Towards Consistent Multi-Task Learning: Unlocking the Potential of Task-Specific Parameters

Xiaohan Qin **Xiaoxing Wang**‡ **Junchi Yan**‡
School of Computer Science & School of Artificial Intelligience, Shanghai Jiao Tong University
{galaxy-1,figure1_wxx,yanjunchi}@sjtu.edu.cn
https://github.com/jianke0604/MTLlib

Abstract

Multi-task learning (MTL) has gained widespread application for its ability to transfer knowledge across tasks, improving resource efficiency and generalization. However, gradient conflicts from different tasks remain a major challenge in MTL. Previous gradient-based and loss-based methods primarily focus on gradient optimization in shared parameters, often overlooking the potential of task-specific parameters. This work points out that task-specific parameters not only capture task-specific information but also influence the gradients propagated to shared parameters, which in turn affects gradient conflicts. Motivated by this insight, we propose ConsMTL, which models MTL as a bi-level optimization problem: in the upper-level optimization, we perform gradient aggregation on shared parameters to find a joint update vector that minimizes gradient conflicts; in the lower-level optimization, we introduce an additional loss for task-specific parameters guiding the k gradients of shared parameters to gradually converge towards the joint update vector. Our design enables the optimization of both shared and task-specific parameters to consistently mitigate gradient conflicts. Extensive experiments show that ConsMTL achieves state-of-the-art performance across various benchmarks with task numbers ranging from 2 to 40, demonstrating its superior performance.

1. Introduction

Multi-task learning (MTL) [6, 31, 40] is a framework in which a unified model is designed to address multiple tasks concurrently. By sharing representations and leveraging cross-task information, MTL can boost the generalization capabilities of the model and enhance overall task performance [4, 28, 35]. This approach is particularly advantageous in contexts with limited computational resources, as it mitigates the need for separate models per task. Its applications span a variety of domains, including computer vision [1, 25, 41], natural language processing [7, 24, 30], and robotics [12, 37]. Despite its advantages, MTL faces a significant challenge: gradient conflicts in shared parameters [21, 32, 36, 38]. Gradients from different tasks exhibit varying directions and magnitudes, making it essential to determine a final update that mitigates these conflicts effectively while ensuring fair optimization across tasks.

To address this challenge, previous works have concentrated on two main approaches: gradient-based [3, 21, 29, 38] and loss-based [18, 19, 22] methods. Gradient-based methods perform k rounds of backpropagation to obtain the gradients $\nabla_{\theta_s}\ell_1, \nabla_{\theta_s}\ell_2, \ldots, \nabla_{\theta_s}\ell_k$ corresponding to the k tasks for the shared parameters θ_s. These methods carefully design aggregation algorithms $\mathcal{A}$ to compute a final joint gradient $g_{\text{share}} = \mathcal{A}(\nabla_{\theta_s}\ell_1, \nabla_{\theta_s}\ell_2, \ldots, \nabla_{\theta_s}\ell_k)$. Benefiting from access to precise gradients from all tasks in each update, gradient-based approaches often exhibit impressive results. In contrast, loss-based methods do not directly obtain the gradients of shared parameters; instead, they assign weights $(\omega_1, \omega_2, \ldots, \omega_k)$ to each task based on the current task losses $(\ell_1, \ell_2, \ldots, \ell_k)$, creating an aggregated loss. Backpropagation is then performed on this aggregated loss, allowing these methods to preserve training-time efficiency as they require only a single backpropagation per update.

However, both approaches primarily focus on gradient aggregation within shared parameters, while overlooking gradient optimization in task-specific parameters, and some works [29, 33] even assume that task-specific parameters do not contribute to the gradient aggregation for shared parameters. Specifically, previous methods directly retain the gradients computed from the corresponding losses for task-specific parameters, as shown in Fig. 1. In gradient-based methods, the joint gradient for shared parameters is obtained by aggregating $g_{\text{share}} = \mathcal{A}(\nabla_{\theta_s}\ell_1, \nabla_{\theta_s}\ell_2, \ldots, \nabla_{\theta_s}\ell_k)$, while the gradient for each

‡ Corresponding author. This work was partly supported by NSFC (62222607) and Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

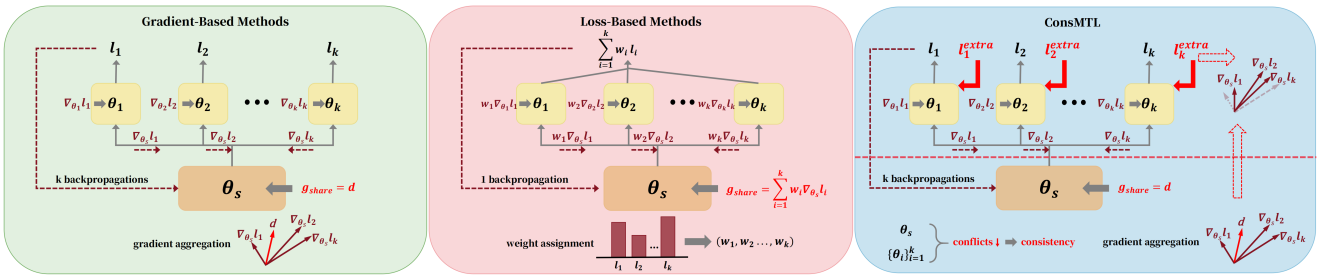


Figure 1. Comparison of multi-task learning methods. Gradient-based methods aggregate gradients for shared parameters to find the joint update vector d, while loss-based methods assign weights to balance the loss. Both approaches primarily focus on the shared parameters, while ConsMTL utilizes the optimization of both shared and task-specific parameters to consistently mitigate gradient conflicts.

task-specific parameters θ_i remains $\nabla_{\theta_i}\ell_i$. Loss-based methods exhibit a similar issue: the gradient for shared parameters is calculated from the weighted combination of losses $\sum_{i=1}^{k}\omega_i\ell_i$, while the gradient for each task-specific parameter θ_i is simply retained as $\omega_i\nabla_{\theta_i}\ell_i$. This simple gradient retention causes task-specific parameters to only receive the gradients from the corresponding tasks, while being unaware of the overall gradient conflicts in MTL. In reality, task-specific parameters not only capture task-specific information but also influence the gradients propagated back to shared parameters, thereby affecting the phenomenon of gradient conflicts. Building on this, we further highlight the importance of ensuring consistency between the optimization of shared parameters and task-specific parameters in order to effectively mitigate gradient conflicts, which is overlooked by previous methods.

