Pareto Multi-Task Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Multi-task learning is a powerful method for solving multiple correlated tasks simultaneously. However, it is often difficult to combine all the tasks together for finding a single best solution, since different tasks might conflict with each other. Recently, a novel method is proposed to find one solution with good trade-off among different tasks by formulating multi-task learning as multi-objective optimization. In this paper, we generalize this idea and propose a novel Pareto multi-task learning algorithm (Pareto MTL) to find a set of widely distributed Pareto solutions which can represent different trade-offs among different tasks. The proposed algorithm first formulates multi-task learning as a multi-objective optimization problem, and then decomposes it into a set of constrained multi-objective subproblems with different trade-off preferences. By solving these multi-objective subproblems in parallel, Pareto MTL can provide a set of well-representative Pareto solutions with different trade-offs among all tasks. Practitioners can easily select their preferred solution based on all Pareto solutions' performance, or use different trade-off solutions for different situations. Experimental results confirm that the proposed algorithm can generate well-representative solutions and outperform some state-of-the-art algorithms on many multi-task learning applications.

1 Introduction

Multi-task learning (MTL) [1], which aims at learning multiple correlated tasks at the same time, is a popular research topic in the machine learning community. By solving multiple relative tasks together, MTL can further improve the performance of each task and reduce the inference time for conducting all the tasks in many real-world applications. Many MTL approaches have been proposed in the current years, and they achieve great performances in a broad domain, such as in computer vision [2], neural language processing [3] and speech recognition [4].

Most MTL approaches focus on finding a single best solution to improve the overall performance of all tasks [5, 6]. However, it is observed in many applications that some tasks could be conflicted with each other, and no single optimal solution can optimize the performance of all tasks at the same time [7]. In real-world applications, MTL practitioners usually have to make a trade-off among different tasks, such as in self-driving car [8], AI assistance [9] and network architecture search [10, 11].

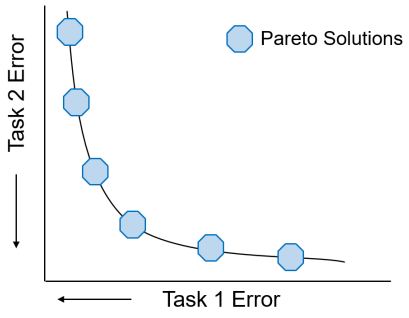


Figure 1: **Pareto MTL** can find a set of widely distributed Pareto solutions with different trade-offs for a given MTL. Then the practitioners can easily select their preferred solution(s).

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

How to combine different tasks together and make a proper trade-off among them is a difficult problem. In a large number of MTL applications, especially those using deep multi-task neural network, all tasks are first combined into a single surrogate task through linear weighted scalarization. A set of fixed weights, which reflects the practitioner's preference or merely balances all tasks, is assigned to each task. Then all tasks are optimized together by minimizing the single surrogate task. Setting proper weights for different tasks is not easy and usually requires exhaustive weights search. In fact, no combination can provide one single best solution if the tasks conflict with each other.

Recently, Sener and Koltun [12] formulate multi-task learning as a multi-objective optimization problem in a novel way. They propose an efficient algorithm to find one Pareto solution with optimal trade-off among different tasks for a MTL problem. However, the MTL problem might have different (even infinite) optimal trade-offs among its tasks, and the single solution obtained by this method might not satisfy the MTL practitioner's need.

In this paper, we generalize the multi-objective optimization idea [12] and propose a novel Pareto Multi-Task Learning (Pareto MTL) algorithm to provide a set of well representative Pareto solutions for a given MTL problem. As shown in Fig. 1, MTL practitioners can easily select their preferred solution(s) among the set of obtained Pareto solutions with different optimal trade-offs, rather than exhaustively searching for a set of proper weights for all tasks.

The main contributions of this paper are: [1]

- We propose a novel method to decompose a MTL problem into multiple subproblems with different preferences. By solving these subproblems in parallel, we can obtain a set of widely distributed Pareto solutions with different trade-offs for the original MTL.
- We show that the proposed Pareto MTL can be reformulated as a linear scalarization approach to solve MTL with dynamically adaptive weights. We also propose a scalable optimization algorithm to solve all constrained subproblems with different preferences.
- Experimental results confirm that the proposed Pareto MTL algorithm can successfully find a set of well representative solutions for different MTL applications.

2 Related Work

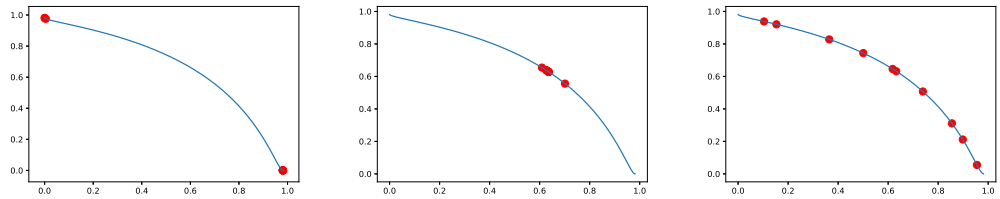
Multi-task learning (MTL) aims at improving the performance of multiple related tasks by learning them at the same time. Compared with the single task counterpart, MTL algorithms usually construct shared parameters representation to combine multiple tasks together. They achieve promising performance in many machine learning applications. However, most MTL algorithms mainly focus on building efficient architectures rather than making trade-offs among multiple tasks [5, 6].

Linear tasks scalarization, together with grid search or random search of the weight vectors, is the current default practice when a MTL practitioner wants to obtain a set of different trade-off solutions. This approach is straightforward but could be extremely inefficient. Some recent works [7, 13] show that a single run of an algorithm with well-designed weights adaption strategy can outperform the random search approach with more than one hundred runs. These adaptive weight methods focus on balancing different tasks during the optimization process and are not suitable for finding solutions with different trade-offs.

Multi-objective optimization [14] aims at finding a set of Pareto solutions with different trade-offs rather than one single solution. This method has been used in many machine learning applications such as reinforcement learning [15], Bayesian optimization [16, 17, 18] and Neural Architecture search [10, 19]. In these applications, the gradient information is usually not available. The population-based and gradient-free multi-objective evolutionary algorithm [20, 21] is a popular method to find a set of widely distributed Pareto solutions in a single run. However, this method can not be used for solving large scale and gradient-based MTL.

Multi-objective gradient descent [22, 23, 24] provides a good way for gradient-based multi-objective optimization. Currently, Sener and Koltun [12] propose a novel method for solving MTL by treating it as a multi-objective optimization problem. However, similar to the adaptive weight methods, this method can not efficiently incorporate preference information, and can not generate a set of well-representative solutions. In this paper, we generalize [12] for further preference selection.

[1] We provide example codes, more experimental results and discussions in the supplementary material.



(a) Random Linear Scalarization (b) MOO MTL (c) Pareto MTL (Ours)

Figure 2: The convergence behaviors of different algorithms on a toy example. (a) The obtained solutions of random linear scalarization after 100 runs. (b) The obtained solutions of the MOO-MTL [12] method after 10 runs. (c) The obtained solutions of the Pareto MTL method proposed by this paper after 10 runs. The proposed Pareto MTL successfully generates a set of widely distributed Pareto solutions with different trade-offs. More results can be found in the supplementary material.

