# *GO4Align*: Group Optimization for Multi-Task Alignment

**Jiayi Shen**[1], **Qi (Cheems) Wang**[2], **Zehao Xiao**[1]*, **Nanne Van Noord**[1], **Marcel Worring**[1]
[1]University of Amsterdam, Amsterdam, the Netherlands
[2]Department of Automation, Tsinghua University, Beijing, China

## Abstract

This paper proposes ***GO4Align***, a multi-task optimization approach that tackles task imbalance by explicitly aligning the optimization across tasks. To achieve this, we design an adaptive group risk minimization strategy, comprising two techniques in implementation: (i) dynamical group assignment, which clusters similar tasks based on task interactions; (ii) risk-guided group indicators, which exploit consistent task correlations with risk information from previous iterations. Comprehensive experimental results on diverse benchmarks demonstrate our method's performance superiority with even lower computational costs.

## 1 Introduction

Multi-task learning is a promising paradigm for handling several tasks simultaneously using a unified architecture. It can achieve data efficiency, improve generalization, and reduce computation costs compared with addressing each task individually [1]. Due to these benefits, there is a growing surge of applications with multi-task learning in several domains, *e.g.*, natural language processing [2–4], computer vision [1, 5, 6] and reinforcement learning [7, 8]. The crux of multi-task learning is to enable positive transfer among tasks while avoiding negative transfer, which usually exists among irrelevant tasks [9–11].

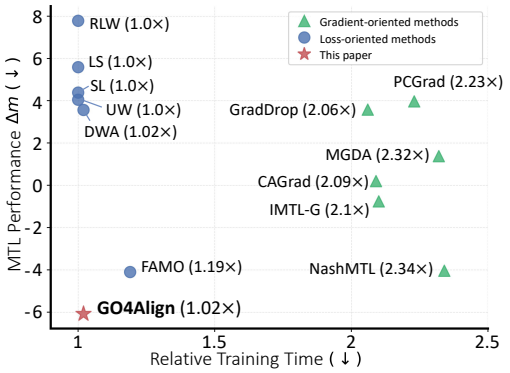


Figure 1: **Performance and computational efficiency evaluation for MTO methods evaluated on** `NYUv2`. Each method's training time is relative to a baseline method, which minimizes the sum of task-specific empirical risks. Left-bottom marks comprehensive optimal results.

**Existing Challenges:** In avoiding the negative transfer, numerous multi-task optimization (MTO) methods [5, 9, 12–14] have emerged and attracted rising attention in recent years. A lasting concern in MTO is the *task imbalance issue*. It describes a phenomenon where some tasks are severely under-optimized [1], which can lead to worse overall performance with larger convergence differences across tasks.

To deal with the task imbalance issue, scaling methods are proposed for MTO. According to differences in scaling manipulations, we roughly divide MTO methods into *gradient-oriented* [12, 15–18] and *loss-oriented* [5, 6, 19, 20]. The former tends to exhibit impressive results at the expense of higher computational or memory requirements during training time due to the assessment of per-task

*Correspondence to: Zehao Xiao <zxiao4ai@gmail.com>.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).


https://doi.org/10.52202/079017-3537

gradients [19].[2] In contrast, the latter preserves training-time efficiency but usually suffers from unsatisfactory overall performance. As shown in Fig. 1, most existing methods cannot simultaneously achieve superior performance and computational efficiency.

**Proposed Solution:** To improve the overall performance and maintain computational and memory efficiency, we propose **G**roup **O**ptimization **for** multi-task **Align**ment (***GO4Align***), a novel and effective loss-oriented method in MTO. As shown in Fig. 2, this work identifies *multi-task alignment* as a crucial factor in solving task imbalance, which means learning progress across tasks should synchronously achieve superior performance over all tasks. The proposed model dynamically aligns learning progress across tasks by exploiting group-based task interactions for multi-task empirical risk minimization. The rationale behind this is that groupings can implicitly capture task correlations for more effective multi-task alignment and thus help multi-task optimizers benefit from positive interactions among relevant tasks. The primary contribution is two-fold:

- As a new member of the loss-oriented MTO branch, ***GO4Align*** recasts the task imbalance issue to a bi-level optimization problem, yielding an adaptive group risk minimization principle for MTO. Such a principle allocates weights over task losses at a group level to achieve learning progress alignment among relevant tasks.
- We develop a heuristic optimization pipeline in ***GO4Align*** to tractably achieve the principle, involving *dynamical group assignment* and *risk-guided group indicators*. The pipeline incorporates beneficial task interactions into the group assignments and exploits task correlations for multi-task alignment, improving overall multi-task performance.

Experimental results show that our approach can outperform existing state-of-the-art baselines in extensive benchmarks. Moreover, it does not sacrifice computational efficiency.

## 2 Preliminary

**Notations.** This work considers a multi-task problem over an input space $\mathcal{X}$ and a collection of task target spaces $\{\mathcal{Y}^m\}_{m=1}^M$, where $M \geq 2$ denotes the number of tasks. A composite dataset for multi-task learning is $\{(\boldsymbol{x}_n, y_n^1, ..., y_n^M)\}_{n=1}^N$, where $N$ is the number of training samples. Let $\boldsymbol{\theta}^s$ and $\boldsymbol{\theta}^m$ respectively be the shared and task-specific parameters in a given multi-task model, thus we have a parametric hypothesis class for the $m$-th task as $f(\boldsymbol{x}_n; \boldsymbol{\theta}^s, \boldsymbol{\theta}^m) : \mathcal{X} \to \mathcal{Y}^m$. Then the empirical risk for the $m$-th task can be written as $\hat{\mathcal{L}}^m(\boldsymbol{\theta}^s, \boldsymbol{\theta}^m) = \frac{1}{N}\sum_{n=1}^N \ell^m(f(\boldsymbol{x}_n; \boldsymbol{\theta}^s, \boldsymbol{\theta}^m), y_n^m)$, where $\ell^m(\cdot,\cdot) : \mathcal{Y}^m \times \mathcal{Y}^m \to \mathbb{R}_+$ denotes the task-specific loss function. The ultimate goal of MTO is to achieve superior performance over all tasks.

**Scale Empirical Risk Minimization (Scale-ERM).** As preliminary, we recap a representative and related strategy in MTO, Scale-ERM, through the lens of risk minimization. Scale-ERM introduces a task-specific weight $\lambda^m \geq 0$ to scale the corresponding empirical risk. For conciseness, this principle is formulated using vector notations. Here, $\boldsymbol{\lambda} = \left[\lambda^1, \lambda^2, \cdots, \lambda^M\right]^\top \in \mathbb{R}^M$ represents a $M$-dimensional vector comprising all task-specific weights. And $\hat{\boldsymbol{L}}(\boldsymbol{\theta}) = \left[\hat{\mathcal{L}}^1(\boldsymbol{\theta}^s, \boldsymbol{\theta}^1), \hat{\mathcal{L}}^2(\boldsymbol{\theta}^s, \boldsymbol{\theta}^2), \cdots, \hat{\mathcal{L}}^M(\boldsymbol{\theta}^s, \boldsymbol{\theta}^M)\right]^\top \in \mathbb{R}^M$ denotes the corresponding vector of empirical risks, where $\boldsymbol{\theta} = \{\boldsymbol{\theta}^s, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \cdots, \boldsymbol{\theta}^M\}$ represents all learnable parameters in a multi-task backbone network. Thus, we obtain the objective of Scale-ERM as follows:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\lambda}} \boldsymbol{\lambda}^\top \hat{\boldsymbol{L}}(\boldsymbol{\theta}) + \Omega(\boldsymbol{\lambda}), \tag{1}$$

where $\Omega(\boldsymbol{\lambda})$ is a regularization term over task weights, designed to prevent the rapid collapse of these weights to zero, as discussed in Kendall et al. [5]. When all task weights are the same in scale, Scale-ERM will degenerate to the most simple strategy in MTO, where each task is treated equally during the joint training.

In Scale-ERM, each task weight controls task-specific learning progress, either by adapting the task-specific weights with loss information (loss-oriented) or by operating on task-specific gradients (gradient-oriented). As previously indicated, there still remains a research gap in MTO to improve multi-task performance without affecting computational efficiency.

[2] In MTO, computational efficiency refers to training time efficiency.

## 3 Methodology

In resolving the task imbalance issue effectively and efficiently, we develop ***GO4Align*** in this section. Our approach relies on grouping-based task interactions to align learning progress across tasks. As a new member of the loss-oriented branch, ***GO4Align*** holds the advantage of computational efficiency without the requirements of per-task gradients.

**Motivation of Multi-Task Alignment.** Empirically, we can observe that the overall performance is worse when there is a larger *convergence difference* between tasks. The convergence difference is measured by the standard deviation of the task-specific epoch numbers to reach convergence. As shown in Fig. 2, UW [5] underperforms FAMO [14] in terms of a overall MTL metric $\boldsymbol{\Delta m}\%$ (lower is better); while UW has a larger convergence difference than FAMO. Intuitively, a larger convergence difference means that per-task training dynamics are more asynchronous, usually leading to worse overall performance.