Motivated by the inconsistent optimization in previous works, we propose ConsMTL, aiming to fully leverage the potential of task-specific parameters by enabling them to be aware of gradient conflicts, thereby aligning their optimization with shared parameters towards consistently mitigating these conflicts. To this end, we revisit the optimization process in MTL and model it as a bi-level optimization. In the upper-level optimization, we focus on the shared parameters and identify an update vector that minimizes gradient conflicts while maximizing fairness in optimization. In the lower-level optimization, we concentrate on task-specific parameters and introduce an additional loss term for each task-specific parameter, which guides the k gradients back-propagated to the shared parameters to gradually converge towards the joint update vector, consistently mitigating gradient conflicts. The contributions are outlined as follows:

- We point out the inconsistent optimization in previous methods and show that task-specific parameters also count since they influence the gradients propagated back to shared parameters. Thus, we consider utilizing both shared parameters and task-specific parameters to consistently mitigate gradient conflicts.
- We propose ConsMTL, a new multi-task learning paradigm that models MTL as a bi-level optimization problem whereby for the first time the shared parameters and task-specific parameters are optimized alternatively. We establish optimization objectives for shared and task-specific parameters respectively, which complement each other to jointly address the issue of gradient conflicts, thereby achieving consistency.
- Extensive experiments show that ConsMTL achieves SOTA performance across various benchmarks. In particular, ConsMTL is the only method that surpasses the corresponding Single-Task Learning baseline across all three tasks on the NYUv2 benchmark. On the more challenging QM9 benchmark, ConsMTL reduces the average performance drop by over 30%.

2. Related Works

In multi-task learning, previous work can be broadly categorized into loss-based and gradient-based approaches. Loss-based methods assign weights to different tasks to address gradient conflicts and task imbalance. These methods are efficient as they only require backpropagation on the aggregated loss, but the varying loss scales pose a significant challenge in task balancing. Various strategies have been proposed to handle task weighting, such as uncertainty-based weighting [18], task prioritization [15], dynamic weight averaging [25], self-paced learning [27], geometric loss formulations [10], random loss weighting [20], and adaptive optimization techniques [22]. Although loss-oriented methods preserve training-time efficiency, they usually suffer from unsatisfactory overall performance.

Gradient-based methods address the gradient conflict problem by fully leveraging the gradient information across tasks and performing gradient aggregation for shared parameters. Works have shown significant performance gains by applying methods such as Pareto optimization [32], gradient normalization [8], conflict projection [38], gradient sign dropout [9], conflict-averse gradients [21], independent gradient alignment [33], Bayesian uncertainty gradients [2], Nash bargaining solution [29] and fair resource allocation

[3]. In our approach, the upper-level optimization also employs gradient-based methods.

Remarks. Among existing loss-based and gradient-based methods, to the best of our knowledge, only IMTL [23] considers the learning of task-specific parameters and leverages task-specific parameters to balance loss scales. However, this approach remains the issue of only allowing task-specific parameters to be aware of the current task loss and does not contribute to mitigating gradient conflicts.

3. Method

3.1. Rethinking MTL as Bi-Level Optimization

Notations. We utilize a single model to optimize $k \geq 2$ tasks simultaneously. The input space is defined as $\mathcal{X}$, and the model consists of shared parameters θ_s and task-specific parameters $\{\theta_i\}_{i=1}^k$. For task i and input sample $\boldsymbol{x} \in \mathcal{X}$, the task-specific loss is represented as $l_i = l_i(\boldsymbol{x}, \theta_s, \theta_i)$. The gradient contribution of task i to the shared parameters is given by $\nabla_{\theta_s}\ell_i = \nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)$, while the gradient contribution to the task-specific parameters θ_i is $\nabla_{\theta_i}\ell_i = \nabla_{\theta_i} l_i(\boldsymbol{x}, \theta_s, \theta_i)$. Loss from task i does not contribute gradients to other task-specific parameters $\{\theta_j\}_{j\neq i}$. Additionally, we denote the shared representation obtained from input sample $\boldsymbol{x}$ after applying the shared parameters as $\boldsymbol{z}$.

Design of ConsMTL. In ConsMTL, we address the gradient conflicts issue through a bi-level optimization. The upper-level optimization focuses on the gradient aggregation in shared parameters, aiming to find an update vector $d \in \mathbb{R}^n$ in the ball of radius ϵ centered around zero, B_ϵ whose dimension is the same as the shared parameters. Given $\{\nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)\}_{i=1}^k$, the upper-level optimization objective is shown as follows:

$$d = \arg\min_{d\in B_\epsilon} \mathcal{A}(d, \{\nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)\}_{i=1}^k), \tag{1}$$

where $\mathcal{A}$ is an aggregation function to ensure optimization fairness and mitigate gradient conflicts, which is discussed in detail in Section 3.2.

