RECON: REDUCING CONFLICTING GRADIENTS FROM THE ROOT FOR MULTI-TASK LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A fundamental challenge for multi-task learning is that different tasks may conflict with each other when they are solved jointly, and a cause of this phenomenon is *conflicting gradients* during optimization. Recent works attempt to mitigate the influence of conflicting gradients by directly altering the gradients based on some criteria. However, our empirical study shows that "gradient surgery" cannot reduce the occurrence of conflicting gradients. In this paper, we take a different approach to reduce conflicting gradients *from the root*. In essence, we investigate the task gradients w.r.t. each *shared* network layer, select the layers with high conflict scores, and set them *task-specific*. Our experiments show that with only a slight increase in model parameters, such a simple approach can effectively reduce the occurrence of conflicting gradients in the remaining shared layers and achieve better performance. We demonstrate the generality of our approach by combining it with state-of-the-art approaches including gradient manipulation methods and branched architecture search methods. Comprehensive experiments on various benchmarks show that our approach can substantially improve their performance.

1 INTRODUCTION

Multi-task learning (MTL) is a learning paradigm in which multiple different but correlated tasks are jointly trained with a shared model (Caruana, 1997; Ruder, 2017; Zhang & Yang, 2021), in the hope of achieving better performance with an overall smaller model size than learning each task independently. By discovering shared structures across tasks and leveraging domain-specific training signals of related tasks, MTL can achieve efficiency and effectiveness. Indeed, MTL has been successfully applied in many domains including natural language processing (Hashimoto et al., 2016), reinforcement learning (Parisotto et al., 2016; D'Eramo et al., 2020) and computer vision (Vandenhende et al., 2021).

A major challenge for multi-task learning is *negative transfer* (Ruder, 2017), which refers to the performance drop on a task caused by the learning of other tasks, resulting in worse overall performance than learning them separately. This is caused by *task conflicts*, i.e., tasks compete with each other and unrelated information of individual tasks may impede the learning of common structures. From the optimization point of view, a cause of negative transfer is *conflicting gradients* (Yu et al., 2020), which refers to two task gradients pointing away from each other and the update of one task will have a negative effect on the other. Conflicting gradients make it difficult to optimize the multi-task objective, since task gradients with larger magnitude may dominate the average update vector, making the optimizer prioritize some tasks over others and struggle to converge to a desirable solution.

Prior works address task/gradient conflicts mainly by balancing the tasks via task reweighting or gradient manipulation. Task reweighting methods adaptively re-weight the loss functions by homoscedastic uncertainty (Kendall et al., 2018), balancing the pace at which tasks are learned Chen et al. (2018); Liu et al. (2019), or learning a loss weight parameter (Liu et al., 2021b). Gradient manipulation methods reduce the influence of conflicting gradients by directly altering the gradients based on different criteria (Sener & Koltun, 2018; Yu et al., 2020; Chen et al., 2020; Liu et al., 2021a) or rotating the shared features (Javaloy & Valera, 2022). While these methods have demonstrated effectiveness in different scenarios, in our empirical study, we find that they cannot reduce the occurrence of conflicting gradients (see Sec. 3.3 for more discussion).

We propose a different approach to reduce conflicting gradients for MTL. Specifically, we investigate layer-wise conflicting gradients, i.e., the task gradients w.r.t. each shared network layer. We first train the network with a regular MTL algorithm (e.g., joint-training) for a number of iterations and compute the conflict scores for all shared layers and select those with highest conflict scores (indicating severe conflicts). We then set the selected shared layers task-specific and continue to train the modified network. As demonstrated by comprehensive experiments and analysis, our simple approach Recon has the following key advantages:

- Recon can effectively reduce the occurrence of conflicting gradients with only a slight increase in model parameters (less than 1% in some cases) and lead to better performance.
- Recon can be combined with gradient manipulation methods and branched architecture search methods to further improve their performance with various network backbones.
- Recon can achieve better performance than branched architecture search methods with a much smaller model size.

2 Related Works

In this section, we briefly review related works in multi-task learning in three categories: architecture design, architecture search, and task balancing. For a comprehensive review, please refer to Zhang & Yang (2021); Vandenhende et al. (2021). Architecture design methods include *hard parameter sharing* methods (Kokkinos, 2017; Long et al., 2017; Bragman et al., 2019), which learn a common feature extractor and task-specific decoders, and *soft parameters sharing* methods (Misra et al., 2016; Ruder et al., 2019; Gao et al., 2019), in which each task has a portion of parameters to do cross-task talk through some sharing mechanism. Differently, MTAN (Liu et al., 2019) adopts task-specific attention to extract shared information for each task.

Instead of designing a fixed network structure, some methods (Rosenbaum et al., 2017; Meyerson & Miikkulainen, 2017; Yang et al., 2020) propose to dynamically self-organize the network for different tasks. Among them, **branched architecture search** (Guo et al., 2020; Bruggemann et al., 2020) methods are more related to our work. They proposes an automated architecture search algorithm to build a tree-structured network by learning the branch position of the network. In contrast, our method Recon decides which layers to share across tasks by considering the severity of layer-wise conflicting gradients, which leads to a better and more compact architecture with lower time cost.

Another line of works is **task balancing** methods. To address task/gradient conflicts, some methods attempt to re-weight the multi-task loss function using homoscedastic uncertainty (Kendall et al., 2018), task prioritization (Guo et al., 2018), or similar learning pace (Liu et al., 2019; 2021b). GradNorm Chen et al. (2018) learns a weight parameter to ensure the similar learning paces across tasks. MGDA (Sener & Koltun, 2018) adopts Frank-Wolfe algorithm to find the weights such that the weighted sum of task gradients has a minimum norm. To reduce the influence of conflicting gradients, PCGrad (Yu et al., 2020) projects each gradient onto the normal plane of another gradient and uses the average projected gradient as the update vector. Graddrop (Chen et al., 2020) randomly drops the elements of gradients based on element-wise conflict. CAGrad (Liu et al., 2021a) ensures convergence to a minimum of the average loss by gradient manipulation. RotoGrad (Javaloy & Valera, 2022) reduces the influence of conflicting gradients by rotating the shared feature space. Instead of manipulating gradients, our method Recon leverages gradient information to modify network structure to mitigate task conflicts from the root.

