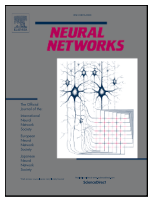
Neural Networks 195 (2026) 108317

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

Dual-balancing for multi-task learning

Baijiong Lin [a,b], Weisen Jiang [c], Feiyang Ye [d], Yu Zhang [d], Pengguang Chen [e], Ying-Cong Chen [a,b,f,*], Shu Liu [e,*], Ivor W. Tsang [g], James T. Kwok [f]

[a] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, 510000, China
[b] HKUST(GZ) - SmartMore Joint Lab, Guangzhou, 510000, China
[c] The Chinese University of Hong Kong, Hong Kong, 999077, China
[d] Southern University of Science and Technology, Shenzhen, 518055, China
[e] SmartMore, Shenzhen, 518000, China
[f] The Hong Kong University of Science and Technology, Hong Kong, 999077, China
[g] Centre for Frontier AI Research, A*STAR, 138632, Singapore

ARTICLE INFO

Keywords:
Multi-task learning
Loss balancing
Gradient balancing

ABSTRACT

Multi-task learning aims to learn multiple related tasks simultaneously and has achieved great success in various fields. However, the disparity in loss and gradient scales among tasks often leads to performance compromises, and the balancing of tasks remains a significant challenge. In this paper, we propose Dual-Balancing Multi-Task Learning (DB-MTL) to achieve task balancing from both the loss and gradient perspectives. Specifically, DB-MTL achieves loss-scale balancing by performing logarithm transformation on each task loss, and rescales gradient magnitudes by normalizing all task gradients to comparable magnitudes using the maximum gradient norm. Extensive experiments on a number of benchmark datasets demonstrate that DB-MTL consistently performs better than the current state-of-the-art.

1. Introduction

Multi-task learning (MTL) (Caruana, 1997; Chen et al., 2025; Zhang & Yang, 2022) jointly learns multiple related tasks using a single model, improving parameter-efficiency and inference speed compared to learning a separate model for each task. By sharing the model, MTL can extract common knowledge to improve each task's performance. It has demonstrated its superiority in various fields, such as computer vision (Lin et al., 2025, 2024; Luo et al., 2025; Vandenhende et al., 2021; Ye & Xu, 2022), natural language processing (Chen et al., 2024; Liu et al., 2017, 2019b; Sun et al., 2020; Wang et al., 2021), and recommendation systems (Hazimeh et al., 2021; Tang et al., 2020; Wang et al., 2023; Yi et al., 2025).

To learn multiple tasks simultaneously, equal weighting (EW) (Zhang & Yang, 2022) is a straightforward method that minimizes the sum of task losses with equal task weights. However, it usually suffers from the challenging *task balancing problem* (Lin et al., 2022; Vandenhende et al., 2021), in which some tasks perform well while others do not (Standley et al., 2020). To alleviate this problem, a number of methods have been recently proposed by dynamically tuning the task weights. They can be categorized as *loss balancing* (Kendall et al., 2018; Liu et al., 2022, 2019a; Ye et al., 2024a, 2021, 2024b) and *gradient balancing* (Chen et al., 2018b, 2020; Fernando et al., 2023; Liu et al., 2021a,b; Navon et al., 2022; Sener & Koltun, 2018; Wang et al., 2021; Yu et al., 2020). Loss balancing methods balance the tasks based on the learning speed (Liu et al., 2019a) or validation performance (Liu et al., 2022; Ye et al., 2024a, 2021) *at the loss level*, while gradient balancing methods balance the gradients by mitigating gradient conflicts (Yu et al., 2020) or enforcing gradient norms to be close (Chen et al., 2018b) *at the gradient level*. However, recently, multiple extensive empirical studies (Kurin et al., 2022; Lin et al., 2022; Xin et al., 2022) demonstrate that the performance of these existing methods is still unsatisfactory, indicating that task balancing is still an open problem.