The lower-level optimization concentrates on the task-specific parameters $\{\theta_i\}_{i=1}^k$. Notably, the gradient of task i with respect to the shared parameters is represented by $\nabla_{\theta_s}\ell_i = \nabla_{\theta_s}\ell_i(\boldsymbol{x}, \theta_s, \theta_i)$, which can also be influenced by θ_i. Our objective is to leverage this potential influence so that optimizing θ_i will not only reduce the corresponding loss ℓ_i but also gradually guide the gradients $\{\nabla_{\theta_s}\ell_i\}_{i=1}^k$ to gradually converge towards the joint update vector, so that the shared parameters can learn a more unified representation. In the process of the lower-level optimization, we assume that the update vector d obtained from the upper-level optimization is fixed. Thus, the objective is given by:

$$\{\theta_i\}_{i=1}^k = \arg\min_{\{\theta_i\}_{i=1}^k} \sum_{i=1}^k \mathcal{L}(\theta_i, d, \nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)), \tag{2}$$

where $\mathcal{L}$ is a loss function specifically designed for θ_i to balance the goals of reducing ℓ_i and aligning the gradients of shared parameters, which is explained in detail in Section 3.3. Overall, the objective of the bi-level optimization in ConsMTL can be formulated as follows:

$$d = \arg\min_{d\in B_\epsilon} \mathcal{A}(d, \{\nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)\}_{i=1}^k),$$
$$s.t.\ \{\theta_i\}_{i=1}^k = \arg\min_{\{\theta_i\}_{i=1}^k} \sum_{i=1}^k \mathcal{L}(\theta_i, d, \nabla_{\theta_s} l_i(\boldsymbol{x}, \theta_s, \theta_i)).$$

3.2. Upper-Level Optimization

During the optimization process, we represent updates of shared parameters at each iteration as $\theta_s^{t+1} = \theta_s^t - \eta d$, where η is the current step size and d is the computed update direction. Considering a first-order Taylor expansion $\ell_i(\boldsymbol{x}, \theta_s^{t+1}, \theta_i^t) - \ell_i(\boldsymbol{x}, \theta_s^t, \theta_i^t) \approx -\eta \nabla_{\theta_s^t}\ell_i^\top d$, we interpret $\nabla_{\theta_s}\ell_i^\top d$ as the utility of task i at the current step following recent works [3, 22, 29], and propose to minimize the mean of the inverse utilities for each task:

$$\arg\min_{d\in B_\epsilon} \frac{1}{k}\sum_{i=1}^k \frac{1}{\nabla_{\theta_s}\ell_i^\top d} \qquad \text{s.t.} \quad \nabla_{\theta_s}\ell_i^\top d > 0, \forall i. \tag{3}$$

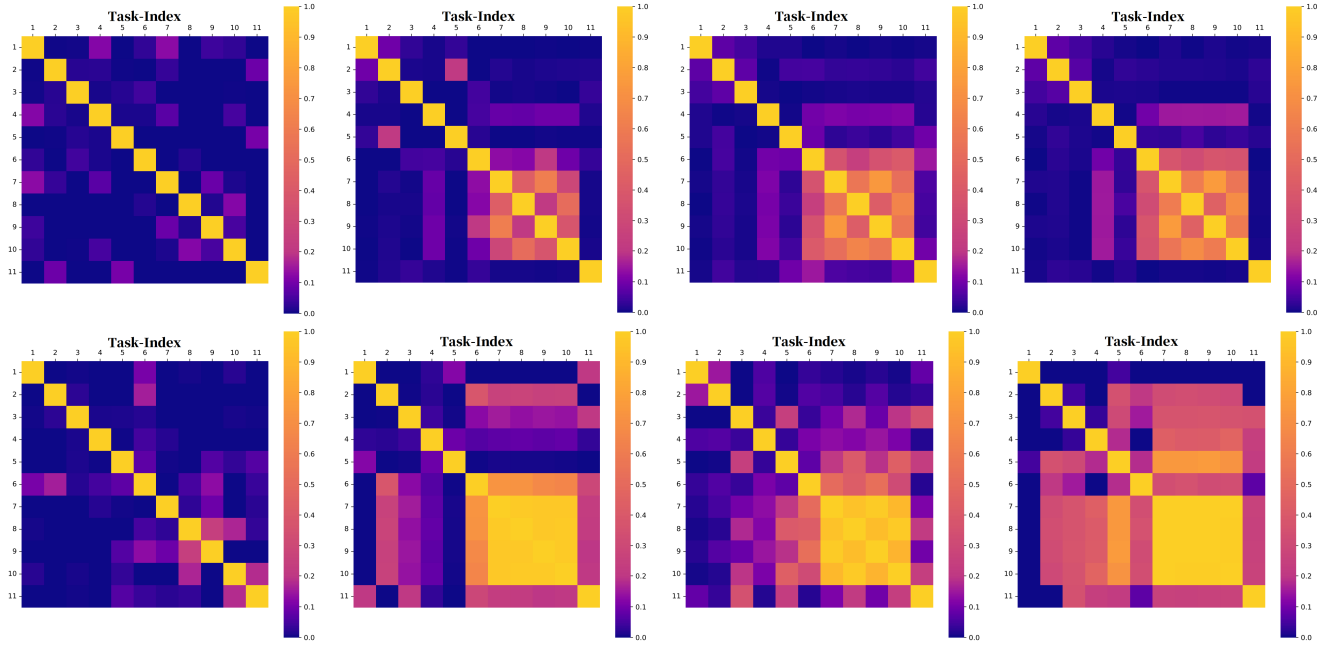
This approach emphasizes tasks with lower utilities, which helps prevent tasks with higher utilities from dominating the optimization. Additionally, $\nabla_{\theta_s}\ell_i^\top d$ serves as an estimate of the immediate loss reduction for task i, effectively representing its current optimization speed. Conversely, $1/\nabla_{\theta_s}\ell_i^\top d$ indicates the approximate number of steps needed for a unit of progress in task i. Thus, the objective in (3) aims to minimize the average steps required for unit progress across all tasks.

Remarks. Similar to MGDA [32] and Nash-MTL [29], the optimization objective in (3) can be viewed as a special case of the FairGrad framework [3], which is a powerful gradient-based baseline and has been extensively compared with our approach in experiments. However, our motivation for minimizing the average steps required for unit progress across all tasks differs from its perspective of fair resource allocation. Furthermore, we introduce an innovative approach by deriving the analytical solution for d using Lagrange multipliers as follows.