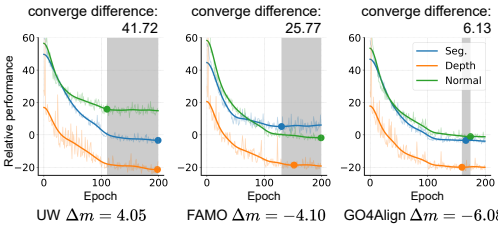


Figure 2: **Multi-task alignment and effects on performance.** We visualize relative task performance curves (lower is better) over training epochs. Better overall performance usually occurs with lower convergence differences. Our method effectively reduces the convergence difference and achieves a better overall performance.

To perform MTO, we consider aligning tasks with group information in the multi-task risk minimization. We first present an adaptive group risk minimization principle for MTO, which targets the alignment of tasks' learning progress from grouping-based task interactions. Then, in tractable problem-solving, we decompose the whole optimization process into two entangled phases: *(i) dynamical group assignment* and *(ii) risk-guided group indicators*. The pseudo-code of ***GO4Align*** is provided in Appendix A.

### 3.1 Adaptive Group Risk Minimization Principle

Recent advances [21, 22] have explored incorporating multi-task grouping in feature sharing and revealed its benefit of aligning the learning progress through task interactions. Nevertheless, their grouping mechanisms ignore monitoring the learning progress, *e.g.*, failure to capture variations in loss scales among tasks, weakening the effectiveness of multi-task alignment.

As a result, we design a new grouping mechanism for MTO with *task-specific learning dynamics*, which directly impacts the *convergence behaviors in optimization*. This induces the adaptive group risk minimization principle suitable for multi-task alignment. The hypothesis is that the dynamical grouping tends to implicitly exploit task correlations [21] and encourages beneficial task interactions from empirical risk information along the learning progress. Meanwhile, such a principle retains computational efficiency as it avoids the computations of per-task gradients.

**Adaptive Group Risk Minimization (AGRM).** We first achieve beneficial task interactions by producing task weights with a grouping mechanism, which is adaptive to various loss scales and their learning dynamics over time. Then, we recast the task imbalance issue with the grouping mechanism into a bi-level optimization problem: (i) In the *lower-level* optimization, the model aims to cluster $M$ tasks of interest into $K$ groups. This implicitly exploits task correlations, where similar tasks should be clustered into one group, yielding more beneficial task interactions. With group assignments, group weights are designed to conduct learning progress alignment at the group level. The group weights in the multi-task objective are motivated as an extension of the observation that similar tasks benefit greatly from training together through parameter sharing [21]. (ii) In the *upper-level* optimization, the proposed principle updates model parameters from the grouped empirical risks, which implicitly relies on the lower-level optimized results. We illustrate ***GO4Align***'s bi-level optimization with grouping-based task interactions in Fig. 3.

Given $K$ as the number of groups, we denote the assignment matrix as $\boldsymbol{\mathcal{G}}_t \in \mathbb{R}^{K\times M}$, where $\mathcal{G}_t(k,m)$ equals 1 if the $k$-th group contains the $m$-th task and 0 otherwise. Note that $\boldsymbol{\mathcal{G}}_t$ is updated in the optimization, with $t$-th indexing iteration step. The group number generally is smaller than the task number, *e.g.*, $1 < K \leq M$. To balance different groups, we place weights over groups $\boldsymbol{\omega}_t = \left[\omega_t^1, \omega_t^2, \cdots, \omega_t^K\right]^\top \in \mathbb{R}^K$, where $\omega_t^k \geq 0$ is specific to the $k$-th group at the $t$-th iteration.


https://doi.org/10.52202/079017-3537

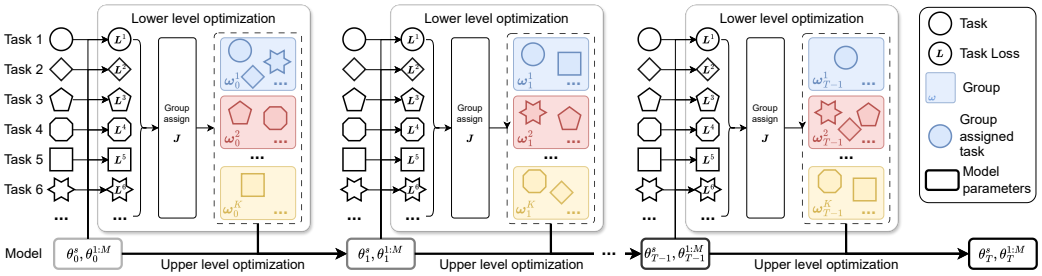


Figure 3: ***GO4Align* using adaptive group risk minimization in the bi-level optimization framework**. In the lower-level optimization, the model assigns tasks to groups with different group weights, encouraging task interactions and aligning learning progress. Such group information is nested into the upper-level optimization for updating the multi-task model's parameters.

Formally, we formulate the bi-level optimization problem as:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\omega}_t^\top \boldsymbol{\mathcal{G}}_t \hat{\boldsymbol{L}}(\boldsymbol{\theta}) \quad s.t.\ \{\boldsymbol{\omega}_t, \boldsymbol{\mathcal{G}}_t\} = \arg\min_{\boldsymbol{\omega}, \boldsymbol{\mathcal{G}}} \boldsymbol{J}(\boldsymbol{\omega}, \boldsymbol{\mathcal{G}}; \boldsymbol{\theta}_t), \tag{2}$$

where $\boldsymbol{\omega}_t$ and $\boldsymbol{\mathcal{G}}_t$ reflect adaptive group information in the lower-level optimization. $\boldsymbol{J}(\boldsymbol{\omega}, \boldsymbol{\mathcal{G}}; \boldsymbol{\theta}_t)$ is the corresponding optimization objective for aligning the learning progress across tasks at the group levels, which is further explained in Sec. 3.2.

To be specific, we perform the bi-level optimization in Eq. (2) according to the following steps. For the $t$-th iteration, we first compute the group information $\boldsymbol{\omega}_t$ and $\boldsymbol{\mathcal{G}}_t$ in the lower-level optimization; and then we update the model's parameter in the upper-level optimization. As a result, we obtain the updated parameter $\boldsymbol{\theta}_{t+1}$, which is used to compute task-specific risk information at the next iteration. In the proposed principle, intra-group tasks share the same scaling weight $\omega^k \geq 0$ to prevent similar tasks from inconsistent learning progress, improving knowledge sharing among similar tasks.

Important in AGRM is to accommodate the grouping during training dynamically. In practice, the proposed principle is compatible with any gradient-based optimizer, such as SGD and Adam [23], yielding dynamical training for each task. As a new member of the loss-oriented branch, the grouping assignment matrix and group weights in ***GO4Align*** can sufficiently utilize the loss information over time to adaptively assign tasks and weight groups.

Unlike prior works on multi-task grouping [21, 22], which require group-specific architectures, our proposed principle focuses on group-specific scaling and adaptively executing grouping operations. Considering the differences in architecture and optimization within grouping mechanisms, we provide two insights: (i) the criteria for grouping tasks should take both *learning dynamics* and *loss scales* into consideration so that similar tasks can benefit from each other's intermediate feature information and boost performance; (ii) adaptive grouping *aligns learning progress across tasks* and provides a more effective way for across-task information transfer.

### 3.2 Dynamical Group Assignment

In solving the optimization problem in Eq. (2), the main challenge lies in the involvement of discrete and continuous variables, which are implicitly entangled in the objective. In detail, the lower-level optimization requires adaptively adjusting the discrete variable $\boldsymbol{\mathcal{G}}_t$ and the continuous variable $\boldsymbol{\omega}_t$ for the learning progress alignment such that the model's parameter $\boldsymbol{\theta}$ updates from the lastest grouping information.

Before executing the *lower-level* optimization, we need to introduce task-specific group indicators $\boldsymbol{\gamma}_t(\boldsymbol{\theta}_t) = \left[\gamma_t^1(\boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^1), \gamma_t^2(\boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^2), \cdots, \gamma_t^M(\boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^M)\right]^\top \in \mathbb{R}^M$. In general, this involves the entanglement of the model's parameters, which is obtained from high-level optimization. These group indicators work for exploiting cross-task correlations along the learning progress and provide group information to enable task interactions in the *lower-level* optimization. We will further discuss the design of the group indicator in Sec. 3.3.

