Research Article

Proactive Gradient Conflict Mitigation in Multi-Task Learning:A Sparse Training Perspective

Zhi Zhang[1], Jiayi Shen[1], Shanghang Zhang[2], Ekaterina Shutova[1]

1. ILLC, University of Amsterdam, Netherlands; 2. State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University, China

Advancing towards generalist agents necessitates the concurrent processing of multiple tasks using a unified model, thereby underscoring the growing significance of simultaneous model training on multiple downstream tasks. A common issue in multi-task learning is the occurrence of gradient conflict, which leads to potential competition among different tasks during joint training. This competition often results in improvements in one task at the expense of deterioration in another. Although several optimization methods have been developed to address this issue by manipulating task gradients for better task balancing, they cannot decrease the incidence of gradient conflict. In this paper, we systematically investigate the occurrence of gradient conflict across different methods and propose a strategy to reduce such conflicts through sparse training (ST), wherein only a portion of the model's parameters are updated during training while keeping the rest unchanged. Our extensive experiments demonstrate that ST effectively mitigates conflicting gradients and leads to superior performance. Furthermore, ST can be easily integrated with gradient manipulation techniques, thus enhancing their effectiveness.

Corresponding authors: Shanghang Zhang, shanghang@pku.edu.cn; Ekaterina Shutova, e.shutova@uva.nl

1. Introduction

Attaining the status of a generalist agent necessitates addressing multiple tasks within a unified architecture, thereby emphasizing the significance of multi-task learning (MTL) [1], which involves

concurrently acquiring proficiency in multiple tasks and striving for superior overall performance compared to learning these tasks separately.

The primary concern for MTL lies in the phenomenon of task competition when the model is jointly trained by optimizing the average loss across all tasks. As a result, a subset of tasks demonstrates superior performance while others remain sub-optimized compared to their individual learning counterparts. One of the reasons behind it, from an optimization perspective, is gradient conflict (GC) [2], wherein the direction and magnitude of gradients between tasks differ significantly. This can result in the average gradient biasing towards optimizing one task while providing relatively smaller and sometimes even negative optimization for other tasks when updating the network [2][3].

Numerous works have employed the gradient manipulation method to directly or indirectly adjust the gradients of tasks to mitigate the issue of gradient conflict in tasks. The former involves direct alteration of task gradients through manually designed criteria when conflicts arise [2][4][5], while the latter modifies task gradients by adjusting weights of loss for each task [6][7][8][3]. Although these methods effectively modify the gradients conflicting with each other, they do not decrease the occurrence of conflicting gradients during training [9].

A simple approach to mitigate the occurrence of conflicting gradients is to convert those layers in which gradient conflict frequently arises into task-specific layers, thereby reducing the likelihood of gradient conflicts within the remaining shared layers [9]. However, this strategy introduces additional modules and disrupts the internal structure of the original model, resulting in increased computational costs. Furthermore, identifying frequently conflicting layers adds extra computational costs. This becomes prohibitively expensive as the model size continues to expand, and thus prompting our fundamental inquiry:

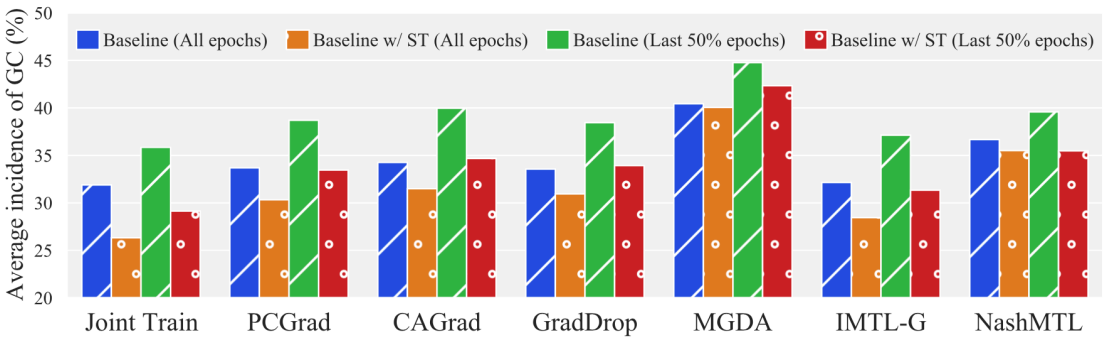


Figure 1. The average occurrence percentage of gradient conflict over epochs (all epochs/last 50% epochs) during training on the SAM model with NYUv2 datasets is evaluated using various methods, including joint training and gradient manipulation techniques.

(*Q*) *Is there a universally applicable approach to proactively mitigate the occurrence of gradient conflicts as well as preserve architectural integrity for MTL?*

To tackle this issue, we propose a novel perspective on mitigating gradient conflict in MTL, termed Sparse Training (ST), wherein a subset of parameters from the original model are selected to learn multiple tasks simultaneously while keeping the remaining parameters frozen. The intuition behind this lies in the reduction of a high-dimensional optimization problem to a low-dimensional one, which effectively alleviates the optimization complexity. Moreover, restricting the gradient updates of individual tasks to influence only a subset of parameters, rather than all parameters, effectively reduces potential interference between tasks.

Our key findings demonstrate that ST can effectively reduce the incidence of gradient conflict, particularly during the later stages of training, as illustrated in Fig. 1. A summary of our contributions is as follows: i) We provide a novel perspective, sparse training, for proactively reducing the incidence of gradient conflict during training while keeping the architecture intact; ii) Sparse training can be easily applied to improve various gradient manipulation methods by reducing the occurrence the gradient conflict over different datasets and architectures; iii) In addition to conventional research that primarily focuses on smaller models (MTAN[10] and SegNet[11]), we provide a comprehensive assessment of larger pre-trained models, including SAM[12], ViT[13], Swin Transformer[14], using various gradient manipulation techniques, such as PCGrad[2], CAGrad[5], GradDrop[4], MGDA[6], IMTL-G[7] and NashMTL[8], to stimulate research in the field of sparse training for MTL. Our findings demonstrate that

doi.org/10.32388/XLQW1L

as the model size increases, the issue of gradient conflict becomes more exacerbated, as shown in Fig. 5a, underscoring the significance of investigating the gradient conflict in large-scale models.

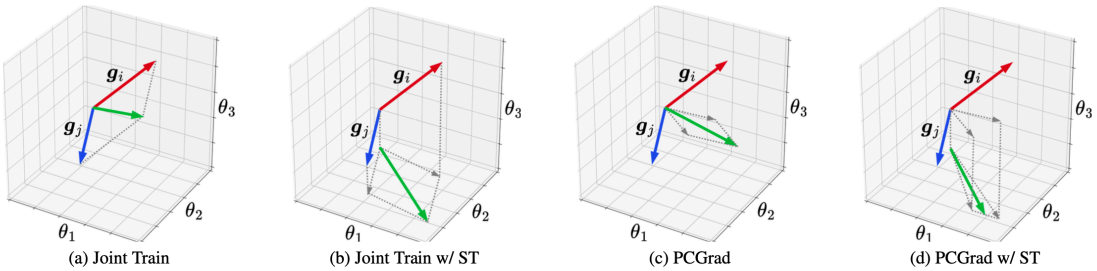
2. Related work

Multi-task optimization for MTL

The recent works[2][5][4][6][7][8][3] have achieved impressive results in addressing task imbalance issues in MTL by directly or indirectly modifying conflicting task gradients. Specifically, some works[2][5][4] propose to form a new update gradient at each training step by directly altering gradients based on certain criteria. Other works[6][7][8][3][15] learn dynamic loss scale to balance different tasks during training, and thus indirectly altering the gradient of tasks. However, these methods only address GC when it occurs and do not proactively prevent it. In this paper, we sparsely train an MTL model, effectively reducing the incidence of GC.

Training with subset of parameters

Several methods have already been proposed in single-task learning. Some of them select a subset of parameters based on a certain pre-defined rule, such as gradient[16][17] and magnitude of parameters[18]. In addition to selecting parameters by hand design, the works in[19][20][21] automatically select the subset of parameters through optimization. Although sparse training has been extensively investigated in single-task learning, its application in MTL remains relatively unexplored.[22] and[23] learn to share information between tasks using a sparse model instead of sparse training. Differently, we research the gradient conflict via the sparse training perspective.



(a) Joint Train (b) Joint Train w/ ST (c) PCGrad (d) PCGrad w/ ST

Figure 2. Visualization of gradients change for different methods. g_i and g_j are two conflicting gradients, and the green arrow is the actual update vector. The process of sparse training can be interpreted as performing an orthographic/coordinate projection of conflicting gradients onto the subspace defined by the selected parameters, resulting in better alignment of the projected gradients.

3. Approach

3.1. Background

Multi-task learning (MTL)

aims to learn multiple tasks simultaneously within a single model. Formally, given $\{\mathcal{T}_t\}_{t=1}^{T}$ tasks (≥ 2) and a model Θ with parameters $\Theta = (\theta_{sha}, \theta_{sep})$, where θ_{sha} and θ_{sep} are shared parameter with all tasks and task-specific parameters $\theta_{sep} = \{\theta_{sep}^t\}_{t=1}^{T}$ respectively, the commonly used optimization method for MTL (referred to as *Joint Train*) is based on computing the average loss across all tasks with equal weights:

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta), \tag{1}$$

$$\mathcal{L}(\Theta) = \mathcal{L}(\theta_{sha}, \theta_{sep}) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_t(\theta_{sha}, \theta_{sep}^t) \tag{2}$$

where each task t is associated with a corresponding loss function $\mathcal{L}_t(\theta_{sha}, \theta_{sep}^t)$.

Gradient conflict (GC)

However, optimizing all tasks by aggregating their losses indiscriminately (Eq. (2)) may lead to task competition, wherein certain tasks demonstrate improvement while others exhibit a decline compared to training them separately. From an optimization perspective, one of the reasons stems from conflicts in

gradients. Formally, the update of task $\mathcal{T}_i$ may potentially exert a detrimental impact on another task $\mathcal{T}_j$, namely:

$$\Delta\mathcal{L}_j = \mathcal{L}_j(\hat{\theta}_{sha}, \theta^j_{sep}) - \mathcal{L}_j(\theta_{sha}, \theta^j_{sep}), \tag{3}$$

$$\hat{\theta}_{sha} = \theta_{sha} - \alpha \mathbf{g}_i \tag{4}$$

where $\mathbf{g}_i = \nabla_{\theta_{sha}} \mathcal{L}_i(\theta_{sha}, \theta^i_{sep})$ is the gradient of loss on task $\mathcal{T}_i$ with respect to θ_{sha} and α is the learning rate. After the first-order Taylor approximation, Eq. (3) can be expressed as $-\alpha \mathbf{g}_i \cdot \mathbf{g}_j + o(\alpha)$. Gradient conflict arises when $\mathbf{g}_i \cdot \mathbf{g}_j < 0$, leading to $\Delta\mathcal{L}_j > 0$, indicating that task $\mathcal{T}_i$ has a detrimental impact on task $\mathcal{T}_j$. Following[2], we provide the definition of gradient conflict:

Definition 1 (Gradient Conflict). *If $\cos\phi_{ij} < 0$, where ϕ_{ij} is the angle between gradients of two tasks $\mathbf{g}_i$ and $\mathbf{g}_j$ $(i \neq j)$, then $\mathbf{g}_i$ and $\mathbf{g}_j$ are deemed to exhibit gradient conflict.*

Gradient manipulation

To alleviate the issue of gradient conflict, gradient manipulation methods adjust conflicting gradients based on specific criteria and utilize these modified gradients for model updating. Instead of updating the model on the average gradient in Eq. (1) and Eq. (2):

$$\nabla_{\theta_{sha}} \mathcal{L}(\Theta) = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\theta_{sha}} \mathcal{L}_t(\theta_{sha}, \theta^t_{sep}), \tag{5}$$

the gradients of all tasks in gradient manipulation methods are modified as follows:

$$\nabla_{\theta_{sha}} \mathcal{L}_{gm}(\Theta) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t \nabla_{\theta_{sha}} \mathcal{L}_t(\theta_{sha}, \theta^t_{sep}), \tag{6}$$

$$\mathbf{w}_t = f\left(\nabla_{\theta_{sha}} \mathcal{L}_1, \ldots, \nabla_{\theta_{sha}} \mathcal{L}_T\right) \tag{7}$$

where $\mathbf{w}_t$ can be either pre-defined or dynamically computed for tasks via f and thus achieve the aim of adjusting the task gradient[3][8][6][7][2][4][5]. However, the results of our experiment suggest that these methods can only modify gradients when conflicts occur, rather than proactively reducing the occurrence of GC during training, compared with *Joint Train*, as shown in Fig. 1.

