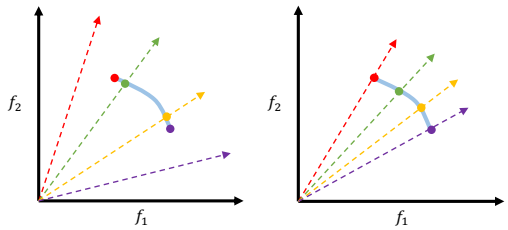
1 Introduction

Deep learning models are often evaluated under multiple, potentially conflicting, criteria. For example, in multi-task learning [6], a single model is required to perform well on multiple tasks. In some scenarios, besides accuracy, model fairness is also important so as to ensure that the model is not biased against gender and race. These problems can all be formulated as multi-objective optimization (MOO) [36] problems, and have attracted attention from various fields such as energy resource optimization [8] and signal processing [4].

Since the multiple objectives usually cannot be optimized simultaneously by a single solution, the goal of MOO is to find a set of solutions with different trade-offs to approximate the true Pareto front (PF). The past decades have witnessed the birth of a large number of gradient-free MOO algorithms, such as evolutionary multi-objective optimization algorithms [10, 49] and Bayesian multi-objective algorithms [23, 3]. These algorithms perform well on small-scale problems but fail to provide useful solutions when facing the huge number of parameters in deep learning models.

Recently, gradient-based MOO algorithms [13] have demonstrated promising performance in deep learning. Based on the pioneering work in [42], Lin et al. [28] and Mahapatra et al. [34] propose to generate multiple solutions using a set of reference vectors (RV). These algorithms try to search for solutions on the PF closest to each RV. Some strategies are further proposed to improve the efficiency by training a hypernetwork [37, 27] or reference-vector-conditioned network [41]. However, note that a set of uniformly distributed RVs may not lead to a set of uniformly distributed solutions. Hence, an important limitation of these algorithms is that solutions generated using fixed RVs may not cover some parts of the PF, thus failing to provide enough information about the PF. As an illustration, in Figure 1a, solutions obtained by the fixed RVs are close to the ends of the PF. If the RVs are properly positioned, it is possible that the obtained solution set can uniformly cover the PF (Figure 1b).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Fixed RVs. (b) Adaptive RVs.

Figure 1: Illustration of the difference between fixed and adaptive reference vectors. The dashed lines are reference vectors and the points are solutions on the PF closest to each reference vector.

Since it is impossible to know the true PF of the problem before optimization, how to generate a proper RV set is the main challenge. In this paper, we propose to learn the set of RVs simultaneously with the model parameters. This is formulated as a bilevel optimization problem [12], in which the lower-level optimization problem obtains the Pareto-optimal solutions with the given set of RVs, while the upper-level problem optimizes the RVs based on some quality measure. Moreover, we propose an inexpensive solver for this optimization problem, while still showing nice theoretical convergence properties. Experiments on an extensive set of learning scenarios demonstrate the superiority of the proposed algorithm.

2 Related Work: Multi-Objective Optimization (MOO)

In multi-objective optimization (MOO) [36], one aims to minimize $m \geq 2$ objectives $\{f_1(\phi), f_2(\phi), \dots f_m(\phi)\}$, or, equivalently, the vector-valued function:

$$\min_{\phi} f(\phi) = [f_1(\phi), \dots, f_m(\phi)] \in \mathbb{R}^m. \tag{1}$$

A solution ϕ_1 is *dominated* by another solution ϕ_2 if and only if $f_i(\phi_1) \geq f_i(\phi_2)$ for $i \in [m] \equiv \{1, \dots, m\}$, and $\exists i \in [m], f_i(\phi_1) > f_i(\phi_2)$. A solution ϕ^* is *Pareto-optimal* if and only if it is not dominated by any other ϕ'. A *Pareto front* (PF) is the set of multi-objective values of all Pareto optimal solutions. A PF is *regular* [45] if its shape is simplex-like (i.e., all vectors with positive directions intersect it when they start at the origin), and irregular otherwise. It is shown that regular PFs are not very realistic [22].

Since the number of Pareto-optimal solutions is usually large or even infinite, a set of n Pareto-optimal solutions $\Phi = \{\phi_1, \dots, \phi_n\}$ is often used to approximate the PF. Denote the corresponding multi-objective values as $\mathcal{F} = \{f(\phi_1), \dots, f(\phi_n)\}$. The quality of Φ can be evaluated from two perspectives: convergence and diversity [24]. Convergence refers to the distance between $\mathcal{F}$ and the true PF, while diversity measures whether the solutions are well-distributed in the space of objectives. A popular measure, which evaluates both convergence and diversity, is the hypervolume (HV) [50]. Given a reference point $z \in \mathbb{R}^m$, the HV of $\mathcal{F}$ is:

$$HV(\mathcal{F}; z) = \lambda(\cup_{f_i \in \mathcal{F}} [z, f_i]), \tag{2}$$

where $[z, f_i] \equiv \{q \in \mathbb{R}^m | f_i \leq q \leq z\}$, and $\lambda(\cdot)$ is the Lebesgue measure of a set.