Intuitively, we conduct the group assignment as a clustering process based on these group indicators $\boldsymbol{\gamma}_t(\boldsymbol{\theta}_t)$. In this case, each cluster represents a group, and the cluster center is set to the group weight. Many clustering algorithms are available to achieve this. In this work, we take the K-means clustering algorithm [24–26] as a practical clustering implementation. Thus, we specify the optimization

objective of the dynamical group assignment as:

$$\min_{\boldsymbol{\omega},\boldsymbol{\mathcal{G}}} J(\boldsymbol{\omega},\boldsymbol{\mathcal{G}};\boldsymbol{\theta}_t) \coloneqq \|\boldsymbol{\gamma}_t^\top(\boldsymbol{\theta}_t) - \boldsymbol{\omega}^\top\boldsymbol{\mathcal{G}}\|^2, \tag{3}$$

where $\boldsymbol{\omega}_t^\top = \boldsymbol{\gamma}_t^\top(\boldsymbol{\theta}_t)\boldsymbol{\mathcal{G}}_t^{-1}$ indicates that the cluster center closely relates to the group assignment matrix. It is worth noting that $\boldsymbol{\mathcal{G}}_t^{-1}$ is a generalized inverse, especially one-sided right inverse, $\boldsymbol{\mathcal{G}}^{-1} = \boldsymbol{\mathcal{G}}^\top(\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{G}}^\top)^{-1}$. The designed dynamical group assignment plays an important role in the lower-level optimization of the proposed AGRM, and it tends to cluster similar tasks into the same group while scattering dissimilar ones in clusters.

By integrating the dynamical group assignment in Eq. (3) and the group indicators into Eq. (2), we provide an instantiation for the AGRM's optimization objective:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\omega}_t{}^\top\boldsymbol{\mathcal{G}}_t\hat{\boldsymbol{L}}(\boldsymbol{\theta}) \; s.t. \; \{\boldsymbol{\omega}_t,\boldsymbol{\mathcal{G}}_t\} = \arg\min_{\boldsymbol{\omega},\boldsymbol{\mathcal{G}}} \|\boldsymbol{\gamma}_t^\top(\boldsymbol{\theta}_t) - \boldsymbol{\omega}^\top\boldsymbol{\mathcal{G}}\|^2. \tag{4}$$

Moreover, the dynamic group assignment heuristically clusters tasks from the group indicators, which avoids the exhausted search of appropriate task combinations for performance gains like previous works [21].

### 3.3 Risk-guided Group Indicators

This subsection discusses the appropriate design of the group indicators for dynamic group assignment introduced in Sec. 3.2.

The misalignment of learning across tasks can usually be attributed to various risk scales and asynchronous learning dynamics among tasks over time. To address this, we take two operations with risk information, *scale-balance* and *smooth-alignment*, into consideration and then obtain the risk-guided group indicators by combining them. The role of the group indicators is to use the risk information to explore the relationships among tasks without incurring the expensive computational cost associated with gradients. Compared with other loss-oriented methods, our group indicator can capture the differences in the per-task risk scale and fully utilize the learning dynamics over time, yielding better representations of risk information.

**Scale-balance.** To alleviate the misalignment caused by differences in per-task risk scales, we introduce *scale-balance*, which enlarges the importance of tasks with smaller risks in optimization. Given task-specific risks at iteration $t$, we normalize them to their average risk for efficient scale balancing. In each iteration, the scale vector for all tasks is denoted as $\mathcal{P}_t(\boldsymbol{\theta}_t) = \left[p_t^1(\boldsymbol{\theta}_t), p_t^2(\boldsymbol{\theta}_t), \ldots, p_t^M(\boldsymbol{\theta}_t)\right]^\top \in \mathbb{R}^M$, which can be calculated as:

$$\mathcal{P}_t(\boldsymbol{\theta}_t) = \mathrm{diag}(\hat{\boldsymbol{L}}(\boldsymbol{\theta}_t))^{-1}\left[\bar{\hat{\boldsymbol{L}}}(\boldsymbol{\theta}_t)\right]_M, \tag{5}$$

where $\mathrm{diag}(\cdot)$ constructs a diagonal matrix with the elements of the vector placed on the diagonal. $\bar{\hat{\boldsymbol{L}}}(\boldsymbol{\theta}_t)$ is a scalar to represent the average risk, and $[\cdot]_M$ represents the construction of an $M$-dimentional vector whose elements are all equal to the average risk. To avoid the upper-level optimization degenerating into a fixed scalar, the gradients of empirical risks in the lower-level optimization are not being computed through them. However, in practice, the learning dynamics over time tend to make the scale vector inconsistent over iterations [12, 16], which is not conducive to aligning learning progress. We therefore introduce *smooth-alignment* to update the scale vector with historical information from previous iterations.

**Smooth-alignment**. To avoid sudden fluctuations of scale vectors over iterations, we introduce the smoothness vector $\mathcal{Q}_t(\boldsymbol{\theta}_{1:t}) = \left[q_t^1(\boldsymbol{\theta}_{1:t}), q_t^2(\boldsymbol{\theta}_{1:t}), \ldots, q_t^M(\boldsymbol{\theta}_{1:t})\right]^\top \in \mathbb{R}^M$, which smooths the updating of the scale vector with previous risk information. Thus, the smoothness vector can deal with asynchronous learning dynamics over time, which helps the model reduce the imbalance of training across tasks. To be specific, we compute the smoothness vector by a normalized exponential moving average as follows:

$$\mathcal{Q}_t(\boldsymbol{\theta}_{1:t}) = \sigma\left[\mathcal{Q}_{t-1}(\boldsymbol{\theta}_{1:t-1}) \odot \exp\left(-\beta\hat{\boldsymbol{L}}(\boldsymbol{\theta}_t)\right)\right], \tag{6}$$

where $\odot$ denotes the element-wise multiplication and $\sigma[\cdot]$ normalizes the sum of all smoothness elements to be 1. $\beta$ is a temperature hyperparameter to control the influence of current risk information.


https://doi.org/10.52202/079017-3537

Note that when $\beta$ is close to zero, each element in the smoothness vector will degrade to a fixed value $\frac{1}{M}$, which does not capture any historical information to group indicators.

**Risk-guided Group Indicators**. By element-wise multiplying the scale vector in Eq. (5) and the smoothness vector in Eq. (6), we obtain the group indicators with sufficient risk information as:

$$\gamma_t(\boldsymbol{\theta}_t) = \mathcal{P}_t(\boldsymbol{\theta}_t) \odot \mathcal{Q}_t(\boldsymbol{\theta}_{1:t}). \tag{7}$$

Then, with the group indicators $\gamma_t(\boldsymbol{\theta}_t)$, we optimize the dynamical group assignment in Eq. (3) to assign tasks into groups in the lower-level optimization.

For each group indicator, the role of $\mathcal{P}_t(\boldsymbol{\theta}_t)$ in Eq. (5) and $\mathcal{Q}_t(\boldsymbol{\theta}_{1:t})$ in Eq. (6) differs in optimization: the smoothness vector requires the accumulated loss information from previous iterations, while the scale vector are independent of iterations. Thus, the smoothness vector can iteratively exploit more consistent task correlations to better align learning progress across tasks. The experimental section will show that the risk-guided group indicators empirically boost the proposed adaptive group risk minimization in aligning learning tasks.

## 4 Related Work

**Multi-Task Optimization.** Multi-task optimization addresses the task imbalance issue in multi-task learning, where each task usually influences a shared network differently. Task-imbalance in MTO [1, 27] refers to imbalanced optimization rather than uneven data distributions in the task space. According to different manipulations in optimization, we roughly divide MTO methods into two branches: (i) *gradient-oriented* methods, which solve the task balancing problem by fully utilizing the gradient information of the shared network from different tasks. Some studies report impressive performance based on Pareto optimal solutions [18], gradient normalization [27], gradient conflicting [12], gradient sign Dropout [17], conflict-averse gradient [16], Nash bargaining solution [13]. However, most gradient manipulation methods usually suffer from high computational cost [19]. (ii) *loss-oriented* methods, which reweight task-specific losses with the help of inductive biases from the loss space, e.g., using homoscedastic uncertainty [5], task prioritization [28], self-paced learning [29], similar learning paces [6, 14], random loss weight [20]. Although loss-oriented methods are more computationally efficient, they often underperform gradient-oriented ones in most multi-task benchmarks. This paper tries to trade off the overall performance and computational efficiency.

Recent work [30] weights tasks under the meta-learning setup but has lower training-time efficiency for large-scale systems with high dimensional parameter space, such as deep neural networks, limiting their applications for dense prediction tasks in MTL. The closest method to ours is the recent work FAMO [14], which balances task-specific losses by decreasing task loss approximately at an equal rate. However, ***GO4Align*** proposes a new MTL optimizer that dynamically aligns learning progress across tasks by introducing group-based task interactions.

**Multi-Task Grouping.** Multi-task grouping [11, 21, 31] assigns tasks into different groups and trains intra-group tasks together in a shared multi-task network. Previous work [11] first evaluates the transferring gains for $2^M - 1$ candidate multi-task networks ($M$ is the number of tasks) and then conducts the brute-force search for the best grouping. Some works follow high-order approximation (HOA) [11] to reduce the prohibitive computational cost. However, they also suffer from inaccurate estimations due to non-linear relationships between high-order gains and corresponding pairwise gains [31]. Meanwhile, Yao et al. [32] represents a clustered multi-task learning method, which clusters tasks into several groups by learning the representative tasks. The benefit of multi-task grouping is performance gains by training similar tasks together, and this inspires us to capture helpful group information in multi-task optimization. Also, rather than employing different multi-task networks, ***GO4Align*** introduce group-based task interactions in scaling for multi-task alignment. Moreover, we share a high-level idea of task clustering with [33]. However, task clustering in [33] is limited to pairwise relationships among tasks; meanwhile, our work allows grouping-based task interactions, thus capturing more complex relationships among tasks.