3 Pilot Study: Task Conflicts in Multi-Task Learning

3.1 Multi-task Learning: Problem Definition

Multi-task learning (MTL) aims to learn a set of correlated tasks $\{\mathcal{T}_i\}_{i=1}^T$ simultaneously. For each task $\mathcal{T}_i$, the empirical loss function is $\mathcal{L}_i(\theta_{\text{sh}}, \theta_i)$, where θ_{sh} are parameters shared among all tasks and θ_i are task-specific parameters. The goal is to find optimal parameters $\theta = \{\theta_{\text{sh}}, \theta_1, \theta_2, \cdots, \theta_T\}$

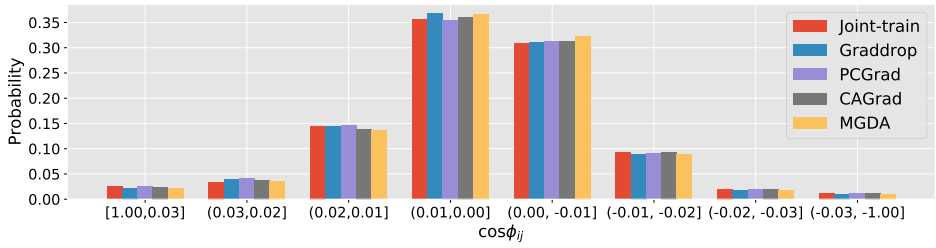


Figure 1: The distributions of gradient conflicts (in terms of $\cos\phi_{ij}$) of the joint-training baseline and state-of-the-art gradient manipulation methods on Multi-Fashion+MNIST benchmark.

to achieve high performance across all tasks. Formally, it aims to minimize a multi-task objective:

$$\theta^* = \arg\min_{\theta} \sum_{i}^{T} w_i \mathcal{L}_i(\theta_{\text{sh}}, \theta_i), \tag{1}$$

where w_i are pre-defined or dynamically computed weights for different tasks. A popular choice is to use the average loss (i.e., equal weights). However, optimizing the multi-task objective is difficult, and a known cause is conflicting gradients.

3.2 Conflicting Gradients

Let $\mathbf{g}_i = \nabla_{\theta_{\text{sh}}} \mathcal{L}_i(\theta_{\text{sh}}, \theta_i)$ denote the gradient of task $\mathcal{T}_i$ w.r.t. the shared parameters θ_{sh} (i.e., a vector of the partial derivatives of $\mathcal{L}_i$ w.r.t. θ_{sh}) and $g_i^{\text{ts}} = \nabla_{\theta_i} \mathcal{L}_i(\theta_{\text{sh}}, \theta_i)$ denote the gradient w.r.t. the task-specific parameters θ_i. A small change of θ_{sh} in the direction of negative $\mathbf{g}_i$ is $\theta_{\text{sh}} \leftarrow \theta_{\text{sh}} - \alpha \mathbf{g}_i$, with a sufficiently small step size α. The effect of this change on the performance of another task $\mathcal{T}_j$ is measured by:

$$\Delta\mathcal{L}_j = \mathcal{L}_j(\theta_{\text{sh}} - \alpha \mathbf{g}_i, \theta_j) - \mathcal{L}_j(\theta_{\text{sh}}, \theta_j) = -\alpha \mathbf{g}_i \cdot \mathbf{g}_j + o(\alpha), \tag{2}$$

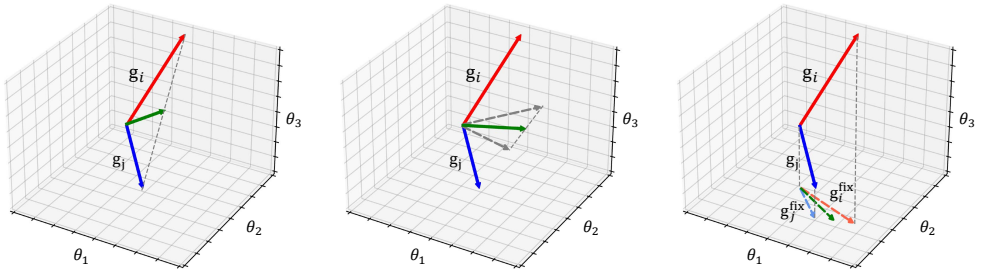
where the second equality is obtained by first order Taylor approximation. Likewise, the effect of a small update of θ_{sh} in the direction of the negative gradient of task $\mathcal{T}_j$ (i.e., $-\mathbf{g}_j$) on the performance of task $\mathcal{T}_i$ is $\Delta\mathcal{L}_i = -\alpha \mathbf{g}_i \cdot \mathbf{g}_j + o(\alpha)$. Notably, the model update for task $\mathcal{T}_i$ is considered to have a negative effect on task $\mathcal{T}_j$ when $\mathbf{g}_i \cdot \mathbf{g}_j < 0$, since it increases the loss of task $\mathcal{T}_j$, and vice versa. A formal definition of conflicting gradients is given as follows (Yu et al., 2020).

Definition 1 (Conflicting Gradients)**.** *The gradients $\mathbf{g}_i$ and $\mathbf{g}_j (i \neq j)$ are said to be conflicting with each other if $\cos\phi_{ij} < 0$, where ϕ_{ij} is the angel between $\mathbf{g}_i$ and $\mathbf{g}_j$.*

As shown in Yu et al. (2020), conflicts in gradient pose serious challenges for optimizing the multi-task objective (Eq. 1). Using the average gradient (i.e., $\frac{1}{T}\sum_{i=1}^{T} \mathbf{g}_i$) for gradient decent may hurt the performance of individual tasks, especially when there is a large difference in gradient magnitudes, which will make the optimizer struggle to converge to a desirable solution.

3.3 Gradient Surgery cannot reduce conflicting gradients

To mitigate the influence of conflicting gradients, several methods (Yu et al., 2020; Chen et al., 2020; Liu et al., 2021a) have been proposed to perform "gradient surgery". Instead of following the average gradient direction, they alter conflicting gradients based on some criteria and use the modified gradients for model update. However, we find that gradient manipulation cannot reduce the occurrence of conflicting gradients, i.e., they remain undiminished after gradient surgery. In our pilot experiment, we count and plot the distribution of conflicting gradients of the joint-training baseline (i.e., training all tasks jointly with equal loss weights and all parameters shared) and several state-of-the-art gradient manipulation methods including GradDrop (Chen et al., 2020), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), and MGDA (Sener & Koltun, 2018) on the Multi-Fashion+MNIST dataset (Lin et al., 2019). As shown in Figure 1, the distributions of gradient manipulation methods are very close to that of joint-training, suggesting that conflicting gradients are prevalent during optimization and gradient manipulation cannot effectively reduce their occurrence.



(a) Joint-train (b) PCGrad (c) Recon

Figure 2: Illustration of the differences between joint-training, gradient manipulation, and our approach. (a) In joint-training, the update vector (in green) is the average gradient $\frac{1}{2}(\mathbf{g}_i + \mathbf{g}_j)$. Due to the conflict between $\mathbf{g}_i$ and $\mathbf{g}_j$, the update vector is dominated by $\mathbf{g}_i$ (in red). (b) PCGrad (Yu et al., 2020) projects each gradient onto the normal plane of the other one and uses the average of the projected gradients (indicated by dashed grey arrows) as the update vector (in green). As such, the update vector is less dominated by $\mathbf{g}_i$. (c) Our approach Recon finds the parameters contributing most (e.g., θ_3) to gradient conflicts and turns them into task specific ones. In effect, it performs an orthographic/coordinate projection of conflicting gradients to the space of the rest parameters (e.g., θ_1 and θ_2) such that the projected gradients $\mathbf{g}_i^{\text{fix}}$ and $\mathbf{g}_j^{\text{fix}}$ are better aligned.

4 Our Approach: Reducing Conflicting Gradients from the Root

Our pilot study shows that adjusting gradients for model update cannot prevent the occurrence of conflicting gradients in MTL, which suggests that the root causes of this phenomenon may be closely related to the nature of different tasks and the way how model parameters are shared among them. Therefore, to mitigate task conflicts for MTL, in this paper, we take a different approach to reduce the occurrence of conflicting gradients from the root.