A reference vector (RV) [49], sometimes called the weight vector or preference vector, can be used to guide the optimization algorithm by indicating the preferred point on the PF. Usually, an algorithm is expected to obtain the Pareto-optimal solution closest to the given RV in the objective space. Given a RV $r \in \mathbb{R}^m_+$, the MOO problem in (1) can be converted to a single-objective problem by using a scalarization function $s(\phi; r)$. The most straightforward construction is linear scalarization:

$$s(\phi; r) = \sum_{j=1}^{m} r(j) f_j(\phi), \tag{3}$$

which weights the m objectives with the elements $r(j)$'s of r. It is known that the minimizer of $s(\phi; r)$ is also Pareto-optimal for the original problem in (1) (Ch 4.7, [5]). However, the solution obtained may be far way from the given RV. To encourage the solutions to be closer to this vector, a penalty term can be added to the linear scalarization function [41]:

$$s(\phi; r) = \sum_{j=1}^{m} r(j) f_j(\phi) + \gamma \frac{r^\top f(\phi)}{\|r\| \cdot \|f(\phi)\|}, \tag{4}$$

where γ is a constant. Given a (discrete) set of solutions, Ma et al. [33] generates exploration directions to spawn new solutions on the PF, leading to a continuous PF.

By using n RVs from a subset $\mathcal{R} \subseteq \mathbb{R}_+^{m\times n}$, a set of solutions can be obtained to approximate the entire PF. Usually, $\mathcal{R}$ is simply set to $\mathbb{R}_+^{m\times n}$. However, sometimes the decision-makers may only be interested in a specific region of the PF [2]. For example, we may limit the angle between any RV and each coordinate axis. $\mathcal{R}$ can then be changed to:

$$\left\{ \{r_1, \dots, r_n\} \;\middle|\; \cos\varphi_2 \le \frac{r_j^\top u_i}{\|r_j\| \cdot \|u_i\|} \le \cos\varphi_1,\ \forall i \in [m], j \in [n] \right\}, \tag{5}$$

where u_i is the ith coordinate axis and φ_1, φ_2 are the maximum and minimum allowable angles, respectively. However, note that the set of pre-specified RVs may not fit the problem's PF (e.g., some of them may not intersect the underlying PF), leading to an undesirable solution distribution.

Gradient-Free MOO. Evolutionary MOO algorithms (e.g., NSGA-II[10], MOEA/D[49]) and Bayesian MOO algorithms (e.g., BMOA[23], USeMO[3]) are widely used for small-scale black-box problems. These algorithms assume that gradient information is not available. Hence, they often fail to converge on deep learning problems where the solution space can be very large.

Gradient-Based MOO. Gradient-based MOO algorithms are more efficient when problems have differentiable objectives and a large number of parameters. Sener and Koltun [42] propose to apply Multiple-Gradient Descent Algorithm (MGDA) [13] to multi-task learning. Liu and Vicente [30] provide theoretical analysis of stochastic MGDA. MGDA can be further extended to incorporate RVs (e.g., EPO [34, 35], ParetoMTL [28]). Some algorithms (e.g., MOO-SVGD [31], HIGA [46]) can output a solution set without using RVs. However, they optimize several neural networks simultaneously, and so are computationally expensive and need large GPU memory (especially when the neural network is large).

3 Proposed Algorithm

As mentioned in Section 1, the fixed uniformly distributed RVs used in common practice may result in undesirable solution distributions. Instead of using a fixed set of n RVs ($R = [r_1, \dots, r_n] \in \mathcal{R}$), we propose to adapt them so that the resultant solution set is well-distributed. In Section 3.1, we first introduce a reference vector-conditioned neural network so that reference vectors can be easily handled without using a lot more parameters. Section 3.2 then formulates reference vector adaptation as a bilevel optimization problem, and an efficient solver is proposed in Section 3.3. Its convergence properties are then studied in Section 3.4

3.1 Reference Vector-Conditioned Neural Network

In deep learning models, ϕ corresponds to the network parameters and optimizing $\Phi = \{\phi_1, \dots, \phi_n\}$ means optimizing n neural networks (as in EPO [34] and MOO-SVGD [31]), which is highly inefficient. To alleviate this problem in deep MOO algorithms, Navon et al. [37] and Lin et al. [27] propose to train a single hypernetwork [20] that can output neural network parameters based on the RV. In particular, the Pareto Hypernetwork (PHN) in [37] proposes two ways to optimize the hypernetwork, leading to (i) PHN-LS, which uses linear scalarization, and (ii) PHN-EPO, which uses EPO. However, the hypernetwork incurs significant computational overhead. For example, the hypernetwork in [37] is around 100 times larger than the base neural network. In this paper, we use the more efficient conditioned network [14, 41]. Specifically, Ruchte and Grabocka [41] concatenates the RV and data sample, and treat this as a joint input to the network. On the other hand, YOTO [14] is originally developed for use with a family of parameterized loss functions, in which the loss parameter is incorporated with the sample into the network via FilM layers [39]. It is theoretically shown that YOTO is as powerful as using n neural networks [14]. In this work, we adapt the architecture in [14] by replacing the loss parameter with the RV. In this way, the proposed model only has a small parameter overhead compared to the single deep network.