Table 1: **Results on NYUv2 (3 tasks).** The upper and lower tables categorize baseline methods into gradient-oriented and loss-oriented types, respectively. Each experiment is repeated over 3 random seeds, and the mean is reported. The best average result is marked in **bold**. **MR** and $\boldsymbol{\Delta m}\%$ are the main metrics for overall MTL performance. Metrics with ↓ denote that the lower the better.

| Method | Segmentation | | Depth | | Surface Normal | | | | | MR ↓ | $\boldsymbol{\Delta m}\%$ ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Angle Dist ↓ | | Within $t^\circ$ ↑ | | | | |
| | mIoU ↑ | Pix Acc ↑ | Abs Err ↓ | Rel Err ↓ | Mean | Median | 11.25 | 22.5 | 30 | | |
| STL | 38.30 | 63.76 | 0.6754 | 0.2780 | 25.01 | 19.21 | 30.14 | 57.20 | 69.15 | - | - |
| MGDA | 30.47 | 59.90 | 0.6070 | 0.2555 | 24.88 | 19.45 | 29.18 | 56.88 | 69.36 | 7.00 | 1.38 |
| PCGRAD | 38.06 | 64.64 | 0.5550 | 0.2325 | 27.41 | 22.80 | 23.86 | 49.83 | 63.14 | 9.00 | 3.97 |
| GRADDROP | 39.39 | 65.12 | 0.5455 | 0.2279 | 27.48 | 22.96 | 23.38 | 49.44 | 62.87 | 7.89 | 3.58 |
| CAGRAD | 39.79 | 65.49 | 0.5486 | 0.2250 | 26.31 | 21.58 | 25.61 | 52.36 | 65.58 | 5.33 | 0.20 |
| IMTL-G | 39.35 | 65.60 | 0.5426 | 0.2256 | 26.02 | 21.19 | 26.20 | 53.13 | 66.24 | 4.56 | -0.76 |
| NASHMTL | 40.13 | **65.93** | **0.5261** | 0.2171 | 25.26 | 20.08 | 28.40 | 55.47 | 68.15 | 2.89 | -4.04 |
| LS | 39.29 | 65.33 | 0.5493 | 0.2263 | 28.15 | 23.96 | 22.09 | 47.50 | 61.08 | 9.89 | 5.59 |
| SI | 38.45 | 64.27 | 0.5354 | 0.2201 | 27.60 | 23.37 | 22.53 | 48.57 | 62.32 | 8.78 | 4.39 |
| RLW | 37.17 | 63.77 | 0.5759 | 0.2410 | 28.27 | 24.18 | 22.26 | 47.05 | 60.62 | 12.22 | 7.78 |
| DWA | 39.11 | 65.31 | 0.5510 | 0.2285 | 27.61 | 23.18 | 24.17 | 50.18 | 62.39 | 8.67 | 3.57 |
| UW | 36.87 | 63.17 | 0.5446 | 0.2260 | 27.04 | 22.61 | 23.54 | 49.05 | 63.65 | 8.33 | 4.05 |
| FAMO | 38.88 | 64.90 | 0.5474 | 0.2194 | 25.06 | 19.57 | 29.21 | 56.61 | 68.98 | 4.33 | -4.10 |
| ***GO4Align*** | **40.42** | 65.37 | 0.5492 | **0.2167** | **24.76** | **18.94** | **30.54** | **57.87** | **69.84** | **2.11** | **-6.08** |

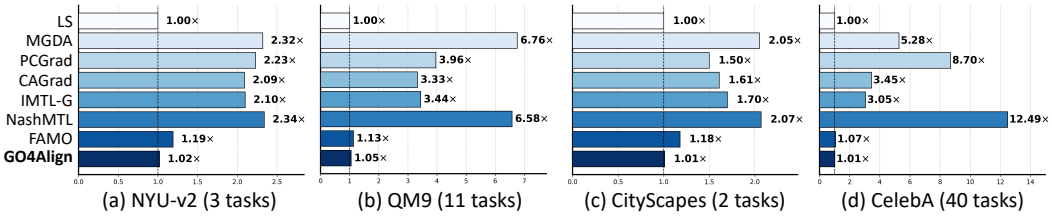


Figure 4: **Efficiency comparisons on training time.** Each method's training time is relative to a simple baseline method with Eq. (1), which minimizes the sum of task-specific empirical risks.

# 5 Experiments

## 5.1 Comparisons on MTL Benchmarks

**Datasets and Settings.** We conduct experiments on four benchmarks commonly used in multi-task optimization literature [6, 13, 14, 16]: NYUv2 [34], CityScapes [35], QM9 [36], and CelebA [37]. For all benchmarks, we follow the training and evaluation protocol in [13, 14].

**Baselines.** We compare ***GO4Align*** with a single-task learning baseline, six gradient-oriented methods, and six loss-oriented methods. Note that single-task learning (STL) trains an independent deep network for each task. The gradient-oriented methods include MGDA [18], PCGRAD [12], CAGRAD [16], IMTL-G [38], GRADDROP [17], and NASHMTL [13]. As for the loss-oriented methods, they are Linear scalarization (LS), Scale-invariant (SI), Dynamic Weight Average (DWA) [6], Uncertainty Weighting (UW) [5], Random Loss Weighting (RLW) [39], and FAMO [14]. Detailed information about datasets and baselines is in Appendix B.

**Evaluations.** Following previous work [13, 16, 40], we report two MTL metrics that demonstrate the overall performance over various task-specific metrics: (1) $\boldsymbol{\Delta m}\%$ is the average per-task performance drop relative to STL. We assume there are $S$ metrics for all tasks. $\mathcal{M}^s$ denote the $s$-th metric value of a multi-task method, while $\mathcal{B}^s$ is the corresponding metric value of the STL baseline. Thus, we formulate the average relative performance drop as: $\boldsymbol{\Delta m}\% = \frac{1}{S}\sum_{s=1}^{S}(-1)^{\delta^s}\frac{(\mathcal{M}^s - \mathcal{B}^s)}{\mathcal{B}^s}$, where $\delta^s = 1$ if higher values for the $s$-th metric are better and 0 otherwise. (2) **MR** $= \frac{1}{S}\sum_{s=1}^{S}\text{rank}(\mathcal{M}^s)$ is the average rank of all task-specific metrics, where $\text{rank}(\mathcal{M}^s)$ denotes the ranking of the $s$-th metric value of the model among all comparison methods. Note that in practice *the lower* $\boldsymbol{\Delta m}\%$ *and* **MR**, *the better overall performance*.


https://doi.org/10.52202/079017-3537

**Effectiveness Comparison.** We provide performance comparisons on NYUv2 in Table 1. In this benchmark, our method achieves the best overall MTL performance among both gradient-oriented and loss-oriented methods. We observe that our work is the only one that improves each task's performance relative to the corresponding STL performance. This suggests that grouping-based task interactions can adequately alleviate the imbalance of learning progress across tasks.

Table 2: **Comparisons on QM9 (11 tasks), CityScapes (2 tasks) and CelebA (40 tasks).** Detailed results are in Appendix C.

| Method | QM9 | | CityScapes | | CelebA | |
|---|---|---|---|---|---|---|
| | MR ↓ | $\Delta m\%\downarrow$ | MR ↓ | $\Delta m\%\downarrow$ | MR ↓ | $\Delta m\%\downarrow$ |
| MGDA | 7.73 | 120.5 | 10.00 | 44.14 | 10.05 | 14.85 |
| PCGRAD | 6.09 | 125.7 | 6.25 | 18.29 | 6.05 | 3.17 |
| CAGRAD | 7.09 | 112.8 | 5.00 | 11.64 | 5.65 | 2.48 |
| IMTL-G | 5.91 | 77.2 | 4.00 | 11.10 | 4.08 | **0.84** |
| NASHMTL | **3.64** | 62.0 | **2.50** | 6.82 | 4.53 | 2.84 |
| LS | 8.00 | 177.6 | 8.50 | 14.11 | 5.55 | 4.15 |
| SI | 5.09 | 77.8 | 8.50 | 14.11 | 7.10 | 7.20 |
| RLW | 9.36 | 203.8 | 7.75 | 24.38 | 4.60 | 1.46 |
| DWA | 7.64 | 175.3 | 6.00 | 21.45 | 6.25 | 3.20 |
| UW | 6.64 | 108.0 | 5.75 | **5.89** | 5.18 | 3.23 |
| FAMO | 4.73 | 58.5 | 5.50 | 8.13 | 4.10 | 1.21 |
| ***GO4Align*** | 4.55 | **52.7** | 7.00 | 8.11 | **3.10** | 0.88 |

The experimental results on QM9, CityScapes and CelebA are reported in Table 2. ***GO4Align*** obtains the lowest $\boldsymbol{\Delta m}\%$ on QM9. It also shows comparable performance with FAMO on CityScapes, one possible reason could be that this dataset only contains 2 tasks, which limits the potential of the grouping mechanism in our method. In CelebA, even though our work does not achieve the lowest average performance drop, it outperforms all loss-oriented methods, which further verifies the effectiveness of the proposed method.