For each task i, both the objective function and constraints in (3) are convex, thus we can define the Lagrangian for this problem:

$$L = \frac{1}{k}\sum_{i=1}^k \frac{1}{\nabla_{\theta_s}\ell_i^\top d} + \lambda(\|d\|^2 - \epsilon^2) - \sum_{i=1}^k \mu_i(\nabla_{\theta_s}\ell_i^\top d),$$

where $\lambda \geq 0$ and $\mu_i \geq 0$ are the Lagrange multipliers corresponding to the constraints. The multiplier λ enforces the constraint on the norm of d, while μ_i enforces the positivity condition on the inner products $\nabla_{\theta_s}\ell_i^\top d > 0$. From



(a) Epoch=0 (b) Epoch=20 (b) Epoch=40 (d) Epoch=60

Figure 2. The cosine similarity matrix of gradients for shared representation $\boldsymbol{z}$ across different tasks during training on QM9 (11-task). Negative values are clipped to 0. The top row represents the case using only upper-level optimization, which can be considered as a naive gradient-based method. The bottom row introduces lower-level optimization for task-specific parameters, i.e., ConsMTL.

the Complementary Slackness condition of the KKT conditions, we have $\mu_i \nabla_{\theta_s} \ell_i^\top d = 0$, $\lambda(\|d\|^2 - \epsilon^2) = 0$. Since $\nabla_{\theta_s} \ell_i^\top d > 0$ and $\|d\| \leq \epsilon$, it follows that $\|d\| = \epsilon$ and $\mu_i = 0$ for all i. Additionally, the gradient of the Lagrangian with respect to d at the optimal point is:

$$\nabla_d L = -\frac{1}{k} \sum_{i=1}^{k} \frac{1}{(\nabla_{\theta_s} \ell_i^\top d)^2} \nabla_{\theta_s} \ell_i + 2\lambda d - \sum_{i=1}^{k} \mu_i \nabla_{\theta_s} \ell_i = 0.$$

Given that $\mu_i = 0$, the stationarity condition simplifies to:

$$\sum_{i=1}^{k} \frac{1}{(\nabla_{\theta_s} \ell_i^\top d)^2} \nabla_{\theta_s} \ell_i = 2k\lambda d. \tag{4}$$

Consistent with prior works [3, 29], we assume that the gradients of different tasks are linearly independent, as otherwise, it would indicate convergence to a Pareto stationary point. As a result, the vector d can be expressed as a linear combination of the gradients from each task $d = \sum_{i=1}^{k} \alpha_i \nabla_{\theta_s} \ell_i$. Neglecting the factor $2k\lambda$ in Eq. 4, which can be adjusted using the step size η_t, we obtain $\alpha_i = \frac{1}{(\nabla_{\theta_s} \ell_i^\top d)^2}$, or equivalently, $(\nabla_{\theta_s} \ell_i^\top d)^2 = \frac{1}{\alpha_i}$.

Let $\boldsymbol{G} = (\nabla_{\theta_s} \ell_1, \nabla_{\theta_s} \ell_2, \ldots, \nabla_{\theta_s} \ell_k) \in \mathbb{R}^{n \times k}$ represent the matrix of task gradients. We can then rewrite this equation in matrix form as:

$$\boldsymbol{G}^\top \boldsymbol{G} \boldsymbol{\alpha} = \sqrt{\mathbf{1} \oslash \boldsymbol{\alpha}}, \tag{5}$$

where the square root is applied element-wise. We treat Eq. 5 as a constrained nonlinear least squares problem, as can be efficiently solved by the `scipy` library to obtain the coefficient α, thereby deriving the update vector d.

3.3. Lower-Level Optimizatoin

In the context of lower-level optimization, we consider the following problem: since $\nabla_{\theta_s} \ell_i = \nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$, changes in θ_i inherently influence $\nabla_{\theta_s} \ell_i$. *How can we exploit this relationship to mitigate gradient conflicts?* A natural idea is to leverage $\nabla_{\theta_s} \ell_i$ to set an additional loss for θ_i. Specifically, given the current gradients of the k tasks with respect to the shared parameters, $\nabla_{\theta_s} \ell_1, \nabla_{\theta_s} \ell_2, \ldots, \nabla_{\theta_s} \ell_k$, and the update vector d obtained from the upper-level optimization, we aim to align these k gradients with d. An intuitive approach is to maximize the inner product between each $\nabla_{\theta_s} \ell_i$ and d, which we define as:

$$\mathcal{L}_i^{extra} = -\lambda < \nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i), d >,$$

where λ is a hyperparameter that controls the magnitude of the loss, and d is the target update vector detached from the computation graph.

However, calculating $\nabla_{\theta_i} \mathcal{L}_i^{extra}$ is memory exhausted. Although $\nabla_{\theta_s} \ell_i = \nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$ can be used to compute the gradient with respect to θ_i, this approach would include the shared parameters θ_s in the newly created computation graph. Since the number of parameters in θ_s is often much

larger than that in θ_i, this leads to substantial memory and time overhead. Therefore, we seek an alternative objective by leveraging the shared representation $\boldsymbol{z} = \mathcal{N}et(\boldsymbol{x}, \theta_s)$, which is obtained by applying the shared parameters θ_s to input $\boldsymbol{x}$. Then, by applying the chain rule of differentiation:

$$\nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i) = \nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i) \cdot \nabla_{\theta_s} \boldsymbol{z}. \tag{6}$$

Note that in Eq. 6, $\nabla_{\theta_s} \boldsymbol{z}$ is the same for all tasks. Therefore, in order to align $\nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$, we can consider aligning $\nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$ instead. For $\nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$, we aim for alignment with the update vector d; whereas for $\nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$, the following proposition provides the corresponding target d_z:

Proposition 3.1. *Given the update vector d and the weight coefficients $\{\alpha_i\}_{i=1}^k$ obtained from the upper-level optimization, the gradient alignment target corresponding to the shared representation $\boldsymbol{z}$ is given by:*

$$d_{\boldsymbol{z}} = \sum_{i=1}^{k} \alpha_i \nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i).$$

Intuitively, the vector d obtained from the upper-level optimization is the joint gradient for the shared parameters. By the chain rule, we have:

$$\begin{aligned} d_{\boldsymbol{z}} \cdot \nabla_{\theta_s} \boldsymbol{z} &= [\sum_{i=1}^{k} \alpha_i \nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)] \cdot \nabla_{\theta_s} \boldsymbol{z} \\ &= \sum_{i=1}^{k} \alpha_i \nabla_{\theta_s} \ell_i(\boldsymbol{x}, \theta_s, \theta_i) = d. \end{aligned}$$