4.1 Recon: Removing Layer-wise Conflicting Gradients

Our approach is extremely simple and intuitive. We first identify the shared network layers where conflicts occur most frequently and then turn them into task-specific parameters. Suppose the shared model parameters θ_{sh} are composed of n layers, i.e., $\theta_{\text{sh}} = \{\theta_{\text{sh}}^{(k)}\}_{k=1}^{n}$, where $\theta_{\text{sh}}^{(k)}$ is the kth shared layer. Let $\mathbf{g}_i^{(k)}$ denote the gradient of task $\mathcal{T}_i$ w.r.t. the kth shared layer $\theta_{\text{sh}}^{(k)}$, i.e., $\mathbf{g}_i^{(k)}$ is a vector of the partial derivatives of $\mathcal{L}_i$ w.r.t. the parameters of $\theta_{\text{sh}}^{(k)}$. Let $\phi_{ij}^{(k)}$ denote the angle between $\mathbf{g}_i^{(k)}$ and $\mathbf{g}_j^{(k)}$. We define layer-wise conflicting gradients and S-conflict score as follows.

Definition 2 (Layer-wise Conflicting Gradients)**.** *The gradients $\mathbf{g}_i^{(k)}$ and $\mathbf{g}_j^{(k)}$ $(i \neq j)$ are said to be conflicting with each other if* $\cos\phi_{ij}^{(k)} < 0$.

Definition 3 (S-Conflict Score)**.** *For any $-1 < S \leq 0$, the S-conflict score for the kth shared layer is the number of different pairs $(i, j)(i \neq j)$ s.t. $\cos\phi_{ij}^{(k)} < S$, denoted as $s^{(k)}$.*

S indicates the severity of conflicts, and setting S smaller means we care about cases of more severe conflicts. The S-conflict score $s^{(k)}$ indicates the occurrence of conflicting gradients at severity level S for the kth shared layer. If $s^{(k)} = \binom{T}{2}$, it means that for any two different tasks, there is a conflict in their gradients w.r.t. the kth shared layer. By computing S-conflict scores, we can identify the shared layers where conflicts occur most frequently.

We describe our method Recon in Algorithm 1. First, we train the network for I iterations and compute S-conflict scores for each shared layer $\theta^{(k)}$ in every iteration, denoted by $\{s_i^{(k)}\}_{i=1}^{I}$. Then, we sum up the scores in all iterations, i.e., $s^{(k)} = \sum_{i=1}^{I} s_i^{(k)}$, and find the layers with highest $s^{(k)}$ scores. Next, we set these layers to be task-specific and continue to train the modified network. We demonstrate the effectiveness of Recon by a theoretical analysis in Sec. 4.2 and comprehensive ex-

Algorithm 1: Recon: Removing Layer-wise Conflicting Gradients

Input: Model parameters θ, learning rate α, a set of tasks $\{\mathcal{T}_i\}_{i=1}^T$, number of iterations I for computing conflict scores, conflict severity level S, number of selected layers K.

```
// Train the network and compute conflict scores for all layers
```

for iteration $i = 1, 2, \ldots, I$ **do**
  for $i = 1, 2, \ldots, T$ **do**
    Compute the gradients of task $\mathcal{T}_i$ w.r.t. all shared layers, i.e., $\{\mathbf{g}_i^{(k)}\}_{k=1}^n$;
  end
  Calculate the S-conflict scores for all shared layers in the current iteration, i.e., $\{s_i^{(k)}\}_{k=1}^n$;
  Update θ with joint-training or any gradient manipulation method ;
end

```
// Set layers with top conflict scores task-specific
```

For each layer k, calculate the sum of S-conflict scores in all iterations, i.e., $s^{(k)} = \sum_{i=1}^I s_i^{(k)}$;
Select the top K layers with highest $s^{(k)}$ and set them task-specific;

```
// Continue to train the modified network
```

for iteration $i = 1, 2, \ldots$ **do**
  Update θ with joint-training or any gradient manipulation method;
end

Output: Model parameters θ.

periments in Sec. 5. The results show that Recon can effectively reduce the occurrence of conflicting gradients in the remaining shared layers and lead to substantial improvements over state-of-the-art.

4.2 Theoretical Analysis

Here, we provide a theoretical analysis of Recon. Let $\theta_{\text{sh}} = \{\theta_{\text{sh}}^{\text{fix}}, \theta_{\text{sh}}^{\text{cf}}\}$, where $\theta_{\text{sh}}^{\text{fix}}$ are the remaining shared parameters, and $\theta_{\text{sh}}^{\text{cf}}$ are those that will be turned to task-specific parameters $\theta_1^{\text{cf}}, \theta_2^{\text{cf}}, \cdots, \theta_T^{\text{cf}}$. Notice that $\theta_1^{\text{cf}}, \theta_2^{\text{cf}}, \cdots, \theta_T^{\text{cf}}$ will all be initialized with $\theta_{\text{sh}}^{\text{cf}}$. Therefore, after applying Recon, the model parameters are $\theta_r = \{\theta_{\text{sh}}^{\text{fix}}, \theta_1^{\text{cf}}, \ldots, \theta_T^{\text{cf}}, \theta_1^{\text{ts}}, \ldots, \theta_T^{\text{ts}}\}$. An one-step gradient update of θ_r is:

$$\hat{\theta}_{\text{sh}}^{\text{fix}} = \theta_{\text{sh}}^{\text{fix}} - \alpha \sum_{i=1}^T w_i \mathbf{g}_i^{\text{fix}}, \quad \hat{\theta}_i^{\text{cf}} = \theta_i^{\text{cf}} - \alpha \mathbf{g}_i^{\text{cf}}, \quad \hat{\theta}_i^{\text{ts}} = \theta_i^{\text{ts}} - \alpha \mathbf{g}_i^{\text{ts}}, \quad i = 1, \ldots, T, \tag{3}$$

where w_i are weight parameters, $\mathbf{g}_i^{\text{ts}} = \nabla_{\theta_i^{\text{ts}}} \mathcal{L}_i$, $\mathbf{g}_i^{\text{cf}} = \nabla_{\theta_{\text{sh}}^{\text{cf}}} \mathcal{L}_i$ and $\mathbf{g}_i^{\text{fix}} = \nabla_{\theta_{\text{sh}}^{\text{fix}}} \mathcal{L}_i$. Notice that different methods such as joint-training, MGDA Sener & Koltun (2018), PCGrad Yu et al. (2020), and CAGrad Liu et al. (2021a) choose different w_i dynamically.