**Efficiency Comparison.** To show the computational efficiency, in Fig. 4, we report the average training time per epoch over 5 epochs for each method. We choose LS as the relative baseline for training time (cf. RLW [39] and FAMO [14]) as it is a commonly used MTL baseline with equal weights for each task, and it does not require additional loss-oriented or gradient-oriented techniques. We note that we run all experiments on an NVIDIA A100 and the code of baseline methods comes from Liu et al. [14] and Navon et al. [13].

As shown in this figure, the proposed method ***GO4Align***, as a new member of the loss-oriented branch, can perform more efficiently than most gradient-oriented methods. Moreover, when the number of tasks scales up from 2 to 40, the reduction in computational cost between our method and other gradient-oriented methods becomes increasingly significant, e.g., NashMTL (2.07) versus Ours (1.01) with 2 tasks, NashMTL (12.49) versus Ours (1.01) with 40 tasks. The main reason is that the training time of gradient-oriented methods is proportional to the number of tasks, but our work can avoid this. More experimental results are provided in Appendix C.

### 5.2 Ablation Study

The effectiveness and efficiency of our proposed method are shown in Sec. 5.1. Next, we answer the following questions with our ablation study: (1) Can we quantify the contributions of each phrase? (2) Can we disentangle the roles of the group assignment and group weights? (3) Can the proposed AGRM principle seamlessly integrate with existing MTO methods? (4) Are there practical ways to appropriately configure hyperparameters, *e.g.*, group number $K$? (5) Why do we choose the K-means clustering in the proposed method?

**Contributions of Each Phase.** To quantify the contributions of each phase in achieving the proposed AGRM principle on NYUv2, we report the detailed performance of our method in each phase. As the grouping is performed by Eq.(4), the first two rows in Table 3 are the variants of our method without grouping, and the last row is our method with grouping. Table 3 empirically examines the performance gains of the variant with task grouping over without grouping.

Table 3: **Effectiveness of each phase in *GO4Align* on NYUv2.** ✓ denote whether the component joins the pipeline.

| Eq.(5) | Eq.(6) | Eq.(4) | $\Delta_{seg.}\%\downarrow$ | $\Delta_{depth}\%\downarrow$ | $\Delta_{normal}\%\downarrow$ | $\Delta m\%\downarrow$ |
|---|---|---|---|---|---|---|
| ✓ | | | -0.02 | **-21.76** | 13.14 | 2.46 |
| ✓ | ✓ | | 14.22 | -15.27 | 2.52 | 1.16 |
| ✓ | ✓ | ✓ | **-4.03** | -20.37 | **-1.18** | **-6.08** |

In detail, compared with the scale vector in Eq. (5), the smoothness vector in Eq. (6) can compromise the performance of the "Normal" and "Depth" tasks, however, scarifying that of "Seg.". Based on the scale and smoothness vectors, the proposed method employs dynamical group assignment in Eq. (4) to exploit the grouping-based task interactions, thus well aligning the learning progress of similar tasks "Depth" and "Seg.". We also observe that our method with both phases can improve the task-specific performance relative to STL. This demonstrates that each phase in the method complements each other, resulting in more balanced performance across tasks.

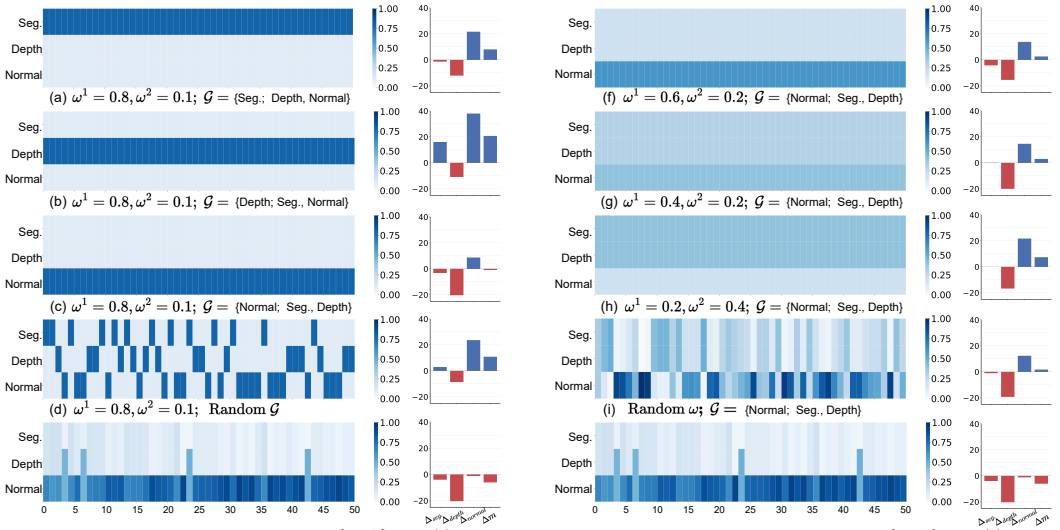


Figure 5: **Comparative analysis of the influence of the group assignment matrix and group weights on** NYUv2. The x-axis in the subplots denotes the epoch, and the intensity of the color indicates the weight value. **(a-d)** have fixed group weights $\boldsymbol{\omega} = [\omega^1, \omega^2]$ but various group assignment matrices $\mathcal{G}$. **(f-i)** have various group weights $\boldsymbol{\omega}$ but a fixed group assignment matrix $\mathcal{G}$. **(e)** is our method that dynamically exploits a group assignment matrix and group weights for each iteration. The right side of each method shows relative performance drops on each task and their average one.

**Influence of Group Assignment Matrix.** To explore the influence of the group assignment matrix $\boldsymbol{\mathcal{G}}$, we assume the group number is 2 on NYUv2 and make comparisons with several variants, which have various group assignments with fixed group weights $\boldsymbol{\omega} = [0.8, 0.1]$. As shown in Fig. 5 (a-c), grouping "Seg." and "Depth" outperforms other options. The main reason could be that these two tasks are very similar and far away from the "Normal" task [21]. We observe that variant (d) with a random grouping strategy shows lower performance than the fixed grouping options (a) and (c), which further implies the importance of appropriate group assignment. It is worth mentioning that the proposed method in (e) without the prior information of the appropriate group assignment also captures such task correlations and each task can get performance gains compared with STL. This demonstrates that group assignment plays an important role in exploring task correlations over time in the proposed method.

**Influence of Group Weights.** To study the influence of group weights, we conduct another visualization in Fig. 5 (f-i), where we focus on various group weights $\boldsymbol{\omega}$ with the "optimal" group assignment matrix $\boldsymbol{\mathcal{G}} = \{\text{Normal}; \text{Seg}, \text{Depth}\}$. We observe that with the fixed group assignment matrix, group weights have effects on the extent of compromising among different groups. On the NYUv2 dataset, lower weights for the first groups obtain better overall performance. The variant method (i) with random group weights achieves surprising performance, $1.75\%$, in terms of the average relative performance drop. (e) shows that our method also tends to dynamically weight the first group with a high value. This demonstrates that group weights are necessary to align the learning progress of different groups over time.

**Effect of Adaptive Group Risk Minimization Principle.** Empirically, the proposed adaptive group risk minimization principle (AGRM) in Sec. 3.1 can be seamlessly integrated with existing MTO methods, taking their updated task weights as group indicators. As detailed in Table 4, MTO methods combined with AGRM consistently show improved performance. MGDA with adaptive group risk minimization achieves the biggest improvement gap. Moreover, our method still outperforms others. The reason could be that the designed risk-guided group indicators are more suitable for AGRM by balancing risk scales and exploiting historical information from previous iterations.

Table 4: **Comparisons of existing MTO methods with the proposed AGRM on** NYUv2.

| **Methods** | $\Delta_{seg.}\% \downarrow$ | $\Delta_{depth}\% \downarrow$ | $\Delta_{normal}\% \downarrow$ | $\Delta_m\% \downarrow$ |
|---|---|---|---|---|
| MGDA | 13.25 | -9.11 | 0.83 | 1.38 |
| MGDA + AGRM | 6.06 | -11.69 | -1.14 | -1.89 |
| NashMTL | -4.09 | **-22.01** | 3.15 | -4.04 |
| NashMTL + AGRM | **-7.75** | -20.14 | 3.60 | -4.20 |
| FAMO | -1.65 | -20.02 | 1.29 | -4.10 |
| FAMO + AGRM | 1.76 | -21.17 | -0.03 | -4.32 |
| ***GO4Align*** | -4.03 | -20.37 | **-1.18** | **-6.08** |


https://doi.org/10.52202/079017-3537

Table 5: **Comparisons between different clustering methods in the proposed framework.**

| Methods | $\Delta_{seg.}\% \downarrow$ | $\Delta_{depth}\% \downarrow$ | $\Delta_{normal}\% \downarrow$ | $\Delta_m\% \downarrow$ | Relative runtime ↓ |
|---|---|---|---|---|---|
| SDP-based clustering [41] | -2.97 | -18.76 | -1.09 | -5.44 | 1.20× |
| Spectral clustering [42] | -1.78 | -18.58 | -0.06 | -4.56 | 1.17× |
| Ours | **-4.03** | **-20.37** | **-1.18** | **-6.08** | **1.02**× |

**Influence of Group Number.** In the proposed method, group number $K$ is an important hyper-parameter, especially when we instantiate the clustering process in dynamical group assignment with $K$-means. In this case, there are many different techniques for choosing the right $K$. To be visualizable, here we apply the conventional elbow method. As shown in Fig. 6, the overall performance (lower is better) of our method in (a) and (b) drops at 2 and 5, respectively, after that both reach a plateau when the group numbers increase. Thus, in this paper we set $K = 2$ and $K = 5$ for `NYUv2` and `QM9`.