This implies that by leveraging the parameter feedback through $\{\alpha_i\}_{i=1}^k$, $d_{\boldsymbol{z}}$ precisely represents the joint gradient of the shared representation $\boldsymbol{z}$, corresponding to the alignment target for the gradients $\nabla_{\boldsymbol{z}} \ell_i(\boldsymbol{x}, \theta_s, \theta_i)$ across all tasks. Consequently, the extra loss added to θ_i is redefined as:

$$\mathcal{L}_i^{extra} = -\lambda < \nabla_z \ell_i(\boldsymbol{x}, \theta_s, \theta_i), d_{\boldsymbol{z}} >, \tag{7}$$

and the overall loss for θ_i is given by:

$$\mathcal{L}_i(\theta_i) = \ell_i(\boldsymbol{x}, \theta_s, \theta_i) + \mathcal{L}_i^{extra}. \tag{8}$$

By incorporating $\mathcal{L}_i^{extra}$, θ_i not only optimizes the corresponding task loss ℓ_i via gradient descent, but also gains awareness of the overall gradient conflicts. For gradient $\nabla_{\theta_s} \ell_i$ that deviates from the joint direction, the computed inner product will be smaller, resulting in a larger $\mathcal{L}_i^{extra}$. This, in turn, enforces the optimization of θ_i towards a direction that promotes gradient alignment and increases the inner product. As a result, the update direction of the shared parameters θ_s becomes more coherent, leading to the learning of more unified shared features, while the task-specific parameters θ_i effectively capture task-specific information.

Algorithm 1 ConsMTL Procedure

Input: Shared parameters θ_s^0; Task-specific parameters $\{\theta_i^0\}_{i=1}^k$; Learning rate $\{\eta_t\}_{t=1}^T$.
for $t = 1$ **to** $T - 1$ **do**
  // Upper-level optimization
  Compute gradients $\boldsymbol{G}(\theta_s^t) = [\nabla_{\theta_s} \ell_1, \ldots, \nabla_{\theta_s} \ell_k]$
  Solve Eq. 5 to obtain $\boldsymbol{\alpha}_t$ and d_t
  Update shared parameters $\theta_s^{t+1} = \theta_s^t - \eta_t d_t$
  // Lower-level optimization
  Compute task-specific losses $\{\mathcal{L}_i(\theta_i^t)\}_{i=1}^k$ using $\boldsymbol{\alpha}_t$ and d_t through Eq. 7 and Eq. 8
  for $i = 1$ **to** k **do**
    Update task-specific parameters through $\theta_i^{t+1} = \theta_i^t - \eta_t \nabla_{\theta_i^t} \mathcal{L}_i(\theta_i^t)$
  end for
end for
return θ_s^T and $\{\theta_i^T\}_{i=1}^k$

It is important to note that λ serves as a critical hyperparameter in our approach, governing the relative magnitudes of the two loss terms and requiring specific tuning for each task. In Section 4.5, our ablation study provides a detailed examination of the influence of various λ values on experimental outcomes, along with empirical guidelines for selecting appropriate λ values.

3.4. Overall Framework

In ConsMTL, we solve the bi-level optimization by alternatively updating the upper- and lower-level objectives. Specifically, after obtaining the precise gradients through backpropagation, we first perform gradient aggregation in the upper-level optimization to obtain the update vector d and weight coefficients $\{\alpha_i\}_{i=1}^k$, which are then fed back to the lower-level optimization to compute task-specific losses and perform gradient descent to update $\{\theta_i\}_{i=1}^k$.

It is important to note that, while our method requires additional computation of gradients with respect to the shared representation $\boldsymbol{z}$, this backpropagation passes only through the task-specific parameters $\{\theta_i\}_{i=1}^k$, which generally have a small parameter count, thus incurring little overhead in memory and computational cost. As shown in Fig. 3, on the CelebA benchmark with as many as 40 tasks, our approach does not exhibit noticeable disadvantages in memory usage or computation time compared to other gradient-based methods. Our complete algorithmic procedure is summarized in Algorithm 1.

In Fig. 2, we also visualize the cosine similarity matrix of the gradients of different tasks with respect to the shared representation $\boldsymbol{z}$ during the optimization process. By incorporating the lower-level optimization, the gradient similarity across tasks significantly improves, which effectively mitigates gradient conflicts and allows the shared parame-

Table 1. Results on NYU-v2 (3-task) dataset. Each experiment is repeated 3 times with different random seeds and the average is reported. The detailed standard error is reported in the supplementary material. The best scores are reported in **gray**.

Method	Segmentation		Depth		Surface Normal					MR ↓	$\Delta m\%$ ↓
					Angle Distance ↓		Within t° ↑				
	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		
LS	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	12.44	5.59
SI	38.45	64.27	0.5354	0.2201	27.60	23.37	22.53	48.57	62.32	11.11	4.39
RLW [19]	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62	15.22	7.78
DWA [25]	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39	11.44	3.57
UW [17]	36.87	63.17	0.5446	0.2260	27.04	22.61	23.54	49.05	63.65	11.00	4.05
MGDA [32]	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36	8.56	1.38
PCGrad [39]	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	11.89	3.97
GradDrop [9]	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	10.56	3.58
CAGrad [21]	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58	7.44	0.20
IMTL-G [23]	39.35	65.60	0.5426	0.2256	26.02	21.19	26.20	53.13	66.24	6.67	-0.76
MoCo [13]	40.30	**66.07**	0.5575	**0.2135**	26.67	21.83	25.61	51.78	64.85	7.00	0.16
Nash-MTL [29]	40.13	65.93	**0.5261**	0.2171	25.26	20.08	28.40	55.47	68.15	4.22	-4.04
Aligned-MTL [33]	40.15	66.05	0.5520	0.2291	25.37	19.89	28.30	55.29	67.95	6.56	-3.12
FAMO [22]	38.88	64.90	0.5474	0.2194	25.06	19.57	29.21	56.61	68.98	5.56	-4.10
FairGrad [3]	39.74	66.01	0.5377	0.2236	24.84	19.60	29.26	56.58	69.16	3.67	-4.66
ConsMTL	**40.33**	65.32	0.5491	0.2151	**24.35**	**18.80**	**31.06**	**58.28**	**70.31**	**2.78**	**-6.72**