To mitigate the task balancing problem, in this paper, we consider simultaneously balancing both the loss scales (at the loss level) and gradient magnitudes (at the gradient level). Since the loss scales/gradient magnitudes among tasks can be different, those with large values can dominate the update direction of the model, causing unsatisfactory performance on some other tasks (Liu et al., 2021b; Standley et al., 2020). Therefore, we propose a simple yet effective Dual-Balancing Multi-Task Learning (**DB-MTL**) method that consists of both loss-scale and gradient-magnitude balancing. First, we perform a logarithm transformation on

* Corresponding authors.
E-mail addresses: bj.lin.email@gmail.com (B. Lin), yingcongchen@ust.hk (Y.-C. Chen), liushuhust@gmail.com (S. Liu).

https://doi.org/10.1016/j.neunet.2025.108317
Received 5 February 2025; Received in revised form 25 September 2025; Accepted 8 November 2025
Available online 11 November 2025

each task loss to make all task losses have a similar scale. This is non-parametric and can recover the loss transformation in IMTL-L (Liu et al., 2021b). We find that the logarithm transformation also benefits existing gradient balancing methods. Second, we normalize all task gradients to the same magnitude as the maximum gradient norm. This is training-free and guarantees all gradients' magnitude are the same compared with GradNorm (Chen et al., 2018b). Empirically, we find that the magnitude of normalized gradients plays an important role in performance, and setting it as the maximum gradient norm among tasks performs the best. Extensive experiments are performed on a number of benchmark datasets. Results demonstrate that DB-MTL consistently outperforms the current state-of-the-art.

Our contributions can be summarized as follows:

1. We propose DB-MTL, a novel dual-balancing approach that simultaneously addresses both loss-scale and gradient-magnitude imbalances in multi-task learning through:
   - A parameter-free logarithm transformation for loss-scale balancing that effectively equalizes loss scales across tasks;
   - A maximum-norm gradient normalization strategy that rescales all task gradients to comparable magnitudes for balanced model updates.
2. We conduct extensive experiments across diverse benchmarks demonstrating that DB-MTL consistently outperforms state-of-the-art MTL methods.

Notations. For clarity, we summarize the key notations used throughout this paper. We use T to denote the number of tasks, $\mathcal{D}_t$ for the training dataset of task t, $\boldsymbol{\theta}$ and $\{\boldsymbol{\psi}_t\}_{t=1}^T$ for task-sharing and task-specific parameters respectively, γ_t for task weights, and ℓ_t for the loss function of task t. $\mathbf{g}_{t,k}$ and $\tilde{\mathbf{g}}_k$ represent the gradient and aggregated gradient at iteration k, with α_k as the scaling factor.

2. Related works

In an MTL problem with T tasks, we aim to learn a model from $\{\mathcal{D}_t\}_{t=1}^T$, where $\mathcal{D}_t$ is the training dataset of task t. The MTL model parameters can be divided into two parts: (i) task-sharing parameter $\boldsymbol{\theta}$, and (ii) task-specific parameters $\{\boldsymbol{\psi}_t\}_{t=1}^T$. For example, in computer vision tasks, $\boldsymbol{\theta}$ usually represents a feature encoder (e.g., *ResNet* (He et al., 2016)) to extract common features among tasks, while $\boldsymbol{\psi}_t$ corresponds to the task-specific output module (e.g., a fully-connected layer). For parameter efficiency, $\boldsymbol{\theta}$ contains most of the MTL model parameters, and is crucial to the performance.

Let $\ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$ be the loss on task t's data $\mathcal{D}_t$ using parameter $(\boldsymbol{\theta}, \boldsymbol{\psi}_t)$. The training objective of MTL is $\sum_{t=1}^T \gamma_t \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$, where γ_t is the weight for task t. Equal weighting (EW) (Zhang & Yang, 2022) is a simple MTL approach that sets $\gamma_t = 1$ for all tasks. However, EW usually suffers from the task balancing problem in which some tasks have unsatisfactory performance (Standley et al., 2020). To improve its performance, many other MTL methods have been proposed to dynamically tune the task weights $\{\gamma_t\}_{i=1}^T$ during training. They can be categorized as loss balancing, gradient balancing, or hybrid balancing.

2.1. Loss balancing methods

This approach weights the task losses with $\{\gamma_t\}_{i=1}^T$ that are computed dynamically. $\{\gamma_t\}_{i=1}^T$ affect the update of both the task-sharing parameter $\boldsymbol{\theta}$ and task-specific parameter $\{\boldsymbol{\psi}_t\}_{t=1}^T$. They can be set based on measures such as homoscedastic uncertainty (Kendall et al., 2018), learning speed (Liu et al., 2019a), validation performance (Ye et al., 2024a, 2021), and improvable gap (Dai et al., 2023). Alternatively, IMTL-L (Liu et al., 2021b) encourages the weighted losses $\{\gamma_t \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)\}_{t=1}^T$ to have similar loss scale across all tasks by transforming each loss $\ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$ as $e^{s_t}\ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t) - s_t$, where $\{s_t\}_{t=1}^T$ are learnable parameters and obtained by gradient descent at each iteration.