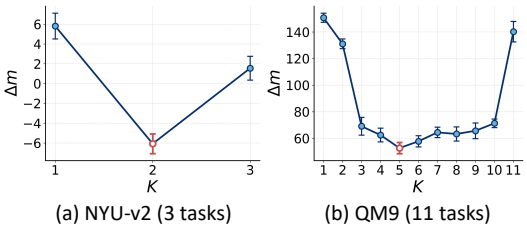


Figure 6: **Identification of "elbow" points on `NYUv2` and `QM9`.** According to the conventional elbow method, we set the group number of the two datasets as 2 and 5, respectively.

**Effect of different clustering methods.** In our main experiments, we employed standard K-means for instantiation. K-means is a widely used clustering approach. To investigate the effect of different clustering methods, we evaluate the impact of using alternative clustering algorithms.

Specifically, we tested our proposed method on `NYUv2` by substituting K-means with SDP-based clustering [41] and spectral clustering [42]. As demonstrated in Table 5, these alternative clustering methods also outperform state-of-the-art approaches (FAMO, $-4.10\%$), particularly by enhancing the performance of each task over STL. Interestingly, our experiments show that the K-means clustering algorithm outperforms spectral and SDP-based clustering methods.

## 6 Conclusion

**Technical Discussion.** This paper focuses on the task imbalance issue in MTO. Previous MTO methods suffer from either intensive computations or non-competitive performance. Our proposed ***GO4Align*** addresses the issue by aligning learning progress across tasks with the help of the AGRM principle. In problem-solving, we present a tractable optimization pipeline, which incorporates grouping-based task interactions into the loss scaling of MTO.

**Limitation.** The main limitation of this work is the heuristic configuration of the group numbers. Although the search space is significantly smaller than some grouping multi-task methods [31], it still needs maximum $M$ runs to find the best number. Some related techniques [43] automatically set the group number can be added to avoid this limitation in future work.

**Broader Impact.** This paper is the first to consider task grouping in multi-task optimization with deep multi-task models. We propose a simple and principled way to fasten multi-task optimization with better training-time efficiency, which has many potential societal impacts, especially in dense prediction tasks. We provide the code for our method to encourage follow-up work.[3]

## Acknowledgment

This work is financially supported by the Inception Institute of Artificial Intelligence, the University of Amsterdam and the allowance Top consortia for Knowledge and Innovation (TKIs) from the Netherlands Ministry of Economic Affairs and Climate Policy.

[3] https://github.com/autumn9999/GO4Align

## References


[1] 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*, 2021.

[2] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *arXiv preprint arXiv:2109.09138*, 2021.

[3] Jonathan Pilault, Christopher Pal, et al. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. In *International Conference on Learning Representations*, 2020.

[4] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017.

[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*, pages 7482–7491, 2018.

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

[7] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[8] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. *arXiv preprint arXiv:2102.06177*, 2021.

[9] Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? *Advances in Neural Information Processing Systems*, 35:13597–13609, 2022.

[10] Shikun Liu, Stephen James, Andrew J Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *arXiv preprint arXiv:2202.03091*, 2022.

[11] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[12] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[13] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

[14] Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 33, 2023.

[15] Dmitry Senushkin, Nikolay Patakin, Arseny Kuznetsov, and Anton Konushin. Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20083–20093, 2023.

[16] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

[17] 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. *arXiv preprint arXiv:2010.06808*, 2020.

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

[19] Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183, 2022.


 https://doi.org/10.52202/079017-3537

[20] 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.

[21] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

[22] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528, 2011.

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

[24] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

[25] Trupti M Kodinariya, Prashant R Makwana, et al. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[26] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020.

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

[28] 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.

[29] Keerthiram Murugesan and Jaime Carbonell. Self-paced multitask learning with shared knowledge. *arXiv preprint arXiv:1703.00977*, 2017.

[30] Cuong C. Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Task weighting in meta-learning with trajectory optimisation. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[31] Xiaozhuang Song, Shun Zheng, Wei Cao, James Yu, and Jiang Bian. Efficient and effective multi-task grouping via meta learning on task combinations. *Advances in Neural Information Processing Systems*, 35: 37647–37659, 2022.

[32] Yaqiang Yao, Jie Cao, and Huanhuan Chen. Robust task grouping with representative tasks for clustered multi-task learning. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 1408–1417, 2019.

[33] Sebastian Thrun and Joseph O'Sullivan. Clustering learning tasks and the selective cross-task transfer of knowledge. In *Learning to learn*, pages 235–257. Springer, 1998.

[34] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[35] 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*, pages 3213–3223, 2016.

[36] L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

[37] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[38] 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*, 2020.

[39] Baijiong Lin, Feiyang Ye, and Yu Zhang. A closer look at loss weighting in multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

[40] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1860, 2019.

[41] Mariano Tepper, Anirvan M Sengupta, and Dmitri Chklovskii. The surprising secret identity of the semidefinite relaxation of k-means: manifold learning. *arXiv preprint arXiv:1706.06028*, 2017.

[42] Anil Damle, Victor Minden, and Lexing Ying. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA*, 8(1):181–203, 2019.

[43] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8: 80716–80727, 2020.

[44] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[45] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39 (12):2481–2495, 2017.

[46] 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.

[47] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343, 2017.

[48] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[49] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214, 2019.

[50] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

[51] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International Conference on Machine Learning*, pages 3854–3863. PMLR, 2020.

[52] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.

[53] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S Yu. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems*, 30, 2017.

[54] Jiayi Shen, Xiantong Zhen, Marcel Worring, and Ling Shao. Variational multi-task learning with gumbel-softmax priors. *Advances in Neural Information Processing Systems*, 34:21031–21042, 2021.

[55] Jiayi Shen, Xiantong Zhen, Qi Wang, and Marcel Worring. Episodic multi-task learning with heterogeneous neural processes. *Advances in Neural Information Processing Systems*, 36:75214–75228, 2023.

 https://doi.org/10.52202/079017-3537

# Appendix

## A Algorithm of *GO4Align*

The pseudo-code of ***GO4Align*** is provided in Algorithm 1. For clarity, we also illustrate the optimization process in Fig. 7.

**Algorithm 1** Group Optimization for Multi-Task Alignment (***GO4Align***)

**Input**: Maximum iteration number $T$; Batch size $N_{\text{bz}}$; Learning rate $\alpha$; Temperature hyperparameter $\beta$; Task number $M$; Group number $(1 < K \leq M)$;
Initialize model parameters $\boldsymbol{\theta}_0 = \{\boldsymbol{\theta}_0^s, \boldsymbol{\theta}_0^1, \boldsymbol{\theta}_0^2, \cdots, \boldsymbol{\theta}_0^M\}$;
**for** $t = 0 : T$ **do**
  Randomly sample a batch of training samples:
  $\{(\boldsymbol{x}_n, y_n^1, ..., y_n^M)\}_{n=1}^{N_{\text{bz}}}$, where $\boldsymbol{x}_n \in \mathcal{X}, y_n^m \in \mathcal{Y}^m$;
  Compute the empirical risk for each task at the $t$-th iteration:
  $\hat{\mathcal{L}}^m(\boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^m) = \frac{1}{N_{\text{bz}}} \sum_{n=1}^{N_{\text{bz}}} \ell^m(f(\boldsymbol{x}_n; \boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^m), y_n^m)$;
  `// Lower-level optimization for grouping-based task interactions.`
  Compute the scale vector $\mathcal{P}_t(\boldsymbol{\theta}_t)$ in Eq. (5);
  Compute the smoothness vector $\mathcal{Q}_t(\boldsymbol{\theta}_{1:t})$ in Eq. (6);
  Obtain the group indicators $\boldsymbol{\gamma}_t(\boldsymbol{\theta}_{1:t})$ with Eq.(7);
  Update the group information $\boldsymbol{\omega}_t$ and $\boldsymbol{\mathcal{G}}_t$ with Eq. (3) and the obtained $\boldsymbol{\gamma}_t(\boldsymbol{\theta}_{1:t})$.
  `// Upper-level optimization for model parameters.`
  Update the model parameters $\boldsymbol{\theta}$ with Eq. (4): $\boldsymbol{\theta}_{t+1} \leftarrow \text{argmin}_{\boldsymbol{\theta}} \boldsymbol{\omega}_t{}^{\top} \boldsymbol{\mathcal{G}}_t \hat{\boldsymbol{L}}(\boldsymbol{\theta})$.
**end for**