ters to learn a more unified representation. It is worth noting that we only visualize the cosine similarity matrix for the first 60 out of 300 epochs, as gradient similarities are typically more pronounced in the early stages of training. Towards the end of training, the gradient similarity stabilizes or even decreases, and the gradient directions ultimately become linearly correlated on the Pareto frontier, which aligns with the observation in [16]. Overall, through gradient aggregation in the upper-level optimization and gradient alignment in the lower-level optimization, ConsMTL effectively leverages both shared and task-specific parameters to jointly mitigate gradient conflicts, thereby achieving consistent multi-task learning.

4. Experiments

4.1. Protocols

We evaluate the proposed ConsMTL on a variety of multi-task learning benchmarks, varying in both the number of tasks and their types. First, we validate the performance of ConsMTL on scene understanding benchmarks, including NYUv2 [34] and Cityscapes [11]. Second, we conduct experiments on regression tasks from the QM9 [5] benchmark. Finally, we apply the method to image-level classification using the CelebA [26] dataset. Additionally, we perform an ablation study to explore the impact of the hyperparameter λ on the experimental results.

Baselines: We compare our proposed ConsMTL described in Section 3 with the following methods: Single-task learning (STL), Linear Scalarization (LS), Scale-Invariant (SI), Dynamic Weight Average (DWA) [25], Uncertainty Weighting (UW) [17], Multi-Gradient Descent Algorithm (MGDA) [32], Random Loss Weighting (RLW) [19], PCGrad [39], GradDrop [9], CAGrad [21], IMTL-G [23], Nash-MTL [29], Moco [13], Aligned-MTL [33], FAMO [22] and FairGrad [3].

Evaluation Metrics: Following previous works [22, 29, 32, 38], since MTL lacks a singular objective and performance metrics can differ across tasks, we focus on two overall performance metrics: (1) $\boldsymbol{\Delta m\%}$ is the average performance drop relative to the single task learning baseline:

$$\Delta m\% = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} (-1)^{\delta_i} (M_{m,i} - M_{b,i}) / M_{b,i} \times 100\%,$$

where $\mathcal{K}$ is the number of total metrics. $M_{b,i}$ is the value of metric M_i obtained by the STL baseline and $M_{m,i}$ denotes the value from the compared MTL method. $\delta_i = 1$ if a higher value is better for the metric M_i and 0 otherwise. (2) **Mean Rank (MR)**: MR reports the average rank of each method across metrics (lower is better). If a method ranks first in all tasks, its MR will equal 1.

4.2. Experiments on Scene Understanding

Following previous works [3, 22, 29], we evaluate ConsMTL on the NYUv2 and Cityscapes datasets. The NYUv2 dataset [34] consists of 1449 densely labeled indoor images, encompassing three pixel-level tasks: semantic segmentation, depth estimation, and surface normal prediction. Similarly, the Cityscapes dataset [11] contains 5000 street-view images with two tasks: semantic segmentation and depth estimation. We follow the setup in [22, 29] using MTAN [23].

Table 2. Results on Cityscapes (2-task) dataset. Each experiment is repeated 3 times with different random seeds and the average is reported. The detailed standard error is reported in the supplementary material. The best scores are reported in **gray**.

Method	CityScapes					
	Segmentation		Depth		MR ↓	$\Delta m\%$ ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓		
STL	74.01	93.16	0.0125	27.77		
LS	75.18	93.49	0.0155	46.77	10.00	22.60
SI	70.95	91.73	0.0161	33.83	12.50	14.11
RLW [19]	74.57	93.41	0.0158	47.79	12.50	24.38
DWA [25]	75.24	93.52	0.0160	44.37	9.75	21.45
UW [17]	72.02	92.85	0.0140	30.13	8.50	5.89
MGDA [32]	68.84	91.54	0.0309	33.50	13.00	44.14
PCGrad [39]	75.13	93.48	0.0154	42.07	10.25	18.29
GradDrop [9]	75.27	93.53	0.0157	47.54	9.25	23.73
CAGrad [21]	75.16	93.48	0.0141	37.60	8.75	11.64
IMTL-G [23]	75.33	93.49	0.0135	38.41	7.00	11.10
MoCo [13]	75.42	93.55	0.0149	34.19	5.75	9.90
Nash-MTL [29]	75.41	93.66	**0.0129**	35.02	4.00	6.82
Aligned-MTL [33]	**75.77**	**93.69**	0.0133	32.66	**2.00**	5.27
FAMO [22]	74.54	93.29	0.0145	32.59	9.00	8.13
FairGrad [3]	75.72	93.68	0.0134	32.25	2.25	5.18
ConsMTL	75.57	93.32	0.0131	**26.41**	4.00	**-0.59**

Table 3. Results on CelebA (40-task) dataset. Each experiment is repeated 3 times with different random seeds and the average is reported. The detailed standard error is reported in the supplementary material. The best scores are reported in **gray**.

Method	MR ↓	$\Delta m\%$ ↓
STL	22.60	7.65
LS	7.62	4.15
SI	9.43	7.20
RLW [19]	6.65	1.46
DWA [25]	8.38	3.20
UW [17]	6.95	3.23
MGDA [32]	12.90	14.85
PCGrad [39]	8.05	3.17
GradDrop [9]	9.45	3.29
CAGrad [21]	7.60	2.48
IMTL-G [23]	5.80	0.84
Nash-MTL [29]	6.12	2.84
FAMO [22]	5.97	1.21
FairGrad [3]	6.67	0.37
ConsMTL	**3.40**	**-1.42**

To align with previous works, the model is trained for 200 epochs with a learning rate of 10^{-4} for the first 100 epochs, decaying by half for the remaining epochs. The hyperparameter λ is set to 1 for NYUv2 and 50 for CityScapes.