Without applying Recon, the model parameters are $\theta = \{\theta_{\text{sh}}^{\text{fix}}, \theta_{\text{sh}}^{\text{cf}}, \theta_1^{\text{ts}}, \ldots, \theta_T^{\text{ts}}\}$. An one-step gradient update of θ is given by

$$\hat{\theta}_{\text{sh}}^{\text{fix}} = \theta_{\text{sh}}^{\text{fix}} - \alpha \sum_{i=1}^T w_i \mathbf{g}_i^{\text{fix}}, \quad \hat{\theta}_{\text{sh}}^{\text{cf}} = \theta_{\text{sh}}^{\text{cf}} - \alpha \sum_{i=1}^T w_i \mathbf{g}_i^{\text{cf}}, \quad \hat{\theta}_i^{\text{ts}} = \theta_i^{\text{ts}} - \alpha \mathbf{g}_i^{\text{ts}}, \quad i = 1, \ldots, T. \tag{4}$$

After the one-step updates, the loss functions with the updated parameters $\hat{\theta}_r$ and $\hat{\theta}$ respectively are:

$$\mathcal{L}(\hat{\theta}_r) = \sum_{i=1}^T \mathcal{L}_i \left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_i^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right), \text{ and, } \mathcal{L}(\hat{\theta}) = \sum_{i=1}^T \mathcal{L}_i \left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_{\text{sh}}^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right), \tag{5}$$

where $\mathcal{L}_i$ is the loss function of task $\mathcal{T}_i$. Denote the set of indices of the layers turned task-specific by $\mathbb{P}$, then $\theta_{\text{sh}}^{\text{cf}} = \{\theta_{\text{sh}}^{(k)}\}, k \in \mathbb{P}$. Assume that for any two different tasks $\mathcal{T}_i$ and $\mathcal{T}_j$, and $\forall k \in \mathbb{P}$, we have $\cos \phi_{ij}^{(k)} < A$, where $-1 < A \leq 1$. We have the following theorem.

Theorem 4.1. *Assume that $\mathcal{L}$ is differentiable and A satisfies that for all task pairs,*

$$A \|\mathbf{g}_i^{cf}\| \leq \|\mathbf{g}_j^{cf}\|, \tag{6}$$

Table 1: Multi-task learning results on Multi-Fashion+MNIST dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Single-task	RotoGrad	BMTAS	Joint-train	w/ Recon	MGDA	w/ Recon	PCGrad	w/ Recon	GradDrop	w/ Recon	CAGrad	w/ Recon
T1 Acc↑	98.37	98.10	98.20	97.42	98.13	95.19	**98.33**	97.37	98.30	97.38	98.25	97.47	98.28
T2 Acc↑	89.63	88.25	89.71	88.82	89.26	89.46	89.28	88.68	**89.77**	88.57	89.51	88.85	89.65
$\Delta m\%\uparrow$	-	-0.91	-0.04	-0.94	-0.33	-1.71	-0.22	-1.04	**0.04**	-1.10	-0.13	-0.90	-0.04
#P.	85.62	42.89	85.61	42.81	43.43	42.81	43.43	42.81	43.43	42.81	43.43	42.81	43.43

then for any sufficiently small learning rate $\alpha > 0$,

$$\mathcal{L}(\hat{\theta}_r) < \mathcal{L}(\hat{\theta}). \tag{7}$$

The smaller A is, the larger the gap between $\mathcal{L}(\hat{\theta}_r)$ and $\mathcal{L}(\hat{\theta})$.

The theorem indicates that a single gradient update on the model parameters of Recon achieves lower loss than that on the original model parameters. The proof is provided in Appendix A

5 Experiments

In this section, we conduct extensive experiments to evaluate our approach Recon for multi-task learning and demonstrate its effectiveness, efficiency and generality.

5.1 Experimental Setup

Datasets. We evaluate Recon on 4 multi-task datasets, namely **Multi-Fashion+MNIST** (Lin et al., 2019), **CityScapes** (Cordts et al., 2016), **NYUv2** (Couprie et al., 2013), and **PASCAL-Context** (Mottaghi et al., 2014). The tasks of each dataset are described as follows. 1) Multi-Fashion+MNIST contains two image classification tasks. Each image consists of an item from FashionMNIST and an item from MNIST. 2) CityScapes contains 2 vision tasks: 7-class semantic segmentation and depth estimation. 3) NYUv2 contains 3 tasks: 13-class semantic segmentation, depth estimation and normal prediction. 4) PASCAL-Context consists of 5 tasks: semantic segmentation, human parts segmentation and saliency estimation, surface normal estimation, and edge detection.

Baselines. The baselines include 1) single-task learning (single-task): training all tasks independently; 2) joint-training (joint-train): training all tasks together with equal loss weights and all parameters shared; (3) gradient manipulation methods: MGDA (Sener & Koltun, 2018), PCGrad (Yu et al., 2020), GradDrop (Chen et al., 2020), CAGrad (Liu et al., 2021a), RotoGrad (Javaloy & Valera, 2022) (4) branched architecture search methods: BMTAS (Bruggemann et al., 2020). For a fair comparison, all methods use same configurations and random seeds. We run all experiments 3 times with different random seeds. More experimental details are provided in Appendix B.

Relative task improvement. Following Maninis et al. (2019), we compute the relative task improvement with respect to the single-task baseline for each task. Given a task $\mathcal{T}_j$, the relative task improvement is $\Delta m_{\mathcal{T}_j} = \frac{1}{K}\sum_{i=1}^{K}(-1)^{l_i}(M_i - S_i)/S_i$, where M_i, S_i refer to metrics for the ith criterion obtained by objective model and single-task model respectively, $l_i = 1$ if a lower value for the criterion is better and 0 otherwise. The average relative task improvement is $\Delta m = \frac{1}{T}\sum_{j=1}^{T}\Delta m_{\mathcal{T}_j}$.

5.2 Comparison with the State-of-the-Art

Recon improves the performance of all base models. The main results on Multi-Fashion+MNIST, CityScapes, PASCAL-Context and NYUv2 are presented in Table 1, Table 2, Table 3 and Table 4 respectively. **(1)** Compared to gradient manipulation methods, Recon consistently improves their performance in most evaluation metrics, and achieve comparable performance on the rest of evaluation metrics. **(2)** Compared with branched architecture search methods, Recon can further improve the performance of BMTAS. Besides, Recon combined with other gradient manipulation methods

Table 2: Multi-task learning results on CityScapes dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Segmentation (Higher Better)		Depth (Lower Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err		
Single-task	74.36	93.22	0.0128	29.9803		190.59
RotoGrad	73.38	92.97	0.0147	82.31	-47.81	103.43
Joint-train	74.13	93.13	0.0166	116.00	-79.32	95.43
w/ Recon	74.17	**93.21**	0.0136	43.18	-12.63	108.44
MGDA	70.74	92.19	0.0130	47.09	-16.22	95.43
w/ Recon	71.01	92.17	**0.0129**	**33.41**	**-4.46**	108.44
Graddrop	74.08	93.08	0.0173	115.79	-80.48	95.43
w/ Recon	74.17	93.11	0.0134	41.37	-10.69	108.44
PCGrad	73.98	93.08	0.0200	114.50	-78.39	95.43
w/ Recon	74.18	93.14	0.0136	46.02	-14.92	108.44
CAGrad	73.81	93.02	0.0153	88.29	-53.81	95.43
w/ Recon	**74.22**	93.10	0.0130	38.27	-7.38	108.44

with small model size can achieve better results than branched architecture search methods with much bigger models.