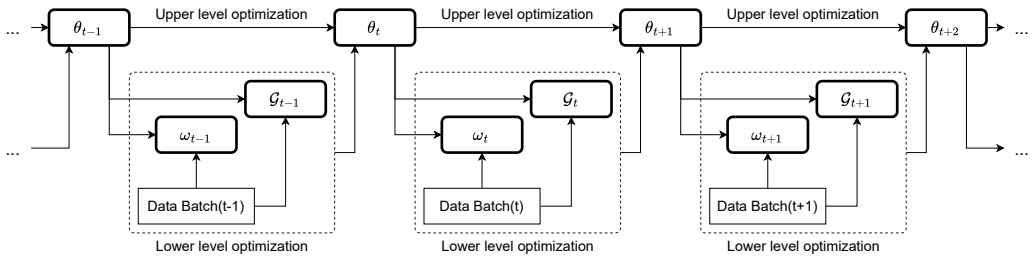


Figure 7: **Optimization process of the proposed adaptive group risk minimization principle.** At each iteration, given the randomly sampled mini-batch data, we first compute the group information $\boldsymbol{\omega}$ and $\boldsymbol{\mathcal{G}}$ in the lower-level optimization and then update the model's parameter $\boldsymbol{\theta}$ in the upper-level optimization.

## B Experimental Set-Up & Implementation Details

### B.1 Benchmark Descriptions

`NYUv2` [34] is an indoor scene dataset consisting of 1449 RGBD images and dense per-pixel labeling with 13 classes. The learning objectives include 3 different dense prediction tasks: image segmentation, depth prediction, and surface normal prediction based on any scene image.

`CityScapes` [35] contains 5000 street-view RGBD images with per-pixel annotations. It needs to predict 2 dense prediction tasks: image segmentation and depth prediction.

`QM9` [36] is a benchmark for group neural networks to predict 11 properties of molecules. It consists of >130K molecules represented as graphs annotated with node and edge features. We use $110K$ molecules from the QM9 example in PyTorch Geometric [44], $10K$ molecules for validation, and the rest of $10K$ molecules as a test set.

CelebA [37] contains 200K face images of 10K different celebrities, and each face image is provided with 40 facial binary attributes. As the protocol provided in previous work [14], each attribute corresponds to one task. Thus, we consider CelebA as a 40-task MTL problem.

This work follows the same experimental setting used in NashMTL [13] and FAMO [14], including the dataset partition for training, validation, and testing. The benchmark partition is shown in Table 6. We also note that NYUv2 and Cityscapes do not have validation sets. Following the protocol in [13, 14], we report the test performance averaged over the last ten epochs.

Table 6: **Benchmark partition for training, validation, and testing.**

| Datasets | Total | Training | Validation | Test |
|---|---|---|---|---|
| NYUv2 | 1449 | 795 | N/A | 654 |
| CityScapes | 3475 | 2975 | N/A | 500 |
| QM9 | ~130k | ~110k | 10k | 10k |
| CelebA | 202,599 | 162,770 | 19,867 | 19,962 |

### B.2 Compared Multi-task Learning Baselines

From the gradient manipulation branch, **(1)** MGDA [18] that finds the equal descent direction for each task; **(2)** PCGRAD [12] proposes to project each task gradient to the normal plan of that of other tasks and combining them together in the end; **(3)** CAGRAD [16] optimizes the average loss while explicitly controls the minimum decrease across tasks; **(4)** IMTL-G [38] finds the update direction with equal projections on task gradients; **(5)** GRADDROP [17] that randomly dropout certain dimensions of the task gradients based on how much they conflict; **(6)** NASHMTL [13] formulates MTL as a bargaining game and finds the solution to the game that benefits all tasks.

From the loss scaling branch, **(1)** Linear scalarization (LS) is the sum of empirical risk minimization; **(2)** Scale-invariant (SI) is invariant to any scalar multiplication of task losses; **(3)** Dynamic Weight Average (DWA) [6], a heuristic for adjusting task weights based on rates of loss changes; **(4)** Uncertainty Weighting (UW) [5] uses task uncertainty as a proxy to adjust task weights; **(5)** Random Loss Weighting (RLW) [39] that samples task weighting whose log-probabilities follow the normal distribution; **(6)** FAMO [14] decreases task losses approximately at equal rates.

### B.3 Neural Architectures & Training Details

For NYUv2 and CityScapes, we follow the training and evaluation protocol in [13], which adds data augmentations during training for all compared methods. We train each method for 200 epochs with an initial learning rate of $1e-4$ and reduce the learning rate to $5e-5$ after 100 epochs. The architecture is Multi-Task Attention Network (MTAN) [6] built upon SegNet [45]. Batch sizes for NYUv2 and CityScapes are set as 2 and 8 respectively. To make a fair comparison with previous works [6, 12, 14, 16], we report the test performance averaged over the last 10 epochs.

We follow the protocol in Navon et al. [13] to normalize each task target at the same scale for fairness. We train each method for 300 epochs with a batch size of 120 and search for the best learning rate in $\{1e-3, 5e-4, 1e-4\}$. We take ReduceOnPlateau [13] as the learning-rate scheduler to decrease the lr once the validation overall performance stops improving. The validation set is also used for early stopping.

Following [14], we use a neural network with five convolutional and two fully connected layers as the shared encoder. The decoder of each task is implemented by another fully connected layer. We train the model for 15 epochs with a batch size of 256. We adopt Adam as the optimizer with a fixed learning rate of $1e-3$. Similar to QM9, we use the validation set for early stopping and hyperparameter selection, such as the number of groups $K$ and the step size of smoothness value $\beta$. We conduct all experiments on a single NVIDIA A100 GPU.


https://doi.org/10.52202/079017-3537

Table 7: **Detailed results on** `QM9`. Each experiment is repeated over 3 random seeds and the mean is reported. The best average result is marked in bold. **MR** and $\boldsymbol{\Delta m}\%$ are the main metrics for MTL performance.

| **Method** | $\mu$ | $\alpha$ | $\epsilon_{\text{HOMO}}$ | $\epsilon_{\text{LUMO}}$ | $\langle R^2 \rangle$ | ZPVE | $U_0$ | $U$ | $H$ | $G$ | $c_v$ | **MR** ↓ | $\boldsymbol{\Delta m}\%$ ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE ↓ | | | | | | | | | | | | |
| STL | 0.07 | 0.18 | 60.6 | 53.9 | 0.50 | 4.53 | 58.8 | 64.2 | 63.8 | 66.2 | 0.07 | - | - |
| MGDA | 0.22 | 0.37 | 126.8 | 104.6 | 3.23 | 5.69 | 88.4 | 89.4 | 89.3 | 88.0 | 0.12 | 7.73 | 120.5 |
| PCGRAD | 0.11 | 0.29 | 75.9 | 88.3 | 3.94 | 9.15 | 116.4 | 116.8 | 117.2 | 114.5 | 0.11 | 6.09 | 125.7 |
| CAGRAD | 0.12 | 0.32 | 83.5 | 94.8 | 3.22 | 6.93 | 114.0 | 114.3 | 114.5 | 112.3 | 0.12 | 7.09 | 112.8 |
| IMTL-G | 0.14 | 0.29 | 98.3 | 93.9 | 1.75 | 5.70 | 101.4 | 102.4 | 102.0 | 100.1 | 0.10 | 5.91 | 77.2 |
| NASHMTL | **0.10** | **0.25** | 82.9 | **81.9** | 2.43 | 5.38 | 74.5 | 75.0 | 75.1 | 74.2 | **0.09** | **3.64** | 62.0 |
| LS | 0.11 | 0.33 | **73.6** | 89.7 | 5.20 | 14.06 | 143.4 | 144.2 | 144.6 | 140.3 | 0.13 | 8.00 | 177.6 |
| SI | 0.31 | 0.35 | 149.8 | 135.7 | **1.00** | **4.51** | 55.3 | 55.8 | 55.8 | 55.3 | 0.11 | 5.09 | 77.8 |
| RLW | 0.11 | 0.34 | 76.9 | 92.8 | 5.87 | 15.47 | 156.3 | 157.1 | 157.6 | 153.0 | 0.14 | 9.36 | 203.8 |
| DWA | 0.11 | 0.33 | 74.1 | 90.6 | 5.09 | 13.99 | 142.3 | 143.0 | 143.4 | 139.3 | 0.13 | 7.64 | 175.3 |
| UW | 0.39 | 0.43 | 166.2 | 155.8 | 1.07 | 4.99 | 66.4 | 66.8 | 66.8 | 66.2 | 0.12 | 6.64 | 108.0 |
| FAMO | 0.15 | 0.30 | 94.0 | 95.2 | 1.63 | 4.95 | 70.82 | 71.2 | 71.2 | 70.3 | 0.10 | 4.73 | 58.5 |
| ***GO4Align*** | 0.17 | 0.35 | 102.4 | 119.0 | 1.22 | 4.94 | **53.9** | **54.3** | **54.3** | **53.9** | 0.11 | 4.55 | **52.7** |