The results presented in Table 1 and Table 2 highlight the outstanding performance of our approach. On the NYUv2 dataset, while existing methods typically outperform the STL baseline on segmentation and depth estimation tasks, they fall short in surpassing STL on the surface normal prediction task, revealing an imbalance between tasks. In contrast, by leveraging the potential of task-specific parameters to mitigate gradient conflicts, our approach achieves an impressive MR of **2.78** and the best $\Delta m\%$ of **-6.72%**, and is the only method that consistently outperforms STL across all 3 tasks and 9 evaluation metrics.

On the Cityscapes benchmark, previous methods generally show superior optimization on the segmentation task but fall short on the depth estimation task. In comparison,

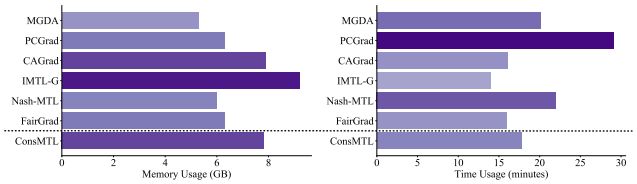


Figure 3. Comparison of memory usage and one-epoch runtime across different gradient-based methods on CelebA (40-task).

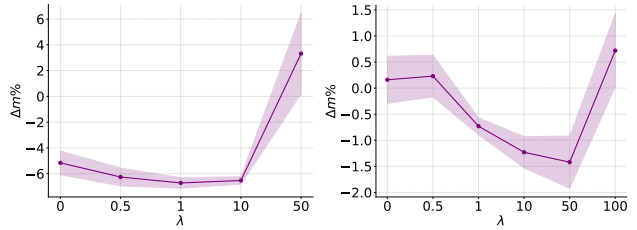


Figure 4. Ablation study on different λ values on NYUv2 (left) and CelebA (right). Each experiment was repeated three times with different random seeds, with error bars shown.

ConsMTL delivers a more balanced performance and is the only method to achieve a negative $\Delta m\%$ of **-0.59%**, further demonstrating the effectiveness of our approach.

4.3. Experiments on Image-Level Classification

The CelebA benchmark [26] is a large-scale facial attributes dataset containing more than 200,000 images, each annotated with 40 binary attributes. This benchmark represents a multi-task learning classification problem with 40 distinct tasks, where each task predicts the presence or absence of a specific attribute. Following the experimental setup in [3, 22], we employ a 9-layer convolutional neural network as the model backbone, augmented with task-specific linear layers. The model is trained for 15 epochs using the Adam optimizer with λ setting to 50.

The results are shown in Table 3. Our proposed method, ConsMTL, achieves SOTA performance and is the only approach to obtain a negative performance drop of **-1.42%**. This indicates that, compared to other methods, ConsMTL is the only one to outperform the STL baseline in terms of the average performance across all 40 tasks. Additionally, Fig. 3 presents detailed information on memory usage and time consumption. It can be observed that, by incorporating the gradient of the intermediate representation $\boldsymbol{z}$ instead of the gradient of shared parameters to impose an additional loss, ConsMTL does not exhibit noticeable disadvantages, even in scenarios with as many as 40 tasks.

4.4. Experiments on Multi-Task Regression

For multi-task regression, we evaluate the performance of ConsMTL on the QM9 dataset [5], which consists of over 130,000 stable organic molecules represented as graphs. Each molecule is annotated with one of 11 molecular properties, which vary significantly in both scale and difficulty.

Table 4. Results on QM9 (11-task) dataset. Each experiment is repeated 3 times with different random seeds and the average is reported. The detailed standard error is reported in the supplementary material. The best scores are reported in **gray**.

METHOD	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	**MR**↓	$\Delta m\%\downarrow$
	MAE ↓												
STL	0.067	0.181	60.57	53.91	0.502	4.53	58.8	64.2	63.8	66.2	0.072		
LS	0.106	0.325	**73.57**	89.67	5.19	14.06	143.4	144.2	144.6	140.3	0.128	9.09	177.6
SI	0.309	0.345	149.8	135.7	**1.00**	4.50	55.3	55.75	55.82	55.27	0.112	5.91	77.8
RLW [19]	0.113	0.340	76.95	92.76	5.86	15.46	156.3	157.1	157.6	153.0	0.137	10.64	203.8
DWA [25]	0.107	0.325	74.06	90.61	5.09	13.99	142.3	143.0	143.4	139.3	0.125	8.82	175.3
UW [17]	0.386	0.425	166.2	155.8	1.06	4.99	66.4	66.78	66.80	66.24	0.122	7.27	108.0
MGDA [32]	0.217	0.368	126.8	104.6	3.22	5.69	88.37	89.4	89.32	88.01	0.120	8.91	120.5
PCGRAD [39]	0.106	0.293	75.85	88.33	3.94	9.15	116.36	116.8	117.2	114.5	0.110	7.27	125.7
CAGRAD [21]	0.118	0.321	83.51	94.81	3.21	6.93	113.99	114.3	114.5	112.3	0.116	8.36	112.8
IMTL-G [23]	0.136	0.287	98.31	93.96	1.75	5.69	101.4	102.4	102.0	100.1	0.096	7.18	77.2
NASH-MTL [29]	**0.102**	0.248	82.95	81.89	2.42	5.38	74.5	75.02	75.10	74.16	0.093	4.55	62.0
FAMO [22]	0.15	0.30	94.0	95.2	1.63	4.95	70.82	71.2	71.2	70.3	0.10	5.82	58.5
FAIRGRAD [3]	0.117	0.253	87.57	84.00	2.15	5.07	70.89	71.17	71.21	70.88	0.095	4.91	57.9
CONSMTL	0.115	**0.202**	82.69	**67.58**	1.61	**3.33**	**48.84**	**49.04**	**49.07**	**49.63**	**0.077**	**2.00**	**23.2**

Following established protocols [3, 22, 29], we employ a graph-based model architecture, specifically the message-passing neural network (MPNN) [14]. The task-specific parameters for each task consist of only a single linear layer. The models are trained for 300 epochs, with a learning rate scheduler to adjust the learning rate throughout training, consistent with prior works. The parameter λ is set to 1e-3.