Small increases in model parameters can lead to good performance gains. Note that Recon only changes a small part of shared parameters to task-specific parameters. As presented in Table 1 to Table 4, Recon increases the model size by nearly 0.52% to 26.87%. We select nearly 1.42%, 12.77%, 0.26%, 9.80% shared parameters to be turned into task-specific parameters on Multi-Fashion+MNIST, CityScapes, NYUv2 and PASCAL-Context respectively. The results suggest that the conflicts of only a small part of shared parameters (less than 13% with high conflict scores) impede the training of the model in multi-task learning.

Searching for layers with severe conflicts is fast. In general, it only requires extra 25% of training iterations to compute conflicting scores. It is worth mentioning that the layers found by Recon with different MTL algorithms (e.g., CAGrad and PCGrad) are surprisingly consistent (see more discussion in Appendix C). Hence, in our experiments, we only need to perform one search and apply the model modified by Recon to different baselines.

Recon is compatible with various neural network architectures. In Multi-Fashion+MNIST benchmark, we build a ResNet18 as the shared backbone and two linear layers as the task-specific decoders. In the CityScapes dataset, we apply SegNet (Badrinarayanan et al., 2017). In the NYUv2 dataset, the network is MTAN (Liu et al., 2019), a MTL architecture that is combined with task-specific attention in the encoder of SegNet. In PASCAL-Context dataset, following Bruggemann et al. (2020), we use MobileNetV2 (Sandler et al., 2018) as backbone. Recon improves the performance of baselines on all these neural network architectures. Note that BMTAS finds a tree-like structure for multi-task learning, and Recon can still improve the performance of BMTAS on the tree-like structure. We also provide the experimental results on CityScapes using MTAN architecture in Appendix C. In conclusion, the results indicate that Recon can be applied with various neural network architectures for reducing conflicting gradients.

5.3 Ablation Study and Analysis

Recon significantly reduces the occurrence of conflicting gradients. In Fig 5, we compare the distribution of $\cos\phi_{ij}$ before and after applying Recon on the CityScapes dataset (the results on Multi-Fashion+MNIST dataset are provided in Appendix C). It can be seen that with Recon, the numbers of gradient pairs with no conflicts or small conflicts (i.e., $\cos\phi_{ij} \in [1, 0]$, $\cos\phi_{ij} \in (0, -0.02]$) are greatly increased while those with severe gradient conflicts (i.e, $\cos\phi_{ij} \in (-0.02, -1]$) are greatly reduced. Further, in the ablation study on S, through empirical experiments, we find that removing gradients with small conflicts (i.e, $\cos\phi_{ij} \in (0, -0.02]$) leads to performance drop.

Table 3: Multi-task learning results on PASCAL-Context dataset with 4-task setting. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates Recon improves the result of the base model. The best average result is marked in bold.

Method	SemSeg (Higher Better)		PartSeg (Lower Better)		saliency (Higher Better)	Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	mIoU	Pix Acc	mIoU	Mean	Median	11.25	22.5		
Single-task	65.00	90.53	59.59	92.61	65.61	14.55	12.36	46.51	81.29		30.09
Joint-train	64.06	90.45	57.91	92.17	62.71	16.40	14.23	39.38	75.93	-4.82	8.04
w/ Recon	64.73	**90.50**	59.00	92.44	66.17	14.99	12.68	44.82	80.11	-0.66	10.20
MGDA	46.05	86.62	54.82	91.39	64.76	15.77	13.54	41.98	77.82	-7.67	8.04
w/ Recon	55.82	87.73	56.31	91.67	64.91	15.12	12.88	44.36	79.81	-4.14	10.20
PCGrad	63.91	90.45	58.01	92.19	63.09	16.34	14.19	39.62	76.06	-4.59	8.04
w/ Recon	**65.02**	90.45	59.22	92.46	66.14	14.95	**12.73**	44.96	80.22	-0.55	10.20
Graddrop	64.14	90.34	57.62	92.12	62.64	16.46	14.28	39.29	75.71	-5.00	8.04
w/ Recon	64.48	90.45	59.08	92.46	**66.23**	14.94	12.72	45.03	80.25	-0.63	10.20
CAGrad	63.37	90.17	57.49	92.07	64.16	16.30	14.12	39.80	76.23	-4.37	8.04
w/ Recon	64.60	90.40	59.27	92.47	65.67	14.92	12.71	45.10	80.33	-0.76	10.20
BMTAS	64.89	90.44	58.87	92.36	63.42	15.66	13.44	42.29	78.14	-2.89	15.18
w/ Recon	64.78	90.46	**59.96**	**92.58**	65.96	**14.74**	**12.57**	**45.62**	**80.84**	**-0.19**	16.83

Table 4: Multi-task learning results on NYUv2 dataset. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Segmentation (Higher Better)		Depth (Lower Better)		Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)			$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30		
Single-task	38.67	64.27	0.6881	0.2788	24.8683	18.9919	30.43	57.81	69.7		
Joint-train	39.48	65.23	0.5491	0.2235	27.87	23.76	22.68	47.91	61.58	0.75	168.72
w/ Recon	39.54	65.20	0.5312	0.2234	26.55	21.40	26.53	52.60	65.31	4.14	169.59
MGDA	29.28	60.30	0.6027	0.2515	**24.89**	**19.32**	**29.85**	**57.18**	**69.38**	-2.26	168.72
w/ Recon	32.82	61.26	0.5884	0.2295	25.17	19.72	28.18	56.49	68.96	0.53	169.59
Graddrop	38.70	64.97	0.5565	0.2333	27.41	23.00	23.79	49.45	62.87	0.49	168.72
w/ Recon	**40.14**	66.08	0.5265	0.2241	26.51	21.45	26.51	52.48	65.26	4.67	169.59
PCGrad	38.55	65.07	0.54	0.23	26.90	22.05	24.98	51.36	64.41	2.02	168.72
w/ Recon	38.61	65.48	0.5350	0.2271	26.31	21.11	26.90	53.21	65.95	3.87	169.59
CAGrad	39.89	**66.47**	0.5496	0.2281	26.36	21.47	25.50	52.68	65.90	3.74	168.72
w/ Recon	39.92	66.07	**0.5320**	**0.2200**	25.80	20.59	27.60	54.31	67.05	**5.80**	169.59

Randomly selecting conflict layers does not work. To show that the performance gain of Recon comes from selecting the layers with most severe conflicts instead of merely increasing model parameters, we further compare Recon with the following two baselines. RSL: randomly selecting same number of layers as Recon and set them task-specific. CMS: randomly selecting similar amount of parameters as Recon and set them task-specific. The results in Table 5 show that both RSL and CMS lead to significant performance drops, which verifies the effectiveness of the selection strategy of Recon. To further support the conclusion, we also compare Recon with the baselines that selects the first or last K layers in Appendix C .