3.2 Reference Vector Adaption via Bilevel Optimization

Consider a RV-conditioned neural network $f(\phi, r)$ with parameter ϕ and RV r as input. With a set of RVs $R = [r_1, \dots, r_n]$, the multi-objective values of a solution set can be written as $[f(\phi, r_1), \dots, f(\phi, r_n)]$. We use a function $\hat{Q}(\cdot)$ to measure its quality. Two choices are considered in this paper. The first one encourages the $f(\phi, r_i)$'s are far away from each other (and thus

more uniformly distributed in the space of objectives):

$$\hat{Q}(R,\phi) = -\sum_{i,j=1}^{n} \exp\left(-\frac{1}{h^2}\|f(\phi,r_i) - f(\phi,r_j)\|^2\right), \tag{6}$$

where h is a constant. The second one is:

$$\hat{Q}(R,\phi) = HV(\{f(\phi,r_1),\ldots,f(\phi,r_n)\};z), \tag{7}$$

which encourages the maximization of HV in (7). Note that the HV-optimal solution is usually not uniformly distributed [19, 44].

To obtain the set of RVs R that generates ϕ, we formulate it as a bilevel optimization problem [12]. Recently, bilevel optimization has gained great popularity in many machine learning problems such as meta-learning [16], neural architecture search [29], and hyperparameter optimization [17]. We consider the following bilevel optimization problem:

$$\min_{R\in\mathcal{R}} \quad Q(R,\phi^*(R)) \tag{8}$$

$$\text{s.t.} \quad \phi^*(R) = \arg\min_{\phi} S(R,\phi), \tag{9}$$

where $Q(R,\phi^*(R)) \equiv -\hat{Q}(R,\phi^*(R))$, $S(R,\phi) \equiv \sum_{i=1}^{n} s(\phi;r_i)$, and $s(\phi;r)$ is the scalarization function in (3) or (4). The lower-level optimization problem (9) obtains the Pareto-optimal solutions with the given set of RVs R, while the upper-level optimization problem (8) optimizes R to maximize the corresponding solution quality.

The idea of RV adaption is also used in some evolutionary algorithms [40, 26]. However, they update the RVs using information from the current population and archive, and cannot be directly used in gradient-based MOO algorithms. Moreover, they do not have any theoretical guarantees.

3.3 Solving the Bilevel Optimization Problem

There are various bilevel optimization solvers for (8). Many of them involve propagation through the inner loop [16, 18], which has a large computational overhead compared to [41]. Here, we propose an inexpensive solver that still has theoretical guarantees (Section 3.4). Essentially, it performs only one stochastic gradient descent step in both the inner and outer loops.

The proposed procedure, shown in Algorithm 1, is called Gradient-based Multi-Objective Optimization algorithm with Adaptive Reference vectors (GMOOAR). In each iteration k, mini-batches ξ_k and π_k are randomly sampled from the data and then used to estimate the stochastic gradients $\nabla_\phi S(R_k,\phi_k;\xi_k)$ and $\nabla_R Q(R_k,\phi_{k+1};\pi_k)$. $\text{proj}_{\mathcal{R}}(\cdot)$ is the Euclidean projection operator onto $\mathcal{R}$ that ensures that RVs are inside $\mathcal{R}$. In the sequel, the algorithm using uniformity-related quality function (6) is denoted GMOOAR-U, while the one using (7) is denoted GMOOAR-HV.

The proposed algorithm has $O(w+m)$ memory and $O(w+m)$ time complexity per iteration, where w is the dimension of ϕ. Since $m \ll w$ in most cases, the proposed algorithm has comparable time and space complexity with COSMOS, which is $O(w)$.

Algorithm 1 GMOOAR

Input: learnable RVs R, learning rates $\{\alpha_k,\beta_k\}$, initial parameter ϕ, number of iterations K.

for $k=1$ to K **do**
  sample a mini-batch ξ_k of samples;
  $\phi_{k+1} \leftarrow \phi_k - \beta_k \nabla_\phi S(R_k,\phi_k;\xi_k)$; /* optimize network parameters */
  sample a mini-batch π_k of samples;
  $R_{k+1} \leftarrow \text{proj}_{\mathcal{R}}(R_k - \alpha_k \nabla_R Q(R_k,\phi_{k+1};\pi_k))$; /* optimize reference vectors */
end for

3.4 Convergence

In this section, we provide convergence analysis for Algorithm 1. As in [21], we make the following assumptions on $S(R,\phi)$ and $Q(R,\phi)$.