3 Multi-Task Learning as Multi-Objective Optimization

3.1 MTL as Multi-Objective Optimization

Generally, a MTL problem involves solving a set of m correlated tasks with a loss vector:

$$\min \mathcal{L}(\theta) = (\mathcal{L}_1(\theta), \mathcal{L}_2(\theta), \cdots, \mathcal{L}_m(\theta))^{\mathrm{T}}, \tag{1}$$

where $\mathcal{L}_i(\theta)$ is the loss of the i-th task. A MTL algorithm usually aims at optimizing all tasks simultaneously by exploiting the shared structure and information among them.

The problem (1) is a multi-objective optimization problem. For non-trivial cases, however, no single solution can optimize all objectives at the same time. What we can obtain instead is an optimal set of solutions, called Pareto set, which provides different optimal trade-offs among all objectives. For a minimization problem, we have the following definitions [25]:

Pareto dominance. Let θ^a, θ^b be two points in Ω, θ^a is said to dominate θ^b ($\theta^a \prec \theta^b$) if and only if $\mathcal{L}_i(\theta^a) \leq \mathcal{L}_i(\theta^b), \forall i \in \{1, ..., m\}$ and $\mathcal{L}_j(\theta^a) < \mathcal{L}_j(\theta^b), \exists j \in \{1, ..., m\}$.

Pareto optimality. θ^* is a Pareto optimal point and $\mathcal{L}(\theta^*)$ is a Pareto optimal objective vector if it does not exist $\hat{\theta} \in \Omega$ such that $\hat{\theta} \prec \theta^*$. The subset of all Pareto optimal points is called the Pareto set.

In this paper, we focus on finding a set of well-representative Pareto solutions which can well approximate the Pareto front. This idea and the results of three methods are presented in Fig. 2.

3.2 Linear Scalarization

Linear scalarization is the most common approach for solving multi-task learning as well as multi-objective optimization problem. This approach simply uses a linear weighted sum method to combine the losses of all tasks into a single surrogate loss:

$$\min \mathcal{L}(\theta) = \sum_{i=1}^{m} \boldsymbol{w}_i \mathcal{L}_i(\theta), \tag{2}$$

where $\boldsymbol{w}_i$ is the weight for the i-th task. Although this approach is simple and straightforward, it suffers many drawbacks from both multi-task learning and multi-objective optimization perspective.

In a typical multi-task learning application, the weights $\boldsymbol{w}_i$ are needed to be assigned manually before optimization, and the overall performance is highly depended on the assigned weights. Assigning a proper weight could be very difficult even for a senior MTL practitioner who has expertise on the given problem. Some current work also shows that exhaustive weights search can be easily outperformed by a single run of adaptive weight method [7, 13].

Solving a set of linear scalarization problems with different weight assignments is also not a good choice for multi-objective optimization problem. As pointed out in [26], this method can only provide solutions on the convex part of the Pareto front. The linear scalarization method with different weight assignments fails to locate the most part of a concave Pareto front as shown in Fig. 2.

3.3 Gradient-based method for multi-objective optimization

Many gradient-based multi-objective optimization methods have been proposed [22, 23]. [24] has proposed a simple gradient-based method as a general form of single objective steepest descent algorithm. The update rule of the algorithm is $\theta_{t+1} = \theta_t + \eta d_t$ where η is the step size and the search direction can be obtained as:

$$(d_t, \alpha_t) = \arg \min_{d \in R^n, \alpha \in R} \alpha + \frac{1}{2}||d||^2, s.t. \quad \nabla \mathcal{L}_i(\theta_t)^T d \leq \alpha, i = 1, ..., m. \tag{3}$$

It can be proved that the solution of the above subproblem is closely related to Pareto critical:

Lemma 1 [24]: Let (d^k, α^k) be the solution of problem (3).

1. If θ_t is Pareto critical, then $d_t = 0 \in \mathbb{R}^n$ and $\alpha_t = 0$.
2. If θ_t is not Pareto critical, then

$$\begin{aligned} &\alpha_t \leq -(1/2)||d_t||^2 < 0, \\ &\nabla \mathcal{L}_i(\theta_t)^T d_t \leq \alpha_t, i = 1, ..., m. \end{aligned} \tag{4}$$

In other words, by solving problem (3), we either obtain $d_t \neq \mathbf{0}$ as a descent direction for the multi-objective problem (1), or obtain $d_t = \mathbf{0}$ and confirm that θ_t is a Pareto critical point.

A recent work [12] uses the multiple gradient descent algorithm (MGDA) [22] for solving MTL problems and achieves promising results. However, as shown in Fig. 2, this method can not incorporate preference information, and running the algorithm multiple times can not provide a set of well-representative solutions on the Pareto front. In this paper, we generalize this method and propose a novel Pareto MTL algorithm to find a set of Pareto solutions with different trade-offs among all tasks.

4 Pareto Multi-Task Learning

4.1 MTL Decomposition

The main idea of Pareto MTL is to decompose the multi-objective optimization problem (MOP) into several constrained multi-objective subproblems with different trade-off preferences among the tasks in the original MTL. By solving these subproblems in parallel, a MTL practitioner can obtain a set of well-representative solutions with different trade-offs.

Decomposition-based multi-objective evolutionary algorithm [27, 28], which decomposes a MOP into several subproblems and solves them simultaneously, is one of the most popular methods in multi-objective optimization. However, most decomposition-based methods are only suitable for solving small scale gradient-free problem. Our proposed Pareto MTL algorithm generalizes the decomposition idea to solve gradient-based MTL.

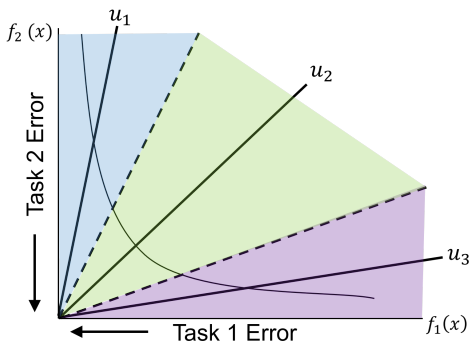


Figure 3: **MTL Decomposition:** Pareto MTL decomposes a given MTL problem into several subproblems with a set of preference vectors. Each subproblem aims at finding one Pareto solution in the restricted region corresponding to its assigned preference vector.

We adopt the idea from [29] and decompose the MTL into K subproblems with a set of widely distributed unit preference vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_K\}$ in R^m_+. Suppose all objectives in the MOP are non-negative, the multi-objective subproblem corresponding to the preference vector $\boldsymbol{u}_k$ is:

$$\min \mathcal{L}(\theta) = (\mathcal{L}_1(\theta), \mathcal{L}_2(\theta), \cdots, \mathcal{L}_m(\theta))^{\mathrm{T}}, s.t. \quad \mathcal{L}(\theta) \in \Omega_k, \tag{5}$$

where $\Omega_k (k = 1, ..., K)$ is a subregion in the objective space:

$$\Omega_k = \{\boldsymbol{v} \in R^m_+ | \boldsymbol{u}_j^T \boldsymbol{v} \leq \boldsymbol{u}_k^T \boldsymbol{v}, \forall j = 1, ..., K\} \tag{6}$$

and $\boldsymbol{u}_j^T \boldsymbol{v}$ is the inner product between the preference vector $\boldsymbol{u}_j$ and a given vector $\boldsymbol{v}$. That is to say, $\boldsymbol{v} \in \Omega_k$ if and only if $\boldsymbol{v}$ has the smallest acute angle to $\boldsymbol{u}_k$ and hence the largest inner product $\boldsymbol{u}_k^T \boldsymbol{v}$ among all K preference vectors.

