EVOGRAD: AUTOMATICALLY EVOLVING GRADIENT AGGREGATION RULES FOR MULTI-TASK LEARNING USING GENETIC PROGRAMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-task learning (MTL) aims to improve data efficiency and generalization by leveraging shared information across tasks. Many existing gradient balancing methods developed for MTL limit their aggregation rules (namely programs) to a linear search space spanned by task gradients, and determine the optimal programs merely based on a local decrease in training loss, which may not lead to model improvement. In this paper, we propose EvoGrad, an evolutionary framework based on multi-objective genetic programming (GP) that automatically discovers optimal programs. By composing nonlinear operations over task gradients, EvoGrad simultaneously explores more expressive search spaces beyond their linear span and identifies optimal programs. To accelerate the exploration, we adopt a hybrid selection strategy that first efficiently selects diverse and balanced candidate programs based on loss-decrease rates, and then identifies the best program using dataset-specific evaluation metrics. Experiments on four benchmark datasets with task numbers ranging from 2 to 40, together with three toy examples, demonstrate the effectiveness of EvoGrad in MTL.

1 INTRODUCTION

Many real-world applications require the simultaneous optimization of multiple task objectives, which may be inherently conflicting with each other. For example, in the field of Computer Vision, a model may be required to perform multiple tasks simultaneously, such as semantic segmentation, depth estimation, and surface normal prediction (Vandenhende et al., 2022). This has motivated a shift from single-task learning (STL) to multi-task learning (MTL) (Zhang & Yang, 2022). The goal of MTL is to learn a single model that can handle multiple tasks while effectively balancing their respective learning objectives. By exploiting shared information across tasks, MTL can potentially achieve better generalization and performance than STL. Consequently, MTL has demonstrated significant potential in a wide range of applications, including Natural Language Processing (Chen et al., 2024), Computer Vision (Vandenhende et al., 2022), autonomous driving (Yu et al., 2020a), robotics (Shu et al., 2018), and general scientific problems (Wu et al., 2018).

As the gradients associated with different tasks in MTL can vary substantially in magnitude and may even conflict with one another (Sener & Koltun, 2018; Yu et al., 2020c; Liu et al., 2021a), simply averaging the losses of all tasks during optimization can cause the resulting update direction to be dominated by tasks with larger gradient magnitudes. This, in turn, biases the optimization process toward certain tasks and potentially degrades the overall performance. Motivated by these observations, various MTL approaches have been proposed, which can be broadly categorized into two groups: Loss Balancing Methods and Gradient Balancing Methods (Chen et al., 2026). Loss Balancing Methods are computationally efficient, as they typically require only a single backward pass per iteration. However, they often exhibit weaker performance than Gradient Balancing Methods. Even among Gradient Balancing Methods, two problems remain: 1) Most of them seek a set of task weights to construct a gradient aggregation direction $d^* = \sum_{i=1}^{t} \alpha_i g_i$, where $\mathbf{g}_i$ denotes the gradient of the i-th task and α_i is its corresponding weight. Consequently, their search space is restricted to the linear span of the task gradients, i.e., $\mathrm{span}\{g_1, \ldots, g_t\}$, which may limit their ability to discover more effective aggregation directions; and 2) The selection of better gradient aggregation directions

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

merely relies on a local decrease in the optimization loss, which does not necessarily correspond to a model improvement on the task-specific evaluation metric.

To address these issues, we propose EvoGrad, an evolutionary framework based on Genetic Programming (GP) to automatically discover (approximately) optimal aggregation directions in a nonlinear span of the task gradients. By composing nonlinear operations over task gradients, our method explores expressive search spaces that extends beyond the linear search space. Specifically, the search space exploration is considered as an evolution process of GP, in which large-scale programs are generated by applying four operations (two linear and two nonlinear) to task gradients. Here, a program represents an aggregation rule that maps the task gradients to a gradient aggregation direction. To efficiently discover the optimal program, we first employ three objectives to select candidate programs, and then apply a dataset-specific evaluation metric to choose the optimal one. These three objectives capture different perspectives of loss-decrease rates across all tasks, thereby providing balanced and diverse programs for subsequent selection.

Our main contributions are as follows:

• A GP-based evolutionary framework is proposed, which automatically searches for approximately optimal gradient aggregation directions beyond linear search spaces. By formulating the direction search as a GP problem, where the evolutionary process searches over compositions of (non)linear operations applied to task gradients, the framework jointly discover the optimal search space and program.

• A hybrid selection strategy is proposed to effectively and efficiently search for the optimal program. It first discovers balanced and diverse candidate programs using three objectives based on loss-decrease rates, and then employs a dataset-specific evaluation metric to identify the best one as the optimal program.

• Extensive experiments on multiple MTL benchmarks validate the effectiveness of the proposed GP-based evolutionary framework. Compared with existing MTL methods, our approach achieves competitive or superior performance across multiple evaluation metrics, demonstrating its ability to discover effective gradient aggregation rules.

2 Related Work

Multi-task Learning. The goal of MTL is to learn a single model that effectively balances all task objectives. The most straightforward approach averages the gradients of all tasks, commonly referred to as Equal Weighting (EW) (Zhang & Yang, 2022). However, task objectives in MTL may conflict with one another, in which case EW may degrade the performance of some tasks (Standley et al., 2020). Existing approaches for improving multi-task performance can be broadly divided into two categories: Loss Balancing Methods (Liu et al., 2022; 2021b; 2019; Chen et al., 2018) and Gradient Balancing Methods Qin et al. (2025); Wang et al. (2026).

Loss Balancing Methods (Cipolla et al., 2018; Lin et al., 2026; Ye et al., 2024a;b) dynamically learn task weights at the loss level during training. Although computationally efficient, they generally exhibit weaker convergence properties than Gradient Balancing Methods (Chen et al., 2026). On the other hand, most Gradient Balancing Methods (Désidéri, 2012; Liu et al., 2021a; Navon et al., 2022; Ban & Ji, 2024; Qin et al., 2025; Wang et al., 2026) adaptively learn task-specific weights to aggregate task gradients at each iteration, aiming to identify a common descent direction that simultaneously improves all task objectives. In addition, some methods (Yu et al., 2020b; Wang et al., 2021) employ gradient projection to alleviate conflicts between task gradients. Although recent studies (Navon et al., 2022; Ban & Ji, 2024; Qin et al., 2025; Wang et al., 2026) have investigated multi-task optimization from a fairness perspective, their search spaces remain restricted to the linear span of task gradients. Meanwhile, some methods have explored nonlinear aggregation of task gradients. For example, GradDrop (Chen et al., 2020) applies coordinate-wise masks based on gradient-sign consistency, while BayesAgg-MTL (Achituve et al., 2024) assigns task- and dimension-specific uncertainty weights. However, the functional forms of their aggregation mechanisms are predefined rather than automatically discovered, as proposed in our study.

Genetic Programming. As a variant of Genetic Algorithms (GA), GP (Koza, 1999) has been widely applied to job-shop scheduling (Geng et al., 2026), symbolic regression (Zhang et al., 2025), feature

construction (Jia et al., 2024) and classification tasks (Meng et al., 2026) due to its global search capability and representational flexibility. Furthermore, its inherent capabilities for constructing explicit relationships among input features make GP a useful approach to interpretable machine learning (Mei et al., 2023). To the best of our knowledge, using GP to directly evolve gradient aggregation rules for MTL remains largely unexplored, which motivates the focus of this paper.

3 PRELIMINARIES

Multi-task Learning. Consider an MTL problem consisting of $t \geq 2$ tasks. Let $\theta \in \mathbb{R}^d$ denote the shared model parameters and $\ell_i(\theta)$ denote the loss associated with task i. The gradient of task i is $g_i = \nabla_\theta \ell_i(\theta)$. MTL aims to learn a single model that jointly minimizes the loss over all tasks:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{t} \sum_{i=1}^{t} \ell_i(\theta). \tag{1}$$

Common Gradient Balancing Methods seek task weights $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_t)$ to construct a gradient aggregation direction $d^* = \sum_{i=1}^{t} \alpha_i g_i$, with the aim of obtaining a common descent direction that improves all tasks. Assume that for the k-th iteration, we update the $\theta^{k+1} \leftarrow \theta^k - \eta d^k$, where the η is a step size. Then, for a small step size $\eta > 0$, a first-order Taylor expansion is defined as:

$$\frac{1}{\eta} \left[\ell_i(\theta) - \ell_i(\theta - \eta d) \right] \approx g_i^\top d. \tag{2}$$

where the $g_i^\top d$ measures the first-order decrease rate of $\ell_i(\theta)$ induced by using d as the update direction. A larger value of $g_i^\top d$ indicates a greater decrease in $\ell_i(\theta)$ and is therefore preferable. Recent Gradient Balancing Methods (Navon et al., 2022; Ban & Ji, 2024; Qin et al., 2025; Wang et al., 2026) have mainly relied on $g_i^\top d$ to optimize the loss decreasing rate of individual tasks in a fairer manner. A detailed derivation of Eq. 2 is provided in Appendix A.

Genetic Programming. Genetic programming (GP) is a widely used evolutionary computation paradigm that automatically discovers solutions through an evolutionary search. Such a search maintains a population of individuals, each representing a candidate solution to the problem of interest. During each evolutionary process, the individuals are evaluated according to a problem-specific fitness criterion, after which selection and variation operators, such as crossover and mutation, are applied to generate the next population. In standard tree-based GP, an individual is represented as an expression tree whose internal nodes are operators drawn from a predefined function set and whose leaf nodes are selected from a terminal set. Applying GP to a specific problem therefore primarily requires defining the function set, terminal set, and fitness criterion, which jointly determine the representation and guide the evolutionary search. Further details on GP initialization, selection, crossover, mutation, and the overall evolutionary procedure are provided in Appendix D.