Ablation study on hyperparameters. We study the influence of the conflict severity S and the number of selected layers K on the performance of CAGrad combined with Recon on the Multi-Fashion+MNIST benchmark, as shown in Fig. 4. We report the relative task improvement Δm w.r.t. varying S and K. It can be seen that on Multi-Fashion+MNIST, the best results are achieved for $K = 25$ and $S = -0.10$. A smaller K (e.g., 5) leads to a significant performance drop, which indicates that there are still some shared network layers suffering from severe gradient conflicts. A larger K (e.g., $K \in [30, 35]$) will not lead to further performance improvement since severe conflicts

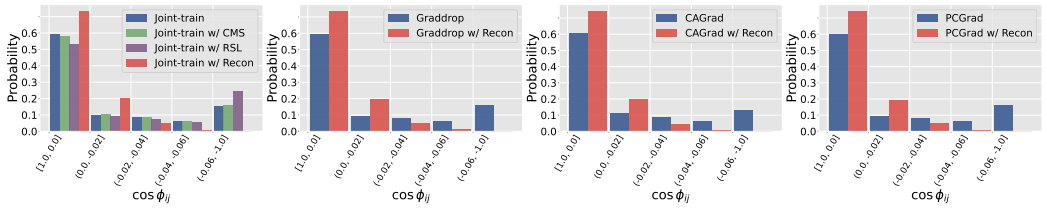
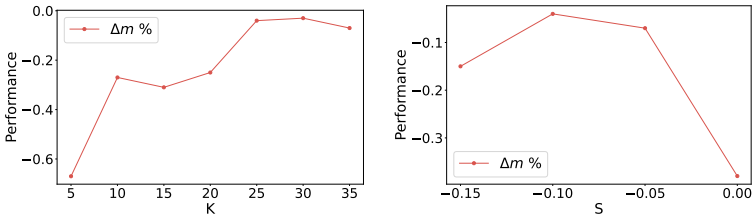


Figure 3: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) w.r.t. the shared parameters on CityScapes. RSL: randomly selecting same number of layers as Recon and set them task-specific. CMS: randomly selecting similar amount of parameters as Recon and set them task-specific.

Table 5: Comparison of Recon with RSL and CMS. PD: performance drop compared to Recon.

Seed	w/ RSL	w/ CMS	w/ Recon	CAGrad					PCGrad				
				Task 1		Task2		#P.	Task 1		Task2		#P.
				Acc↑	PD	Acc↑	PD		Acc↑	PD	Acc↑	PD	
0	✓			97.60	**0.68**	64.39	**25.26**	73.02	97.43	**0.87**	65.57	**24.21**	73.02
1	✓			97.11	**1.18**	87.61	**2.04**	83.63	94.92	**3.39**	87.31	**2.46**	83.63
2	✓			94.62	**3.66**	87.68	**1.96**	76.33	92.90	**5.40**	87.41	**2.36**	76.33
0		✓		97.11	**1.18**	85.57	**4.07**	52.25	96.93	**1.38**	88.16	**1.62**	52.25
1		✓		97.81	**0.47**	88.28	**1.36**	51.96	97.63	**0.68**	88.55	**1.22**	51.96
2		✓		81.18	**17.10**	76.56	**13.09**	47.50	88.71	**9.59**	84.51	**5.27**	47.50
-	-	-	✓	98.28	0	89.65	0	43.42	98.30	0	89.77	0	43.42



(a) (b)

Figure 4: The performance of CAGrad combined with Recon on the Multi-Fashion+MNIST benchmark with (a) different number of selected layers K (b) different severity value S for computing conflict scores.

have been resolved. For the conflict severity S, we find that a high value of S (e.g., 0.0) leads to performance drops since it includes too many gradient pairs with small conflicts, while some of them are helpful for learning common structures and should not be removed. In the meantime, a too small S (e.g., -0.15) also leads to performance degradation because it ignores too many gradient pairs with large conflicts, which may be detrimental to learning.

6 Conclusion

We have proposed a very simple yet effective approach, namely Recon, to reduce the occurrence of conflicting gradients for multi-task learning. By considering layer-wise gradient conflicts and identifying the shared layers with severe conflicts and setting them task-specific, Recon can significantly reduce the occurrence of severe conflicting gradients and boost the performance of existing gradient manipulation methods and branched architecture search methods with only a slight increase in model parameters. We have demonstrated the effectiveness, efficiency, and generality of Recon via extensive experiments and analysis.

REFERENCES

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.

Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C Alexander, and Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1385–1394, 2019.

David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search for resource-efficient branched multi-task networks. *arXiv preprint arXiv:2008.10292*, 2020.

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

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*, pp. 794–803. PMLR, 2018.

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

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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, Jan Peters, et al. Sharing knowledge in multi-task deep reinforcement learning. In *8th International Conference on Learning Representations*, pp. 1–11. OpenReview. net, 2020.

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*, pp. 3205–3214, 2019.

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)*, pp. 270–287, 2018.

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

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. In *ICLR*, 2022.

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*, pp. 7482–7491, 2018.

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*, pp. 6129–6138, 2017.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

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, 2021a.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. ICLR, 2021b.

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*, pp. 1871–1880, 2019.

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.

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*, pp. 1851–1860, 2019.

Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017.

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*, pp. 3994–4003, 2016.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR (Poster)*, 2016.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

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

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4822–4829, 2019.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

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

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.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

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.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

A PROOF OF THEOREM A.1

Theorem A.1. *Assume that $\mathcal{L}$ is differentiable and A satisfies that for all task pairs,*

$$A\|\mathbf{g}_i^{cf}\| \leq \|\mathbf{g}_j^{cf}\|, \tag{8}$$

then for any sufficiently small learning rate $\alpha > 0$,

$$\mathcal{L}(\hat{\theta}_r) < \mathcal{L}(\hat{\theta}). \tag{9}$$

The smaller S is, the larger the gap between $\mathcal{L}(\hat{\theta}_r)$ and $\mathcal{L}(\hat{\theta})$.

Proof. We consider the first order Taylor approximation of $\mathcal{L}_i$. For normal update, we have

$$\begin{aligned}\mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_{\text{sh}}^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) =& \mathcal{L}_i\left(\theta_{\text{sh}}^{\text{fix}}, \theta_{\text{sh}}^{\text{cf}}, \theta_i^{\text{ts}}\right) + (\hat{\theta}_{\text{sh}}^{\text{fix}} - \theta_{\text{sh}}^{\text{fix}})^\top \mathbf{g}_i^{\text{fix}} && (10)\\ &+ (\hat{\theta}_{\text{sh}}^{\text{cf}} - \theta_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + (\hat{\theta}_i^{\text{ts}} - \theta_i^{\text{ts}})^\top \mathbf{g}_i^{\text{ts}} + o(\alpha). && (11)\end{aligned}$$

For Recon update, we have

$$\begin{aligned}\mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_i^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) =& \mathcal{L}_i\left(\theta_{\text{sh}}^{\text{fix}}, \theta_{\text{sh}}^{\text{cf}}, \theta_i^{\text{ts}}\right) + (\hat{\theta}_{\text{sh}}^{\text{fix}} - \theta_{\text{sh}}^{\text{fix}})^\top \mathbf{g}_i^{\text{ts}} && (12)\\ &+ (\hat{\theta}_i^{\text{cf}} - \theta_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + (\hat{\theta}_i^{\text{ts}} - \theta_i^{\text{ts}})^\top \mathbf{g}_i^{\text{ts}} + o(\alpha). && (13)\end{aligned}$$