3.2. Sparse training for multi-task learning

In this study, we investigate the gradient conflict commonly observed in multi-task learning from a novel perspective: sparse training, which selectively trains only a subset of the model parameters as opposed to full parameter training. This perspective is based on the intuition that by converting a high-dimensional

space optimization problem into a lower-dimensional one, the complexity of optimization can be effectively reduced. Additionally, by limiting the impact of gradient updates to only a subset of parameters for each task instead of all parameters, potential interference between tasks can be mitigated.

Sparse training (ST)

entails the initial parameter selection from the original model, and then updating only these parameters while keeping other parameters fixed during model training. To clarify potential misunderstandings regarding ST—often confused with sparse networks, where parameters are abandoned for model compression—we provide the following definition to ensure consistency and ease of understanding throughout this paper.

Definition 2 (Sparse Training). *Given a model Θ and a binary mask matrix M indicating whether parameters in Θ are selected, where $M \in \mathbb{R}^{|\Theta|\times|\Theta|}$, $M_{ii} \in \{0,1\}$ and $M_{ij} = 0 \ (\forall i \neq j)$, the model is updated by $\hat{\Theta} = \Theta - \alpha M \nabla_{\Theta} \mathcal{L}(\Theta)$. We define this training strategy as sparse training.*

Typically, the model architecture in multi-task learning includes a shared encoder as a feature extractor with task-specific decoders for multiple tasks. Therefore, sparse training is used in the encoder, and full parameters training for the decoders. We detail how the mask is computed in section 3.4. We now apply sparse training for multi-task learning (Joint Train). The visualization of the gradient change can be viewed in Fig. 2 and the update with the reformulated gradient from Eq. (5) is as follows

$$\begin{aligned} \hat{\theta}_{sha} &= \theta_{sha} - \nabla_{\theta_{sha}} \mathcal{L}(\Theta) \\ &= \theta_{sha} - M \frac{1}{T} \sum_{t=1}^{T} \nabla_{\theta_{sha}} \mathcal{L}_t(\theta_{sha}, \theta_{sep}^t). \end{aligned} \tag{8}$$

Combination with gradient manipulation methods

The application of sparse training can be seamlessly and effectively extended to improve various gradient manipulation methods in MTL. The update with the reformulated gradient from Eq. (6) is as follows

$$\begin{aligned} \hat{\theta}_{sha} &= \theta_{sha} - \nabla_{\theta_{sha}} \mathcal{L}_{gm}(\Theta) \\ &= \theta_{sha} - M \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t \nabla_{\theta_{sha}} \mathcal{L}_t(\theta_{sha}, \theta_{sep}^t). \end{aligned} \tag{9}$$

3.3. Theoretical analysis for sparse training

After introducing sparse training into MTL, the optimization objective in Eq. (1) can be formed:

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta), s.t. \left\| (I - M) \left(\theta_{sha} - \theta_{sha}^{in} \right) \right\|^2 = 0, \tag{10}$$

where θ_{sha}^{in} is the initialized original model for θ and I is identity matrix. According to Lagrangian duality, Eq. (10) can be reformulated as:

$$L = \min_{\Theta} \max_{\lambda} \mathcal{L}(\Theta) + \lambda \|(I - M)(\theta_{sha} - \theta_{sha}^{in})\|^2. \tag{11}$$

This can be transformed to optimize the upper bound L of regularized problem:

$$L_r = \min_{\Theta} \mathcal{L}(\Theta) + \|(I - M)(\theta_{sha} - \theta_{sha}^{in})\|^2 \leq L. \tag{12}$$

Please see the supplemental material for proof. [17] demonstrates that Eq. (12) has better stability and smaller generalization bound than only optimizing Eq. (1), resulting in better performance.

3.4. Parameter selection per neuron (PSN)

Several promising sparse training methods exist for single-task learning, but they are either time-consuming, requiring mask updates at each iteration[19][20][21], or memory-intensive due to gradient calculations for all parameters[16][17]. In MTL, where multiple tasks are trained simultaneously, time efficiency is crucial. Thus, we adopt a one-time selection method, choosing parameters before training and keeping the mask fixed throughout. We consider the following two aspects for selection, magnitude of the parameter and involvement of all neurons in the network.

The magnitude of parameters

Several studies have focused on model compression through the elimination of parameters with lower magnitudes[24][25]. This highlights the significance of parameters with larger magnitudes in neural networks, which is consistent with our experimental findings (See Fig. 5c). The intuition behind this phenomenon lies in the fact that parameters with larger magnitudes exert a greater influence on altering neuron activation states through the activation function, wherein a neuron becomes active once the input surpasses a predefined threshold. Therefore, we exclusively select parameters with the highest magnitude for training multiple tasks.

The involvement of all neurons

A simple idea is to select a certain proportion of parameters with the highest magnitude from the neural network (NN), but this may prevent some neurons from being engaged during training and hinder effective model training due to the dependence of the NN state on neuron activation. Motivated by studies highlighting distinct roles for different components in NN[26][16][27], we posit that engaging all

neurons is crucial for effective model training. The rationale is that each neuron within the network possesses the inherent capability to finely adjust its activation state, thereby effectively adapting the overall NN state to the tasks, especially for learning multiple tasks simultaneously. Our experiments further substantiate this assertion, as shown in Fig. 5c.

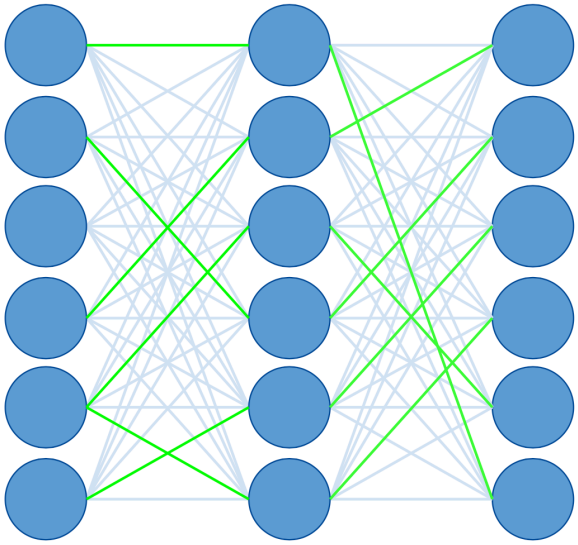


Figure 3. PSN. Top-1 highest-magnitude parameter among all input connections of each neuron is selected.

PSN

By integrating the two aspects, we select the top-K connections (weight/parameters) with the highest magnitude among all input connections for each neuron in the network (Please see Fig. 3 for top-1 example). This approach facilitates the training process for fitting tasks by ensuring that every neuron possesses activation potential, while parameters with higher magnitudes facilitate easier activation of neurons. In this paper, sparse training refers to using this method to select parameters and training the selected parameter, unless otherwise specified.

4. Experiments

Our experiments are conducted on comprehensive MTL benchmarks to evaluate the effectiveness of sparse training. First, we investigate if sparse training reduces gradient conflict. Subsequently, we

examine its impact on performance across various MTL setups. The more details of the experiment are provided in Appendix D.

4.1. Experimental Setup

Dateset

Our MTL datasets are categorized into three groups: i) *Dense prediction tasks:* ***NYUv2***[28]: An indoor scene understanding dataset containing 1449 RGBD images with per-pixel labels across 13 classes, including semantic segmentation, depth estimation, and surface normal prediction. ***CityScapes***[29]: 5000 street-view RGBD images with per-pixel annotations for 7-class semantic segmentation and depth estimation. ii) *Multiple binary-classification tasks:* ***CelebA***[30]: 200,000 facial images of 10,000 celebrities, each with 40 binary attributes for facial features. We use the first 10 attributes for 10 binary classification tasks due to limited computation. iii) *Multiple multi-class classification tasks:* ***VTAB***[31]: Containing 24 image understanding tasks with 1000 training examples per task. We use four tasks from it to create two multi-task benchmarks: ***Clevr***: Simple 3D shapes with counting and depth prediction tasks. ***SmallNORB***: Artificial objects with object azimuth and camera elevation prediction tasks.

Baseline

We evaluate our approach using various baselines including i) single-task learning (***STL***): Each task is trained independently; ii) ***Joint Train***: Training all tasks with average task loss; and 6 gradient manipulation methods including 3 direct and 3 indirect modification techniques. The former includes: iii) ***PCGrad***: Projecting each task gradient onto the normal plane of other tasks[2]; iv) ***CAGrad***: Enhancing the optimization of average loss by explicitly regulating the minimum decrease across tasks[5]; and v) ***GradDrop***: Stochastically dropping specific dimensions of the gradients based on their level of conflict. The latter includes vi) ***MGDA***: Identifying the same descent direction for each task[6]; vii) ***IMTL-G***: Determining the update direction by ensuring equal projections on gradients[7]; viii) ***NashMTL***: Treating MTL as a bargaining game to optimize all tasks[8].

Model

We experiment with several architectures including: i) *CNN-based:* ***MTAN***[10] incorporates an attention mechanism into the SegNet[11]. ii) *Transformer-based.* ***SAM***[32] is a strong visual foundation model for

segmentation. ***ViT-B/16***[13] and ***Swin Transformer***[14] are vision classification models pre-trained on ImageNet21K[33]. All experiments were conducted on pre-trained SAM, ViT and Swin (except for randomly initialized MTAN), unless otherwise specified.