Assumption 1. *(i) $S(R,\phi)$ is twice-differentiable in (R,ϕ). (ii) $\nabla_\phi S(R,\phi)$, $\nabla^2_{R\phi}S(R,\phi)$, $\nabla^2_{\phi\phi}S(R,\phi)$, $\nabla_R Q(R,\phi)$, and $\nabla_\phi Q(R,\phi)$ are Lipschitz continuous w.r.t. ϕ with constants $L_s, L_{s,1}$, $L_{s,2}$, $L_{q,1}$, and $L_{q,2}$. (iii) $\nabla^2_{R\phi}S(R,\phi)$, $\nabla^2_{\phi\phi}S(R,\phi)$, and $\nabla_\phi Q(R,\phi)$ are Lipschitz continuous w.r.t. R with constants $L_{s,3}$, and $L_{s,4}$, and $L_{q,3}$. (iv) $S(R,\phi)$ are μ_s-strongly convex in ϕ. (v) $\|\nabla^2_{R\phi}S(R,\phi)\| \le C_s$ and $\|\nabla_\phi Q(R,\phi)\| \le C_q$.*

Given $\phi^*(R)$, the gradient of upper-level objective $u(R) \equiv Q(R,\phi^*(R))$ can be obtained as

$$\nabla u(R) = \nabla_R Q(R,\phi^*(R)) - \nabla^2_{R\phi}S(R,\phi^*(R))[\nabla^2_{\phi\phi}S(R,\phi^*(R))]^{-1}\nabla_R Q(R,\phi^*(R)).$$

In [21], it is shown that the outer gradient is the stochastic estimate of

$$\overline{\nabla}_R Q(R_k,\phi_{k+1}) \equiv \nabla_R Q(R_k,\phi_{k+1}) - \nabla^2_{R\phi}S(R_k,\phi_{k+1})[\nabla^2_{\phi\phi}S(R_k,\phi_{k+1})]^{-1}\nabla_R Q(R_k,\phi_{k+1}). \tag{10}$$

However, this involves computing the Hessian and is expensive. The proposed algorithm uses $h_q^k \equiv \nabla_R Q(R_k,\phi_{k+1};\pi_k)$, which is the stochastic estimate of $\nabla_R Q(R_k,\Phi_{k+1})$. Note that $\nabla_R Q(R_k,\Phi_{k+1})$ is the first-order approximation of the outer gradient in (10). This approximation greatly reduces the time and memory complexities, but leads to a bias that can be bounded by a constant.

$$\|\overline{\nabla}_R Q(R,\phi) - \mathbb{E}_\pi[h_q^k]\| = \|\nabla^2_{R\phi}S(R,\phi)[\nabla^2_{\phi\phi}S(R,\phi)]^{-1}\nabla_R Q(R_k,\phi)\| \le C_s C_q/\mu_s. \tag{11}$$

Next, we also make the following assumption similar to [21]. Let $h_s^k \equiv \nabla_\phi S(R_k,\phi_k;\xi_k)$.

Assumption 2. *For any $k \ge 0$, there exist constants σ_s, σ_q, and b_q such that:*

$$\begin{aligned}
&\mathbb{E}_\xi[h_s^k] = \nabla_\phi S(R_k,\phi_k),\ \ \mathbb{E}_\pi[h_q^k] = \overline{\nabla}_R Q(R_k,\phi_{k+1}) + B_k,\ \ \|B_k\| \le b_q,\\
&\mathbb{E}_\xi[\|h_s^k - \nabla_\phi S(R_k,\phi_k)\|^2] \le \sigma_s^2 \cdot \{1 + \|\nabla_\phi S(R_k,\phi_k)\|^2\},\\
&\mathbb{E}_\pi[\|h_q^k - B_k - \overline{\nabla} Q(R_k,\phi_{k+1})\|^2] \le \sigma_q^2.
\end{aligned}$$

Denote the expected gap between ϕ_k and the optimal network parameter given reference vectors R_{k-1} by $\Delta_\phi^k \equiv \mathbb{E}_\xi[\|\phi_k - \phi^*(R_{k-1})\|]$. Similarly, denote the gap between R_k and the optimal reference vectors R^* in (8) by $\Delta_R^k \equiv \mathbb{E}_\pi[\|R_k - R^*\|]$.