The difference between the two loss functions after the update is

$$\begin{aligned}\mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_i^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) - \mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_{\text{sh}}^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) =& (\hat{\theta}_i^{\text{cf}} - \hat{\theta}_{\text{sh}}^{\text{cf}})^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) && (14)\\ =& -\alpha\left(\mathbf{g}_i^{\text{cf}} - \sum_{j=1}^{T} w_j \mathbf{g}_j^{\text{cf}}\right)^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) && (15)\\ =& -\alpha \sum_{j=1}^{T} w_j \left(\mathbf{g}_i^{\text{cf}} - \mathbf{g}_j^{\text{cf}}\right)^\top \mathbf{g}_i^{\text{cf}} + o(\alpha) && (16)\\ =& -\alpha \sum_{j=1}^{T} w_j \left(\|\mathbf{g}_i^{\text{cf}}\|^2 - \mathbf{g}_j^{\text{cf}\top} \mathbf{g}_i^{\text{cf}}\right) + o(\alpha), && (17)\end{aligned}$$

where

$$\begin{aligned}\left\|\mathbf{g}_i^{\text{cf}}\right\|^2 - \mathbf{g}_j^{\text{cf}\top}\mathbf{g}_i^{\text{cf}} &= \sum_{k\in\mathbb{P}}\left(\left\|\mathbf{g}_i^{(k)}\right\|^2 - \mathbf{g}_i^{(k)\top}\mathbf{g}_j^{(k)}\right) && (18)\\ &= \sum_{k\in\mathbb{P}}\left\|\mathbf{g}_i^{(k)}\right\|\left(\left\|\mathbf{g}_i^{(k)}\right\| - \cos\phi_{ij}^{(k)}\left\|\mathbf{g}_j^{(k)}\right\|\right) && (19)\\ &> \sum_{k\in\mathbb{P}}\left\|\mathbf{g}_i^{(k)}\right\|\left(\left\|\mathbf{g}_i^{(k)}\right\| - A\left\|\mathbf{g}_j^{(k)}\right\|\right) && (20)\\ &= \left\|\mathbf{g}_i^{\text{cf}}\right\|\left(\left\|\mathbf{g}_i^{\text{cf}}\right\| - A\left\|\mathbf{g}_j^{\text{cf}}\right\|\right) && (21)\\ &\geq 0. && (22)\end{aligned}$$

Hence, the above difference is negative, if α is sufficiently small. As such, the difference between the multi-task loss functions is also negative, if α is sufficiently small.

$$\mathcal{L}(\hat{\theta}_r) - \mathcal{L}(\hat{\theta}) = \sum_{i=1}^{T}\mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_i^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) - \sum_{i=1}^{T}\mathcal{L}_i\left(\hat{\theta}_{\text{sh}}^{\text{fix}}, \hat{\theta}_{\text{sh}}^{\text{cf}}, \hat{\theta}_i^{\text{ts}}\right) < 0 \tag{23}$$

Eq. (21) indicates that the smaller S is, the larger the gap between $\mathcal{L}(\hat{\theta}_r)$ and $\mathcal{L}(\hat{\theta})$. □

B EXPERIMENTAL SETUP

B.1 MULTI-FASHION+MNIST.

Model. We adopt ResNet18 without pre-training as the backbone and modify the dimension of the output features to 100 for the last linear layer. For the task-specific heads, we define two linear layers followed by a ReLU function.

Tasks, losses, and metrics. Each task is a classification problem with 10 classes and we use the cross-entropy loss as the classification loss. For evaluation, we use the classification accuracy as the metric for each task.

Model hyperparameters. We train the model for 120 epochs with the batch size of 256. We adopt SGD with an initial learning rate of 0.1 and a weight decay of 0.1 at the 60th and 90th epoch.

Baseline hyperparameters. For CAGrad, we set $\alpha = 0.2$. For BMTAS, we set the resource loss weight to 1.0, and we search the architecture for 100 epochs. For RotoGrad, we set $R_k = 100$ which is equal to the dimension of shared features and set the learning rate of rotation parameters as learning rate of the neural networks.

Recon hyperparameters. We compute the conflict scores of each shared layers for 30 epochs with CAGrad as the update algorithm, and we set $S = -0.1$ for computing the scores. We select 25 layers with the highest conflict scores and turn them into task-specific.

B.2 CITYSCAPES

Model. We adopt SegNet (Badrinarayanan et al., 2017) as the backbone where the decoder is split into two convolutional heads.

Model hyperparameters. We train the model for 200 epochs with the batch size of 8. We adopt Adam with an initial learning rate of $5e-5$ and a weight decay of 0.5 at the 100th epoch.

Baselines hyperparameters. For CAGrad, we set $\alpha = 0.2$. For RotoGrad, we set $R_k = 1024$ and set the learning rate of rotation parameters as 10 times less than the learning rate of the neural networks.

Recon hyperparameters. We compute the conflict scores of each shared layers for 40 epochs with joint-train as the update algorithm, and we set $S = 0.0$ for computing the scores. We select 39 layers with the highest conflict scores and turn them into task-specific.

B.3 NYUV2

Model. We adopt MTAN (Liu et al., 2019), the SegNet combined with task-specific attention modules on the encoder.

Model hyperparameters. We train the model for 200 epochs with the batch size of 2. We adopt Adam with an initial learning rate of $1e-4$ and a weight decay of 0.5 at the 100th epoch.

Baseline hyperparameters. For CAGrad, we set $\alpha = 0.4$ similar with Liu et al. (2021a).

Recon hyperparameters. We compute the conflict scores of each shared layers for 40 epochs with joint-train as the update algorithm, and we set $S = -0.02$ for computing the scores. We select 15 layers with the highest conflict scores and turn them into task-specific.

B.4 PASCAL-CONTEXT

Model. Following Bruggemann et al. (2020), we employ MobileNetv2 Sandler et al. (2018) as the backbone with a reduced design of the ASPP module (R-ASPP) (Sandler et al., 2018). We pre-train the model on ImageNet (Deng et al., 2009).

Model hyperparameters. We train the model for 130 epochs with the batch size of 6. We adopt Adam with an initial learning rate of $1e-4$ and a weight decay of 0.1 at the 70th and 100th epoch.

Table 6: Multi-task learning results on CityScapes dataset with MTAN as the backbone network. All experiments are repeated over **3** random seeds and the mean values are reported. $\Delta m\%$ denotes the average relative improvement of all tasks. #P denotes the model size (MB). The grey cell color indicates that Recon improves the result of the base model. The best average result is marked in bold.

Method	Segmentation (Higher Better)		Depth (Lower Better)		$\Delta m\% \uparrow$	#P.
	mIoU	Pix Acc	Abs Err	Rel Err		
Single-task	73.74	93.05	0.0129	27.7121		190.58
Joint-train	75.35	93.55	0.0169	45.6377	-23.26	168.72
w/ Recon	75.72	**93.74**	0.0130	40.8970	-11.36	196.32
MGDA	70.46	91.75	0.0224	34.3330	-26.02	168.72
w/ Recon	72.23	92.60	**0.0122**	**26.9335**	**1.37**	196.32
Graddrop	75.19	93.53	0.0168	46.3521	-23.90	168.72
w/ Recon	75.60	93.72	0.0127	38.5504	-8.71	196.32
PCGrad	75.64	93.54	0.02	43.53	-23.60	168.72
w/ Recon	**75.89**	93.71	0.0129	40.0490	-10.35	196.32
CAGrad	75.26	93.50	0.0176	44.2298	-23.40	168.72
w/ Recon	75.65	93.71	0.0125	36.2316	-6.15	196.32

Table 7: Multi-task learning results on Multi-Fashion+MNIST dataset. LSK refers to turning the fist K layers into task-specific layers. FSK refers to turning the last K layers into task-specific layers. PD denotes the performance drop compared with Recon.