Evaluation

i. ***Relative task drop*** ($\Delta m\%$). Following[34], we evaluate the MTL overall performance for a baseline b by computing the average performance drop against STL s over $\{\mathcal{T}_t\}_{t=1}^{T}$ tasks and $K_{\mathcal{T}_t}$ metrics for each $\mathcal{T}_t$: $\Delta m\% = \left(\frac{1}{T} \sum_{t=1}^{T} \frac{1}{K_{\mathcal{T}_t}} \sum_{k=1}^{K_{\mathcal{T}_t}} (-1)^{\delta_k} \left(M_b^k - M_s^k \right) / M_s^k \right) \times 100$
where M_b^k, M_s^k are the value of metrics k evaluated with b and s respectively. $\delta_k = 1$ if the M^k is higher the better and 0 otherwise.

ii. ***Average incidence of GC*** ($p\%$). We evaluate the extent of gradient conflict for a baseline by calculating the average incidence of GC over epochs during training. Given T tasks, E epochs, and I iterations per epoch, $p\% = \frac{1}{EI} \sum_{e=1}^{E} \sum_{i=1}^{I} \left(N_{gc} / N_{all} \right) \times 100$, where N_{gc} and N_{all} represent the number of occurrence of gradient conflicts between two tasks for all task combinations $\binom{T}{2}$ and the number of the combinations in each iteration during training, respectively.

4.2. Incidence of gradient conflict

We train a MTL model using the *Joint Train* and 6 state-of-the-art gradient manipulation techniques including *PCGrad, CAGrad, GradDrop, MGDA, IMTL-G* and *NashMTL* and then introduce our sparse training strategy to these methods. Throughout the training process, we record instances of GC between any two tasks among all tasks for each training iteration and then calculate the average incidence of GC both over all epochs and the last 50% epochs. The observations of the SAM model on the NYU-v2 dataset are provided below. Similar results on other datasets and models are shown in Appendix F.3, Appendix F.5, Appendix F.7 and Appendix F.6.

Gradient manipulation methods cannot effectively reduce the incidence of gradient conflict

The gradient manipulation methods[2][4][5][8][3][7][6] aim to modify conflicting gradients that are prevalent during the joint training of MTL. As shown in Table 1, the average incidence of GC using *Joint train* is 31.89% across all training epochs and 35.85% over the last 50% epochs. The incidence of GC cannot be effectively reduced by any gradient magnitude methods compared with the *Joint train*, as shown in Fig. 1 and Table 1. The reason is that these methods can only make the conflicting gradients not

conflict when the GC occurs, rather than proactively prevent the occurrence of GC. The incidence of GC is even exacerbated by these methods, particularly MGDA showing a significant increase of 8.55% compared to *Joint Train*. Notably, these findings are consistent with[9], where they provide the distribution of the angles between the two task gradients.

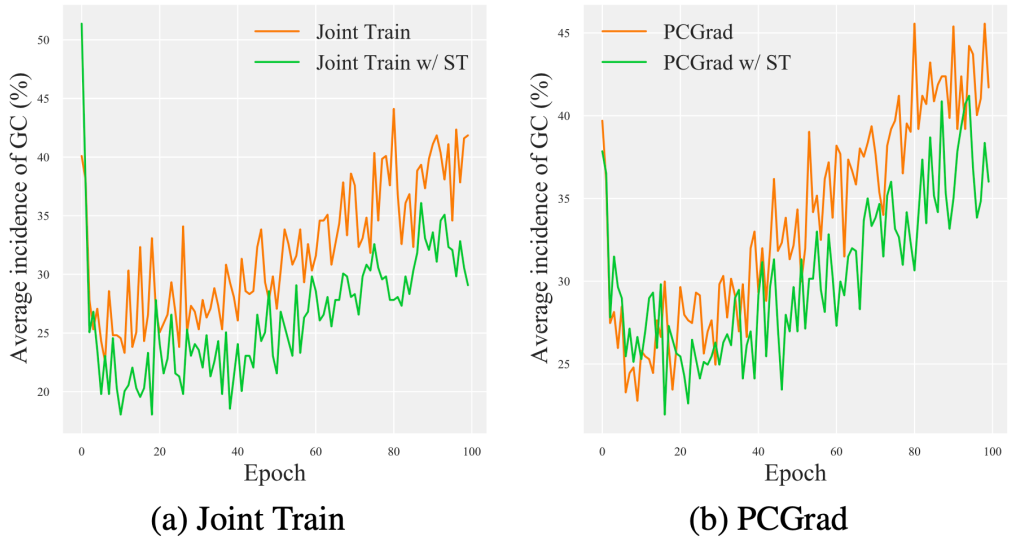
Sparse training effectively decreases the occurrence of gradient conflict

As shown in Fig. 1, after combining sparse training with all methods, including *Joint Train* and gradient manipulation methods, the average incidence of gradient conflict is effectively reduced over all epochs. For example, ST in *Joint Train* reduced the incidence over all epochs by 5.56%. The phenomenon of gradient conflict reduction is consistently observed in nearly every training epoch, as illustrated in Fig. 4, which further demonstrates the effectiveness of ST for decreasing gradient conflict. In addition, all methods with ST exhibit a greater improvement in the average incidence of gradient conflict during the last 50% epochs compared to all epochs, which implies a greater level of prevention of gradient conflict with the progress of sparse training. For instance of NashMTL, there is a threefold improvement in the average incidence of gradient conflict during the last 50% epochs compared to all epochs.

Methods	Average incidence of GC (%)	
	All epochs	Last 50% epochs
Joint Train	31.89	35.85
w/ ST	26.33 **(5.56)**	29.14 **(6.71)**
PCGrad	33.69	38.70
w/ ST	30.33 **(3.36)**	33.46 **(5.24)**
CAGrad	34.26	39.97
w/ ST	31.50 **(2.76)**	34.68 **(5.29)**
GradDrop	33.56	38.45
w/ ST	30.95 **(2.61)**	33.93 **(4.52)**
MGDA	40.44	44.77
w/ ST	40.05 **(0.39)**	42.34 **(2.43)**
IMTL-G	32.15	37.13
w/ ST	28.45 **(3.70)**	31.34 **(5.79)**
NashMTL	36.67	39.58
w/ ST	35.51 **(1.16)**	35.48 **(4.10)**

Table 1. Average incidence of GC between tasks for different methods. We compute the average incidence of GC over all epochs and the last 50% epochs during training SAM on NYUv2. The improvement by sparse training is provided in (•).

doi.org/10.32388/XLQW1L



(a) Joint Train (b) PCGrad

Figure 4. The incidence of GC between tasks during training SAM on NYUv2 dataset. The top and bottom figures are *Joint Train* and *PCGrad* respectively. Please see Fig. 7 in Appendix F.2 for more results on other gradient manipulation methods.

4.3. Performance on diverse benchmarks

It is natural to investigate whether reducing gradient conflict during training through sparsity can enhance performance on common benchmarks. In this section, we present diverse benchmarks to demonstrate the effectiveness of ST.

Methods	Segmentation		Depth		Surface Normal					$\Delta\mathbf{m}\%\downarrow$
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
STL	58.62	79.20	0.3810	0.1553	19.29	12.64	46.37	72.19	80.73	–
Joint Train	59.09	79.61	0.3348	0.1360	22.34	16.33	35.46	64.02	75.20	6.763
w/ ST	60.03	79.96	0.3320	0.1353	21.98	15.92	36.69	64.92	75.82	5.314
PCGrad	59.18	80.12	0.3258	0.1323	21.81	15.72	36.92	65.49	76.26	4.584
w/ ST	59.37	**80.33**	0.3272	0.1330	21.53	15.39	38.02	66.09	76.71	3.741
CAGrad	59.78	80.16	**0.3215**	**0.1305**	19.92	13.40	43.87	70.47	79.59	−1.816
w/ ST	60.33	80.20	0.3232	0.1306	19.74	13.20	44.42	71.04	80.02	**−2.423**
GradDrop	59.02	79.80	0.3283	0.1321	22.03	15.95	36.42	64.90	75.79	5.323
w/ ST	59.74	80.32	0.3278	0.1322	21.81	15.63	37.40	65.50	76.15	4.329
MGDA	37.43	67.58	0.4427	0.1810	19.23	**12.61**	46.43	**72.35**	**80.87**	9.162
w/ ST	41.60	69.96	0.4414	0.1778	**19.22**	**12.61**	**46.44**	72.29	80.80	7.791
IMTL-G	**60.64**	80.29	0.3324	0.1348	19.85	13.37	43.92	70.78	79.9	−1.537
w/ ST	60.35	80.18	0.3347	0.1350	19.65	13.15	44.66	71.25	80.20	−1.955
NashMTL	59.42	80.20	0.3303	0.1341	19.90	13.39	43.86	70.65	79.72	−1.295
w/ ST	59.36	79.98	0.3278	0.1323	19.63	13.02	45.06	71.31	80.15	−2.384

Table 2. The test performance on NYU-v2 dataset training on SAM model. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

doi.org/10.32388/XLQW1L

Methods	CelebA	Clevr			SmallNORB	NYU-v2	CityScapes
	Δm% ↓ (F1)	Counting (Top 1 ↑)	Depth (Top 1 ↑)	Δm% ↓	Δm% ↓	Δm% ↓	Δm% ↓
STL	–	58.64	57.68	–	–	–	–
Joint Train	3.12	54.86	54.68	5.84	10.70	5.59	26.87
w/ ST	2.03	**61.80**	54.81	−0.21	10.11	2.49	17.48
PCGrad	1.70	49.01	53.39	11.93	9.99	3.97	19.96
w/ ST	1.42	59.01	55.29	1.75	9.71	1.98	19.22
CAGrad	1.96	49.33	53.67	11.41	10.50	0.20	16.26
w/ ST	1.23	58.51	55.27	2.19	10.22	-2.76	8.88
GradDrop	1.18	49.02	52.88	12.36	11.73	3.58	20.34
w/ ST	0.83	58.87	54.07	2.94	10.76	1.38	17.45
MGDA	−0.41	49.56	55.97	9.22	10.15	1.38	6.91
w/ ST	**-1.08**	58.23	**56.91**	1.02	9.79	-3.18	**3.17**
IMTL-G	0.97	54.99	54.51	5.87	10.19	-0.76	10.65
w/ ST	0.19	61.05	56.73	**-1.24**	10.15	-3.18	7.10
NashMTL	3.59	47.04	53.07	13.89	10.84	-4.04	6.68
w/ ST	3.22	58.61	54.97	2.37	**9.57**	**-5.11**	3.99

Table 3. The test performance on CelebA, Clevr, SmallNORB, NYU-v2 and CityScapes dataset. CelebA is trained on Swin Transformer. Clevr and SmallNORB are trained on ViT. NYU-v2 and CityScapes are trained on MTAN. We only present $\Delta m\%$ for limited space. Please see 7, 12 and 11 for detailed results in supplemental materials. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

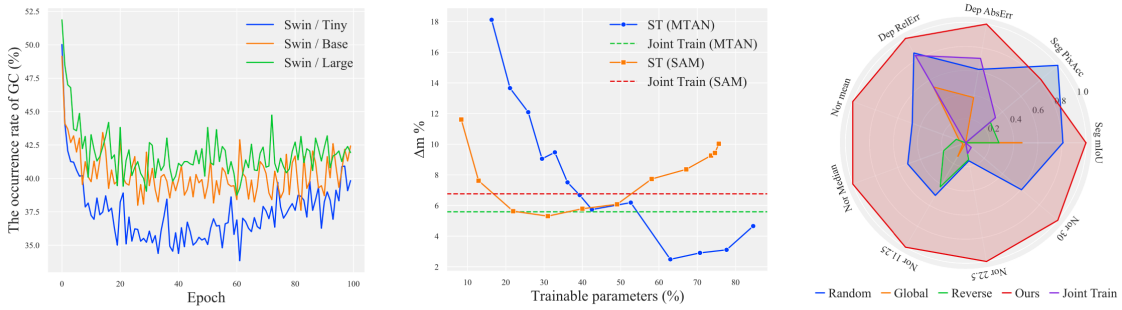
Sparse training improves the performance for all state-of-the-art methods

The performance of *Joint Train* and all gradient manipulation methods is consistently improved by sparse training, as demonstrated in Table 2 for NYU-v2 benchmarks. Specifically, sparse training not only enhances overall task performance but also improves individual task performance for the majority of methods. For example, in Table 2, *Joint Train* demonstrates improvements across all individual tasks through sparse training. Similarly, as shown in Table 3, all methods exhibit notable improvements by sparse training on CelebA, Clevr, SmallNORB and CityScapes benchmarks.