The subproblem (5) can be further reformulated as:

$$\begin{aligned} &\min \mathcal{L}(\theta) = (\mathcal{L}_1(\theta), \mathcal{L}_2(\theta), \cdots, \mathcal{L}_m(\theta))^{\mathrm{T}} \\ &s.t. \quad \mathcal{G}_j(\theta_t) = (\boldsymbol{u}_j - \boldsymbol{u}_k)^T \mathcal{L}(\theta_t) \leq 0, \forall j = 1, ..., K, \end{aligned} \tag{7}$$

As shown in Fig. 3, the preference vectors divide the objective space into different sub-regions. The solution for each subproblem would be attracted by the corresponding preference vector and hence be guided to its representative sub-region. The set of solutions for all subproblems would be in different sub-regions and represent different trade-offs among the tasks.

4.2 Gradient-based Method for Solving Subproblems

4.2.1 Finding the Initial Solution

To solve the constrained multi-objective subproblem (5) with a gradient-based method, we need to find an initial solution which is feasible or at least satisfies most constraints. For a randomly generated solution θ_r, one straightforward method is to find a feasible initial solution θ_0 which satisfies:

$$\min ||\theta_0 - \theta_r||^2 \quad s.t. \quad \mathcal{L}(\theta_0) \in \Omega_k. \tag{8}$$

However, this approach is inefficient even in low dimensional case [30], and is not suitable for optimizing the parameters of a deep neural network. In the proposed Pareto MTL algorithm, we use a sequential gradient-based method to find the initial solution θ_0. For a solution θ_r, we define the index set of activated constraints as $I(\theta_r) = \{j | \mathcal{G}_j(\theta_r) \geq 0, j = 1, ..., K\}$. We can find a valid descent direction d_r to reduce the value of all activated constraints $\{\mathcal{G}_j(\theta_r) | j \in I(\theta_r)\}$ by solving:

$$(d_r, \alpha_r) = \arg \min_{d \in R^n, \alpha \in R} \alpha + \frac{1}{2}||d||^2, s.t. \nabla \mathcal{G}_j(\theta_r)^T d \leq \alpha, j \in I(\theta_r). \tag{9}$$

This approach is similar to the unconstrained gradient-based method (3) but now has the activated constraints as objectives to be minimized. The gradient-based update rule is $\theta_{r_{t+1}} = \theta_{r_t} + \eta_r d_{r_t}$ and will be stopped once a feasible solution is found (all constraints are inactivated) or a predefined number of iterations is met.

4.2.2 Solving the Subproblem

Once we have the initial solution, we can use gradient-based method to solve the constrained subproblem. For a constrained multiobjective optimization problem, the Pareto optimality restricted on Ω_k can be defined as [24]:

Restricted Pareto Optimality. θ^* is a Pareto optimal point for $\mathcal{L}(\theta)$ restricted on Ω_k if $\theta^* \in \Omega_k$ and it does not exist $\hat{\theta} \in \Omega_k$ such that $\hat{\theta} \prec \theta^*$.

According to [24, 30], we can find a descent direction for this constrained MOP by solving a subproblem similar to the subproblem (3) for the unconstrained case:

$$\begin{aligned} (d_t, \alpha_t) = &\arg \min_{d \in R^n, \alpha \in R} \alpha + \frac{1}{2}||d||^2 \\ &s.t. \quad \nabla \mathcal{L}_i(\theta_t)^T d \leq \alpha, i = 1, ..., m. \\ &\nabla \mathcal{G}_j(\theta_t)^T d \leq \alpha, j \in I_\epsilon(\theta_t), \end{aligned} \tag{10}$$

where $I_\epsilon(\theta)$ is the index set of activated constraints:

$$I_\epsilon(\theta) = \{j \in I | \mathcal{G}_j(\theta) \geq -\epsilon\} \tag{11}$$

Like the unconstrained case, for a feasible solution θ_t, by solving problem (10), we either obtain $d_t \neq \mathbf{0}$ as a descent direction for the constrained multi-objective problem (7), or obtain $d_t = \mathbf{0}$ and confirm that θ_t is a Pareto critical point restricted on Ω_k.

Lemma 2 [30]: Let (d^k, α^k) be the solution of problem (10).

1. If θ_t is Pareto critical restricted on Ω_k, then $d_t = 0 \in \mathbb{R}^n$ and $\alpha_t = 0$.
2. If θ_t is not Pareto critical restricted on Ω_k, then

$$\begin{aligned} &\alpha_t \leq -(1/2)||d_t||^2 < 0, \\ &\nabla \mathcal{L}_i(\theta_t)^T d_t \leq \alpha_t, i = 1, ..., m \\ &\nabla \mathcal{G}_j(\theta_t)^T d_t \leq \alpha_t, j \in I_\epsilon(\theta_t). \end{aligned} \tag{12}$$

Therefore, we can obtain a restricted Pareto critical solution for each subproblem with simple gradient-based update rule $\theta_{t+1} = \theta_t + \eta_r d_t$. By solving all subproblems, we can obtain a set of Pareto critical solutions restricted on different sub-regions which can represent different trade-offs among all tasks.

4.2.3 Scalable Optimization Method

By solving the constrained optimization problem (10), we can obtain a descent direction for each multi-objective subproblem. However, the optimization problem itself is not scalable well for high dimensional decision space. For example, when training a deep neural network, we often have more than millions of parameters to be optimized, and solving the constrained optimization problem (10) in this scale would be extremely slow. In this subsection, we propose a scalable optimization method to solve the constrained optimization problem.

Inspired by [24], we first rewrite the optimization problem (10) in its dual form. Based on the KKT conditions, we have

$$d_t = -(\sum_{i=1}^{m} \lambda_i \nabla \mathcal{L}_i(\theta_t) + \sum_{j \in I_{\epsilon(\boldsymbol{\varpi})}} \beta_i \nabla \mathcal{G}_j(\theta_t)), \quad \sum_{i=1}^{m} \lambda_i + \sum_{j \in I_{\epsilon(\theta)}} \beta_j = 1, \tag{13}$$

where $\lambda_i \geq 0$ and $\beta_i \geq 0$ are the Larange multipliers for the linear inequality constraints. Therefore, the dual problem is:

$$\begin{aligned} &\max_{\lambda_i, \beta_j} \ -\frac{1}{2}||\sum_{i=1}^{m} \lambda_i \nabla \mathcal{L}_i(\theta_t) + \sum_{j \in I_{\epsilon(\boldsymbol{\varpi})}} \beta_i \nabla \mathcal{G}_j(\theta_t)||^2 \\ &s.t. \ \ \sum_{i=1}^{m} \lambda_i + \sum_{j \in I_{\epsilon(\theta)}} \beta_j = 1, \lambda_i \geq 0, \beta_j \geq 0, \forall i = 1, ..., m, \forall j \in I_\epsilon(\theta). \end{aligned} \tag{14}$$

For this dual optimization problem (14), the decision space is no longer the parameter space but now the objective space and constraint space. For a MOP with 2 objective function and 5 activated constraints, the dimension of problem (14) is 7, which is significant smaller than problem (10).

The algorithm framework of Pareto MTL is shown in **Algorithm** 1. All subproblems can be solved in parallel since there is no communication between them during the optimization process. The only preference information for each subproblem is the set of preference vectors.