Pareto Optimality. MTL can be naturally formulated as a multi-objective optimization (MOO) problem with an objective vector $\boldsymbol{\ell}(\theta) = (\ell_1(\theta), \ldots, \ell_t(\theta))$. Under the minimization convention, a solution θ^a dominates another solution θ^b, denoted by $\theta^a \prec \theta^b$, if

$$\ell_i(\theta^a) \leq \ell_i(\theta^b), \quad \forall i \in \{1, \ldots, t\}, \qquad \text{and} \qquad \exists j : \ell_j(\theta^a) < \ell_j(\theta^b).$$

A solution is said to be Pareto optimal if it is not dominated by any other feasible solution. The collection of all Pareto-optimal solutions is referred to as the Pareto set, while the corresponding objective vectors form the Pareto front. Building on the same dominance relation, Evolutionary Multi-objective Optimization (EMO) (Knowles & Zheng, 2024) has been developed for evolutionary algorithms in settings where each individual is evaluated by more than one objective and must be compared across these objectives within a population. Individuals that are not dominated by any other individual in the current population constitute the nondominated set, whose objective vectors serve as an approximation of the Pareto front.

4 EVOGRAD

In this section, we present our proposed method, EvoGrad, with its pseudocode provided in Algorithm 1. Each iteration consists of three main steps. 1) **Gradient Program Representation**

Algorithm 1: EVOGRAD for MTL

Input: Initial model parameters θ_0; task losses $\{\ell_i\}_{i=1}^t$; learning rates $\{\eta_k\}_{k=1}^K$; initial GP population $\mathcal{P}_0$; number of training iterations K.
Output: Trained model parameters θ_K

Initialize $\mathcal{P} = \mathcal{P}_0$;
for k =*1* **to** K **do**
  Compute normalized task gradients $T_k = \left\{\widehat{g}_{i,k} = \frac{\nabla_\theta \ell_i(\theta_k)}{\|\nabla_\theta \ell_i(\theta_k)\|_2}\right\}_{i=1}^t$;
  Evolve the GP population: $\mathcal{P} = \mathrm{Evolve}(\mathcal{P}, T_k)$;
  Select the best program p_k^* from the evolved population;
  Compile p_k^* into a function: $\tilde{p}_k^* = \mathrm{Compile}(p_k^*)$;
  Compute the aggregation gradient direction: $d_k^* = \tilde{p}_k^*(T_k)$;
  Update parameters $\theta_{k+1} = \theta_k - \eta_k d_k^*$;
end
return θ_K;

formulates MTL as a GP problem. In a common GP problem, each individual represents a candidate program; for clarity, we use only the term program hereafter, where a program represents a mapping from the task gradients to a gradient aggregation direction. 2) **Balance-Based Program Evolution** employs EMO together with three objectives based on loss-decrease rates to conduct program evaluation in GP, producing balanced candidate programs. 3) **Performance-Aware Program Selection** uses a combination of diversity-based candidate program selection and metric-guided best program selection to find approximately optimal programs.

Notice that, both the balance-based program evolution and diversity-based candidate program selection rely on the three objectives and together constitute the coarse-grained stage of our hybrid program selection strategy. In contrast, the metric-guided best program selection is based on dataset-specific evaluation metrics and constitutes the fine-grained stage of this strategy.

4.1 GRADIENT PROGRAM REPRESENTATION

At iteration k in Algorithm 1, let $\mathcal{B}_{i,k}$ denote the mini-batch for task i, $g_{i,k} = \nabla_\theta \ell_i(\theta_k; \mathcal{B}_{i,k})$ denote the gradient of task i over $\mathcal{B}_{i,k}$, and T_k is a set consists of the normalized task gradients.

$$\widehat{g}_{i,k} = \frac{g_{i,k}}{\|g_{i,k}\|_2}, \qquad T_k = \{\widehat{g}_{i,k}\}_{i=1}^t, \tag{3}$$

To search for approximately optimal gradient aggregation directions in a nonlinear search space, we formulate a dedicated GP problem in which each candidate program p is represented as an expression tree. In such a tree, the terminal set is T_k, and the internal nodes are drawn from the primitive set:

$$\mathcal{F} = \{+, -, \odot, \oslash\}, \tag{4}$$

where $+$, $-$, $\odot$ and $\oslash$ denote element-wise addition, subtraction, multiplication and protected element-wise division, respectively. Through program p, the task gradients are mapped to an aggregation direction, $d_{p,k} = p(T_k)$.

In this paper, the primitives support both basic linear combinations and coordinate-wise nonlinear transformations of task gradients, returning a vector of the same dimension as the task gradients. As introducing additional primitives would enlarge the combinatorial search space and may increase the difficulty of evolutionary search and exacerbate program bloat, we deliberately restrict $\mathcal{F}$ to these four primitives to get a balance of representational expressiveness and search efficiency.

4.2 BALANCE-BASED PROGRAM EVOLUTION

In GP, each evolutionary process needs to evaluate programs at a large scale in order to select some of them for subsequent processes. However, direct performance-based evaluation of each program—temporarily updating the training model and then measuring its performance—imposes substantial cost on the evolutionary process. On the other hand, an efficient approach that evaluates

each program by a loss decrease does not necessarily correspond to performance improvement on the task-specific evaluation metric. Moreover, since conflicts may exist among task gradients, enforcing task-wise descent conditions as hard constraints—i.e., loss-decrease rate $r_i(p) > 0$ for every task i—may yield no feasible solution. Therefore, we adopt a balanced program selection strategy and formulate it as an Evolutionary Multi-objective Optimization (EMO) problem with three explicit yet inexpensive surrogate objectives. These three objectives are jointly maximized so as to simultaneously capture different perspectives of loss-decrease rates across all tasks.

To be more specific, motivated by the criteria employed in Gradient Balancing Methods (Ban & Ji, 2024), we define the first-order loss-decrease rate of task i for a program p as $r_i(p) = g_i^\top d_p$, and the three objective as:

$$\begin{aligned} f_{\text{sum}}(p) &= \sum_{i=1}^{t} r_i(p), \\ f_{\text{num}}(p) &= \sum_{i=1}^{t} \max\{sign(r_i(p)), 0\}, \\ f_{\min}(p) &= \min_{i \in \{1,\dots,t\}} r_i(p), \end{aligned} \tag{5}$$

where f_{sum} measures the overall loss-decrease rate across tasks, f_{num} counts the number of tasks for which d_p is a decreasing direction, and $f_{\min}$ is the minimum loss-decrease rate across tasks. A good program should consistently increase all three objectives. However, since a high f_{sum} may come at the expense of some tasks' performance, such a program may not exist—highlighting the necessity of our balanced program selection strategy, where f_{num} indicates the number of tasks that are not compromised, and $f_{\min}$ indicates the severity of the worst compromise.

NSGA-II (Deb et al., 2002), a commonly-used EMO solver, is used to evolve programs with respect to the three objectives. After all evolutions in GP are finished, a nondominated set of programs is produced, whose objective vectors form an approximation to the Pareto front. The detail of evolution process is provided in Appendix C.1.

4.3 Performance-Aware Program Selection

Given the nondominated set of programs produced by GP, an approximately optimal program must be selected to support a model parameter update. As aforementioned, the direct performance-based evaluation is computationally expensive, especially when the nondominated set is large. For instance, on the CelebA dataset, we empirically observe that the nondominated set can contain more than one hundred unique programs.To address this, we propose a two-step selection strategy: (i) Diversity-based Candidate Program Selection, which employs the three objectives to select diverse candidate programs, and (ii) Metric-Guided Best Program Selection, which uses dataset-specific evaluation metrics to identify the best program as the approximately optimal one.

4.3.1 Diversity-Based Candidate Program Selection

To accelerate optimal program selection, a compact and representative candidate set (denoted by $\mathcal{C}_k$) is constructed from the nondominated set $\mathcal{A}_k$, preserving objective-wise extreme along the approximate Pareto front. That is, B programs are selected from $\mathcal{A}_k$ to form $\mathcal{C}_k$ as diversely as possible, where B is set to 10 in this paper.

Specifically, according to Eq. 6, three programs that each maximize one of the three objectives are first selected to initialize $\mathcal{C}_k$. These represent the most promising programs in terms of the total loss-decrease rate, the number of tasks with a positive loss-decrease rate, and the minimum loss-decrease rate across tasks.

$$p_{k,j}^{\max} = \arg\max_{p \in \mathcal{A}_k} f_j(p), \qquad j \in \{\text{sum}, \text{num}, \min\}. \tag{6}$$

The remaining programs, $\mathcal{A}_k \setminus \mathcal{C}_k$, are then ranked by crowding distance (Deb et al., 2002) in descending order, where a higher distance indicates greater diversity. The top-ranked programs are added to $\mathcal{C}_k$ until the desired set size is reached or no programs remain.

4.3.2 METRIC-GUIDED BEST PROGRAM SELECTION

Given the candidate set $\mathcal{C}_k$, the direct performance-based evaluation is used to select the best program. Specifically, for each $p \in \mathcal{C}_k$, its gradient aggregation direction $d_{p,k} = p(\widehat{G}_k)$ is computed, and a temporary one-step model parameter update is made, which changes the original parameter state $\theta_{p,k}$ to a temporary state $\widetilde{\theta}_{p,k}$. According to Eq. 7, the quality of this temporary state is then measured as the average performance on mini-batch $\mathcal{B}_{i,k}$ under a dataset-specific evaluation metric. Finally, the program p_k^* with the best performance is selected as the approximately optimal one.

$$Q_k(p) = D\Big(\widetilde{\theta}_{p,k}; \{\mathcal{B}_{i,k}\}_{i=1}^{t}\Big), \qquad p_k^* \in \begin{cases} \arg\max\limits_{p\in\mathcal{C}_k} Q_k(p), & \text{if } Q_k \text{ is higher-is-better,} \\ \arg\min\limits_{p\in\mathcal{C}_k} Q_k(p), & \text{if } Q_k \text{ is lower-is-better.} \end{cases} \tag{7}$$

where D denotes the model performance on the dataset-specific metric (see Sec. 5.1) and t is the number of tasks.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. The effectiveness of EvoGrad is evaluated on a diverse set of MTL benchmarks, including synthetic toy examples, image classification, and scene understanding tasks. For the two image-level classification benchmarks, CelebA (Liu et al., 2015) (over 200,000 celebrity images) is a 40-task classification dataset, and Office-31 (Liu et al., 2023) contains 4,110 images spanning 3 tasks: Amazon, DSLR, and Webcam. For the two scene understanding benchmarks, Cityscapes (Cordts et al., 2016) (2,975 training and 500 validation images) defines 2 tasks, segmentation and depth estimation, whereas NYU-v2 (Silberman et al., 2012) (795 training and 654 test images) defines 3 tasks: semantic segmentation, depth estimation, and surface normal prediction. The details are provided in Appendix E.2.