2.2. Gradient balancing methods

The update of the task-sharing parameter $\boldsymbol{\theta}$ depends on all task gradients $\{\nabla_{\boldsymbol{\theta}}\ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)\}_{t=1}^T$. Thus, gradient balancing methods aim to aggregate all task gradients in different manners. For example, MGDA (Sener & Koltun, 2018) formulates MTL as a multi-objective optimization problem and selects the aggregated gradient with the minimum norm (Désidéri, 2012). CAGrad (Liu et al., 2021a) improves MGDA by constraining the aggregated gradient to be around the average gradient. MoCo (Fernando et al., 2023) mitigates the bias in MGDA by introducing a momentum-like gradient estimate and a regularization term. GradNorm (Chen et al., 2018b) learns task weights to scale the task gradients to similar magnitudes. PCGrad (Yu et al., 2020) projects the gradient of one task onto the normal plane of the other if their gradients conflict. GradVac (Wang et al., 2021) aligns the gradients regardless of whether the gradients conflict or not. GradDrop (Chen et al., 2020) randomly masks out gradient values with inconsistent signs. IMTL-G (Liu et al., 2021b) learns task weights to enforce the aggregated gradient to have equal projections on each task gradient. Nash-MTL (Navon et al., 2022) formulates gradient aggregation as a Nash bargaining game.

For most gradient balancing methods (such as PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), MoCo (Fernando et al., 2023), GradDrop (Chen et al., 2020), and IMTL-G (Liu et al., 2021b)), the task weight γ_t only affects update of the task-sharing parameter $\boldsymbol{\theta}$, while in some other gradient balancing methods (such as MGDA (Sener & Koltun, 2018), GradNorm (Chen et al., 2018b), and Nash-MTL (Navon et al., 2022)), the task weight γ_t affects the update of both the task-sharing and task-specific parameters.

2.3. Hybrid balancing methods

As loss balancing and gradient balancing are complementary, these two types of methods can be combined to achieve better performance. In this approach, the task weight γ_t is obtained as the product of the loss and gradient balancing weights. For example, the first hybrid balancing method IMTL (Liu et al., 2021b) combines IMTL-L with IMTL-G. Subsequently, various combinations (Dai et al., 2023; Lin et al., 2022; Liu et al., 2022) of loss/gradient balancing methods demonstrate performance improvements. In this paper, we propose DB-MTL that combines the logarithm transformation (for loss balancing) and the maximum-norm gradient normalization (for gradient balancing).

3. Proposed method

In this section, we alleviate the task balancing problem from both the loss and gradient perspectives. First, we balance all loss scales by performing logarithm transformation on each task's loss (Section 3.1). Next, we achieve gradient-magnitude balancing by normalizing each task's gradient to the same magnitude as the maximum gradient norm (Section 3.2). The procedure, called DB-MTL (Dual-Balancing Multi-Task Learning), is shown in Algorithm 1.

3.1. Scale-balancing loss transformation

Tasks with different types of loss functions usually have different scales, leading to the task balancing problem. For example, in the *NYUv2* dataset (Silberman et al., 2012), the cross-entropy loss, L_1 loss, and cosine loss are used as the loss functions of the semantic segmentation, depth estimation, and surface normal prediction tasks, respectively. As observed in Navon et al. (2022), Standley et al. (2020), Yu et al. (2020) and also in our experimental results in Tables 1 and 6, surface normal prediction is affected by the other two tasks (semantic segmentation and depth estimation), causing MTL methods like EW to perform unsatisfactorily.

When prior knowledge of the loss scales is available, we can choose $\{s_t^\star\}_{t=1}^T$ such that $\{s_t^\star \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)\}_{t=1}^T$ have the same scale, and then

Algorithm 1 Dual-balancing multi-task learning.