Effectiveness on both pre-trained and randomly initialized models

Our study primarily focuses on the sparse training for large pre-trained models, because leveraging prior knowledge from these models can be beneficial for MTL and our experimental results demonstrate that

larger models exhibit a more severe gradient conflict, as shown in Fig. 5a. However, in order to ensure a fair comparison with related works that manipulate gradients in small and randomly initialized models, we also conduct experiments under the same setting as theirs to further demonstrate the effectiveness of sparse training. As shown in Table 3, we observe that even for the small randomly initialized models, the performance of joint training and all gradient manipulation methods is improved by sparse training. Please see Tab. 7 and Tab. 12 for the detailed results in the Appendix.



(a) Different model size (b) Different num. of parameters (c) Different sparse methods

Figure 5. Ablation study for *Joint Train* with NYU-v2 dataset. (a) The average incidence of GC during joint training on different sizes of Swin transformers. Please see the numerical statics for all epochs in Tab. 8 in Appendix F.4. (b) The different number of trainable parameters for MTAN and SAM models. (C) Different sparse methods training on SAM. Metrics for all tasks are min-max normalized. Please see Tab. 5 for detailed results in Appendix F.1.

Generalization on different architectures and MTL tasks

To evaluate the generalization across diverse architectures and MTL tasks, we conducted experiments on both CNN-based models and transformer-based models with varying visual MTL capabilities. Specifically, our MTL tasks encompassed visual classification (CelebA, Clevr and SmallNORB) and visual dense prediction (NYU-v2 and CityScapes). For the former, we utilized Swin Transformer and ViT as backbones for multiple binary classification tasks (Tab. 3) and two multi-class classification tasks (Tab. 3, and Tab. 11 in Appendix), respectively. The latter involved predicting dense masks for each task, necessitating an encoder-decoder structure to generate corresponding masks. We explored two types of structures: a symmetrical encoder-decoder structure with a CNN-based model, e.g. MTAN (Tab. 3, and Tabs. 7 and 12 in Appendix) and an asymmetric structure with a heavy-weight encoder and a light-weight decoder using a transformer-based model, e.g. SAM (Tab. 2 in Appendix). As shown in these tables, the

doi.org/10.32388/XLQW1L

efficacy of sparse training in improving all baselines across various architectures and MTL tasks underscores its robust generalization capability.

4.4. Ablation study

The larger the model, the more severe gradient conflicts.

In this paper, we focus more on investigating the gradient conflict in the pre-trained large models as larger models demonstrated a more severe phenomenon of gradient conflict. This can be observed in Fig. 5a, where *Swin/Tiny* demonstrates significantly less gradient conflict compared to *Swin/Base* and *Swin/Large*. It is worth noting that although larger models tend to experience more severe gradient conflicts, this does not necessarily lead to inferior performance compared to smaller models with milder gradient conflicts. This discrepancy can be attributed to differences in model capacity and the prior knowledge embedded through pre-training. Nevertheless, this observation underscores the importance of exploring methods to mitigate gradient conflicts in larger models. Within the same model architecture and size, reducing gradient conflicts has been shown to improve performance, as evidenced by works such as[2][5]. Addressing severe gradient conflicts in larger models may thus unlock their full potential, enabling better utilization of their capacity and capabilities.

Effortless search for the number of trainable parameters.

We explore the effect of trainable parameter numbers for ST. The results in Fig. 5b show that the pre-trained model (SAM) and the randomly initialized model (MTAN) have different optimal trainable parameter numbers. MTAN requires $\sim$60% of the parameters, while SAM needs only $\sim$30%, leveraging information from the pre-trained model. In our paper, most of the experiments use these proportions for ST and achieve better results (please see Tab. 4 in Appendix D.1 for the detailed number). Additionally, ST offers a wide range of trainable parameter options that outperform *Joint Train*, which implies that hyperparameter search for the number of trainable parameters becomes effortless. Specifically, both models have a $\sim$40% probability of yielding superior outcomes.

Effectiveness for both higher magnitude and neural-level selection.

We investigate various parameter selection approaches: *Random:* Randomly selecting parameters from the network; *Global:* Choosing parameters with the highest magnitude from the whole network instead of the input connections of each neuron in the network (*Ours*); *Reverse:* Selecting parameters with the

lowest magnitude among input connections of each neuron. For a fair comparison, we maintain the same selected number. The results in Fig. 5c indicate that higher magnitude values are superior to lower ones (*Ours* $>$ *Reverse*). Furthermore, it is crucial to evenly select parameters from the entire network (*Ours* $>$ *Random* $>$ *Global*), as *Ours* ensure that the parameters of input connection for each neuron are selected, and *Random* guarantees an equal proportion of parameters is selected in each block of the network, whereas this is not the case for *Global* (see Fig. 6 for detailed statistics in Appendix).

5. Conclusion

In this paper, the occurrence of gradient conflict in multi-task learning is extensively investigated from a novel perspective: sparse training. Extensive experiments demonstrate that sparse training transferring high-dimensional space into low-dimensional space effectively reduces the incidence of gradient conflict during training while preserving the integrity of the original model. Furthermore, combining sparse training with other gradient manipulation methods significantly improves performance for multi-task learning.

Supplementary Material

In this supplemental material, we provide extra details about the content in the main body of the paper. First, we provide detailed proof for Equation (12) in Appendix A. Then, we discuss the limitations of our work in Appendix B. Moreover, the broader impacts of our research are discussed in Appendix C. In addition, we present all hyperparameters and experiment settings in Appendix F for a better understanding of the experiments and reproduction to the readers. We also provide the extended related works in Appendix E. Finally, the additional experiment results are demonstrated in Appendix F, which further indicate the effectiveness of our proposed method and the consistency with the claim in the main body of the paper.

A. Proof for Equation (12)

According to Lagrangian duality, Eq. (10) can be reformulated as:

$$\begin{aligned} L &= \min_{\Theta} \max_{\lambda} \mathcal{L}(\Theta) + \lambda \|(I - M)(\theta_{sha} - \theta_{sha}^{in})\|^2 \\ &\geq \max_{\lambda} \min_{\Theta} \mathcal{L}(\Theta) + \lambda \|(I - M)(\theta_{sha} - \theta_{sha}^{in})\|^2 \\ &\geq \min_{\Theta} \mathcal{L}(\Theta) + \|(I - M)(\theta_{sha} - \theta_{sha}^{in})\|^2 \end{aligned}$$

where λ is the Lagrangian multiplier.

B. Limitations

Due to the limited computational resources, we employ grid searches in the *Joint Train* method to determine the optimal hyperparameter for the number of trainable parameters, which is then utilized across all gradient manipulation methods. However, it is possible that these methods may benefit from a more optimized hyperparameter selection for the number of trainable parameters. Furthermore, sparse training can effectively mitigate gradient conflicts between tasks in MTL by reducing the dimensionality of parameter space and limiting their impact on updates between tasks. The regularization constitutes one of the theory's reasons. Nevertheless, we anticipate that our future research will contribute to a deeper comprehension of multi-task learning and subsequently enhance the performance of MTL.

C. Broader Impacts

The nature of our research does not directly contribute to societal impact; however, like any machine learning paper, it has the potential to adversely affect society through automation and job displacement. While it is challenging to predict specific risks, similar to any technology, inadequate regulation may lead to an exacerbation of social and economic inequality. The positive aspect lies in the potential environmental impact of our work, as multi-task learning enables information sharing among tasks, thereby reducing data requirements and further minimizing energy consumption during training.

D. Detailed experiment setting

D.1. Number of trainable parameters

We provide the number of trainable parameters for all experiments conducted in our paper. As shown in Table 4, most of them have the same percentage of trainable parameters within a model across different methods. In addition, in general, we can observe that sparse training for the pre-trained model needs $\sim$ 30% while that for random initialized model needs $\sim$60%.

Method	Pre-trained model				Random initialized model	
	SAM	Swin	ViT		MTAN	
	NYU-v2	CelebA	Clevr	SmallNORB	NYU-v2	CityScapes
Joint Train w/ ST	30.97	37.60	29.38	29.38	62.19	76.02
PCGrad w/ ST	30.97	37.60	29.38	19.63	62.19	76.02
CAGrad w/ ST	30.97	72.85	29.38	29.38	62.19	76.02
GradDrop w/ ST	30.97	49.58	29.38	29.38	62.19	76.02
MGDA w/ ST	30.97	37.60	29.38	29.38	62.19	62.19
IMTL-G w/ ST	30.97	37.60	29.38	29.38	62.19	62.19
NashMTL w/ ST	30.97	37.60	29.38	29.38	62.19	83.48
FAMO w/ ST	30.97	37.60	29.38	29.38	62.19	62.19

Table 4. Number of trainable parameters. The values in the table are expressed as percentages (%). As we select Top-K input parameters among all input connections for each neuron, therefore the same K might lead to different percentages of trainable parameters for different models. For example, K=300 results in 30.97% in SAM, 37.60% in Swin, and 29.38% in ViT for the pre-trained model.

D.2. Implementation details

Following the work of Nash[8], we apply all gradient manipulation techniques to the gradients of the shared weights. We set the hyperparameter c of CAGrad to 0.4, as it has been reported to yield optimal performance for NYUv2 and Cityscapes datasets[5]. The experiments were conducted on the A100 80G GPU. Typically, training with SAM using NYU-v2 and Swin with CelebA requires approximately 1 day for a gradient manipulation method. Training ViT with SmallNORB takes around 18 minutes for a gradient manipulation method, while training ViT with Clevr takes about 30 minutes. On the other hand, training MTAN with NYU-v2 demands roughly 18 hours for a gradient manipulation method, whereas training MTAN with CityScapes necessitates approximately 12 hours.

SAM, ViT, Swin

For all methods, including single-task learning, the gradient manipulation method, and our sparse training, we employed a batch size of 3 and searched for the optimal learning rate from the set {2e-4, 5e-5}, and then the best results are reported. The reason is that we find the optimal learning rate for sparse training is bigger than that for full parameters training. Therefore, for most methods, the optimal learning rate for sparse training is 2e-4 and that for the full parameters training is 5e-5. we also use data augmentations for all methods, following[5]. The batch size used is set to be 3 for NYUv2 dataset, and 256 for CelebA, and 128 for SmallNORB and Clevr.