Theorem 1. *With Assumptions 1–2, assume that $u(R)$ is μ_q-strongly convex, and the step sizes (α_k,β_k) satisfy*

$$\alpha_k \le \mathrm{c}_0\,\beta_k^{3/2},\ \beta_k \le \mathrm{c}_1\alpha_k^{2/3},\ \frac{\beta_{k-1}}{\beta_k} \le 1 + \beta_k\mu_s/8,\ \frac{\alpha_{k-1}}{\alpha_k} \le 1 + 3\alpha_k\mu_q/4, \tag{12a}$$

$$\alpha_k \le \frac{1}{\mu_q},\ \beta_k \le \min\left\{\frac{1}{\mu_s}, \frac{\mu_s}{L_s^2(1+\sigma_s^2)}, \frac{\mu_s^2}{48\mathrm{c}_0^2L^2L_q^2}\right\},\ 8\mu_q\alpha_k \le \mu_s\beta_k,\ \forall\, k \ge 0, \tag{12b}$$

where L, L_q are constants and $\mathrm{c}_0, \mathrm{c}_1 > 0$ are free parameters. For any $k \ge 1$, the iterates generated by Algorithm 1 satisfy

$$\begin{aligned}
\Delta_R^k &\lesssim \left[\prod_{i=0}^{k-1}(1-\alpha_i\mu_q)\right]\left[\Delta_R^0 + \frac{L^2}{\mu_q^2}\Delta_\phi^0\right] + \frac{\mathrm{c}_1L^2}{\mu_q^2}\left[\frac{\sigma_s^2}{\mu_s} + \frac{\mathrm{c}_0^2L_q^2}{\mu_s^2}\tilde{\sigma}_q^2\right]\alpha_{k-1}^{2/3} + \frac{b_q^2}{\mu_q^2},\\
\Delta_\phi^k &\lesssim \left[\prod_{i=0}^{k-1}(1-\beta_i\mu_s/4)\right]\Delta_\phi^0 + \left[\frac{\sigma_s^2}{\mu_s} + \frac{\mathrm{c}_0^2L_q^2}{\mu_s^2}\tilde{\sigma_q}^2\right]\beta_{k-1},
\end{aligned}$$

where $\lesssim$ denotes that numerical constants are omitted.

With diminishing step sizes $\alpha_k = c_\alpha/(k+k_\alpha)$, $\beta_k = c_\beta/(k+k_\beta)^{2/3}$, where

$$k_\alpha = \max\left\{35\Big(\frac{L_s}{\mu_s}\Big)^3(1+\sigma_s^2)^{\frac{3}{2}}, \frac{(512)^{\frac{3}{2}}L^2L_q^2}{\mu_q^2}\right\},\ c_\alpha = \frac{8}{3\mu_q},\ k_\beta = \frac{1}{4}k_\alpha,\ c_\beta = \frac{32}{3\mu_s},$$

the following Corollary shows that the assumptions (12a) and (12b) in Theorem 1 are satified, and thus ϕ_k converges to the optimal solution $\phi^*(R_{k-1})$.

Corollary 1. *ϕ_k converges to the optimal $\phi^*(R_{k-1})$ of (9).*

Since the minimizer of linear scalarization function is Pareto-optimal for the original multi-objective optimization problem [5], ϕ_k converges to a Pareto-optimal solution.

4 Experiments

In this section, extensive experiments are performed, including synthetic problems (Section 4.1), multi-task learning (Section 4.2), accuracy-fairness trade-off (Section 4.3), and usage on larger networks (Section 4.4). Finally, ablation study is presented in Section 4.5.

4.1 Synthetic Problems

Experiments are performed on four commonly used multi-objective benchmark problems [10, 49, 31] with different numbers of objectives: (i) 2-objective DTLZ2 [11], (ii) 3-objective DTLZ2 [11], (iii) 2-objective scaled-DTLZ2 [11], and (iv) 3-objective MaF1 [7]. Their detailed definitions are in Appendix B.1. The PFs of problems (i) and (ii) are regular, while those of (iii) and (iv) are irregular. The number of inputs is set to 30. We aim to get 15 non-dominated solutions for each 2-objective problem, and 36 non-dominated solutions for each 3-objective problem.

The proposed algorithm[1] (GMOOAR-U using quality function (6) and GMOOAR-HV using quality function (7)) is compared with the state-of-the-art COSMOS [41],which uses fixed reference vectors. For the 2-objective problems, reference vectors for COSMOS are generated by following their strategy in [41]. As 3-objective problems are not considered in [41], we generate reference vectors for COSMOS by the method in [9]. For GMOOAR, the reference vectors are initialized randomly. As in [37], a neural network (with 2 hidden layers, each with 20 units) is used. More experimental details can be found in Appendix B.2.

Figure 3 shows the solution sets obtained by GMOOAR-U on 3-objective DTLZ2 when the region of interest is a subspace $\mathcal{R}$ constrained as in (5). Note that this constraint can be difficult for COSMOS as it is unclear how the pre-specified RVs can be uniformly generated in $\mathcal{R}$. Moreover, even with a set of uniformly distributed RVs, they may not lead to a uniformly distributed set of solutions.