Baselines. EvoGrad is compared against state-of-the-art MTL baselines, including UW (Cipolla et al., 2018), DWA (Liu et al., 2019), MGDA (Sener & Koltun, 2018), RLW (Lin et al., 2021), IMTL-G (Liu et al., 2021b), NashMTL (Navon et al., 2022), FAMO (Liu et al., 2023), FairGrad (Ban & Ji, 2024), PIVRG (Qin et al., 2025) and SVFair (Wang et al., 2026).

Evaluation metrics. For the image-classification and scene understanding benchmarks, the task-level results are evaluated with two metrics: relative performance improvement Δp, and Mean Rank (MR). Δp denotes the relative performance improvement measured across all evaluation metrics, and MR denotes the relative improvement in performance rank. A larger Δp and a smaller MR both indicate better overall performance.

For an MTL method $\mathcal{M}$, its relative performance improvement over single-task learning (STL), Δp, is defined as:

$$\Delta_p(\mathcal{M}) = \frac{1}{t}\sum_{i=1}^{t} \Delta_{p,i}(\mathcal{M}), \tag{8}$$

$$\Delta_{p,i}(\mathcal{M}) = \frac{100\%}{N_i}\sum_{m=1}^{N_i} (-1)^{s_{i,m}} \frac{M_{i,m}^{\mathcal{M}} - M_{i,m}^{\mathrm{STL}}}{M_{i,m}^{\mathrm{STL}}}. \tag{9}$$

where t is the number of tasks, N_i is the number of evaluation metrics for task i, $M_{i,m}^{\mathcal{M}}$ and $M_{i,m}^{\mathrm{STL}}$ denote the m-th metric values achieved by method $\mathcal{M}$ and STL respectively, and $s_{i,m}$ is 0 if a larger value of the m-th metric on task i indicates better performance, and 1 otherwise.

Let $r_{i,m}(\mathcal{M})$ denote the rank of method $\mathcal{M}$ among all compared methods according to the m-th metric of task i, where a rank of one corresponds to the best performance. The mean rank is then

computed by averaging across tasks:

$$\mathrm{MR}(\mathcal{M}) = \frac{1}{t}\sum_{t=1}^{t}\left(\frac{1}{N_i}\sum_{i=1}^{N_i} r_{i,m}(\mathcal{M})\right). \tag{10}$$

Dataset-specific metrics. Regarding the dataset-specific evaluation metrics used in our metric-guided best program selection, for datasets with consistent task-level metrics, mean F1 and mean accuracy are used on CelebA and Office-31, respectively. For datasets with inconsistent task-level metrics, including Cityscapes and NYU-v2, Δ_p based on the STL training results is chosen.

Implement details. For CelebA and Office-31, most baseline results are taken from (Liu et al., 2023; Wang et al., 2026), except for PIVRG and FAMO. PIVRG is rerun on both datasets because its gradient weights are obtained using STL test set results, and FAMO is run on Office-31 as its performance is missing. On the other hand, all baselines are reproduced on Cityscapes and NYU-v2 because our batch size settings differ from the original implementations. Further details are provided in Appendix E.2.

5.2 Toy Example

We adopt the two-task synthetic example introduced in Liu et al. (2023), denoted as L_1 and L_2. To illustrate the optimization behavior, we select five different initial parameter points and visualize the optimization trajectories of EvoGrad and the compared MTL baselines in Fig. 1. Further details and more results are provided in the Appendix E.1.

As shown in Fig. 1, FAMO, IMTL-G, and Nash-MTL approach Pareto-stationary points and subsequently exhibit little further movement. In contrast, EvoGrad continues to move smoothly along the trade-off boundary after reaching it, exhibiting trajectory patterns similar to those of FairGrad and SVFair across different initializations.

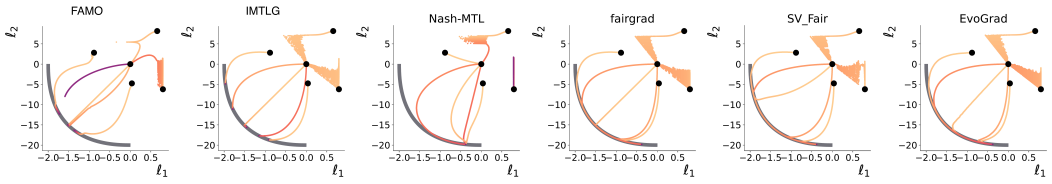

Figure 1: Optimazation trajectories for selected methods on the FAMO two-task example. Starting points are indicated by black dots, and the Pareto front is represented by thick gray lines.

5.3 Performance Comparison

Overall Performance. Table 1 reports the performance of various MTL methods on Office-31, CelebA, Cityscapes, and NYU-v2. The detail results can be found in Appendix E.3

For image-level classification, on the 3-task Office-31 benchmark, EvoGrad achieves the lowest MR (1.33) but performs slightly worse than the best baseline, SVFair, in terms of Δp (1.28% vs. 1.53%). On the more challenging 40-task CelebA benchmark, EvoGrad achieves both the lowest MR (3.95) and the largest Δp (1.01%).

For scene understanding, on Cityscapes, EvoGrad performs best on MR and ranks fourth on Δp. On NYU-v2, however, EvoGrad performs less well on both MR and Δp. For example, for EvoGrad and SVFair, Δp is 8.59% vs. 9.79% on Cityscapes, and 3.89% vs. 6.04% on NYU-v2.

Performance Analysis. EvoGrad achieves the best MR on most benchmarks, but obtains varying Δp, due to two reasons: representation capability with nonlinear search spaces (Sec. 4.1) and balanced program selection with the three objectives (Sec. 4.2).

Table 1: Results on four benchmarks. The best result in each column is highlighted in bold.

METHOD	CELEBA		OFFICE-31		CITYSCAPES		NYU-V2	
	MR↓	$\Delta_p(\%)\uparrow$	MR↓	$\Delta_p(\%)\uparrow$	MR↓	$\Delta_p(\%)\uparrow$	MR↓	$\Delta_p(\%)\uparrow$
RLW	5.93	-1.46	8.00	-0.60	6.38	0.67	9.00	-1.65
DWA	7.23	-3.20	8.33	-0.71	4.38	4.64	8.40	-1.23
UW	6.10	-3.23	7.83	-0.56	7.13	7.42	5.83	0.53
MGDA	10.43	-14.85	7.00	-0.28	9.00	2.18	7.50	-0.35
IMTL-G	5.30	-0.84	9.83	-0.98	4.63	9.12	9.17	-0.24
Nash-MTL	5.53	-2.84	5.50	0.24	6.13	3.55	7.27	-0.72
FAMO	5.43	-1.21	6.50	-0.15	9.00	-0.59	3.90	5.12
FairGrad	5.85	-0.37	6.17	-0.68	4.38	9.61	**3.10**	5.24
PIVRG	5.78	-1.69	3.83	0.49	5.63	9.38	5.00	5.16
SVFair	4.50	0.63	1.67	**1.53**	5.25	**9.79**	3.33	**6.04**
EvoGrad	**3.95**	**1.01**	**1.33**	1.28	**4.13**	8.59	3.50	3.89

Table 2: Percentage of final selected programs containing nonlinear gradient operators.

DATASET	NONLINEAR DIRECTIONS (%)
CELEBA	30.72
OFFICE-31	59.81
CITYSCAPES	41.92
NYU-V2	77.95

To assess whether EvoGrad explores gradient aggregation beyond linear task-gradient weighting, Table 2 reports the percentage of gradient directions whose corresponding programs contain at least one nonlinear operator (element-wise multiplication $\odot$ or protected division $\oslash$) for each benchmark. Across all benchmarks, nonlinear gradient directions account for at least 30%, suggesting that the linear search space adopted in previous studies constrains the representational capacity of gradient aggregation. Particularly, the nonlinear search space of EvoGrad can capture diverse representations occuring in large-scale training data (e.g., over 200,000 training samples in CelebA). However, this strong representational capacity may also cause overfitting (see our Appendix B) when the training data is small (e.g., ~ 800 training samples in NYU-v2), leading to a more severe degradation of Δp.

The effect of our balanced program selection strategy is reflected in the detailed results provided in Appendix E.3. Since the selection strategy does not tend to favor any particular task, the merely decent Δp on a certain task (e.g., "DSLR" in Office-31, "depth estimation" in Cityscapes, "surface normal prediction" in NYU-v2) completely offsets the consistently best Δp on all other tasks, leading to a relatively worse Δp of EvoGrad on the three benchmarks. In contrast, the exceptionally high performance on that certain task achieved by the best baseline, SVFair, leads to its best Δp. This also indicates the necessity of combining the two evaluation metrics, Δp and MR, for a comprehensive evaluation of an MTL method.

5.4 ABLATION STUDY

To investigate the effectiveness of each objectives used in two candidate program selection strategies, balance-based (Sec. 4.2) and diversity-based (Sec. 4.3.1), we design several variants of EvoGrad and evaluate them on the Office-31 and CelebA datasets. Note that: 1) An objective is either used or not used simultaneously in both strategies; 2) When only one objective is used, there is no diversity-based selection strategy.

EvoGrad (f_{sum}) employs only f_{sum} as the objective function, and directly selects the best program with the highest f_{sum} value as final one (see the 1st row in Table 3).