MTAN

Following the works in[8][5], we incorporate data augmentations during training for both Joint Train method and all gradient manipulation methods. Each method is trained for 200 epochs with an initial learning rate of 0.0001, which is then reduced to 0.00005 after 100 epochs. For Multi-Task Learning (MTL) methods, we utilize a Multi-Task Attention Network (MTAN)[10] based on SegNet architecture proposed by[11]. Similar to previous studies[5], the STL baseline refers to training task-specific SegNet models. The batch size used is set to be 2 for NYUv2 dataset and 8 for CityScapes dataset respectively. To align with prior research on MTL including[5][2][11], we report the test performance averaged over the last 10 epochs.

E. Extended related work

Multi-task learning

Multi-task learning[1] aims to improve the overall performance of all tasks. In this work, we focus on a conventional setup of multi-task learning[35]: given a single input, multi-task models perform different and related predictions, such as segmentation, depth and surface normal. In other words, the input is shared by different tasks. In this paper, we roughly divide existing MTL into two categories:

i. **Multi-task optimization.** Recent works[2][5][4][6][7][8][3] provide impressive results in solving the task imbalance during optimization. The rationale behind these works is that re-weighting all task gradients or losses helps multi-task models reduce conflicting gradients among tasks[5][6]. Specifically, some works[4][6][7] propose to form a new update gradient at each optimization by

linearly combining task gradients. Other works[3][15] learn dynamic loss scale to balance different tasks during training. However, it is challenging to scale up most existing optimization works to giant foundation models due to non-trivial computational and memory costs. In this paper, we propose a neuron-based parameter selection to sparsely fine-tune the pre-trained model, which boosts the performance of most optimization methods.

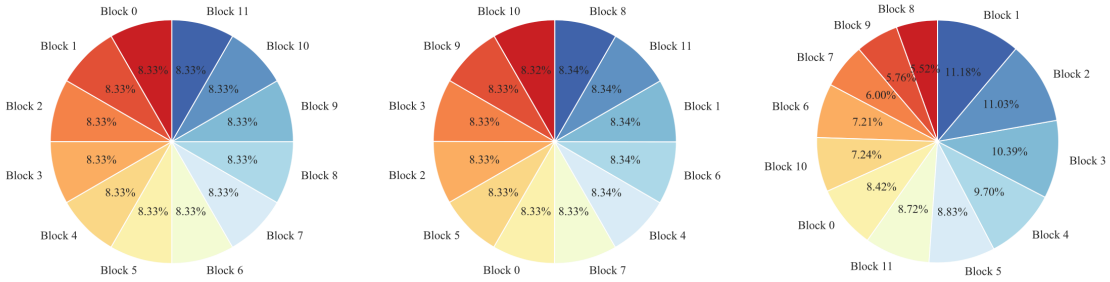
ii. **Multi-task architecture** In this branch, multi-task methods design different architectures to improve the exchanging or sharing of information among tasks[35]. Regarding where tasks interact, multi-task architectures are separated into *encoder-focused* and *decoder-focused*. The former shares the information in the encoder by the transformation of activations among tasks[36], learnable task-specific attention modules[10], branching networks for similar tasks[37] and so on. The latter recursively uses task predictions to improve overall performance[38][39][40]. However, these architectures still suffer from the task imbalance issue during multi-task optimization. In this paper, our work focuses on boosting multi-task optimization. As one of the multi-task optimization methods, our method can seamlessly generalize to different backbone models.

Training with subset of parameters

several methods already proposed in single-task learning. several methods select a subset of parameters based on a certain pre-defined rule, such as gradient[16][17] and magnitude of parameters[18]. In addition to selecting parameters by hand design,[19][20][21] automatically select the subset of parameters through optimization. Although sparse training has been extensively investigated in single-task learning, its application in multi-task learning remains relatively unexplored.[22][23] learning to share information between tasks using a sparse model. Differing from them, in this paper, we systematically research the gradient conflict via the sparse training perspective.

Methods	Segmentation		Depth		Surface Normal					$\Delta m\%$ ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
Random	59.85	80.09	0.3357	0.1359	22.17	16.12	36.08	64.50	75.56	6.014
Global	59.53	79.38	0.3380	0.1373	22.32	16.32	35.62	63.93	75.16	6.855
Reverse	59.35	79.57	0.3417	0.1396	22.31	16.25	35.98	64.07	75.16	6.960
Ours	60.03	79.96	0.3320	0.1353	21.98	15.92	36.69	64.92	75.82	5.314

Table 5. Different sparse training methods on SAM model with NYU-v2 datasets.



(a) Ours (b) Random (c) Global

Figure 6. The distribution of selected trainable parameters for different sparse training methods over different blocks. The experiments are conducted on SAM model with NYU-v2 dataset.

F. Detailed experiment results

In this section, we provide the detailed experiment results conducted in the main body of our paper, including the average incident of gradient conflict, the incident of gradient conflict for all epochs, and visualization of the gradient conflict for *Joint Train* and all gradient manipulation methods.

F.1. Ablation study

The detailed results for various sparse methods are provided in Tab. 5, which is the full version of Fig. 5c. It can be observed that, with the exception of *Pix Acc* in segmentation, our sparse method outperforms other methods. In addition, we provide the distribution of the selected parameters using different sparse training over different blocks of the model. As shown in Fig. 6, the parameters selected by our sparse training method and *Random* are evenly distributed over the whole network. As for *Global* selecting the parameters with the highest magnitude, the distribution of selected parameters is largely different over different blocks

F.2. NYU-v2 on SAM

The incidence of gradient conflict for *Joint Train* and gradient manipulation method over all epochs are shown in Fig. 7, which is the full version of Fig. 4 in the main body of the paper.

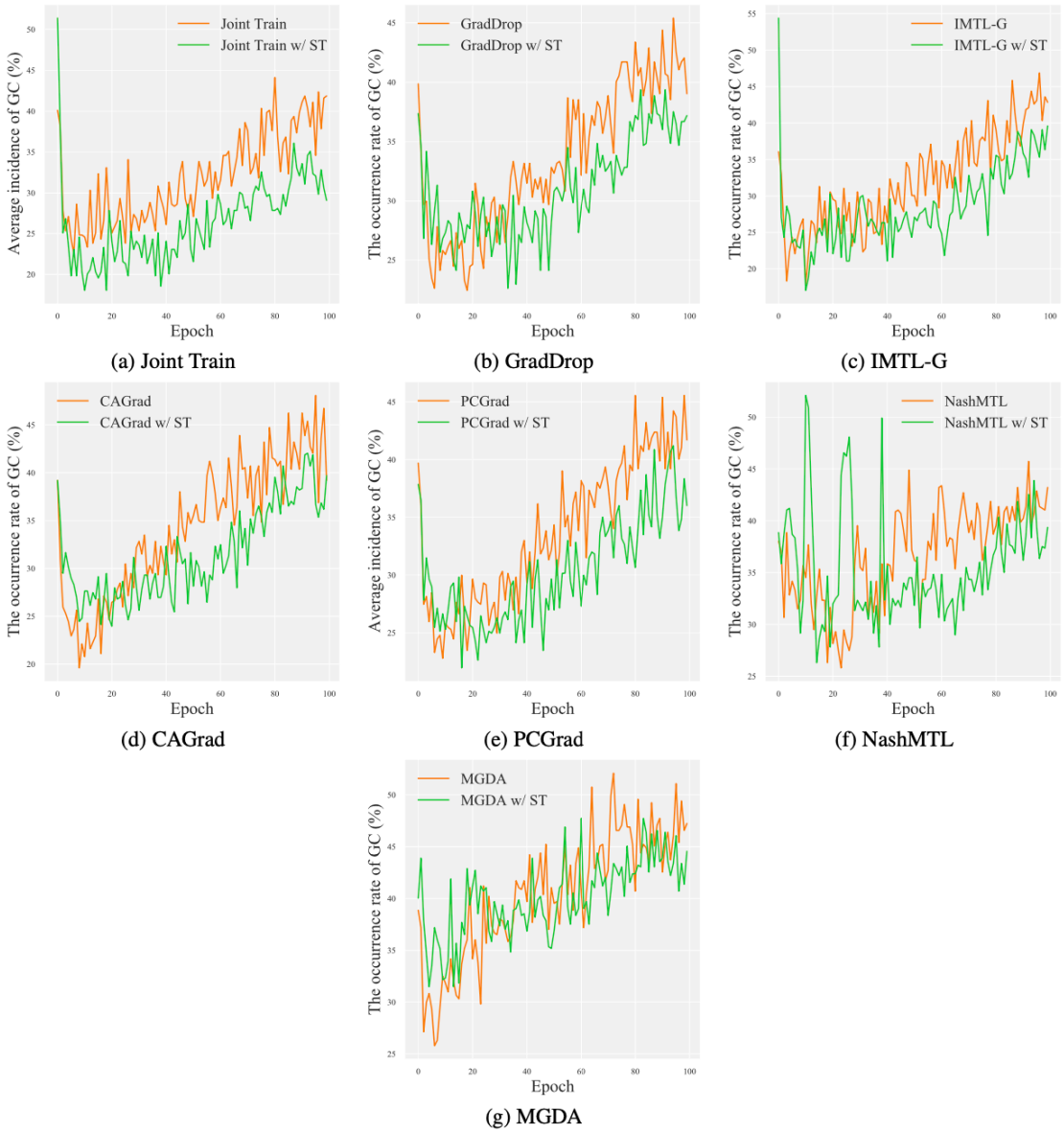


Figure 7. The number of occurrence gradient conflictions between tasks during training SAM on NYUv2 dataset.

doi.org/10.32388/XLQW1L

F.3. NYU-v2 on MTAN

Methods	Average incidence of GC (%)	
	All epochs	Last 50% epochs
Joint Train	36.01	39.87
w/ ST	33.86 **(2.15)**	36.45 **(3.42)**
PCGrad	35.71	39.51
w/ ST	34.05 **(1.66)**	37.25 **(2.26)**
CAGrad	37.21	40.93
w/ ST	34.14 **(3.07)**	37.04 **(3.89)**
GradDrop	36.37	39.71
w/ ST	34.42 **(1.95)**	37.10**(2.61)**
MGDA	37.76	42.1
w/ ST	37.15 **(0.61)**	41.25 **(0.85)**
IMTL-G	37.14	41.22
w/ ST	35.81 **(1.33)**	39.17 **(2.05)**
NashMTL	37.19	40.79
w/ ST	35.83 **(1.36)**	39.0 **(1.79)**

Table 6. Average incidence of gradient conflict between tasks over epochs for different methods. The improvement by sparse training is provided in (•). We calculate the average incidence of gradient conflict over all epochs and the last 50% epochs during training MTAN on NYUv2.

We also conduct experiments on MTAN with NYU-v2 dataset. MTAN is a random initialized model. As we can see in Tab. 6, even for the random initialized model, sparse training can also reduce the incidence of gradient conflict. The visualization of the occurrence of gradient conflict for each epoch is shown in Fig. 9 and the average incidence of gradient conflict across all epochs for different methods is shown in Fig. 8.