LSK	FSK	w/ Recon	CAGrad					PCGrad				
			Task 1		Task2		#P.	Task 1		Task2		#P.
			Acc↑	PD	Acc↑	PD		Acc↑	PD	Acc↑	PD	
✓			97.63	**0.66**	89.14	**0.50**	84.17	97.63	**0.65**	88.98	**0.66**	84.17
	✓		98.21	**0.07**	89.15	**0.50**	48.90	98.19	**0.09**	89.51	**0.13**	48.90
	-	✓	98.28	0	89.65	0	43.42	98.30	0	89.77	0	43.42

Baselines hyperparameters. For CAGrad, we set $\alpha = 0.1$. For BMTAS, we set the resoure loss weight to 0.1, and we search the architecture for 130 epochs.

Recon hyperparameters. We compute the conflict scores of each shared layers for 40 epochs with joint-train as the update algorithm, and we set $S = -0.02$ for computing the scores. We select 85 layers with the highest conflict scores and turn them into task-specific.

C Additional Experiments

Additional experiments on CityScapes. To further illustrate that our algorithm can be adopted to various network structures (e.e., MTL architectures), we provide the results on the CityScapes dataset using MTAN Liu et al. (2019), a MTL architecture that is combined with task-specific attention modules in the encoder of SegNet, in Table 6. The results show that Recon can achieve significant performance improvements on MTAN.

Selecting the first K layers and the last K Layers as conflict layers does not work. To further support the conclusion that the selection of parameters with higher probability of conflicting gradients contributes most to the performance gain rather than the increase in model capacity. We compare Recon with two baselines: (1) Select the first K neural network layers and turn them into task-specific layers. (2) Select the last K neural network layers and turn them into task-specific layers. The multi-task learning results on the Multi-Fashion+MNIST benchmark are presented in Table 7. The results show that if we directly turn the top or the bottom of the neural network into task-specific parameters, it still will lead to performance degradation compared to Recon.

The distribution of gradient conflicts. To further support the conclusion that Recon can indeed reduce the occurrence of conflicting gradients, we provide the distributions of $\cos\phi_{ij}$ when using

Table 8: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) of the shared parameters on Multi-Fashion+MNIST dataset.

$\cos\phi_{ij}$	Joint-train	w/ RSL	w/ CMS	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1,0)	59.55	53.16	58.29	73.62	63.9	78.27	59.56	73.82	59.85	74.52	60.79	74.54
(0,-0.02]	10.14	9.01	10.77	20.13	12.51	12.54	9.61	19.75	9.58	19.43	11.13	19.77
(-0.02,-0.04]	8.52	7.34	8.72	5.13	8.59	5.54	8.19	5.17	7.94	4.89	8.83	4.62
(-0.04,-0.06]	6.45	5.69	6.48	0.94	5.39	2.23	6.49	1.05	6.24	0.96	6.05	0.89
(-0.06,-0.08]	4.79	4.53	4.61	0.14	3.29	0.85	4.76	0.16	4.41	0.15	4.06	0.13
(-0.08,-0.1]	3.27	3.95	3.18	0.02	1.93	0.34	3.32	0.04	3.24	0.04	2.72	0.03
(-0.1,1]	7.27	16.31	7.95	0.01	4.4	0.22	8.06	0.01	8.74	0.01	6.41	0.01

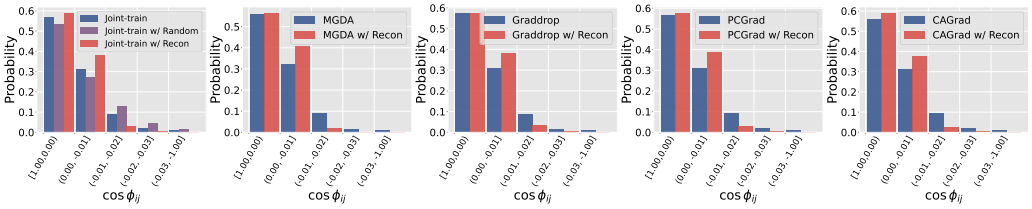


Figure 5: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) of the shared parameters on Multi-Fashion+MNIST dataset.

Recon and other baselines on Multi-Fashion+MNIST, as shown in Table 9 and Fig. 5. The detailed data for Fig. 1 are presented in Table 9

Recon can consistently find similar layers with different MTL methods. Given two layer permutations π_i and π_j, denote the positions of layer l in π_i and π_j by $\pi_i(l)$ and $\pi_j(l)$ respectively. We define the distance between the layers permutation as follows:

$$d(\pi_i, \pi_j) = \frac{1}{|\mathbb{L}|} \sum_{l \in \mathbb{L}} |\pi_i(l) - \pi_j(l)| \tag{24}$$

To evaluate the difference between the conflict permutations of neural network layers obtained by adopting different updating rules during the conflict searching procedure, we compute the distance between them, and the results are presented in Table 10. The empirical results illustrate that even though we adopt different updating rules during the conflict searching procedure, the layers with high conflict scores we find are similar. Therefore, we only perform the conflict search procedure once and apply the same modified structure to all the method in Table 1 to Table 4.

Table 9: The distribution of gradient conflicts (in terms of $\cos\phi_{ij}$) of the shared parameters on Multi-Fashion+MNIST dataset.

$\cos\phi_{ij}$	Joint-train	w/ Random	w/ Recon	MGDA	w/ Recon	Graddrop	w/ Recon	PCGrad	w/ Recon	CAGrad	w/ Recon
[1,0)	56.56	53.46	58.53	56.06	56.50	57.26	57.61	56.72	57.75	56.18	59.06
(0,-0.01]	31.25	27.28	37.67	32.36	40.93	31.06	38.28	31.19	38.76	31.25	37.84
(-0.01,-0.02]	9.26	13.17	3.04	8.87	2.12	8.93	3.32	9.09	2.87	9.37	2.44
(-0.02,-0.03]	2.05	4.32	0.5	1.71	0.26	1.72	0.54	1.90	0.42	2.00	0.41
(-0.03,-1.0]	1.25	1.76	0.25	1.0	0.18	1.03	0.26	1.10	0.2	1.20	0.25

Table 10: The distance between the conflict permutations of neural network layers obtained by adopting different updating rules during the conflict searching procedure on Multi-Fashion+MNIST dataset.

Method	Joint-train	CAGrad	PCGrad	Gradrop	mgd
Joint-train	0	-	-	-	-
CAGrad	1.07	0	-	-	-
PCGrad	0.78	1.17	0	-	-
Gradrop	0.59	0.83	0.68	0	-
mgd	1.71	1.32	**1.90**	1.56	0