Algorithm 1 Pareto MTL Algorithm

Input: A set of evenly distributed vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_K\}$
Update Rule:
(can be solved in parallel)
for $k = 1$ to K **do**
  randomly generate parameters $\theta_r^{(k)}$
  find the initial parameters $\theta_0^{(k)}$ from $\theta_r^{(k)}$ using gradient-based method
  for $t = 1$ to T **do**
    obtain $\lambda_{ti}^{(k)} \geq 0, \beta_{ti}^{(k)} \geq 0, \forall i = 1, ..., m, \forall j \in I_\epsilon(\theta)$ by solving subproblem (14)
    calculate the direction $d_t^{(k)} = -(\sum_{i=1}^{m} \lambda_{ti}^{(k)} \nabla \mathcal{L}_i(\theta_t^{(k)}) + \sum_{j \in I_{\epsilon(\boldsymbol{\varpi})}} \beta_{ti}^{(k)} \nabla \mathcal{G}_j(\theta_t^{(k)})$
    update the parameters $\theta_{t+1}^{(k)} = \theta_t^{(k)} + \eta d_t^{(k)}$
  end for
end for
Output: The set of solutions for all subproblems with different trade-offs $\{\theta_T^{(k)} | k = 1, \cdot, K\}$

4.3 Pareto MTL as an Adaptive Weighted Loss Approach

We have proposed the Pareto MTL algorithm from the multi-objective optimization perspective. In this subsection, we show that the Pareto MTL algorithm can be reformulated as a linear scalarization of tasks with adaptive weights assignment. In this way, we can have a deeper understanding of the differences between Pareto MTL and other MTL algorithms.

We first tackle the unconstrained case. Suppose we do not decompose the multi-objective problem and hence remove all constraints from the problem (14), it will immediately reduce to the update rule proposed from MGDA [22] which is used in [12]. It is straightforward to rewrite the corresponding MTL into a linear combination form:

$$\mathcal{L}_i(\theta_t) = \sum_{i=1}^{m} \lambda_i \mathcal{L}_i(\theta_t), \tag{15}$$

where we adaptively assign the weights λ_i by solving the following problem in each iteration:

$$\max -\frac{1}{2}||\sum_{i=1}^{m} \lambda_i \nabla \mathcal{L}_i(\theta_t)||^2, \quad s.t. \sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0, \forall i = 1, ..., m. \tag{16}$$

In the constrained case, we have extra constrain terms $\mathcal{G}_j(\theta_t)$. If $\mathcal{G}_j(\theta_t)$ is inactivated, we can ignore it. For an activated $\mathcal{G}_j(\theta_t)$, assuming the corresponding reference vector is $\boldsymbol{u}_k$, we have:

$$\nabla \mathcal{G}_j(\theta_t) = (\boldsymbol{u}_j - \boldsymbol{u}_k)^T \nabla \mathcal{L}(\theta_t) = \sum_{i=1}^{m} (\boldsymbol{u}_{ji} - \boldsymbol{u}_{ki}) \mathcal{L}_\rangle(\theta_t) \tag{17}$$

The general Pareto MTL algorithm can be rewritten as:

$$\mathcal{L}_i(\theta_t) = \sum_{i=1}^{m} \alpha_i \mathcal{L}_i(\theta_t), \text{ where } \alpha_i = \lambda_i + \sum_{j \in I_{\epsilon(\theta)}} \beta_j (\boldsymbol{u}_{ji} - \boldsymbol{u}_{ki}), \tag{18}$$

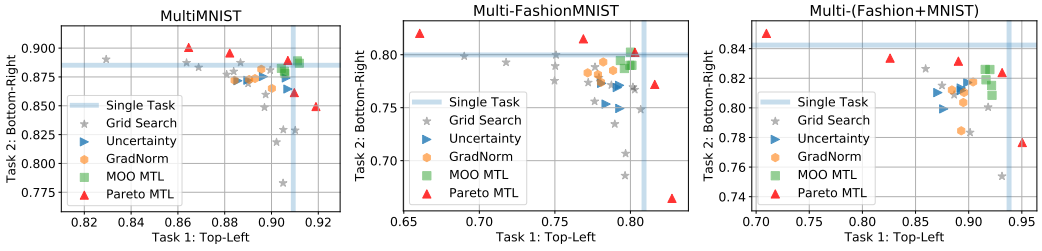
where λ_i and β_j are obtained by solving problem (14) with assigned reference vector $\boldsymbol{u}_k$.

Therefore, although MOO-MTL [12] and Pareto MTL are both derived from multi-objective optimization, they can also be treated as linear MTL scalarization with adaptive weights assignment.

5 Experiments

In this section, we test the proposed Pareto MTL algorithm on different MTL problems with the following algorithm: 1) **Single Task**: the single task baseline; 2) **Grid Search**: linear scalarization with fixed weights; 3) **GradNorm** [13]: gradient normalization; 4) **Uncertainty** [7]: weights with uncertainty balance; 5) **MOO-MTL** [12]: finding one Pareto solution for multi-objective problem. More experimental results and analysis can be found in the supplementary material.

5.1 Multi-Fashion-MNIST



(a) MultiMNIST (b) Multi-FashionMNIST (c) Multi-(Fashion+MNIST)

Figure 4: **The results for three experiments with Task1&2 accuracy:** our proposed Pareto MTL successfully finds a set of widely distributed solutions with different trade-offs for all experiments, and it significantly outperforms Grid Search, Uncertainty and GradNorm. MOO-MTL algorithm can also find promising solutions, but their diversity is worse than the solutions generated by Pareto MTL.

In order to evaluate the performance of Pareto MTL on multi-task learning problem with different tasks relations, we first conduct experiments on MultiMNIST [31] and two MultiMNIST-like datasets. To construct the MultiMNIST dataset, we randomly pick two images with different digits from the original MNIST dataset [32], and then combine these two images into a new one by putting one digit on the top-left corner and the other one on the bottom-right corner. Each digit can be moved up to 4 pixels on each direction. With the same approach, we can construct a Multi-FashionMINST dataset with overlap FashionMNIST items [33], and a Multi-(Fashion + MNIST) with overlap MNIST and FashionMNIST items. For each dataset, we have a two objective MTL problem to classify the item on the top-left (task 1) and to classify the item on the bottom-right (task 2). We build a LeNet [32] based MTL neural network similar to the one used in [12]. The obtained results are shown in Fig. 4.

In all experiments, since the tasks compete with each other, solving each task separately results in a hard-to-beat single task baseline. Our proposed Pareto MTL algorithm can generate multiple widely distributed Pareto solutions for all experiments, which are compatible with the strong task baseline but with different trade-offs among the tasks. Pareto MTL algorithm achieves the overall best performance among all MTL algorithms. These results confirm that our proposed Pareto MTL can successfully provide a set of well-representative Pareto solutions for MTL problem.

It is not surprising to observe that the Pareto MTL's solution for subproblem with extreme preference vector (e.g., $(0,1)$ and $(1,0)$) always have the best performance in the corresponding task. Especially in the Multi-(Fashion-MNIST) experiment, where the two tasks are highly uncorrelated with each other, almost all MTL's solutions are dominated by the strong single task's baseline, but Pareto MTL can still generate solutions with the best performance for each task separately. This case can be treated as auxiliary learning, where the task with the assigned preference vector is the main task and the others are auxiliary tasks.