As for the performance of the overall tasks on NYU-v2, the sparse training improves not only the overall performance ($\Delta m\%$) but also the performance of each task for all methods including *Joint Train* and all gradient manipulation methods, as shown in Tab. 7. In addition, following[8], we conduct the experiments three times with three different seeds. The $mean \pm std$ is presented in Tab. 7, we can observe that the sparse training is robust to the random seed.

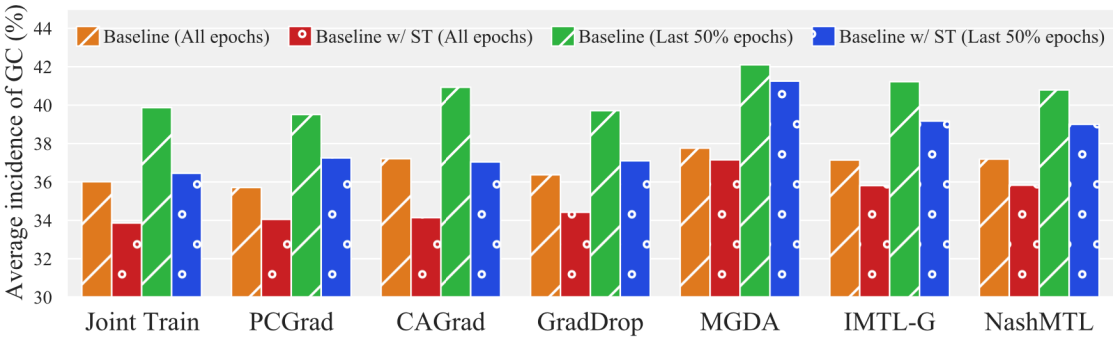


Figure 8. The average occurrence percentage of gradient conflict over epochs (all epochs/last 50% epochs) during training on MTAN model with NYU-v2 datasets was evaluated using various methods, including joint training and gradient manipulation techniques.

doi.org/10.32388/XLQW1L

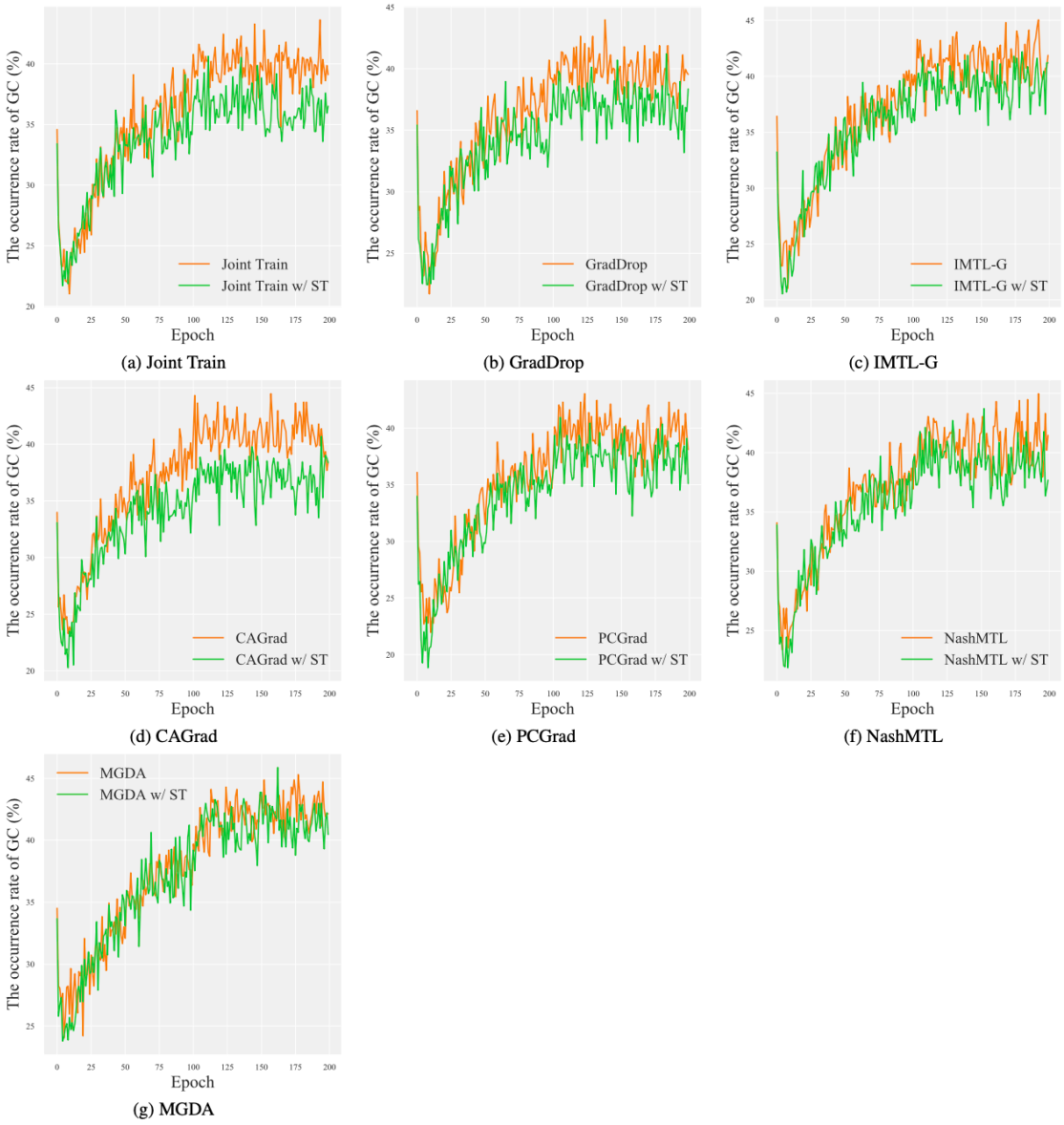


Figure 9. The number of occurrence gradient conflictions between tasks during tuning MTAN on NYUv2 dataset.

doi.org/10.32388/XLQW1L

Methods	Segmentation		Depth		Surface Normal					$\Delta m\% \downarrow$
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
					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	–
Joint Train	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	5.59
w/ ST	41.04 (±0.28)	66.05 (±0.12)	0.5417 (±0.0008)	0.2232 (±0.0011)	27.40(±0.05)	22.90(±0.12)	23.58(±0.13)	49.59(±0.14)	63.01(±0.09)	2.49(±0.11)
PCGrad	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	3.97
w/ ST	40.49 (±0.32)	66.17(±0.23)	0.5441 (±0.0023)	0.2264 (±0.0030)	27.09 (±0.08)	22.55(±0.03)	24.22(±0.12)	50.34(±0.17)	63.63(±0.12)	1.98(±0.12)
CAGrad	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58	0.20
w/ ST	39.93(±0.33)	66.19(±0.16)	0.5299(±0.0025)	0.2097(±0.0038)	25.71(±0.02)	20.70(±0.03)	26.86(±0.13)	54.22(±0.15)	67.30(±0.13)	-2.76(±0.10)
GradDrop	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	3.58
w/ ST	40.84(±0.35)	66.84(±0.24)	0.5288(±0.0021)	0.2209(±0.0021)	27.18(±0.03)	22.56(±0.07)	24.10(±0.11)	50.33(±0.14)	63.67(±0.13)	1.38(±0.12)
MGDA	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36	1.38
w/ ST	32.42(±0.41)	61.61(±0.21)	0.5851(±0.0015)	0.2239 (±0.0032)	24.35(±0.02)	18.61(±0.03)	31.14(±0.12)	58.63(±0.15)	70.62(±0.13)	-3.09(±0.14)
IMTL-G	39.35	65.60	0.5426	0.2256	26.02	21.19	26.20	53.13	66.24	-0.76
w/ ST	40.73(±0.33)	66.00(±0.17)	0.5219(±0.0015)	0.2100(±0.0021)	25.6(±0.05)	20.64(±0.04)	26.81(±0.16)	54.38(±0.15)	67.49(±0.12)	-3.18(±0.11)
NashMTL	40.13	65.93	0.5261	0.2171	25.26	20.08	28.40	55.47	68.15	-4.04
w/ ST	39.75(±0.21)	66.45(±0.05)	0.5156(±0.0006)	0.2121(±0.0009)	24.96(±0.01)	19.80(±0.05)	28.80(±0.11)	56.20(±0.10)	68.93(±0.09)	-5.11(±0.07)

Table 7. The test performance on NYU-v2 dataset training on MTAN model, involving three tasks: semantic segmentation, depth estimation and surface normal. The result is the mean over three random seeds (std is presented in (± ●). The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

F.4. NYU-v2 on Swin

In order to investigate how the incidence of gradient conflict changes with varying model sizes, we conduct experiments on Swin/Tiny, Swin/Base and Swin/Large through the Joint Train. As depicted in Tab. 8, there is an observed increase in the incidence of gradient conflict as the model size increases. Additionally, the performance of tasks improves as the model size increases Tab. 9.

Model / Size	Average incidence of GC (%)
Swin / Tiny	37.42
Swin / Base	40.34
Swin / Large	41.84

Table 8. The average incidence of gradient conflict across all epochs during joint training with NYU-v2 on different sizes of Swin transformer.

doi.org/10.32388/XLQW1L

Model	Segmentation		Depth		Surface Normal				
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑		
					Mean	Median	11.25	22.5	30
Swin/Tiny	55.22	76.54	0.3746	0.1542	27.47	21.70	27.81	52.40	64.05
Swin/Base	59.60	79.16	0.3419	0.1388	25.88	19.74	31.23	56.24	67.32
Swin/Large	61.34	80.28	0.3321	0.1345	25.09	18.73	33.05	58.12	68.86

Table 9. The test performance on NYU-v2 dataset jointly training on Swin models.

F.5. CelebA on Swin

Following[8], we train CelebA on Swin for only 30 epochs, because there are many more tasks in this dataset compared with other datasets, which leads to a significant increase in computation. As we can observe in Fig. 10, most of the methods including Joint Train and gradient manipulation methods can be improved by sparse training in terms of average incidence of gradient conflict between tasks over epochs. It is noted that the improvement by sparse training here is not significant, which is because of the limited training epoch. Specifically, as shown in Fig. 1 and Fig. 6, our sparse training improves more for later epochs. As for the performance of CelebA on Swin, please refer to Tab. 3. The visualization for the occurrence of gradient conflict for each epoch and average incidence of gradient conflict over all epochs for different methods, including Joint Train and all gradient manipulation methods, are shown in Fig. 10 and Fig. 11

Methods	Average incidence of GC (%)	
	All epochs	Last 50% epochs
Joint Train	47.61	48.78
w/ ST	46.96 **(0.65)**	48.48 **(0.30)**
PCGrad	48.48	50.83
w/ ST	47.24 **(1.24)**	48.88 **(1.95)**
CAGrad	48.21	50.23
w/ ST	48.33 **(-0.12)**	50.40 **(-0.17)**
GradDrop	47.36	48.72
w/ ST	47.13 **(0.23)**	48.57 **(0.15)**
MGDA	44.56	45.65
w/ ST	44.30 **(0.26)**	44.26 **(1.39)**
IMTL-G	46.89	47.77
w/ ST	45.03 **(1.86)**	46.32 **(1.45)**
NashMTL	46.83	47.67
w/ ST	46.78 **(0.05)**	47.34 **(0.33)**