EvoGrad ($f_{\text{sum+metric}}$) works in the same way as EvoGrad (f_{sum}), except that the metric-guided best program selection is used.

Table 3: Ablation study of the hybrid program selection strategy in EvoGrad on Office-31 and CelebA. Training time is measured in minutes. The best result in each column is highlighted in bold.

OBJECTIVES			SELECTION		CELEBA			OFFICE-31		
f_{sum}	f_{num}	f_{min}	DIVERSITY-BASED	METRIC-GUIDED	F1 ↑	$\Delta_p(\%)$ ↑	TIME ↓	ACC. ↑	$\Delta_p(\%)$ ↑	TIME ↓
✓	×	×	×	×	68.43	-1.72	**10.53**	93.51	0.44	**0.38**
✓	×	×	×	✓	68.62	-1.64	58.11	93.97	0.94	1.21
✓	✓	×	✓	✓	68.67	-0.94	29.03	94.00	0.97	0.92
✓	✓	✓	×	✓	67.96	-2.68	71.10	**94.33**	**1.35**	1.35
✓	✓	✓	✓	✓	**69.95**	**1.01**	68.40	94.27	1.33	1.39

EvoGrad ($f_{\text{sum}} + f_{\text{num}}$) employs both f_{sum} and f_{num} as objective function, and uses the metric-guided best program selection (see the 3rd row in Table 3).

EvoGrad$_{no_div}$ uses the same objective functions as EvoGrad. However, instead of using the diversity-based candidate program selection strategy, it directly selects the top 10 candidate programs with the highest f_{sum} values for subsequent metric-guided best program selection (see the 4th row in Table 3).

As shown in Table 3, the performance progressively improves as the number of objectives increases from one to three, i.e., EvoGrad (f_{sum})→ EvoGrad ($f_{\text{sum}} + f_{\text{num}}$) → EvoGrad, demonstrating the effectiveness of the three objectives. Moreover, the worse performance achieved by EvoGrad(f_{sum}) confirms that the correspondence between loss decrease and model performance does not always exist, and the performance improvement from EvoGrad (f_{sum}) to EvoGrad ($f_{\text{sum+metric}}$) indicates the importance of dataset-specific evaluation metrics in the best program selection. However, either incorporating additional objectives or adding metric-guided program evaluation also increases the computational cost, with the single-objective variant requiring the shortest training time.

Compared with EvoGrad, EvoGrad$_{no_div}$ achieves comparable performance on the Office-31 dataset (3-task), but performs substantially worse on CelebA (40-task). This difference may be attributed to the increasing complexity of the trade-offs among the three objectives as the number of tasks increases. In such a setting, selecting candidate programs solely according to f_{sum} may result in a set of programs with similar characteristics, failing to preserve the diversity that EvoGrad maintains. However, such program diversity is particularly beneficial when the number of tasks is large.

6 CONCLUSION

In this paper, we propose EvoGrad, an evolutionary framework based on GP for automatically discovering optimal gradient aggregation directions in a nonlinear search space composed of four operations (two linear and two nonlinear) over task gradients. To identify optimal directions efficiently, EvoGrad first discover balanced and diverse candidate programs using three objectives based on loss-decrease rates, and then applies a dataset-specific evaluation metric to select the best program. Experiments show that EvoGrad achieves competitive or improved performance compared with other MTL methods. The results also demonstrate that the composition-evolved and metric-driven programs possess powerful representation capability and exhibit optimization behavior beyond conventional gradient weighting methods.

Limitations and Future Work. EvoGrad incurs higher computational overhead than conventional Gradient Balancing Methods because of its population-based GP search and metric-guided candidate selection. Moreover, the compositional search space may overfit small datasets such as NYU-v2, resulting in unstable program selection and degraded generalization performance. Future work will focus on reducing the computational cost of evolutionary search and developing more robust strategies to mitigate overfitting.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing of the manuscript. We have not used generative AI tools for ideation, research methodology, experimental design, data analysis, or result generation. All research concepts, ideas, analyses, and experiments were developed and conducted by the authors. The use of generative AI was limited to improving the linguistic quality of the manuscript, with no involvement in the scientific content or data analysis. All AI-assisted text was reviewed, edited, and approved by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. In this study, no human subjects, sensitive personal data, or experiments that could potentially lead to ethical issues was involved. All experiments are conducted on publicly available datasets.

REPRODUCIBILITY STATEMENT

Our source code has been made publicly available in an anonymous repository `https://anonymous.4open.science/r/Review_for_ICLR_2027` to facilitate replication and verification. The experimental setup, including training steps, model configurations, and hardware details, is described in detail in the paper. Additionally, all datasets used in the paper are publicly available, ensuring consistent and reproducible evaluation results. We believe these measures will enable other researchers to reproduce our work and further advance the field.

REFERENCES

Idan Achituve, Idit Diamant, Arnon Netzer, Gal Chechik, and Ethan Fetaya. Bayesian uncertainty for gradient aggregation in multi-task learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 117–134. PMLR, 21–27 Jul 2024.

Hao Ban and Kaiyi Ji. Fair Resource Allocation in Multi-Task Learning. In *International Conference on Machine Learning*, volume 235, pp. 2715–2731, 2024.

Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *ACM Comput. Surv.*, 56(12), July 2024.

Weiyu Chen, Baijiong Lin, Xiaoyuan Zhang, Xi Lin, Han Zhao, Qingfu Zhang, and James Kwok. Gradient-based multi-objective deep learning: Algorithms, theories, applications, and beyond. *Transactions on Machine Learning Research*, 2026.

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*, volume 80, pp. 794–803, 10–15 Jul 2018.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

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

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

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

Shaojin Geng, Yi Mei, Fangfang Zhang, Lei Wang, Qidi Wu, and Mengjie Zhang. A novel multiobjective memetic algorithm for multi-task fuzzy flexible job shop scheduling. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 10(2):1765–1778, 2026.

Kaixuan Jia, Jianbin Ma, Xiaoying Gao, and Jiaxin Niu. Archive-based multiple feature construction methods using adaptive genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 503–506, 2024.

Joshua Knowles and Weijie Zheng. Evolutionary multiobjective optimization (emo). In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1432–1459, 2024.

John R Koza. *Genetic programming III: Darwinian invention and problem solving*, volume 3. Morgan Kaufmann, 1999.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, Shu Liu, Ivor W. Tsang, and James T. Kwok. Dual-balancing for multi-task learning. *Neural Networks*, 195:108317, 2026.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 18878–18890, 2021a.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 57226–57243, 2023.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021b.

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

Shikun Liu, Stephen James, Andrew Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

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

Yi Mei, Qi Chen, Andrew Lensen, Bing Xue, and Mengjie Zhang. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation*, 27(3): 621–641, 2023.

Wenyang Meng, Ying Li, Xiaoying Gao, and Jianbin Ma. Ensemble classifiers using multi-objective genetic programming for unbalanced data. *Applied Soft Computing*, 158:111554, 2024.

Wenyang Meng, Ying Li, Fan Zhang, Xiaoying Gao, and Jianbin Ma. Developing distance-based genetic programming classifiers by reconstructing datasets for imbalanced binary classification. *Pattern Recognition*, 173:112825, 2026.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 16428–16446, 17–23 Jul 2022.

Wenbin Pei, Bing Xue, Lin Shang, and Mengjie Zhang. High-dimensional unbalanced binary classification by genetic programming with multi-criterion fitness evaluation and selection. *Evolutionary Computation*, 30(1):99–129, 2022.

Xiaohan Qin, Xiaoxing Wang, and Junchi Yan. Revisiting fairness in multitask learning: A performance-driven approach for variance reduction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20492–20501, 2025.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Computer Vision – ECCV 2010*, pp. 213–226, 2010.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations*, 2018.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision – ECCV 2012*, pp. 746–760, 2012.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 9120–9132, 13–18 Jul 2020.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3614–3633, 2022.

Xiao Wang, Yuying Han, Dazi Li, Fei Zhang, and Min Tang. From gradient volume to shapley fairness: Towards fair multi-task learning. In *International Conference on Learning Representations*, volume 2026, pp. 129987–130040, 2026.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2021.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Feiyang Ye, Baijiong Lin, Xiao-Qun Cao, Yu Zhang, and Ivor W. Tsang. A first-order multi-gradient algorithm for multi-objective bi-level optimization. In *European Conference on Artificial Intelligence*, 2024a.

Feiyang Ye, Baijiong Lin, Zhixiong Yue, Yu Zhang, and Ivor W. Tsang. Multi-objective meta-learning. *Artificial Intelligence*, 335:104184, 2024b.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *2020 IEEE/CVF conference on computer vision and pattern recognition*, pp. 2633–2642, 2020a.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020b.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. In *Neural Information Processing Systems*, 2020c.

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. Las-gp: Harnessing large semantic libraries for evolutionary feature construction in symbolic regression. *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022.

A Derivation of the First-Order Descent Rate

Let $\ell_i(\theta)$ denote the loss of task i, and let

$$g_i = \nabla_\theta \ell_i(\theta)$$

be its gradient at the current parameters θ. Given an aggregation direction d and a step size $\eta > 0$, the parameters are updated according to

$$\theta' = \theta - \eta d.$$

Assuming that ℓ_i is differentiable at θ, its first-order Taylor expansion around θ gives

$$\ell_i(\theta + \Delta\theta) = \ell_i(\theta) + \nabla_\theta \ell_i(\theta)^\top \Delta\theta + o\big(\|\Delta\theta\|_2\big), \tag{11}$$

where $o(\|\Delta\theta\|_2)$ denotes the higher-order remainder. By setting $\Delta\theta = -\eta d$, we obtain

$$\begin{aligned} \ell_i(\theta - \eta d) &= \ell_i(\theta) - \eta \nabla_\theta \ell_i(\theta)^\top d + o\big(\eta \|d\|_2\big) \\ &= \ell_i(\theta) - \eta g_i^\top d + o\big(\eta \|d\|_2\big). \end{aligned} \tag{12}$$

Rearranging Eq. 12 and dividing both sides by η yield

$$\frac{\ell_i(\theta) - \ell_i(\theta - \eta d)}{\eta} = g_i^\top d + o\big(\|d\|_2\big). \tag{13}$$

Therefore, for a sufficiently small step size η,

$$\frac{1}{\eta}\left[\ell_i(\theta) - \ell_i(\theta - \eta d)\right] \approx g_i^\top d, \tag{14}$$

which gives Eq. 2 in the main text.