5.2 Self-Driving Car: Localization

Method	Reference Vector	Translation (m)	Rotation (°)
Single Task	-	8,392	2.461
Grid Search	(0.25,0.75)	9.927	2.177
	(0.5,0.5)	7.840	2.306
	(0.75,0.25)	7.585	2.621
GradNorm	-	7.751	2.287
Uncertainty	-	7.624	2.263
MOO-MTL	-	7.909	2.090
Pareto MTL	(0,1)	**7.285**	2.335
	$(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$	7.724	2.156
	(1,0)	8.411	**1.771**

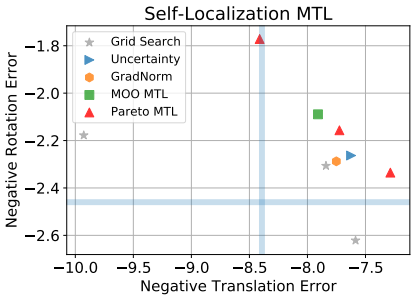


Figure 5: **The results of self-location MTL experiment:** Our proposed Pareto MTL outperforms other algorithms and provides solutions with different trade-offs.

We further evaluate Pareto MTL on an autonomous driving self-localization problem [8]. In this experiment, we simultaneously estimate the location and orientation of a camera put on a driving car based on the images it takes. We use data from the apolloscape autonomous driving dataset [34], and focus on the Zpark sample subset. We build a PoseNet with a ResNet18 [35] encoder as the MTL model. The experiment results are shown in Fig. 5. It is obvious that our proposed Pareto MTL outperforms all MTL approaches.

6 Conclusion

In this paper, we proposed a novel Pareto Multi-Task Learning (Pareto MTL) algorithm to find a set of widely distributed Pareto solutions with different trade-offs for a given MTL problem. MTL practitioners can then easily select their preferred solution among these Pareto solutions. Experimental results confirm that our proposed algorithm outperforms some state-of-the-art MTL algorithm and can successfully find a set of well-representative solutions for different MTL problems.

References

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

[2] 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.

[3] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[4] Zhen Huang, Jinyu Li, Sabato Marco Siniscalchi, I-Fan Chen, Ji Wu, and Chin-Hui Lee. Rapid adaptation for deep neural networks through multi-task learning. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

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

[6] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[7] 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 (CVPR)*, 2018.

[8] Peng Wang, Ruigang Yang, Binbin Cao, Wei Xu, and Yuanqing Lin. Dels-3d: Deep localization and segmentation with a 3d semantic map. In *CVPR*, pages 5860–5869, 2018.

[9] Jaebok Kim, Gwenn Englebienne, Khiet P Truong, and Vanessa Evers. Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning. *arXiv preprint arXiv:1708.03920*, 2017.

[10] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 517–531, 2018.

[11] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[12] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 525–536, 2018.

[13] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 794–803, 2018.

[14] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[15] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[16] Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *International Conference on Machine Learning*, pages 462–470, 2013.

[17] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

[18] Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927, 2016.

[19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. 2018.

[20] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer.

[21] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[22] Jean-Antoine Désidéri. Mutiple-gradient descent algorithm for multiobjective optimization. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012)*, 2012.

[23] 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.

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

[25] 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.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[28] Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2016.

[29] Hai-Lin Liu, Fangqing Gu, and Qingfu Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Trans. Evolutionary Computation*, 18(3):450–455, 2014.

[30] Bennet Gebken, Sebastian Peitz, and Michael Dellnitz. A descent method for equality and inequality constrained multiobjective optimization problems. In *Numerical and Evolutionary Optimization*, pages 29–61. Springer, 2017.

[31] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

[32] 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.

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

[34] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *arXiv: 1803.06184*, 2018.

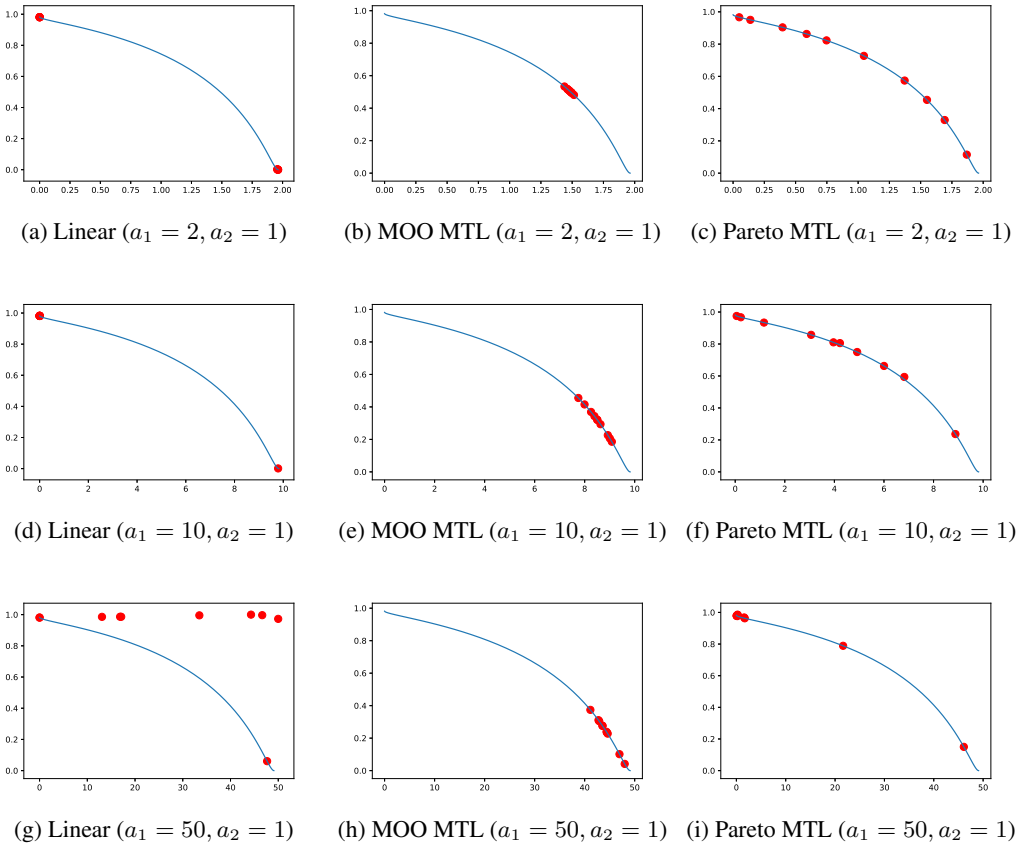
[35] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

Supplementary Material: Pareto MTL

Anonymous Author(s)
Affiliation
Address
email

In this supplementary material, we show and analyze more experimental results of the proposed Pareto MTL algorithm. We also discuss some interesting research topics which arise when we formulate solving MTL as finding Pareto solutions with different trade-offs for multi-objective optimization.