4.2 Multi-Task Learning

In this experiment, we use three benchmark datasets from [28]: Multi-MNIST, Multi-Fashion, and Multi-Fashion+MNIST. In Multi-MNIST, each image is constructed by putting two different MNIST images together, one at the bottom-right (BR) and the other at the top-left (TL). Similarly, Multi-Fashion images are constructed by combining images from FashionMNIST [47], while Multi-Fashion+MNIST images are constructed by combining one MNIST image with one FashionMNIST image. More details can be found in [28]. The goal is to classify both the BR and TL images correctly, by minimizing the two cross-entropy losses using a single neural network. As in [28, 34, 41], we use the LeNet [25] with multi-head as base network.

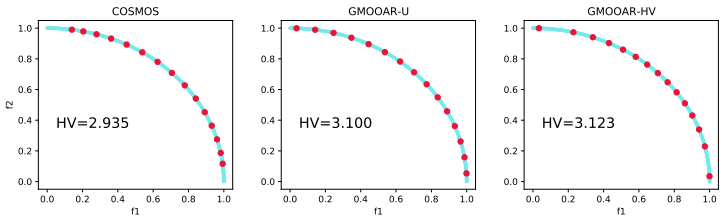
We compare the proposed algorithms (GMOOAR-U and GMOOAR-HV) with (i) EPO [34], (ii) Pareto hypernetworks (PHN-LS and PHN-EPO) [37], (iii) MOO-SVGD [31], and (iv) COSMOS [41]. For EPO, PHN-LS, PHN-EPO and COSMOS, we generate reference vectors following the strategy in [41]. For GMOOAR, the reference vectors are initialized randomly. The experiment is repeated 10 times with different random seeds.

Following common practice [41], we obtain a set of n solutions[2] in each iteration. They are evaluated on the validation set every 5 epochs. We only keep the solutions of iteration k_{best} as the final solution set, where k_{best} is the iteration that yields the solution set with the largest validation HV. Note that the original implementation of MOO-SVGD (obtained from the authors) stores all non-dominated solutions of each iteration in an archive $\mathcal{A}$. On termination, they try all size-n subsets of $\mathcal{A}$ and select the subset with the largest HV on the validation set as the final solution set. As there are $C_n^{|\mathcal{A}|}$ such subsets and $|\mathcal{A}|$ is large when MOO-SVGD terminates, this can be very expensive. In order to be fair to all algorithms being compared, we thus also use the aforementioned commonly practiced strategy on MOO-SVGD.

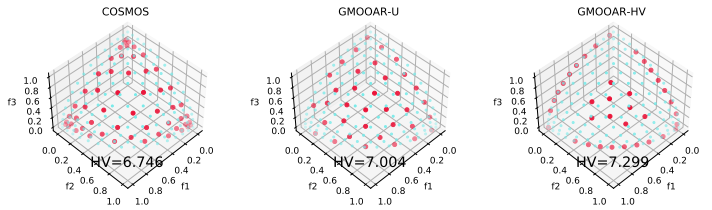
Figure 4 shows the test losses obtained by the solution set with median HV over the 10 runs. As can be seen, on Multi-MNIST and Multi-Fashion, the solution sets obtained by COSMOS are dense in the middle but sparse towards the ends, while the solution sets obtained by GMOOAR-U are more

[1] We use linear scalarization with penalty term in (4).

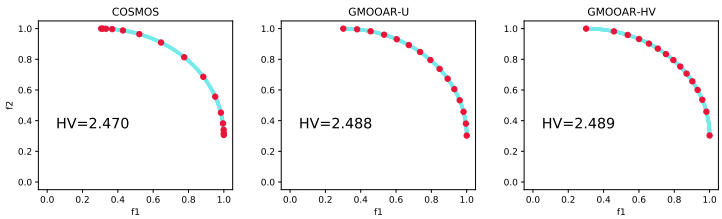
[2] We set n to 15 in all experiments



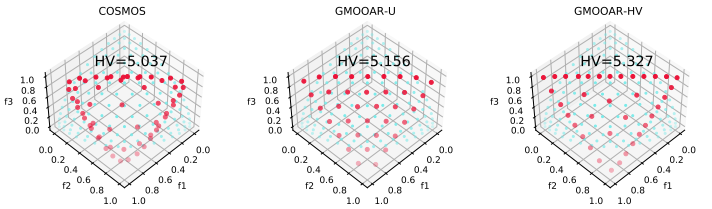
(a) 2-objective DTLZ2.



(b) 3-objective DTLZ2.

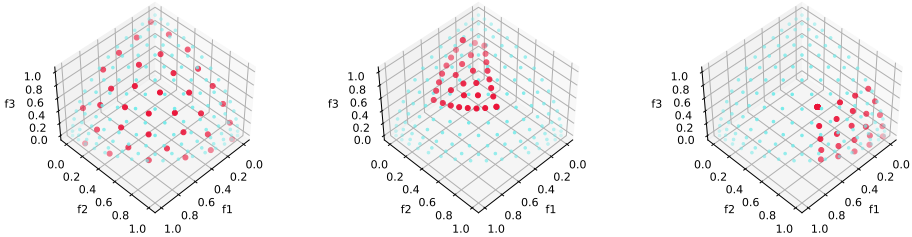


(c) 2-objective scaled-DTLZ2.