B Overfitting in the NYU-v2 Dataset

As shown in Fig. 2, EvoGrad consistently achieves higher training Δp values than SVFair, where Δp is calculated based on the corresponding STL training result. However, this advantage does not generalize to the test set. The test Δp, which is calculated based on the STL test result, is consistently lower for EvoGrad than for SVFair. This discrepancy between the training and test performance suggests that EvoGrad tends to overfit the training data on the NYU-v2 dataset, which contains only 795 training samples. Consequently, the overfitting limits the generalization ability of the programs discovered by EvoGrad, resulting in less satisfactory test Δp performance.

C EvoGrad Training Details

C.1 EvoGrad Evolution Process Details

This section we provide the details of EvoGrad evolution process in Algorithm 2.

C.2 Practical Speedup Strategy

Obtaining task-specific gradients typically requires t backward passes per training iteration. EvoGrad introduces additional overhead by evaluating a population of programs on high-dimensional gradients. Motivated by prior work on low-dimensional gradient representations and periodic task-weight updates (Chang et al., 2025; Navon et al., 2022), we employ coordinate subsampling and periodic evolution.

At iteration k, we uniformly sample a coordinate subset $\mathcal{S}_k \subseteq \{1, \ldots, D\}$ with $|\mathcal{S}_k| = \min(4096, D)$ and compute

$$\widetilde{g}_{i,k} = g_{i,k}[\mathcal{S}_k], \qquad \widetilde{d}_{p,k} = p\big(\{\widetilde{g}_{i,k}\}_{i=1}^{t}\big)\,.$$

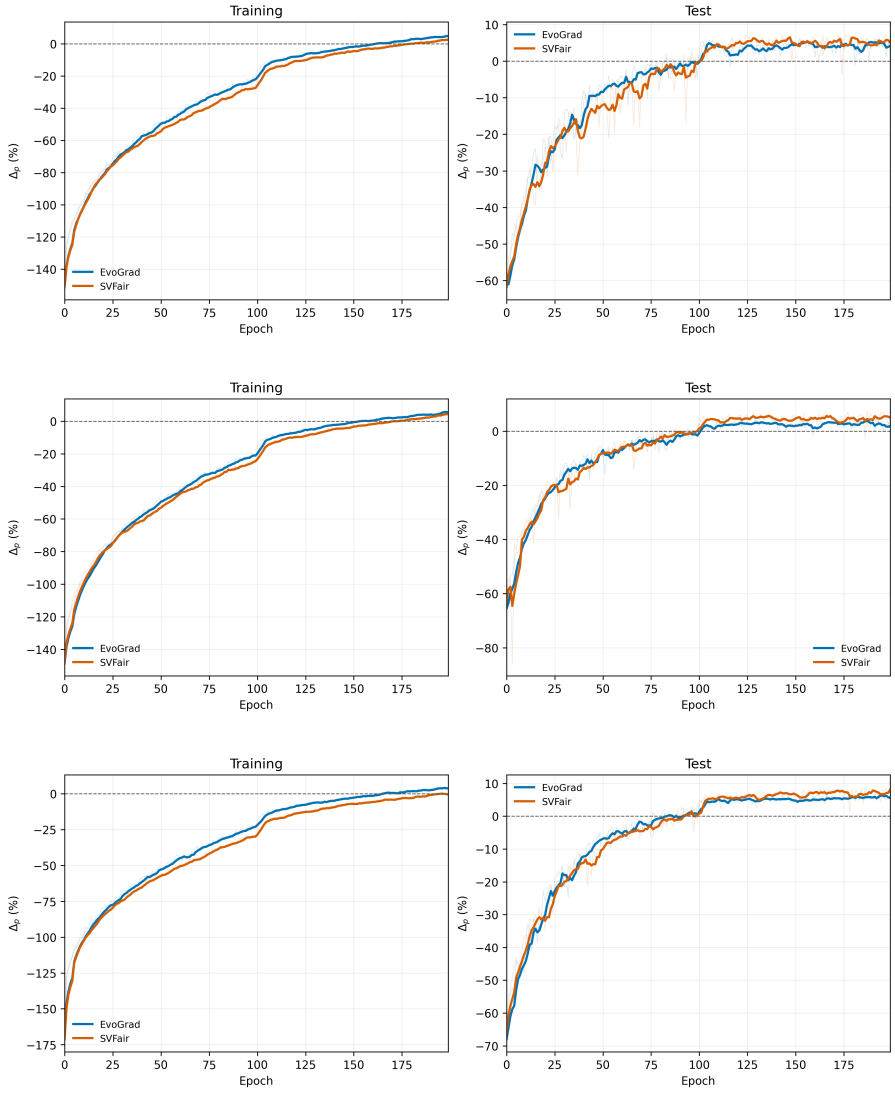

Figure 2: Training and test Δp of EvoGrad and SVFair on NYU-v2 over 200 epochs across three runs with different random seeds. The same random seed is used for both methods in each run.

Algorithm 2: Detailed evolutionary procedure of EVOGRAD

Input: Initial model parameters θ_k; Learning rate η_k; Population size N_{pop}; Number of GP generations E; Normalized task gradients G_k; Current generation e
Output: The best candidate gradient program p_k^*

Randomly initialize a GP population $\mathcal{P}_0$;
Initialize the Pareto archive $\mathcal{A}_0 \leftarrow \emptyset$;
if $e = 1$ **then**
  Evaluate each individual on G_k using three evaluation objectives based on Eq. 5;
  Perform fast non-dominated sorting and assign each individual a fitness equal to its non-domination level;
  Use tournament selection to choose individuals;
  Produce offspring $\mathcal{Q}_e$ through crossover and mutation;
end
for $e = 2$ **to** E **do**
  Merge $\mathcal{Q}_{e-1}$ and the parent population $\mathcal{P}_{e-1}$ to form a combined population $\mathcal{R}_e$;
  Perform fast non-dominated sorting and calculate the crowding distance for $\mathcal{R}_e$;
  Generate a new population $\mathcal{N}_e$ by selecting individuals from $\mathcal{R}_e$ based on non-dominated sorting and crowding distance;
  Let $\mathcal{Q}_e = \mathcal{N}_e$;
end
Remove overlapping individuals from the Pareto set and update the Pareto archive $\mathcal{A}_k$;
Construct a candidate subset $\mathcal{C}_k$; the maximum size of $\mathcal{C}_k$ is set to 10;
if $|\mathcal{A}_k| \leq 10$ **then**
  $\mathcal{C}_k = \mathcal{A}_k$;
end
else
  Use Eq. 6 to select individuals from $\mathcal{A}_k$ and form $\mathcal{C}_k$;
  Overlapping individual removing from $\mathcal{C}_k$;
  Rank the remaining programs $\mathcal{A}_k \setminus \mathcal{C}_k$ by crowding distance in descending order and add them to $\mathcal{C}_k$ until $|\mathcal{C}_k| = 10$;
end
foreach $p \in \mathcal{C}_k$ **do**
  Compute the candidate direction $d_{p,k} = p(G_k)$;
  Perform a temporary one-step update $\widetilde{\theta}_{p,k} = \theta_k - \eta_k d_{p,k}$;
  Evaluate $\widetilde{\theta}_{p,k}$ on the training mini-batch $\mathcal{B}_k$;
end
Select p_k^* based on Eq. 7;
return p_k^*;

The same subset is shared across all tasks and programs within each iteration. The three objectives are evaluated using $\widetilde{g}_{i,k}^{\top} \widetilde{d}_{p,k}$, reducing the cost of population evaluation. The metric-guided final program selection is then performed using the full-dimensional task gradients.

We evolve the population and generate offspring at the first iteration ($k = 1$) and subsequently when $k \bmod \tau = 0$, where $\tau = 5$. For a non-evolutionary iteration k, let k' denote the most recent iteration satisfying $k' = 1$ or $k' \bmod \tau = 0$. The programs in $\mathcal{A}_{k'}$ are reevaluated using the current sampled gradients, and their current nondominated subset is retained as $\mathcal{A}_k$, without performing a new evolutionary step.

D GENETIC PROGRAMING

D.1 GP PARAMETERS SETTINGS

Following commonly used GP settings in the existing literature (Pei et al., 2022; Meng et al., 2024), the GP hyperparameters used by EvoGrad are reported in Table 4.

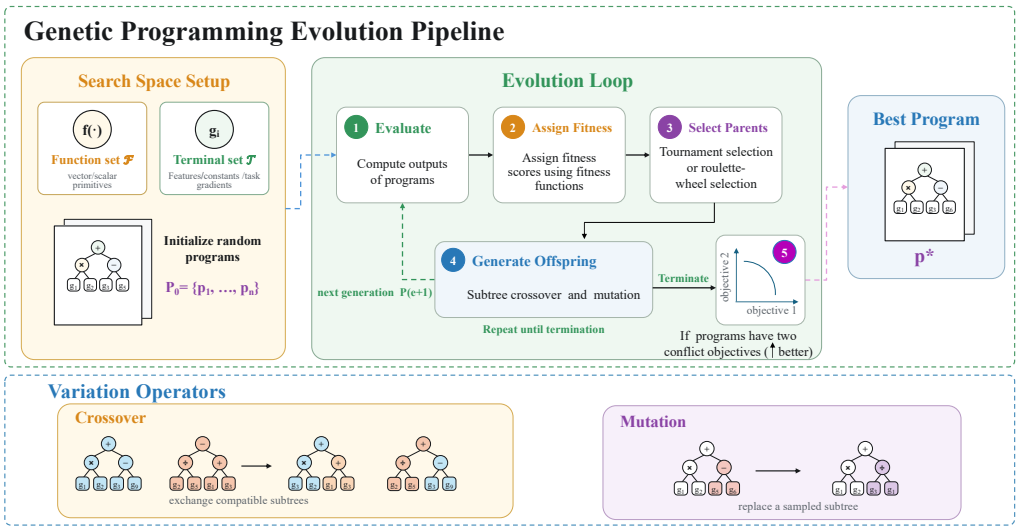

Figure 3: Overall GP evolution workflow.