1 A Synthetic Example with Different Tasks Difficulties



(a) Linear ($a_1 = 2, a_2 = 1$) (b) MOO MTL ($a_1 = 2, a_2 = 1$) (c) Pareto MTL ($a_1 = 2, a_2 = 1$)

(d) Linear ($a_1 = 10, a_2 = 1$) (e) MOO MTL ($a_1 = 10, a_2 = 1$) (f) Pareto MTL ($a_1 = 10, a_2 = 1$)

(g) Linear ($a_1 = 50, a_2 = 1$) (h) MOO MTL ($a_1 = 50, a_2 = 1$) (i) Pareto MTL ($a_1 = 50, a_2 = 1$)

Figure 1: The convergence behaviours of different algorithms on a synthetic example with different tasks difficulties. Pareto MTL can find a set of widely distributed Pareto solutions on problems with low or medium difficulty unbalance level (($a_1 = 2, a_2 = 1$) and ($a_1 = 10, a_2 = 1$)), but its performance gets worse for problem with high difficulty unbalance level ($a_1 = 50, a_2 = 1$).

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

This paper proposes a novel Pareto MTL algorithm for solving deep multi-task learning problems, although it can also be used for solving a regular multi-objective optimization problem. The loss landscapes for a deep MTL problem's different tasks would be complicated and highly non-convex, which can result in complicated global and local Pareto front.

To better analyze the convergence behavior of the proposed Pareto MTL, we first compare it with two commonly used methods, namely the linear scalarization method and the multiple gradient descent algorithm used in MOO-MTL [1], on a simple synthetic multi-objective optimization problem:

$$\begin{aligned} \min f_1(\boldsymbol{x}) &= \alpha_1 - \alpha_1 \exp\left(-\sum_{i=1}^{d}(x_d - \frac{1}{\sqrt{d}})^2\right) \\ \min f_2(\boldsymbol{x}) &= \alpha_2 - \alpha_2 \exp\left(-\sum_{i=1}^{d}(x_d + \frac{1}{\sqrt{d}})^2\right) \end{aligned} \tag{1}$$

where $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are two objective functions to be minimized at the same time and $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ is the d dimensional decision variable. This problem has a concave Pareto front on the objective space. The two objective function can have different difficulty levels controlled by the parameters α_i. If $\alpha_1 = \alpha_2$, the two tasks have similar difficult level. If $\alpha_1 > \alpha_2$, the task one could be "easier" since it has a larger gradient value.

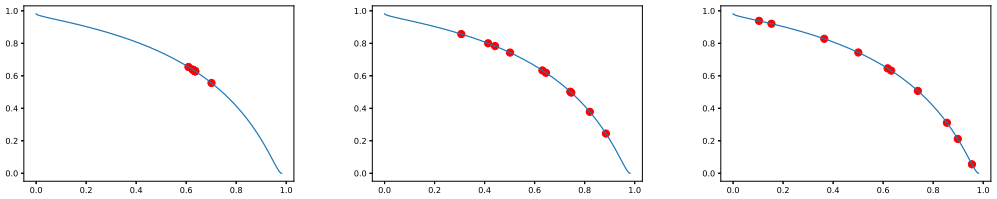
Convergence Behaviour. In the main paper, we show the results on problem with the same difficulty level (Fig.2 in Page 3, with $\alpha_1 = \alpha_2 = 1$). In this case, the proposed Pareto MTL can find a set of widely distributed Pareto solutions with different trade-offs, while MOO-MTL always gets solutions with a balanced trade-off in multiple runs. It is interesting to observe that the linear scalarization method cannot find solutions on most of the Pareto front evenly with 100 runs. According to to [2], this method can only provide solutions on the convex part of the Pareto front. Therefore, it can only reach the two endpoints of the concave Pareto front. This behavior gives another reason for not using linear scalarization with fixed for MTL problem from the multi-objective optimization perspective.

Performance for Problem with Unbalanced Difficulties. The results on problems with different difficulty levels are shown in Fig. 1. As in the balanced difficult case, the linear scalarization and MOO-MTL approach can not obtain a set of well-representative solutions for all problems. Pareto MTL can find a set of widely distributed Pareto solutions on problems with low or medium difficulty unbalance level (($a_1 = 2, a_2 = 1$) and ($a_1 = 10, a_2 = 1$)), but its performance gets worse for problem with high difficulty unbalance level ($a_1 = 50, a_2 = 1$).

Different Biases for MOO-MTL and Pareto MTL. Another interesting observation in this experiment is that MOO-MTL and Pareto MTL will be biased to different tasks in the same unbalance problem. For example, in the highly unbalanced problem ($a_1 = 50, a_2 = 1$), MOO-MTL is biased to the "easier" task 1, but most solutions found by Pareto MTL are biased to the "harder" task 2. The bias of MOO-MTL is straightforward since Task 1 has a larger absolute gradient value and it will contribute most to the combined gradient direction obtained by MOO-MTL. For Pareto MTL, the bias comes from the evenly distributed preference vector. When we do not have any prior information about the tasks, assigning a set of balance preference vectors such as $\{(1, 0), (\sqrt{2}/2, \sqrt{2}/2), (0, 1)\}$, would be a reasonable choice. A MTL practitioner would hope these preference vectors can evenly divide the objective space into several sub-regions. However, when the tasks have different difficulty levels, the divided sub-region would be highly unbalanced, and the solutions would be attracted by a few preference vector much easier. In the ($a_1 = 50, a_2 = 1$) case, most solutions generated by Pareto MTL are attracted by the preference vector $(0, 1)$ corresponding to task 2 rather than task 1.

How to wisely use these different biased behaviors to design a self-adaptive algorithm for solving unbalance MTL would be an interesting research topic in the future. It should be noticed that all experiments here are run on a synthetic example. Study the behavior of different MTL algorithms on real-world complicated MTL applications is also an important research topic.

2 The Importance of Finding the Initial Solution



(a) MOO MTL (b) Pareto MTL w/o Initialization (c) Pareto MTL w/ Initialization

Figure 2: **The convergence behaviours of different algorithms on the synthetic example.** The proposed Pareto MTL algorithm without initialization step can still generate a set of solutions with different trade-offs on the Pareto front, but it fails to find solutions near the end points.

In the paper, we propose a gradient-based method to find an initial solution to solve each constrained multi-objective subproblem. The initial solution would be a feasible solution to the constrained subproblem or at least satisfy most constraints. In this section, we further analyze the importance of finding the initial solution.

We first test the Pareto MTL algorithm without the initialization step on the synthetic example. As shown in Figure. 2, without the initialization step, Pareto MTL can still generate a set of Pareto solutions with different trade-offs and outperforms the MOO-MTL algorithm. However, the diversity of these solutions is worse than those generated by the Pareto MTL with the initialization step. It is obvious that Pareto MTL without initialization fails to find solutions near the endpoints of the Pareto front. The lack of ability to cross the boundary between different sub-regions would be one reason for the inferior performance.

For a subproblem, a randomly generated solution might not be in (or even far away from) its preference sub-region. With the initialization step, the solution can be sequentially updated to get close to the assigned preference sub-region since the constraint values are lowered at each iteration. Many constraints would turn inactivated once the solution crosses its boundary to get close to its assigned preference vector. In contrast, without the initialization step, the solution might stop at some boundary of the preference vectors (with corresponding activated constraints) during the optimization process since now the objective functions are taken into consideration. It is easier to find a descent direction to lower the value of activated constraints than to find a direction to lower both the values of activated constraints and the tasks.