(d) 3-objective MaF1.

Figure 2: Solution sets (red) and HV values obtained on the synthetic datasets. The Pareto optimal solutions is in blue.

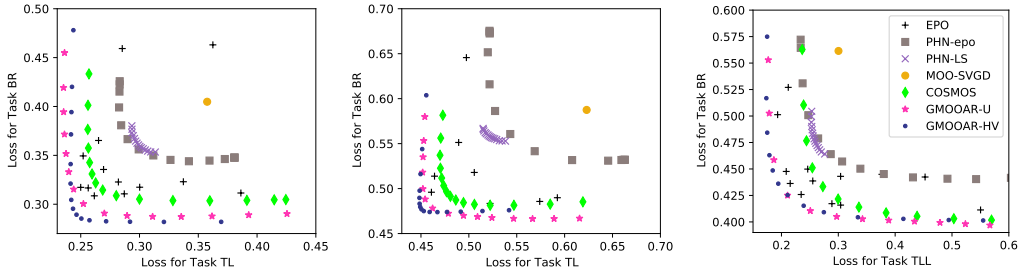


(a) $\varphi_1 = 0, \varphi_2 = \pi/2$. (b) $\varphi_1 = \pi/10, \varphi_2 = 3\pi/10$. (c) $\varphi_1 = 3\pi/10, \varphi_2 = \pi/2$.

Figure 3: Solution sets obtained by GMOOAR-U with different (φ_1, φ_2) settings.

uniform. Moreover, compared to all other baselines, the solution sets obtained by GMOOAR are closer to the bottom-right corner where the underlying true PF resides. For MOO-SVGD, many of its obtained solutions are much inferior, and only one of them is in the range shown in Figure 4. A complete plot of all the MOO-SVGD solution sets is in Appendix C.

Table 1 shows the average HV's of the solution sets over the 10 runs. As can be seen, GMOOAR consistently outperforms all others. Moreover, the number of parameters of GMOOAR is comparable



(a) Multi-MNIST. (b) Multi-Fashion. (c) Multi-Fashion+MNIST.

Figure 4: BR and TL test losses obtained on the real-world multi-task learning datasets.

	HV			# Parameters	Runtime(s)
	Multi-MNIST	Multi-Fashion	Fashion-MNIST		
EPO [34]	2.95±0.02	2.31±0.01	2.86±0.02	478,650	8,370
PHN-EPO [37]	2.82±0.04	2.16±0.05	2.74±0.05	3,243,410	1,515
PHN-LS [37]	2.79±0.04	2.14±0.04	2.67±0.06	3,243,410	644
MOO-SVGD [31]	2.67±0.02	2.02±0.02	2.54±0.04	478,650	8,661
COSMOS [41]	2.95±0.02	2.31±0.03	2.82±0.03	43,058	260
GMOOAR-U	**3.02±0.01**	**2.33±0.10**	2.91±0.02	43,685	288
GMOOAR-HV	**3.02±0.01**	**2.33±0.09**	**2.92±0.02**	43,685	348

Table 1: Average HV and standard deviation obtained on the real-world multi-task learning datasets.

to that of COSMOS, and is much fewer than the other baselines. Compared to the base network, GMOOAR has only 37% more parameters.

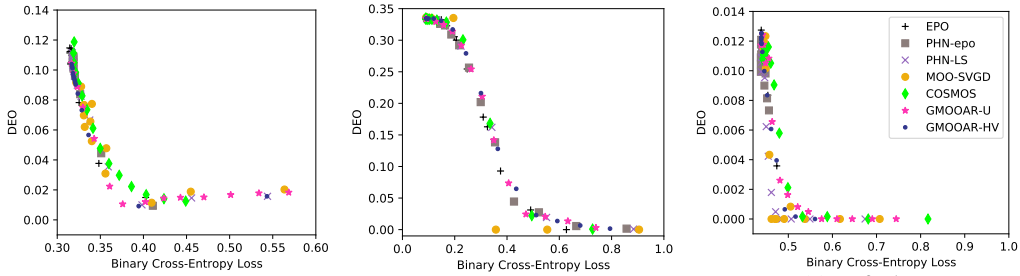
4.3 Accuracy-Fairness Trade-off

In this experiment, we follow [41] and aim to achieve both high accuracy and fairness on three tabular datasets: Adult [15], Compass [1], and Default [48]. The accuracy is measured by the cross-entropy loss, while fairness is measured by a hyperbolic tangent relaxation of the Difference of Equality of Opportunity (DEO) [38]. As in [37, 41], a 2-hidden-layer multilayer perceptron is used as base network. More details can be found in Appendix B.4. The experiment is repeated 10 times.