Table 4: Parameter setting of GP.

Parameters	Parameter value
Function set	$+, -, \odot, \oslash$ (protected element-wise division)
Terminal set	Task gradients
Population size	500
Number of generations	50
Initialization	Ramped half-and-half
Selection method	Tournament selection method (size = 6)
Maximum tree depth	8
Mutation method	Uniform mutation (probability = 30%)
Crossover method	One-point crossover (probability = 80%)

D.2 Genetic Programming Operators and Evolutionary Search

GP is a population-based evolutionary method that searches for executable programs. Starting from a randomly initialized population, GP evaluates the candidate programs and iteratively generates new populations through parent selection, crossover, and mutation. The evolutionary process terminates when a predefined number of generations is reached. Fig. 3 illustrates the overall GP evolution workflow.

Tree-based program representation. In standard GP, each candidate program is typically represented in the form of a tree and can also be represented using other forms, such as linear representations, graphs, logic networks, and algebraic expressions. In the tree-based representation, the leaf nodes are referred to as terminals and may represent input variables, constants, or zero-argument functions. The internal nodes are function nodes, which may include arithmetic operators such as addition, subtraction, multiplication, and division, as well as other operations such as $\sin$, $\cos$, $\log$, logical operators, or conditional functions.

Fitness Functions and Values. Each individual in GP is evaluated using one or more objective functions, which are also commonly referred to as fitness functions. The value obtained by evaluating an individual with a fitness function is called its fitness value. Individuals with better fitness are generally more likely to be selected for reproduction and retained in the population than those with poorer fitness. In single-objective GP (SOGP), each individual is typically evaluated using a single fitness function and therefore has a single fitness value. In contrast, multi-objective GP (MOGP)

typically employs multiple fitness functions, often two or three, resulting in multiple fitness values for each individual.

Selections. Two commonly used selection methods in GP are tournament selection and roulette-wheel selection. In tournament selection, several individuals are randomly sampled from the population, and this group is referred to as a tournament. The individual with the best fitness is then selected from the tournament for reproduction. The tournament size is predetermined, and this process is repeated until a sufficient number of individuals have been selected to form the next generation. Roulette-wheel selection is a probability-based selection method in which individuals with better fitness values have a higher probability of being selected. The process is analogous to a roulette wheel, where each individual occupies a portion of the wheel proportional to its fitness value. Thus, individuals with higher fitness values occupy larger portions of the wheel and consequently have a higher probability of being selected. EvoGrad adopts tournament selection based on the NSGA-II ranking criterion.

Crossover and Mutation. Two primary genetic operators are commonly used to generate offspring in GP: crossover and mutation. The basic operation of the crossover operator is to randomly select two parent individuals and then randomly select a crossover point in each parent. The subtrees rooted at the selected crossover points are exchanged between the two parents to generate new offspring. The basic operation of the mutation operator is to randomly select a node in a parent individual as the mutation point and replace the subtree rooted at this node with a randomly generated subtree.

D.3 An example gredient program

This section provides a numerical example to illustrate how a gradient program maps multiple task gradients to a gradient aggregation direction. Consider a model with five parameters,

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)^\top,$$

and three tasks. At a particular training iteration, suppose that the task gradients are

$$\begin{aligned} g_1 &= (3,\ 2,\ 2,\ 2,\ 2)^\top, \\ g_2 &= (2,\ 3,\ 2,\ -2,\ 2)^\top, \\ g_3 &= (2,\ 2,\ 3,\ 2,\ -2)^\top. \end{aligned}$$

Since

$$\|g_1\|_2 = \|g_2\|_2 = \|g_3\|_2 = 5,$$

their normalized gradients are

$$\begin{aligned} \widehat{g}_1 &= (0.6,\ 0.4,\ 0.4,\ 0.4,\ 0.4)^\top, \\ \widehat{g}_2 &= (0.4,\ 0.6,\ 0.4,\ -0.4,\ 0.4)^\top, \\ \widehat{g}_3 &= (0.4,\ 0.4,\ 0.6,\ 0.4,\ -0.4)^\top. \end{aligned}$$

The gradient program illustrated in Fig. 4 represents the following aggregation rule:

$$d^* = \widehat{g}_1 + \widehat{g}_2 + \widehat{g}_2 \odot \widehat{g}_3 + \widehat{g}_1 \oslash \widehat{g}_3,$$

The intermediate element-wise terms are

$$\begin{aligned} \widehat{g}_2 \odot \widehat{g}_3 &= (0.16,\ 0.24,\ 0.24,\ -0.16,\ -0.16)^\top, \\ \widehat{g}_1 \oslash \widehat{g}_3 &= (1.50,\ 1.00,\ 0.67,\ 1.00,\ -1.00)^\top. \end{aligned}$$

Consequently, the program produces

$$d^* = (2.66,\ 2.24,\ 1.71,\ 0.84,\ -0.36)^\top.$$

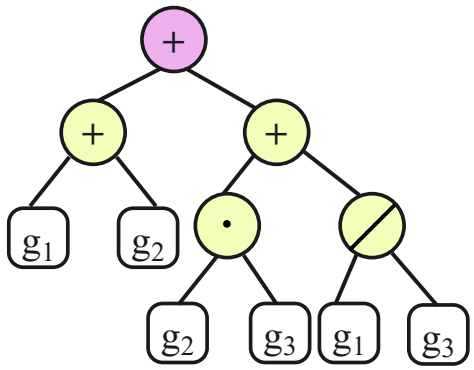

Figure 4: An example gredient program.

E EXPERIMENTS DETAILS

E.1 SYNTHETIC TWO-TASK PROBLEMS

We consider three synthetic two-task optimization problems adopted from previous (Liu et al., 2023; Ban & Ji, 2024; Qin et al., 2025). These examples cover different levels of gradient conflict and optimization difficulty. Each two-task scenario, consisting of $L_1(\theta)$ and $L_2(\theta)$, is defined over $\theta = (\theta_1, \theta_2)^\top \in \mathbb{R}^2$, allowing a comprehensive comparison of MTL methods across different task pairs. We use pink to highlight the differences among the three two-task settings. Following the setup in (Qin et al., 2025), we select five initial points: $\theta_{\text{init}} \in \{(-8.5, 7.5), (0, 0), (9.0, 9.0), (-7.5, -0.5), (9.0, -1.0)\}$. The Adam optimizer is employed with a learning rate of 1×10^{-3}. We compare EvoGrad with RLW (Lin et al., 2021), DWA (Liu et al., 2019), IMTL-G (Liu et al., 2021b), MGDA (Sener & Koltun, 2018), NashMTL (Navon et al., 2022), FAMO (Liu et al., 2023), FairGrad (Ban & Ji, 2024), and SVFair (Wang et al., 2026). The results are presented in Figs. 5–7.

Across the three toy examples, most methods either converge to a stationary point that satisfies the Pareto-stationarity condition and then remain there, or fail to reach the Pareto front. In contrast, EvoGrad consistently reaches the Pareto front and smoothly and comprehensively traces the Pareto front between Task A and Task C. Its behavior is similar to that of the fairness-based methods, FairGrad and SVFair.

Two-Task A: FAMO. Following (Liu et al., 2023), the two objectives are defined as

$$L_1(\boldsymbol{\theta}) = 0.1\,(f_1(\boldsymbol{\theta})g_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta})h_1(\boldsymbol{\theta})),$$
$$L_2(\boldsymbol{\theta}) = f_1(\boldsymbol{\theta})g_2(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta})h_2(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2)$, and

$$f_1(\boldsymbol{\theta}) = \max\{\tanh(0.5\theta_2),\, 0\}\,,$$
$$f_2(\boldsymbol{\theta}) = \max\{\tanh(-0.5\theta_2),\, 0\}\,,$$
$$g_1(\boldsymbol{\theta}) = \log\big(\max\big\{|0.5(-\theta_1 - 7) - \tanh(-\theta_2)|\,, 5 \times 10^{-6}\big\}\big) + 6,$$
$$g_2(\boldsymbol{\theta}) = \log\big(\max\big\{|0.5(-\theta_1 + 3) - \tanh(-\theta_2) + 2|\,, 5 \times 10^{-6}\big\}\big) + 6,$$
$$h_1(\boldsymbol{\theta}) = \frac{(-\theta_1 + 7)^2 + 0.1(-\theta_2 - 8)^2}{10} - 20,$$
$$h_2(\boldsymbol{\theta}) = \frac{(-\theta_1 - 7)^2 + 0.1(-\theta_2 - 8)^2}{10} - 20.$$

Two-Task B: FairGrad Variant. The two objectives are adapted from the formulation of Ban & Ji (2024) and defined as

$$L_1(\boldsymbol{\theta}) = 0.1 f_1(\boldsymbol{\theta})g_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta})h_1(\boldsymbol{\theta}),$$
$$L_2(\boldsymbol{\theta}) = f_1(\boldsymbol{\theta})g_2(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta})h_2(\boldsymbol{\theta}),$$

where

$$
\begin{aligned}
f_1(\boldsymbol{\theta}) &= \max\{\tanh(0.5\theta_2),\, 0\}\,, \\
f_2(\boldsymbol{\theta}) &= \max\{\tanh(-0.5\theta_2),\, 0\}\,, \\
g_1(\boldsymbol{\theta}) &= \log\big(\max\big\{|0.5(-\theta_1-7)-\tanh(-\theta_2)|\,, 5\times 10^{-6}\big\}\big)+6, \\
g_2(\boldsymbol{\theta}) &= \log\big(\max\big\{|0.5(-\theta_1+3)-\tanh(-\theta_2)+2|\,, 5\times 10^{-6}\big\}\big)+6, \\
h_1(\boldsymbol{\theta}) &= \frac{(-\theta_1+7)^2+0.1(-\theta_1-8)^2}{10}-20, \\
h_2(\boldsymbol{\theta}) &= \frac{(-\theta_1-7)^2+0.1(-\theta_1-8)^2}{10}-20.
\end{aligned}
$$