Require: numbers of iterations K, learning rate η, tasks $\{\mathcal{D}_t\}_{t=1}^T$, $\epsilon = 10^{-8}$, β;
1: randomly initialize $\boldsymbol{\theta}_0, \{\boldsymbol{\psi}_{t,0}\}_{t=1}^T$;
2: initialize $\hat{\mathbf{g}}_{t,-1} = \mathbf{0}$, for all t;
3: **for** $k = 0, \dots, K-1$ **do**
4: **for** $t = 1, \dots, T$ **do**
5: sample a mini-batch dataset $\mathcal{B}_{t,k}$ from $\mathcal{D}_t$;
6: $\mathbf{g}_{t,k} = \nabla_{\boldsymbol{\theta}_k} \log(\ell_t(\mathcal{B}_{t,k}; \boldsymbol{\theta}_k, \boldsymbol{\psi}_{t,k}) + \epsilon)$;
7: compute $\hat{\mathbf{g}}_{t,k} = \beta \hat{\mathbf{g}}_{t,k-1} + (1-\beta)\mathbf{g}_{t,k}$;
8: **end for**
9: compute $\tilde{\mathbf{g}}_k = \alpha_k \sum_{t=1}^T \frac{\hat{\mathbf{g}}_{t,k}}{\|\hat{\mathbf{g}}_{t,k}\|_2 + \epsilon}$, where $\alpha_k = \max_{1 \le t \le T} \|\hat{\mathbf{g}}_{t,k}\|_2$;
10: update task-sharing parameter by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \tilde{\mathbf{g}}_k$;
11: **for** $t = 1, \dots, T$ **do**
12: $\boldsymbol{\psi}_{t,k+1} = \boldsymbol{\psi}_{t,k} - \eta \nabla_{\boldsymbol{\psi}_{t,k}} \log(\ell_t(\mathcal{B}_{t,k}; \boldsymbol{\theta}_k, \boldsymbol{\psi}_{t,k}) + \epsilon)$;
13: **end for**
14: **end for**
15: **Return** $\boldsymbol{\theta}_K, \{\boldsymbol{\psi}_{t,K}\}_{t=1}^T$.

minimize the total loss $\sum_{t=1}^T s_t^\star \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$. Previous methods (Kendall et al., 2018; Liu et al., 2021b, 2019a; Ye et al., 2021) implicitly learn $\{s_t^\star\}_{t=1}^T$ when learning the task weights $\{\gamma_t\}_{t=1}^T$. However, obviously the optimal $\{s_t^\star\}_{t=1}^T$ cannot be obtained during training.

Without the availability of $\{s_t^\star\}_{t=1}^T$, the logarithm transformation can be used to alleviate the loss scale problem. Specifically, we transform each task's loss $\ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$ to $\log \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$, and then minimize $\sum_{t=1}^T \log \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)$. Since $\log(\cdot)$ can compress the range of its input, it can reduce the loss scale gap between different tasks.

IMTL-L (Liu et al., 2021b) tackles the loss scale issue using a transformed loss $e^{s_t} \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t) - s_t$, where s_t is a learnable parameter for the t-th task and approximately solved by one-step gradient descent at every iteration. The following Proposition 1 shows that IMTL-L is equivalent to the logarithm transformation when s_t is the exact minimizer in each iteration.

Proposition 1. *For* $x > 0$, $\log(x) = \min_s e^s x - s - 1$.

Proof. Define an auxiliary function $f(s) = e^s x - s - 1$. It is easy to show that $\frac{\mathrm{d}f(s)}{\mathrm{d}s} = e^s x - 1$ and $\frac{\mathrm{d}^2 f(s)}{\mathrm{d}s^2} = e^s x > 0$. Thus, $f(s)$ is convex. By the first-order optimal condition (Boyd & Vandenberghe, 2004), let $e^{s^\star} x - 1 = 0$, the global minimizer is solved as $s^\star = -\log(x)$. Therefore, $f(s^\star) = e^{s^\star} x - s^\star - 1 = e^{-\log(x)} x + \log(x) - 1 = \log(x)$, where we finish the proof. □

Compared to IMTL-L, the logarithm transformation does not require additional parameters and computational cost during training. Thus, the logarithm transformation is simpler and more effective than IMTL-L.

3.2. Magnitude-balancing gradient normalization

In addition to the task losses, task gradients also suffer from the scale issue. As the update direction of $\boldsymbol{\theta}$ is obtained by uniformly averaging all task gradients, it may be dominated by the large task gradients, causing sub-optimal performance (Liu et al., 2021a; Yu et al., 2020).