The experimental results are presented in Fig. 2 and Table 4. As shown in Fig. 2, compared to the naive MTL paradigm with only upper-level optimization, by incorporating lower-level optimization, we observe a notable improvement in gradient similarity across tasks, which effectively mitigates gradient conflicts and enables the shared parameters to learn a more unified representation. On the QM9 benchmark, where task difficulty varies greatly and gradient conflicts are more pronounced, the consistent optimization of both shared and task-specific parameters significantly mitigates these conflicts. As seen in Table 4, ConsMTL reduces the average performance drop by over **30%** compared to prior methods. Addtionally, unlike previous approaches which generally perform worse than the STL baseline across all tasks, our method surpasses the STL in 5 out of 11 tasks and exhibits strong competitiveness on the remaining tasks. This clearly highlights the immense potential of MTL and the superiority of our approach.

4.5. Ablation Study on the Hyperparameter λ

The hyperparameter λ plays a crucial role in our method, as it governs the relative magnitude of the two loss terms, and thus needs to be specified for each task. Figure 4 shows the impact of varying λ on the experimental results. A well-chosen λ can improve performance, whereas an excessively large λ may cause a significant performance decline. This phenomenon can be intuitively explained: smaller values of λ help adjust the gradients from the original loss without introducing large fluctuations. However, when λ becomes too large, leading to $\nabla_{\theta_i}\mathcal{L}_i^{\text{extra}}$ dominating, the ability of $\nabla_{\theta_i}\ell_i$ to reduce task-specific loss is relatively weakened.

Due to the significant differences between $\|\nabla_{\theta_i}\ell_i\|$ and $\|\nabla_{\theta_i}\mathcal{L}_i^{\text{extra}}\|$ across different tasks, providing a unified criterion for setting λ is challenging. However, as shown in Figure 4, the performance drop on NYUv2 and Celeba demonstrates a similar trend of initially decreasing and then increasing with larger λ. Based on this, we recommend gradually increasing λ from 0 in practice to identify an optimal value. In our experiments, we typically print $\|\nabla_{\theta_i}\ell_i\|$ and $\|\nabla_{\theta_i}\mathcal{L}_i^{\text{extra}}\|$ during the initial epochs and compare their approximate magnitudes to determine a suitable range for λ. We generally suggest setting λ that $\|\nabla_{\theta_i}\mathcal{L}_i^{\text{extra}}\|$ is approximately $0.1 \times \|\nabla_{\theta_i}\ell_i\|$ and applying gradient clipping to mitigate excessive fluctuations.

5. Conclusion, Limitations and Future Work

In this work, we point out the inconsistent optimization of previous methods and emphasize the important role of task-specific parameters that influence the gradients propagated back to shared parameters. Based on this key observation, we propose ConsMTL, a novel multi-task learning paradigm that models MTL as a bi-level optimization problem. We introduce optimization objectives for shared and task-specific parameters, respectively, which complement each other to address the issue of gradient conflicts consistently. Extensive experiments demonstrate that ConsMTL achieves SOTA performance across various benchmarks.

Limitations and Future Work. Although introducing additional optimization objectives for task-specific parameters helps mitigate gradient conflicts, the extra gradient computation inevitably introduces memory and computational overhead. Future work can explore more efficient methods to achieve this goal. Additionally, as shown in Section 4.5, different λ can have varying effects on the experimental results, and the appropriate λ may change during different stages of training. We leave the investigation of adaptive methods for dynamically adjusting λ for future work.

References

[1] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 123–133, 2021. 1

[2] Idan Achituve, Idit Diamant, Arnon Netzer, Gal Chechik, and Ethan Fetaya. Bayesian uncertainty for gradient aggregation in multi-task learning. *arXiv preprint arXiv:2402.04005*, 2024. 2

[3] Hao Ban and Kaiyi Ji. Fair resource allocation in multi-task learning. *arXiv preprint arXiv:2402.15638*, 2024. 1, 3, 4, 6, 7, 8

[4] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000. 1

[5] Lorenz C Blum and Jean-Louis Reymond. 970 million drug-like small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25):8732–8733, 2009. 6, 7

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

[7] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, 56(12):1–32, 2024. 1

[8] 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. 2

[9] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020. 2, 6, 7

[10] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir A Rawashdeh. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 2

[11] 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. 6

[12] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017. 1

[13] Heshan Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. International Conference on Learning Representations, 2023. 6, 7

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 8

[15] 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. 2

[16] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*. 6, 1

[17] 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, 7, 8

[18] 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. 1, 2

[19] Baijiong Lin, Feiyang Ye, and Yu Zhang. A closer look at loss weighting in multi-task learning. 2021. 1, 6, 7, 8

[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. 2

[21] 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. 1, 2, 6, 7, 8

[22] Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 3, 6, 7, 8

[23] Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. iclr, 2021. 3, 6, 7, 8

[24] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017. 1

[25] 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. 1, 2, 6, 7, 8

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015. 6, 7

[27] Keerthiram Murugesan and Jaime Carbonell. Self-paced multitask learning with shared knowledge. *arXiv preprint arXiv:1703.00977*, 2017. 2

[28] Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *arXiv preprint arXiv:2007.02693*, 2020. 1

[29] 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. 1, 2, 3, 4, 6, 7, 8

[30] Jonathan Pilault, Amine Elhattami, and Christopher Pal. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *arXiv preprint arXiv:2009.09139*, 2020. 1

[31] S Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 1

[32] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 1, 2, 3, 6, 7, 8

[33] 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. 1, 2, 6, 7

[34] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pages 746–760. Springer, 2012. 6

[35] 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. 1

[36] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*, 2020. 1

[37] Caiming Xiong, SHU Tianmin, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning, 2023. US Patent 11,562,287. 1

[38] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020. 1, 2, 6

[39] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020. 6, 7, 8

[40] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34 (12):5586–5609, 2021. 1

[41] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 56(1):1–37, 2023. 1