Two-Task C: PIVRG. Following Qin et al. (2025), the two objectives are defined as

$$
\begin{aligned}
L_1(\boldsymbol{\theta}) &= f_1(\boldsymbol{\theta})g_1(\boldsymbol{\theta})+f_2(\boldsymbol{\theta})h_1(\boldsymbol{\theta}), \\
L_2(\boldsymbol{\theta}) &= f_1(\boldsymbol{\theta})g_2(\boldsymbol{\theta})+f_2(\boldsymbol{\theta})h_2(\boldsymbol{\theta}),
\end{aligned}
$$

where

$$
\begin{aligned}
f_1(\boldsymbol{\theta}) &= \max\{\tanh(0.5\theta_2),\, 0\}\,, \\
f_2(\boldsymbol{\theta}) &= \max\{\tanh(-0.5\theta_2),\, 0\}\,, \\
g_1(\boldsymbol{\theta}) &= \log\big(\max\big\{|0.5(-\theta_1-7)-\tanh(-\theta_2)|\,, 5\times 10^{-6}\big\}\big)+6, \\
g_2(\boldsymbol{\theta}) &= \log\big(\max\big\{|0.5(-\theta_1+3)-\tanh(-\theta_2)+2|\,, 5\times 10^{-6}\big\}\big)+6, \\
h_1(\boldsymbol{\theta}) &= \frac{(-\theta_1+7)^2+0.1(-\theta_2-8)^2}{10}-20, \\
h_2(\boldsymbol{\theta}) &= \frac{(-\theta_1-7)^2+0.1(-\theta_2-8)^2}{10}-20.
\end{aligned}
$$

E.2 Experiments Details For Image-Level Classification And Scene Understanding

All experiments were conducted using NVIDIA GeForce RTX 4090 and RTX 5090 GPUs. The dataset-specific experimental configurations are described below.

CelebA. CelebA (Liu et al., 2015) is a large-scale facial-attribute dataset containing more than 200,000 celebrity images. Each image is annotated with 40 binary attributes, such as smiling, wavy hair, and mustache. CelebA is formulated as a 40-task image-level classification problem, where each task predicts the presence of a specific attribute. We adopt a shared 9-layer convolutional neural network (CNN) and a task-specific linear classification head for each attribute. The model is trained for 15 epochs using Adam with a learning rate of 3×10^{-4} and a batch size of 256.

Office-31. Office-31 (Saenko et al., 2010) contains 4,110 images collected from three domains: Amazon, DSLR, and Webcam. Each domain contains images from the same 31 object categories and is treated as an individual classification task. An ImageNet-pretrained ResNet-18 is employed as the shared feature extractor, together with a task-specific linear classification head for each domain. All input images are resized to 224×224. The model is trained for 20 epochs using Adam with a learning rate of 10^{-4}, a weight decay of 10^{-5}, and a batch size of 64.

Cityscapes. Cityscapes (Cordts et al., 2016) is an urban scene-understanding dataset containing 2,975 training images and 500 validation images. Two tasks are considered: 7-class semantic segmentation and depth estimation. Following prior work, an MTAN architecture built on a SegNet backbone is adopted, with task-specific attention modules for the individual tasks. The input images are resized to 128×256. The model is trained for 200 epochs using Adam with a batch size of 24. The learning rate is set to 10^{-4} for the first 100 epochs and reduced to 5×10^{-5} for the remaining 100 epochs.

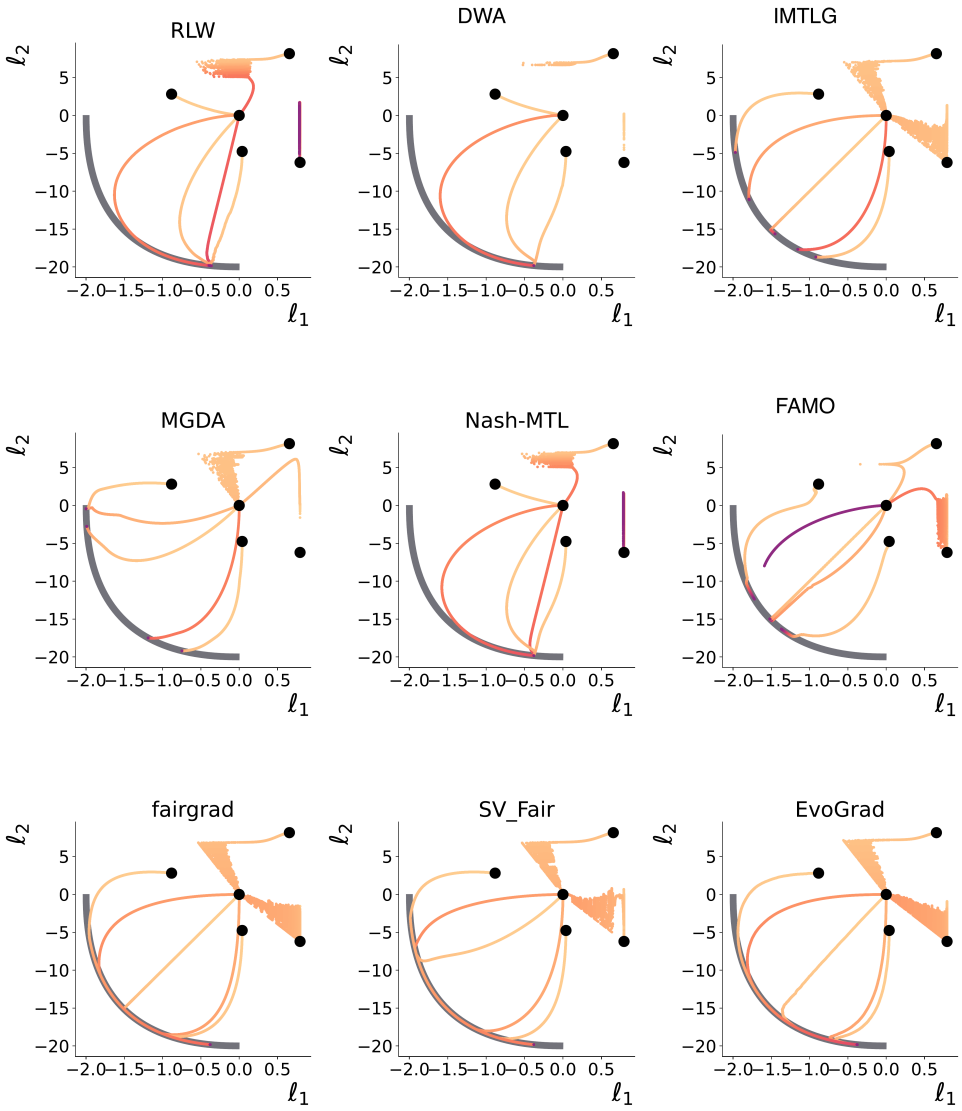

Figure 5: Optimazation trajectories for selected methods on the FAMO two-task example. Starting points are indicated by black dots, and the Pareto front is represented by thick gray lines.

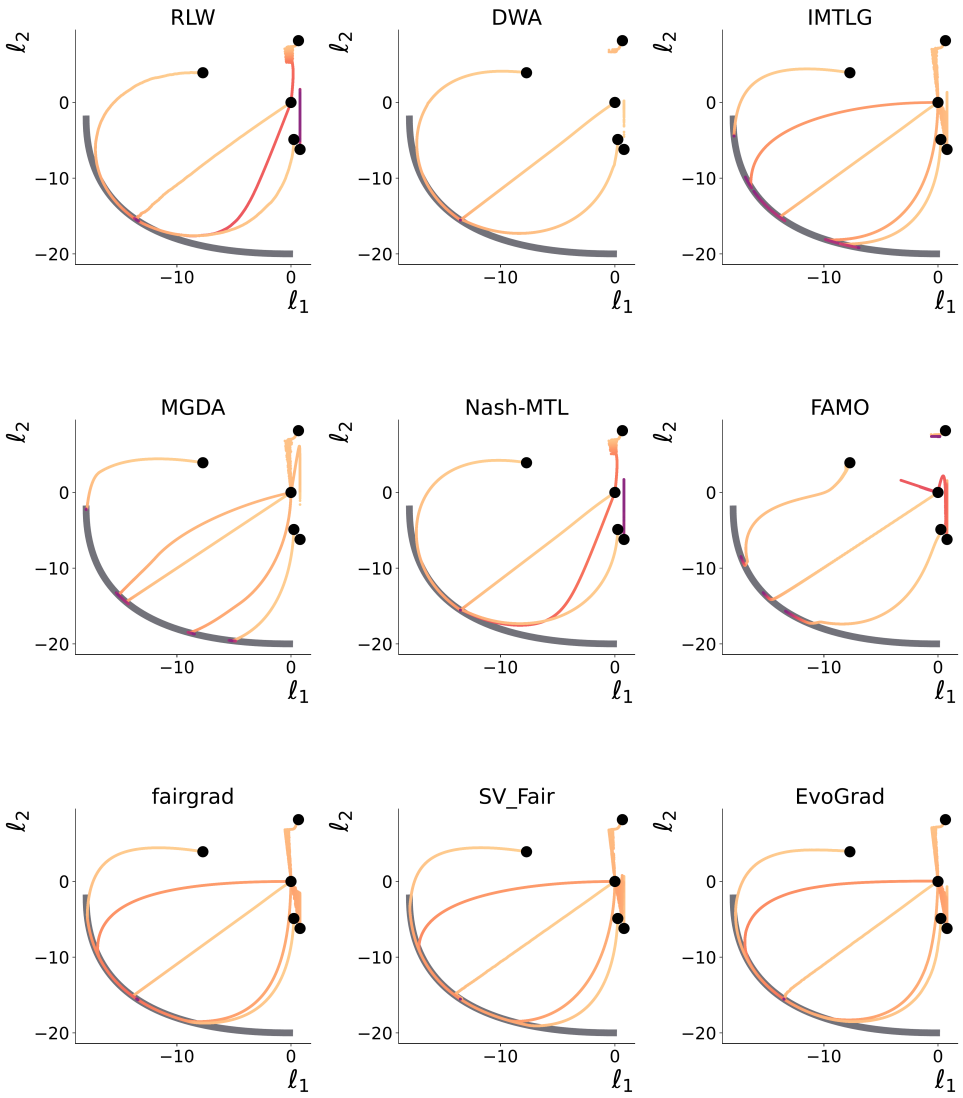

Figure 6: Optimazation trajectories for selected methods on the FairGrad Variant two-task example. Starting points are indicated by black dots, and the Pareto front is represented by thick gray lines.