Table 10. Average incidence of gradient conflict between tasks over epochs for different methods. The improvement by sparse training is provided in (•). We calculate the average incidence of gradient conflict over all epochs and the last 50% epochs during training Swin on CelebA.

doi.org/10.32388/XLQW1L

Figure 10. The number of occurrence gradient conflictions between tasks during tuning Swin on CelebA dataset.

doi.org/10.32388/XLQW1L

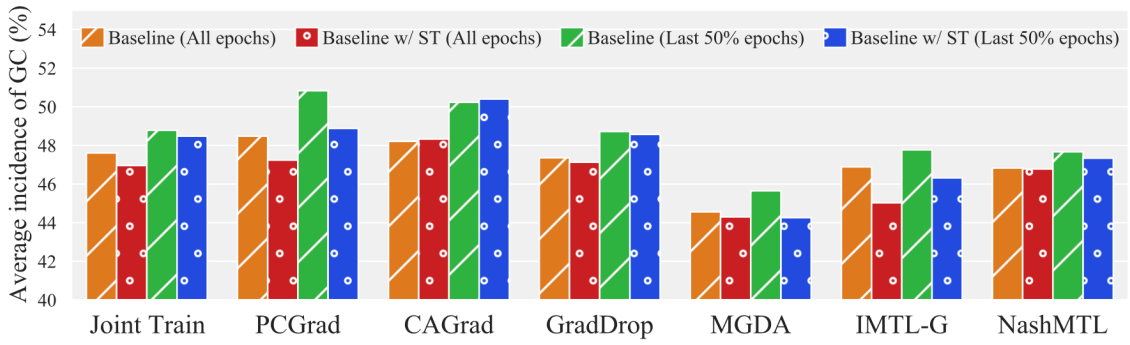


Figure 11. The average occurrence percentage of gradient conflict over epochs (all epochs/last 50% epochs) during training on Swin model with CelebA datasets was evaluated using various methods, including joint training and gradient manipulation techniques.

F.6. SmallNORB on ViT

SmallNORB is a much more difficult benchmark compared to other benchmarks in this paper. It comprises artificial objects observed under varying conditions and includes two tasks: object azimuth and camera-elevation prediction. As shown in Tab. 11, even for the STL, the Top 1 accuracy only achieves $\sim$30%, therefore, we use Top 5 as an extra metric here. We observed that even for this difficult task, sparse training can still achieve better performance compared with Joint Train and all gradient manipulation methods.

Methods	Object Azimuth		Camera Elevation		$\Delta m\% \downarrow$
	Top 1 ↑	Top 5 ↑	Top 1 ↑	Top 5 ↑	
STL	32.92	70.06	36.56	94.67	–
Joint Train	28.01	67.05	29.84	89.75	10.70
w/ ST	27.33	68.35	30.73	89.87	10.11
PCGrad	28.79	67.85	30.10	88.44	9.99
w/ ST	27.539	67.92	31.18	90.18	9.71
CAGrad	28.72	68.42	29.33	87.93	10.50
w/ ST	28.59	68.21	29.82	88.37	10.22
GradDrop	27.50	66.13	29.86	88.50	11.73
w/ ST	28.34	67.79	29.52	88.38	10.76
MGDA	30.82	70.13	27.29	86.16	10.15
w/ ST	28.28	68.88	30.01	89.47	9.79
IMTL-G	29.57	69.92	28.51	86.74	10.19
w/ ST	27.65	69.09	30.01	89.66	10.15
NashMTL	27.02	66.88	30.83	89.74	10.84
w/ ST	28.17	67.93	31.01	89.35	9.57

Table 11. The test performance on SmallNORB dataset trained on ViT. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

F.7. CityScapes on MTAN

We also conduct experiments on MTAN with CityScapes dataset. MTAN is a random initialized model. As we can see in Figure 13, even for the random initialized model, sparse training can also reduce the incidence of gradient conflict. The reduction in the incidence of gradient conflict for CityScapes is observed to be comparatively smaller than that for NYU-v2. This discrepancy can be attributed to the fact that CityScapes, which involves only two tasks, has a lower likelihood of encountering gradient conflicts between tasks compared to NYU-v2, which encompasses three tasks. The visualization of the occurrence of gradient conflict for each epoch is shown in Figure 12 and the average incidence of gradient conflict across all epochs for different methods is shown in Table 13. As for the performance of the overall tasks on CityScapes, the sparse training improves all methods including Joint Train and all gradient manipulation methods, as shown in Table 12.

Methods	Segmentation		Depth		$\Delta m\% \downarrow$
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	
STL	77.61	94.15	0.0122	35.68	–
Joint Train	78.14	94.29	0.0174	59.21	26.87
w/ ST	78.34	94.34	0.0143	55.00	17.48
PCGrad	77.79	94.21	0.0155	51.99	19.96
w/ ST	77.79	94.26	0.0160	51.99	19.22
CAGrad	76.82	93.70	0.0138	53.74	16.26
w/ ST	77.20	94.01	0.0150	39.85	8.88
GradDrop	77.91	94.28	0.0154	55.58	20.34
w/ ST	78.34	94.38	0.0163	48.95	17.45
MGDA	69.91	92.17	0.0124	40.68	6.91
w/ ST	68.38	91.91	0.0128	33.19	3.17
IMTL-G	77.55	94.10	0.0135	47.17	10.65
w/ ST	75.75	93.98	0.0138	40.16	7.10
NashMTL	77.51	94.22	0.0152	36.36	6.68
w/ ST	76.87	94.09	0.0148	33.30	3.99

Table 12. The test performance on CityScapes dataset training on MTAN model. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

doi.org/10.32388/XLQW1L

Methods	Average incidence of GC (%)	
	All epochs	Last 50% epochs
Joint Train	39.72	40.99
w/ ST	38.79 **(0.93)**	40.02 **(0.97)**
PCGrad	39.98	41.06
w/ ST	38.66 **(1.32)**	39.97 **(1.09)**
CAGrad	39.39	40.94
w/ ST	37.77 **(1.62)**	39.42 **(1.52)**
GradDrop	39.32	40.72
w/ ST	39.03 **(0.29)**	40.12 **(0.60)**
MGDA	36.37	39.69
w/ ST	36.14 **(0.23)**	39.38 **(0.31)**
IMTL-G	37.72	39.51
w/ ST	36.83 **(0.89)**	38.72 **(0.79)**
NashMTL	38.40	40.69
w/ ST	38.04 **(0.36)**	40.26 **(0.43)**

Table 13. Average incidence of gradient conflict between tasks over epochs for different methods. The improvement by sparse training is provided in (•). We calculate the average incidence of gradient conflict over all epochs and the last 50% epochs during training MTAN on CityScapes.

doi.org/10.32388/XLQW1L

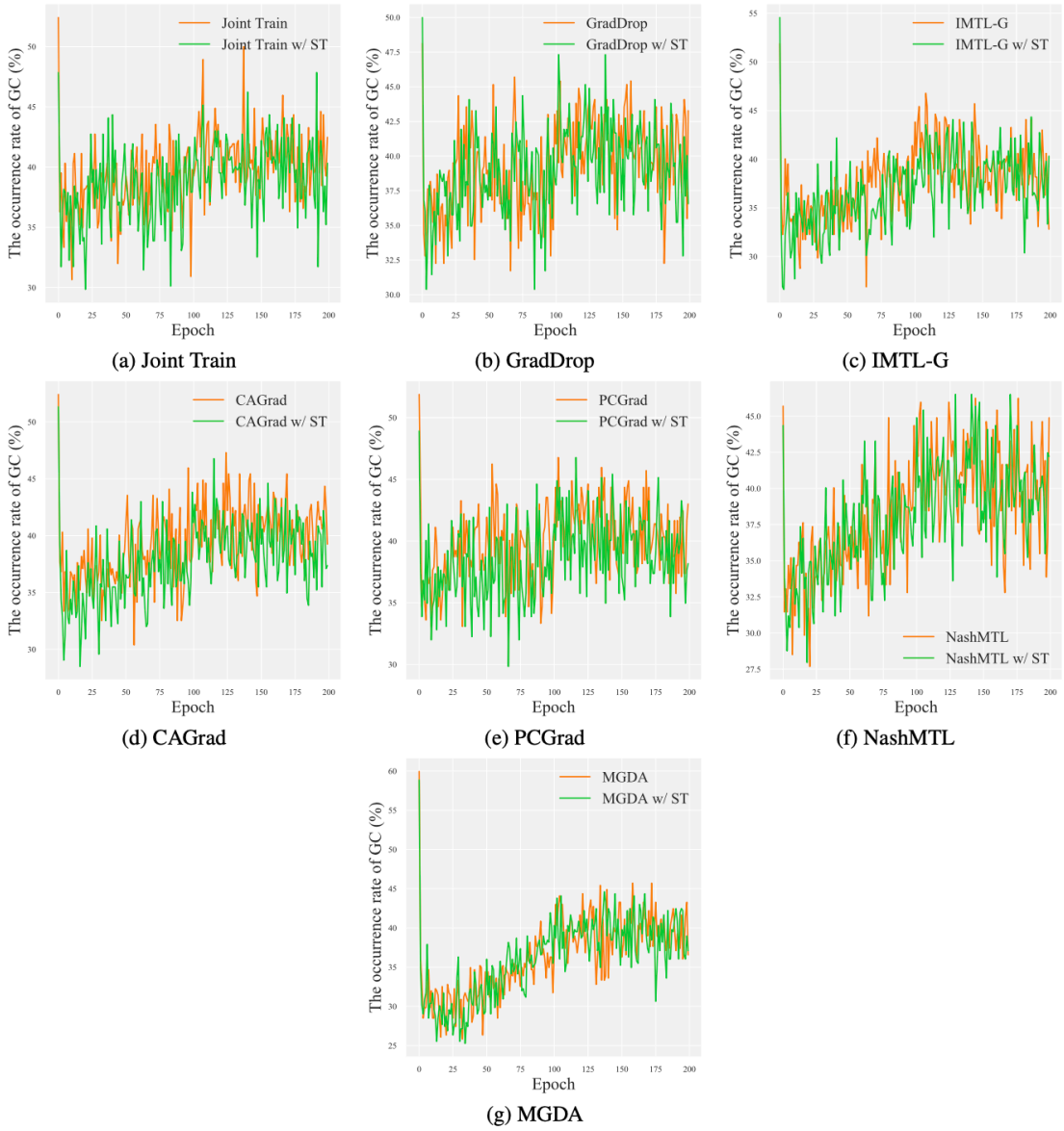


Figure 12. The number of occurrence gradient conflictions between tasks during tuning MTAN on CityScapes dataset.

doi.org/10.32388/XLQW1L

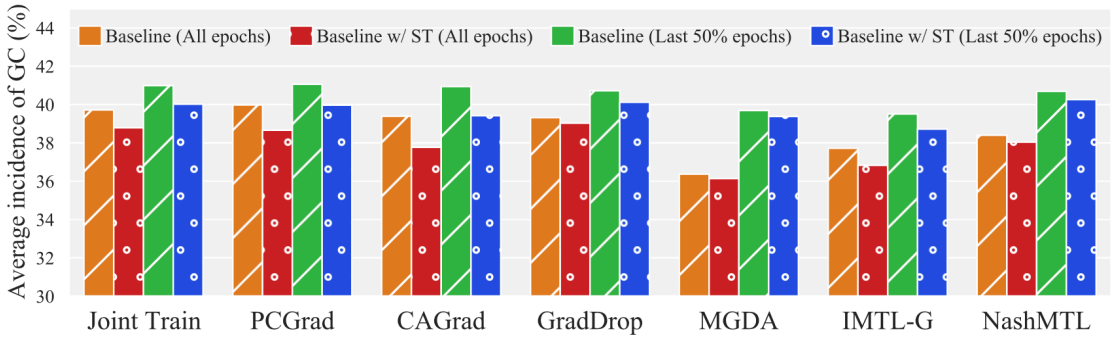


Figure 13. The average occurrence percentage of gradient conflict over epochs (all epochs/last 50% epochs) during training on MTAN model with CityScapes datasets was evaluated using various methods, including joint training and gradient manipulation techniques.