Figure 5 shows the test loss and fairness measure obtained by the solution set with median HV over the 10 runs. Since the datasets are not difficult, the approximated PFs obtained by various algorithms are close. Solutions obtained by GMOOAR are uniformly distributed, while those obtained by COSMOS, EPO, PHN-LS, and PHN-EPO are very dense in the top-left region. Table 2 shows the HV values of the obtained datasets. It can be seen that GMOOAR achieves better HVs than the baselines.

4.4 Larger Networks

To demonstrate that the proposed method can be used on larger networks, we apply GMOOAR on the EfficientNet-B4 [43] with about 17 million parameters. Following [41], we perform experiments



(a) Adult. (b) Compass. (c) Default.

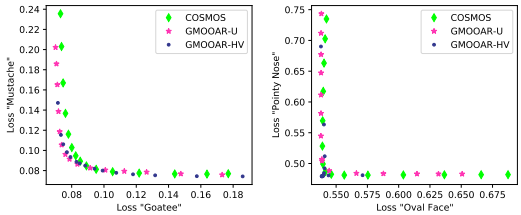
Figure 5: Test losses and fairness measures obtained on the fairness datasets.

	Adult	Compass	Default
EPO[34]	3.342±0.001	3.709±0.002	3.119±0.001
PHN-EPO [37]	3.340±0.006	3.709±0.004	3.111±0.005
PHN-LS [37]	3.341±0.008	3.698±0.007	3.121±0.003
MOO-SVGD [31]	3.330±0.008	3.716±0.011	3.110±0.005
COSMOS [41]	3.336±0.006	3.710±0.004	3.114±0.005
GMOOAR-U	3.344±0.004	**3.719±0.008**	**3.123±0.004**
GMOOAR-HV	**3.345±0.005**	3.714±0.008	**3.123±0.002**

Table 2: Average HV and standard deviation on the fairness datasets.

on two easy tasks ("Goatee" and "Mustache") and two hard tasks ("Oval Face" and "Pointy Nose") from the 32 tasks in CelebA [32]. Since EfficientNet-B4 is around 400 times larger than the LeNet used in previous experiments, PHN-LS, PHN-EPO, and MOO-SVGD cannot be run on our machine (with RTX-2080Ti and 11GB memory). For performance evaluation, the testing cross-entropy loss of each selected task is used. The experiment is repeated 5 times with different random seeds.

Figure 6 shows the test losses obtained by the solution set with median HV over the 5 runs. The corresponding HV values are shown in Table 7. On CelebA-Easy, both GMOOAR-U and GMOOAR-HV outperform COSMOS in terms of HV. On CelebA-Hard, all three algorithms achieve similar HVs, though that of GMOOAR-HV is slightly better.



(a) CelebA-Easy. (b) CelebA-Hard.

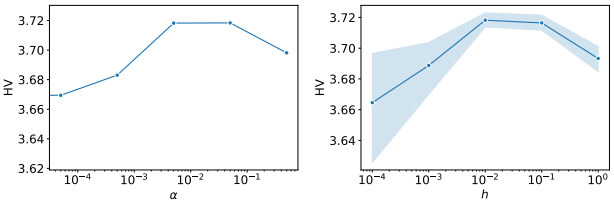
Figure 6: Test losses on the two easy tasks (left) and two hard tasks (right) of CelebA.

	CelebA-Easy	CelebA-Hard
COSMOS [41]	3.700±0.005	2.217±0.002
GMOOAR-U	3.710±0.005	2.217±0.002
GMOOAR-HV	**3.711±0.005**	**2.222±0.006**

Figure 7: Average HV and standard deviation of solution sets obtained on CelebA.

4.5 Ablation Study

In this experiment, we study the effect of RV learning rate α and bandwidth h in (6) on the performance of GMOOAR-U. We use the same setting as in Section 4.3. The experiment is repeated 10 times with different random seeds.



(a) RV learning rate α (b) bandwith h

Figure 8: Averge HV and 95% confidence iterval with different α's and h's on GMOOAR-U using the Compass dataset.

Figure 8a shows the variations of HV with α (h is fixed to 0.01). As can be seen, α too small results in almost no RV adaption and thus poor performance, while α too large may lead to unstable learning. Figure 8b shows the variation of HV with h (α is fixed to 0.005). When h is too small (resp. too large), $Q(R, \phi)$ is close to zero (resp. n) and the gradient vanishes, making learning difficult.

5 Conclusion

In this paper, we present a novel gradient-based MOO algorithm with adaptive RVs. The proposed algorithm can efficiently adapt the RVs during optimization and provide diverse solutions with a small overhead compared to single-objective optimization. Experiments show the ability of the proposed strategies to obtain well-distributed solutions based on the specified quality function. Incorporating other state-of-the-art MOO algorithms into our model can be an interesting direction.