Table 8: **Detailed results on** `CityScapes`. Each experiment is repeated over 3 random seeds and the mean is reported. The best average result is marked in bold. **MR** and $\boldsymbol{\Delta m}\%$ are the main metrics for MTL performance.

| **Method** | Segmentation | | Depth | | **MR** ↓ | $\boldsymbol{\Delta m}\%$ ↓ |
|---|---|---|---|---|---|---|
| | mIoU ↑ | Pix Acc ↑ | Abs Err ↓ | Rel Err ↓ | | |
| STL | 74.01 | 93.16 | 0.0125 | 27.77 | | |
| MGDA | 68.84 | 91.54 | 0.0309 | 33.50 | 10.00 | 44.14 |
| PCGRAD | 75.13 | 93.48 | 0.0154 | 42.07 | 6.25 | 18.29 |
| CAGRAD | 75.16 | 93.48 | 0.0141 | 37.60 | 5.00 | 11.64 |
| IMTL-G | 75.33 | 93.49 | 0.0135 | 38.41 | 4.00 | 11.10 |
| NASHMTL | **75.41** | **93.66** | **0.0129** | 35.02 | **2.50** | 6.82 |
| LS | 70.95 | 91.73 | 0.0161 | 33.83 | 8.50 | 14.11 |
| SI | 70.95 | 91.73 | 0.0161 | 33.83 | 8.50 | 14.11 |
| RLW | 74.57 | 93.41 | 0.0158 | 47.79 | 7.75 | 24.38 |
| DWA | 75.24 | 93.52 | 0.0160 | 44.37 | 6.00 | 21.45 |
| UW | 72.02 | 92.85 | 0.0140 | 30.13 | 5.75 | **5.89** |
| FAMO | 74.54 | 93.29 | 0.0145 | 32.59 | 5.50 | 8.13 |
| ***GO4Align*** | 72.63 | 93.03 | 0.0164 | **27.58** | 7.00 | 8.11 |

# C Additional Experimental Results

## C.1 Detailed Results on `QM9` and `CityScapes`

We provide task-specific performance on `QM9` in Table 7. The proposed ***GO4Align*** obtains the best performance in terms of the average performance drop $\boldsymbol{\Delta m}\%$. And its average rank **MR** is lower than all loss-oriented methods, which demonstrates the proposed method can get a more balanced performance for each task.

As shown in Table 8, our method achieves competitive performance on `CityScapes` with other alternatives except for NASHMTL and UW. The main reason can be that there are only two tasks in the datasets, which constrains the effectiveness of the grouping mechanism in our method.

## C.2 Training Time Comparisons

In Fig. 4, we show the training time of MTO methods relative to a baseline method (LS). Here we provide real training time (seconds) in Table 9, where we compute the average training time of 5 epochs. From this table, we observe that loss-oriented methods in general use less time for one epoch than gradient-oriented methods. ***GO4Align*** requires the second lowest time cost during training on all four datasets, demonstrating its good computational efficiency.

Table 9: **Detailed training time (seconds) for one epoch of different methods.** In addition to LS, our method requires the lowest time cost during training on all 4 datasets.

| Method | NYU-v2 | QM9 | CityScapes | CelebA |
|---|---|---|---|---|
| MGDA | 199.73 | 601.52 | 136.69 | 902.53 |
| PCGRAD | 192.41 | 352.05 | 100.21 | 1486.71 |
| CAGRAD | 180.37 | 296.36 | 107.84 | 590.05 |
| IMTL-G | 180.67 | 306.11 | 113.51 | 522.22 |
| NASHMTL | 201.30 | 585.37 | 138.26 | 2134.75 |
| LS | 86.21 | 88.95 | 66.63 | 170.81 |
| FAMO | 102.87 | 100.40 | 78.70 | 183.46 |
| ***GO4Align*** | 87.78 | 93.50 | 67.30 | 171.69 |

### C.3 Visualizations on Risk Ratios

To investigate the influence of different scaling methods on the training of tasks, we illustrate the ratios between task-specific empirical risk and the sum of all empirical risks before and after scaling on `NYUv2`. In each small figure, the three shadows from top to down represent the ratios of "Normal", "Seg." and "Depth", respectively. The x-axis represents epochs ranging from 1 to 200. Note that MGDA and NASHMTL scale task-specific gradients. For direct comparisons, here we provide the scaled loss ratios, which are equivalent to gradient scaling for the shared network.

As shown in Fig. 8, all methods have similar ratios over epochs on the unscaled risks but perform differently on the scaled risks. We observe that MGDA have a significantly larger ratio on the "Normal" task, which means MGDA prefers to optimize the "Normal" task. Compared with related works, our method has more stable ratios of different tasks. The possible reason could be that our method benefits from historical information, which avoids training instability among tasks.

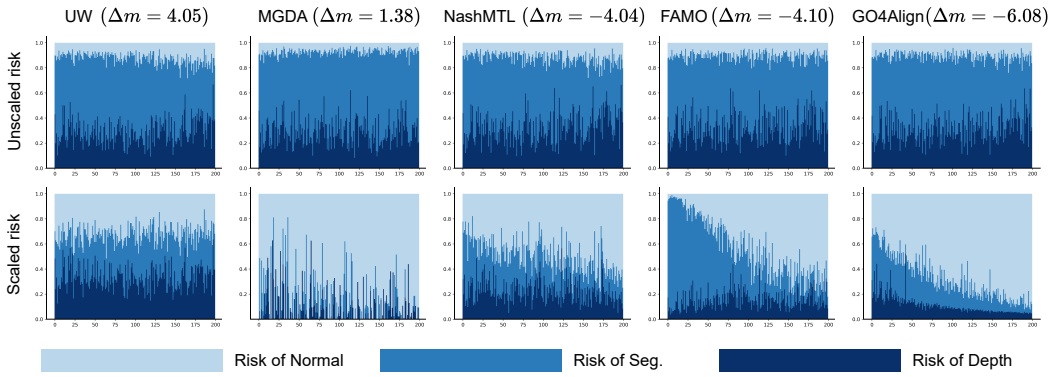


Figure 8: **Analysis on risk ratios.** Compared with other works, the proposed ***GO4Align*** shows more stable ratios among tasks over epochs, indicating that ***GO4Align*** can maintain better alignment throughout the training process.

## D Additional Related Work

In MTL, we group multi-task learning methods into three categories: multi-task optimization, multi-task grouping, and deep multi-task architecture. The first two categories are most related to this paper and are mentioned in the main paper. To be self-contained, here we provide detailed discussions about deep multi-task architecture.

Our method ***GO4Align*** builds on multi-task grouping and multi-task optimization, inheriting advantages from both to balance different tasks in joint learning. In the following, we discuss each category of multi-task learning methods and explain how they relate to our proposed method.


https://doi.org/10.52202/079017-3537

**Deep Multi-Task Architecture.** Multi-task architecture design can be roughly categorized into either a hard-parameter sharing design [46, 47] or a soft-parameter sharing design [6, 48, 49]. The hard-parameter sharing design generally contains a shared encoder and several task-specific decoders. Branching points between the shared encoder and decoders are determined in an ad-hoc way [1, 50], resulting in a suboptimal solution. Some work [47, 51, 52] automatically learns where to share or branch with a network. The soft-parameter sharing design considers all parameters task-specific and instead learns feature-sharing mechanisms to handle the cross-task interactions [48, 53]. Some work [54, 55] are probabilistic multi-task learning methods, which introduce latent variables to encode task-specific information and enable knowledge sharing in the latent spaces. Soft-parameter sharing methods usually struggle with model scalability as the model size grows linearly with the number of tasks. ***GO4Align*** focuses on the optimization of multi-task learning, which adopts a simple hard-parameter sharing architecture as the backbone and subsequently reduces task interference in the gradient space of the shared encoder or the loss space.

https://doi.org/10.52202/079017-3537

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope of this paper are claimed in the abstract. Detailed information can be found in the introduction section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a "limitation" subsection in the conclusion section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]


https://doi.org/10.52202/079017-3537

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the details of the experiment in the experiment section. The algorithm and Python codes of our method can be found in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide all experimental details in the experiment section. The algorithm and Python codes of our method can be found in Appendix A. Details about MTL benchmarks in this paper are provided in Appendix B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We follow the standard experimental setup in MTO, where the data splits, hyperparameters, and optimizer are set as the same as previous works (nashMTL, CAGrad, and FAMO). Detailed information can be found in Appendix B.3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Following the standard experimental setup, we repeat each experiment over 3 random seeds and report the mean of the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).


https://doi.org/10.52202/079017-3537

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computing resources in Appendix B.3. The training-time efficiency is provided in Figure 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We reviewed and followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide the potential broader impacts in the conclusion section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The data and models pose no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers that produced the code package and datasets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

 https://doi.org/10.52202/079017-3537

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The provided Python code cannot be used without the authors' permission.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.