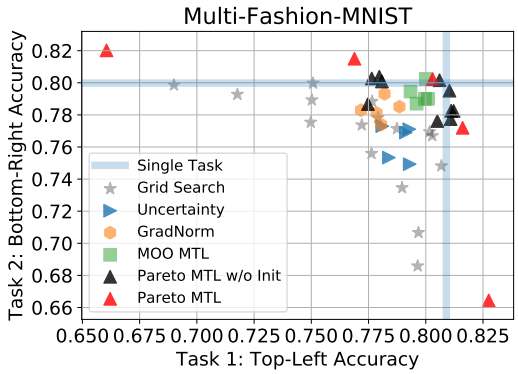


Figure 3: **Pareto MTL with and without Initialization on Multi-Fashion MNIST problem.**

We also test the Pareto MTL without initialization step on the Multi-FashionMNIST problem. As shown in Fig. 3, the Pareto MTL without initialization step can generate solutions with different trade-offs, but the diversity is much worse than Pareto MTL with initialization step.

In the proposed Pareto MTL, we obtain the descent direction for each subproblem by solving:

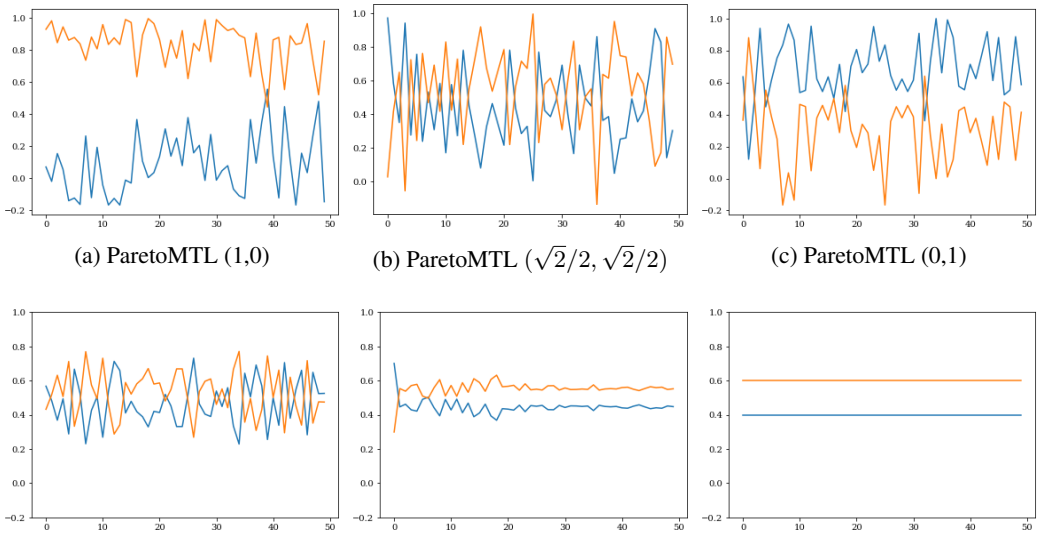
$$
\begin{aligned}
(d_t, \alpha_t) = &\arg \min_{d \in R^n, \alpha \in R} \alpha + \frac{1}{2}||d||^2 \\
s.t. \quad &\nabla \mathcal{L}_i(\theta_t)^T d \leq \alpha, i = 1, ..., m. \\
&\nabla \mathcal{G}_j(\theta_t)^T d \leq \alpha, j \in I_\epsilon(\theta_t),
\end{aligned} \tag{2}
$$

Actually, depended on how to obtain the gradient direction, we have three different algorithms:

1. Only consider the tasks $\mathcal{L}_i(\theta_t)$: MOO-MTL, for convergence;
2. Only consider the constraints $\mathcal{G}_j(\theta_t)$: the initialization step in Pareto MTL, for diversity;
3. Consider both tasks $\mathcal{L}_i(\theta_t)$ and $\mathcal{G}_j(\theta_t)$: the main step of Pareto MTL, which tries to find a set of restricted Pareto points on diverse sub-region, somehow balance the convergence and diversity.

How to choose or switch among these different algorithms would be an interesting research topic. The proposed Pareto MTL first runs step 2 and then runs step 3. An immediate extension is to keep and get a snapshot [3] of all solutions at the end of step 3, then relax all constraints and run step 1. In this way, we can obtain a set of restricted Pareto critical solutions for each subproblem by Pareto MTL (running step 2 and step 3) with good diversity, plus a set of Pareto critical solutions (not restricted) with potential better convergence by running step 1 at the end.

3 The Adaptive Weight Vectors



(a) ParetoMTL (1,0) (b) ParetoMTL $(\sqrt{2}/2, \sqrt{2}/2)$ (c) ParetoMTL (0,1)

(d) MOO-MTL (e) GradNorm (f) Fixed Weight

Figure 4: **The adaptive weight vectors for different algorithms during the training process for MultiMNIST experiment.** Pareto MTL behaves differently with different reference vectors. The other algorithms with adaptive weights assignment try to balance different tasks.

In the paper, we show that MOO-MTL and the proposed Pareto MTL algorithm can be reformulated as a linear scalarization of different tasks with adaptive weights assignment where the Pareto MTL algorithm can be rewritten as:

$$
\mathcal{L}_i(\theta_t) = \sum_{i=1}^{m} \alpha_i \mathcal{L}_i(\theta_t), \text{ where } \alpha_i = \lambda_i + \sum_{j \in I_{\epsilon(\theta)}} \beta_j (\boldsymbol{u}_{ji} - \boldsymbol{u}_{ki}), \tag{3}
$$

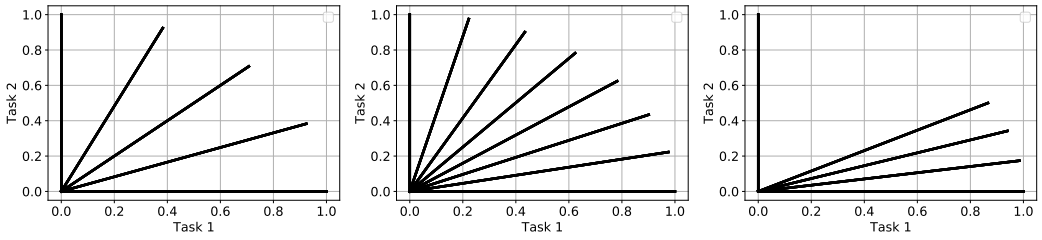
In this section, we compare the adaptive weight vectors for different algorithms during the training process. As shown in Fig. 4, Pareto MTL has clearly different weight adaption strategy for sub-problems with different preference vectors while MOO-MTL and GradNorm always try to balance different tasks.

From the view point of linear scalarization with adaptive weights, the surrogate loss for MOO-MTL, GradNorm and Uncertainty can be written as $\mathcal{L}_i(\theta_t) = \sum_{i=1}^{m} \lambda_i \mathcal{L}_i(\theta_t)$. Different methods have their own strategy to adapt the weight λ_i to balance the loss function $\mathcal{L}_i(\theta_t)$. For Pareto MTL, the weight vector now has the form $\alpha_i = \lambda_i + \sum_{j \in I_{\epsilon(\theta)}} \beta_j(\boldsymbol{u}_{ji} - \boldsymbol{u}_{ki})$. While the parameter λ_i is still for balancing different tasks, the preference term $\sum_{j \in I_{\epsilon(\theta)}} \beta_j(\boldsymbol{u}_{ji} - \boldsymbol{u}_{ki})$ will guide the Pareto MTL to its corresponding preference vector. As shown in Fig. 4, Pareto MTL will bias the search to a specific task when it has extreme preference vectors (e.g., $(0, 1)$ and $(1, 0)$), and it will try to balance different tasks with a balance preference vector (e.g., $(\sqrt{2}/2, \sqrt{2}/2)$).