F.8. FAMO

FAMO[3] is an approximation method for gradient manipulation by using the history of loss to compute the current task weight. We also try our sparse training with FAMO on NYU-v2, CelebA, Clevr, SmallORB datasets with ViT, SAM, MTAN and Swin models. As shown in 14, 16, 17 and 15, even for the approximation method, sparse training method achieves the best results and further show the effectiveness of our sparse training methods.

Methods	Segmentation		Depth		Surface Normal					Δm% ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
FAMO	57.64	78.59	0.3574	0.1463	19.396	12.846	45.61	71.87	80.59	−0.5669
w/ ST	57.68	78.79	0.3520	0.1430	19.279	12.711	46.12	72.06	80.72	−1.353

Table 14. The test performance on NYU-v2 dataset training on SAM model. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

doi.org/10.32388/XLQW1L

Methods	CelebA	Clevr			NYU-v2
	Δm% ↓ (F1)	Counting (Top 1 ↑)	Depth (Top 1 ↑)	Δm% ↓	Δm% ↓
FAMO	2.35	55.83	56.80	3.16	-4.10
w/ ST	2.32	**62.57**	56.04	**-1.93**	-4.46

Table 15. The test performance on CelebA, Clevr and NYU-v2 dataset. CelebA is trained on Swin Transformer and Clevr is trained on ViT. NYU-v2 is trained on MTAN. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

Methods	Segmentation		Depth		Surface Normal					Δm% ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
FAMO	38.88	64.90	0.5474	0.2194	25.06	19.57	29.21	56.61	68.98	-4.10
w/ ST	37.85	65.27	0.5543	0.2215	25.09	19.15	30.03	57.49	69.52	-4.46

Table 16. The test performance on NYU-v2 dataset training on MTAN model. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

Methods	Object Azimuth		Camera Elevation		Δm% ↓
	Top 1 ↑	Top 5 ↑	Top 1 ↑	Top 5 ↑	
FAMO	24.68	63.69	34.35	92.13	10.71
w/ ST	26.38	66.54	32.05	91.02	10.27

Table 17. The test performance on SmallNORB dataset trained on ViT. The green cell color indicates that sparse training improves the performance of joint training or gradient manipulation methods. The best result is highlighted in bold.

doi.org/10.32388/XLQW1L

References

1. a, bZhang Y, Yang Q (2021). "A survey on multi-task learning". IEEE Transactions on Knowledge and Data Engineering. **34** (12): 5586–5609.
2. a, b, c, d, e, f, g, h, i, j, k, l, mYu T, Kumar S, Gupta A, Levine S, Hausman K, Finn C (2020). "Gradient surgery for multi-task learning". Advances in Neural Information Processing Systems. **33**: 5824–5836.
3. a, b, c, d, e, f, g, h, iLiu B, Feng Y, Stone P, Liu Q (2023). "FAMO: Fast Adaptive Multitask Optimization". arXiv. Available from: arXiv:2306.03792.
4. a, b, c, d, e, f, g, hChen Z, Ngiam J, Huang Y, Luong T, Kretzschmar H, Chai Y, Anguelov D (2020). "Just pick a sign: Optimizing deep multitask models with gradient sign dropout". Advances in Neural Information Processing Systems. **33**: 2039–2050.
5. a, b, c, d, e, f, g, h, i, j, k, l, m, n, oLiu B, Liu X, Jin X, Stone P, Liu Q (2021). "Conflict-averse gradient descent for multi-task learning". Advances in Neural Information Processing Systems. **34**: 18878–18890.
6. a, b, c, d, e, f, g, h, i, jSener O, Koltun V (2018). "Multi-task learning as multi-objective optimization". Advances in Neural Information Processing Systems. **31**.
7. a, b, c, d, e, f, g, h, iLiu L, Li Y, Kuang Z, Xue J, Chen Y, Yang W, Liao Q, Zhang W (2021). "Towards impartial multi-task learning". In: International Conference on Learning Representations.
8. a, b, c, d, e, f, g, h, i, j, k, lNavon A, Shamsian A, Achituve I, Maron H, Kawaguchi K, Chechik G, Fetaya E (2022). "Multi-Task Learning as a Bargaining Game". arXiv. arXiv:2202.01017 [cs.LG].
9. a, b, cShi G, Li Q, Zhang W, Chen J, Wu X-M (2023). "Recon: Reducing conflicting gradients from the root for multi-task learning". arXiv preprint arXiv:2302.11289.
10. a, b, c, dLiu S, Johns E, Davison AJ. "End-to-end multi-task learning with attention". In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. p. 1871-1880.
11. a, b, c, dBadrinarayanan V, Kendall A, Cipolla R (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". IEEE Transactions on Pattern Analysis and Machine Intelligence. **39** (12): 2481–2495.
12. ^Chen T, Zhu L, Ding C, Cao R, Zhang S, Wang Y, Li Z, Sun L, Mao P, Zang Y (2023). "SAM Fails to Segment Anything? -- SAM-Adapter: Adapting SAM in Underperformed Scenes: Camouflage, Shadow, and More". arXiv. 2304.09148.
13. a, bDosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. (2020). "An image is worth 16x16 words: Transformers for image recognition at scale". arXiv preprint arXiv:2010.11929.

14. a, b Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021). "Swin transformer: Hierarchical vision trans former using shifted windows". Proceedings of the IEEE/CVF international conference on computer vision. p p. 10012–10022.
15. a, b Kendall A, Gal Y, Cipolla R (2018). "Multi-task learning using uncertainty to weigh losses for scene geome try and semantics". Proceedings of the IEEE conference on computer vision and pattern recognition. pages 7482–7491.
16. a, b, c, d Zhang Z, Zhang Q, Gao Z, Zhang R, Shutova E, Zhou S, Zhang S (2023). "Gradient-based Parameter S election for Efficient Fine-Tuning". arXiv preprint arXiv:2312.10136. **2023**. Available from: https://arxiv.org/ab s/2312.10136.
17. a, b, c, d Fu Z, Yang H, So AM-C, Lam W, Bing L, Collier N (2023). "On the effectiveness of parameter-efficient f ine-tuning". Proceedings of the AAAI Conference on Artificial Intelligence. **37** (11): 12799–12807.
18. a, b Lagunas F, Charlaix E, Sanh V, Rush AM (2021). "Block pruning for faster transformers". arXiv preprint ar Xiv:2109.04838. Available from: https://arxiv.org/abs/2109.04838.
19. a, b, c Sanh V, Wolf T, Rush A (2020). "Movement pruning: Adaptive sparsity by fine-tuning". Advances in neu ral information processing systems. **33**: 20378–20389.
20. a, b, c Mostafa H, Wang X. Parameter efficient training of deep convolutional neural networks by dynamic sp arse reparameterization. In: International Conference on Machine Learning. PMLR; 2019. p. 4646–4655.
21. a, b, c Xu R, Luo F, Zhang Z, Tan C, Chang B, Huang S, Huang F (2021). "Raise a child in large language model: Towards effective and generalizable fine-tuning". arXiv preprint arXiv:2109.05687. Available from: https://ar xiv.org/abs/2109.05687.
22. a, b Sun T, Shao Y, Li X, Liu P, Yan H, Qiu X, Huang X (2020). "Learning sparse sharing architectures for multi ple tasks". Proceedings of the AAAI conference on artificial intelligence. **34** (05): 8936–8943.
23. a, b Calandriello D, Lazaric A, Restelli M (2014). "Sparse multi-task reinforcement learning". Advances in neu ral information processing systems. **27**.
24. ^ Han S, Pool J, Tran J, Dally W (2015). "Learning both weights and connections for efficient neural network". Advances in neural information processing systems. **28**.
25. ^ Frankle J, Carbin M (2018). "The lottery ticket hypothesis: Finding sparse, trainable neural networks". arXi v preprint arXiv:1803.03635. Available from: https://arxiv.org/abs/1803.03635.
26. ^ Wang Z, Huang SL, Kuruoglu EE, Sun J, Chen X, Zheng Y (2021). "Pac-bayes information bottleneck". arXiv preprint arXiv:2109.14509. Available from: https://arxiv.org/abs/2109.14509.

27. ^Fan F, Xiong J, Li M, Wang G (2020). "On Interpretability of Artificial Neural Networks: A Survey". IEEE Transactions on Radiation and Plasma Medical Sciences. 5: 741-760.

28. ^Couprie C, Farabet C, Najman L, LeCun Y (2013). "Indoor Semantic Segmentation using depth information". arXiv. arXiv:1301.3572 [cs.CV].

29. ^Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016). "The cityscapes dataset for semantic urban scene understanding." In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223.

30. ^Liu Z, Luo P, Wang X, Tang X (2015). "Deep learning face attributes in the wild". In: Proceedings of the IEEE international conference on computer vision. pp. 3730–3738.

31. ^Zhai X, Puigcerver J, Kolesnikov A, Ruyssen P, Riquelme C, Lucic M, Djolonga J, Pinto AS, Neumann M, Dosovitskiy A, et al. A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867. 2019.

32. ^Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, Xiao T, Whitehead S, Berg AC, Lo WY, et al. Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. p. 4015–4026.

33. ^Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009). "Imagenet: A large-scale hierarchical image database." In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. Ieee. pp. 248–255.

34. ^Maninis KK, Radosavovic I, Kokkinos I (2019). "Attentive single-tasking of multiple tasks". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 1851--1860.

35. a, bVandenhende S, Georgoulis S, Van Gansbeke W, Proesmans M, Dai D, Van Gool L (2021). "Multi-task learning for dense prediction tasks: A survey". IEEE transactions on pattern analysis and machine intelligence. 44 (7): 3614–3633.

36. ^Misra I, Shrivastava A, Gupta A, Hebert M (2016). "Cross-stitch networks for multi-task learning". Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pages 3994–4003.

37. ^Guo P, Lee CY, Ulbricht D. "Learning to branch for multi-task learning." In: International conference on machine learning. PMLR; 2020. p. 3854-3863.

38. ^Xu D, Ouyang W, Wang X, Sebe N (2018). "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 675–684.

39. ^Zhang Z, Cui Z, Xu C, Yan Y, Sebe N, Yang J (2019). "Pattern-affinitive propagation across depth, surface normal and semantic segmentation". Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 4106–4115.

40. ^Zhang Z, Cui Z, Xu C, Jie Z, Li X, Yang J. Joint task-recursive learning for semantic segmentation and depth estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018. p. 235–251.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.