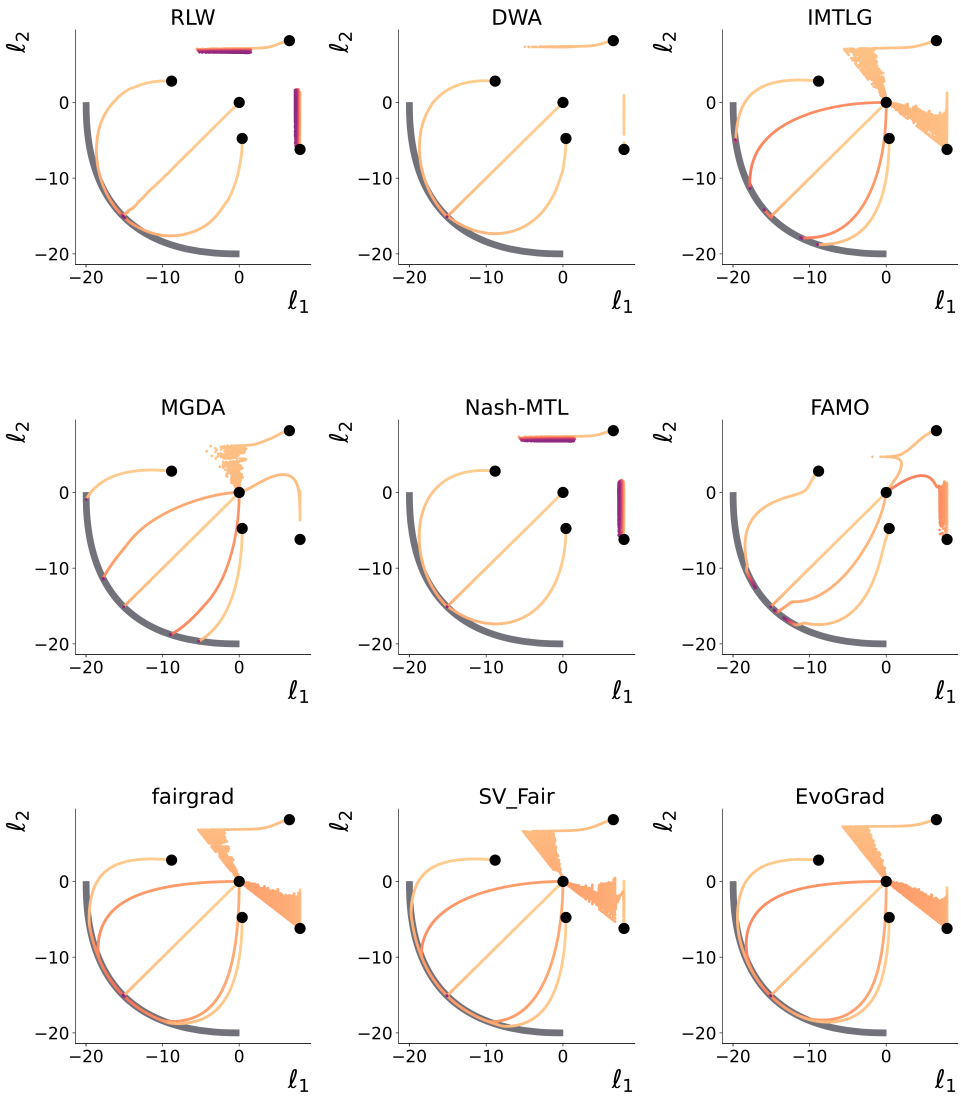

Figure 7: Optimazation trajectories for selected methods on the PIVRG two-task example. Starting points are indicated by black dots, and the Pareto front is represented by thick gray lines.

NYU-v2. NYU-v2 (Silberman et al., 2012) is an indoor scene-understanding dataset containing 795 training images and 654 test images. Three tasks are considered: 13-class semantic segmentation, depth estimation, and surface normal prediction. The same MTAN architecture built on a SegNet backbone is adopted, with task-specific attention modules for the three tasks. The input images are resized to 288×384. The model is trained for 200 epochs using Adam with a batch size of 6. As in the Cityscapes experiment, the learning rate is set to 10^{-4} for the first 100 epochs and reduced to 5×10^{-5} for the remaining 100 epochs.

E.3 Experiments Results Details

We provide the detailed experimental results corresponding to Table 1. Each experiment is repeated three times with different random seeds, and the average is reported. Table 5 reports the results on the Office-31 dataset, Table 6 reports the results on the Cityscapes dataset, and Table 7 reports the results on the NYU-v2 dataset.

Table 5: Results on the Office-31 (3-task) dataset. Each experiment is repeated three times with different random seeds, and the average results are reported. The best result in each column is highlighted in bold.

Method	Office-31				
	Amazon ↑	DSLR ↑	Webcam ↑	MR↓	$\Delta_p(\%)$ ↑
STL	86.61	95.63	96.85	–	–
RLW	83.82	96.99	96.85	8.00	-0.60
DWA	83.87	96.99	96.48	8.33	-0.71
UW	83.82	97.27	96.67	7.83	-0.56
MGDA	85.47	95.90	97.03	7.00	-0.28
IMTL-G	83.41	96.72	96.48	9.83	-0.98
Nash-MTL	85.01	97.54	97.41	5.50	0.24
FAMO	85.07	96.17	97.59	6.50	-0.15
FairGrad	82.11	97.54	97.96	6.17	-0.68
PIVRG	85.24	97.81	97.59	3.83	0.49
SVFair	85.81	**99.45**	98.33	1.67	**1.53**
EvoGrad	**86.21**	98.09	**98.52**	**1.33**	1.28

Table 6: Results on the Cityscapes (2-task) dataset. Each experiment is repeated three times with different random seeds, and the average results are reported. The best result in each column is highlighted in bold.

Method	Segmentation		Depth		MR↓	$\Delta_p(\%)$ ↑
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓		
STL	74.15	93.24	0.0159	62.65	–	–
RLW	73.94	93.17	0.0176	54.05	6.38	0.67
DWA	**74.34**	**93.21**	0.0160	50.77	4.38	4.64
UW	68.47	91.57	0.0150	41.67	7.13	7.42
MGDA	70.70	92.29	0.0188	42.21	9.00	2.18
IMTL-G	72.94	93.17	0.0145	44.24	4.63	9.12
Nash-MTL	74.03	93.16	0.0162	52.42	6.13	3.55
FAMO	70.95	92.45	0.0203	43.55	9.00	-0.59
FairGrad	73.77	93.15	0.0150	41.74	4.38	9.61
PIVRG	71.62	92.75	0.0160	**36.29**	5.63	9.38
SVFair	71.34	92.82	**0.0143**	41.77	5.25	**9.79**
EvoGrad	73.61	93.16	0.0157	41.41	**4.13**	8.59

Table 7: Results on the NYU-v2 (3-task) dataset. Each experiment is repeated three times with different random seeds, and the average results are reported. The best result in each column is highlighted in bold.

Method	Segmentation		Depth		Surface Normal					MR ↓	Δ_p (%) ↑
					Angle Distance ↓		Within t° ↑				
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Mean	Median	11.25	22.5	30		
STL	40.57	64.93	0.6033	0.2656	24.35	17.87	32.18	59.84	71.12	–	–
RLW	40.69	66.37	0.5487	0.2306	27.15	22.34	24.48	50.51	63.64	9.00	-1.65
DWA	40.52	66.54	0.5440	0.2308	27.03	22.09	25.04	50.98	63.99	8.40	-1.23
UW	40.92	66.93	0.5303	0.2269	26.39	21.44	25.92	52.29	65.36	5.83	0.53
MGDA	33.68	63.64	0.5814	0.2389	**23.94**	**17.61**	**33.31**	**60.28**	**71.55**	7.50	-0.35
IMTL-G	39.94	66.16	0.5617	0.2401	25.74	20.23	28.05	54.79	67.23	9.17	-0.24
Nash-MTL	41.05	66.62	0.5378	0.2281	27.06	22.18	25.09	50.80	63.80	7.27	-0.72
FAMO	40.30	66.92	**0.5111**	**0.2126**	24.73	18.82	30.53	57.79	69.79	3.90	5.12
FairGrad	**41.78**	**67.77**	0.5176	0.2198	24.71	18.90	30.42	57.63	69.71	**3.10**	5.24
PIVRG	40.39	66.71	0.5261	0.2231	24.07	18.05	32.30	59.40	71.07	5.00	5.16
SVFair	40.86	66.86	0.5209	0.2171	24.03	17.97	32.57	59.55	71.15	3.33	**6.04**
EvoGrad	41.11	67.38	0.5165	0.2174	25.30	19.72	28.62	55.91	68.40	3.50	3.89