When all constraints are inactivated (e.g., $I_{\epsilon(\theta)} = \emptyset$), Pareto MTL has a feasible solution right in the assigned sub-region for a given subproblem. In this case, the surrogate loss could be reduced to $\mathcal{L}_i(\theta_t) = \sum_{i=1}^{m} \lambda_i \mathcal{L}_i(\theta_t)$ which is the same as MOO-MTL, and Pareto MTL will try to find a balanced solution in the assigned sub-region.

Pareto MTL is not mutually exclusive with other adaptive weight strategies such as GradNorm [4] and Uncertainty [5], especially when it can be reformulated as the linear scalarization method. For MTL problem with highly unbalanced tasks, it is possible to first balance all tasks with some adaptive weight strategies, and then use Pareto MTL to find a set of Pareto solutions for the balanced tasks.

4 Preference Vector Assignment



(a) (b) (c)

Figure 5: **Three different sets of preference vector:** (a) 5 evenly distributed unit preference vectors; (b) 8 evenly distributed preference unit vectors; and (c) 5 biased unit preference vectors.

How to properly assign the set of preference vectors is important for Pareto MTL, since it determines how to divide the objective space. When we do not have any prior information for a given MTL problem, it is reasonable to evenly divide the objective space for different tasks with a set of evenly distributed preference vector as shown in Fig. 5 (a)(b). In the experiments, we separately assign a set of $3, 5, 10$ evenly distributed preference unit vectors for the self-driving car, the MultiMNIST, and the synthetic problem.

When the MTL practitioners have their own preference or any prior knowledge for a given MTL problem, they can feel free to use a set of biased preference vectors as in Fig. 5 (c). How to efficiently set the preference vectors based on the user's preference or any prior information could be an interesting topic. A strategy to adaptively set or change the preference vectors during the multi-task learning process to incorporate the practitioner's preference or better explore the objective space is another possible extension.

5 MTL with Many Tasks

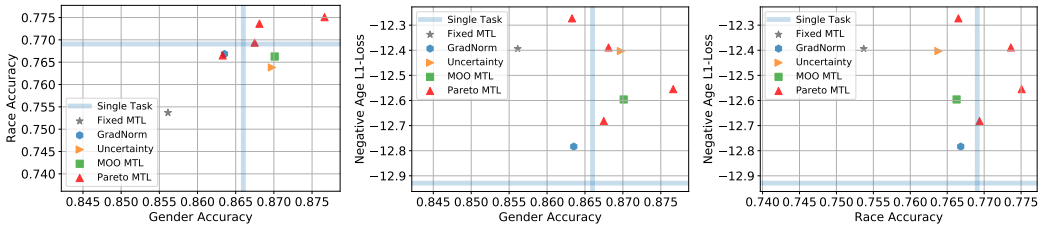
Pareto MTL uses a set of preference vectors to divide a given MTL problem into several constrained multi-objective subproblems. However, to fairly cover the whole objective space of all tasks, the number of required preference vectors would increase exponentially when the MTL have more tasks. In other words, it is impossible for Pareto MTL to provide a set of widely distributed solutions to explore the whole objective space for a MTL problem with many tasks. In this section, we provide some potential methods to use Pareto MTL for solving MTL problem with many tasks.

Finding representative solutions with preferred trade-offs. When the preference vectors are fixed and cannot be adaptively adjusted, MTL practitioners can still directly use Pareto MTL to generate different Pareto solutions with their preferred trade-offs for MTL problem with many tasks. As discussed in the experiment section, in Pareto MTL, the subproblem with extreme preference vector (e.g., $(0, 1)$ and $(1, 0)$) can be explained as auxiliary multi-task learning, where the corresponding task is the preferred main task and the others are auxiliary tasks. Similar explains can also be applied for subproblem with a specific preference vector. In other words, once the MTL practitioners have their preferred trade-off(s) among the tasks, they can directly run Pareto MTL to find diverse Pareto solution(s) corresponding to different preference vectors. If the MTL practitioners do not have any preferred trade-off yet, they can at least run Pareto MTL with a few different preference vectors to obtain a set of diverse Pareto solutions. They can directly choose their preferred solutions or use them to summarize preferred trade-off(s) for another run.

In this section, we run Pareto MTL with only a few preference vectors for solving a MTL with three different tasks. The dataset we use is the UTKFace dataset [6], and the MTL problem is to predict human's gender, race, and age based on one image of their faces. We build a deep MTL network with Resnet18 as the encoder and a task-specific fully connected layer for each task.

Table 1: The gender accuracy, race accuracy, and age L1-loss obtained by different algorithms. The best results are highlighted. Pareto MTL can find widely distributed solutions with diverse trade-offs.

Method	Reference Vector	Gender (Accuracy %)	Race (Accuracy %)	Age (L1-Loss)
Single Task	-	0.865999	0.769067	12.929101
Fixed MTL	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	0.856132	0.753702	12.393776
GradNorm	-	0.863526	0.766833	12.783628
Uncertainty	-	0.869731	0.763821	12.403259
MOO-MTL	-	0.870106	0.766267	12.595977
Pareto MTL	$(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3})$	0.867482	0.769377	12.681999
	(1,0,0)	0.868106	**0.773621**	12.388865
	(0,1,0)	0.863294	0.766513	**12.272779**
	(0,0,1)	**0.876636**	**0.775078**	12.554767



(a) gender v.s. race (b) gender v.s. age (c) race v.s. age

Figure 6: The 2-D projections for the results obtained by different algorithms. We report the negative age L1-loss for the sake of consistency. Pareto MTL can provide solutions with diverse trade-offs.

The experiment result is shown in Table. 1 and Fig. 6. It confirms that Pareto MTL with a few preference vectors can find a set of well-representative solutions for a MTL with three tasks.

Learning to find preferred Pareto solution(s). One advantage of Pareto MTL over MOO-MTL is its ability to incorporate preference information even with only a single run (solving one subproblem, but still need a set of preference vectors to divide the objective space). Recently, some learning-based methods have been proposed to solve MTL problems [7, 8]. It is possible to propose learning-based Pareto MTL for dynamically adjusting the preference vectors to incorporate the MTL practitioner's preference, and to guide the solutions search to their preferred small subspace for a MTL with many tasks.

Methods from the multi-objective optimization community. By formulating the MTL with many tasks as a multi-objective optimization problem, we get a many objective optimization problem, which is indeed a popular research topic in the multi-objective optimization community [9, 10, 11, 12]. How to adopt the techniques proposed from the multi-objective optimization community to solve MTL problem with many tasks is a potential research direction.

References

[1] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 525–536, 2018.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[3] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 794–803, 2018.

[5] 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 (CVPR)*, 2018.

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

[7] Yu Zhang, Ying Wei, and Qiang Yang. Learning to multitask. In *Advances in Neural Information Processing Systems*, pages 5771–5782, 2018.

[8] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287, 2018.

[9] Peter J Fleming, Robin C Purshouse, and Robert J Lygoe. Many-objective optimization: An engineering design perspective. In *International conference on evolutionary multi-criterion optimization*, pages 14–32. Springer, 2005.

[10] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426. IEEE, 2008.

[11] David Hadka and Patrick Reed. Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary computation*, 21(2):231–259, 2013.

[12] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.