A simple approach is to normalize task gradients to the same magnitude. As computing the batch gradient is computationally expensive, mini-batch stochastic gradient descent is often used in practice. Specifically, at iteration k, we sample a mini-batch $\mathcal{B}_{t,k}$ from $\mathcal{D}_t$ for the t-th task (step 5 in Algorithm 1) and compute the mini-batch gradient $\mathbf{g}_{t,k} = \nabla_{\boldsymbol{\theta}_k} \log \ell_t(\mathcal{B}_{t,k}; \boldsymbol{\theta}_k, \boldsymbol{\psi}_{t,k})$ (step 6 in Algorithm 1). Exponential moving average (EMA), which is popularly used in adaptive gradient methods (e.g., RMSProp (Tieleman & Hinton, 2012), AdaDelta (Zeiler, 2012), and Adam (Kingma & Ba, 2015)), is used to estimate $\mathbb{E}_{\mathcal{B}_{t,k} \sim \mathcal{D}_t} \nabla_{\boldsymbol{\theta}_k} \log \ell_t(\mathcal{B}_{t,k}; \boldsymbol{\theta}_k, \boldsymbol{\psi}_{t,k})$ dynamically (step 7 in Algorithm 1) as

$$\hat{\mathbf{g}}_{t,k} = \beta \hat{\mathbf{g}}_{t,k-1} + (1-\beta)\mathbf{g}_{t,k}, \tag{1}$$

where $\beta \in (0, 1)$ controls the forgetting rate. After obtaining the task gradients $\{\hat{\mathbf{g}}_{t,k}\}_{t=1}^T$, we normalize them to have the same ℓ_2-norm, and compute the aggregated gradient as

$$\tilde{\mathbf{g}}_k = \alpha_k \sum_{t=1}^T \frac{\hat{\mathbf{g}}_{t,k}}{\|\hat{\mathbf{g}}_{t,k}\|_2}, \tag{2}$$

where α_k is a scaling factor controlling the update magnitude. After normalization, all tasks contribute with comparable magnitudes to the update direction.

The choice of α_k is critical in alleviating the task balancing problem. Intuitively, when some tasks have large gradient norms and others have small gradient norms, the first group of tasks has not yet converged while the second group of tasks has almost converged. The current model $\boldsymbol{\theta}_k$ is undesirable and can cause the task balancing problem as not all tasks have converged. Hence, α_k should be large to escape this undesirable solution. On the other hand, when all task gradient norms are small, model $\boldsymbol{\theta}_k$ is close to a stationary solution for all tasks, and α_k should be small so that the solution will no longer change. Thus, we choose $\alpha_k = \max_{1 \le t \le T} \|\hat{\mathbf{g}}_{t,k}\|_2$, i.e., α_k is small if and only if all the task gradient norms are small.

After scaling the losses and gradients, the task-sharing parameter is updated as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \tilde{\mathbf{g}}_k$ (step 10), where $\eta > 0$ is the learning rate. For the task-specific parameters $\{\boldsymbol{\psi}_{t,k}\}_{t=1}^T$, as the update of each of them only depends on the corresponding task gradient separately, their gradients do not suffer from the gradient scaling issue. Hence, the update for task-specific parameters is simply $\boldsymbol{\psi}_{t,k+1} = \boldsymbol{\psi}_{t,k} - \eta \nabla_{\boldsymbol{\psi}_{t,k}} \log \ell_t(\mathcal{B}_{t,k}; \boldsymbol{\theta}_k, \boldsymbol{\psi}_{t,k})$ (steps 11–13).

GradNorm (Chen et al., 2018b) also aims to learn $\{\gamma_t\}_{t=1}^T$ so that the scaled gradients have similar norms. However, it has two problems. First, alternating the updates of model parameters and task weights cannot guarantee all task gradients have the same magnitude in each iteration. Second, as will be seen from Fig. 6 inSection 4.5, the choice of the update magnitude α_k can significantly affect performance. However, this is not considered in GradNorm.

4. Experiments

In this section, we empirically evaluate the proposed DB-MTL on a number of tasks, including scene understanding (Section 4.1), molecular property prediction (Section 4.2), and image classification (Section 4.3).

4.1. Evaluation on scene understanding

Datasets. Following RLW (Lin et al., 2022), CAGrad (Liu et al., 2021a), and Nash-MTL (Navon et al., 2022), the following two scene understanding datasets are used: *NYUv2* (Silberman et al., 2012), which is an indoor scene understanding dataset. It has 3 tasks (13-class semantic segmentation, depth estimation, and surface normal prediction) with 795 training and 654 testing images. *Cityscapes* (Cordts et al., 2016), which is an urban scene understanding dataset. It has 2 tasks (7-class semantic segmentation and depth estimation) with 2,975 training and 500 testing images.

Baselines. The proposed DB-MTL is compared with a number of MTL baselines, including (i) equal weighting (EW) (Zhang & Yang, 2022); (ii) GLS (Chennupati et al., 2019), which minimizes the geometric mean loss $\sqrt[T]{\prod_{t=1}^T \ell_t(\mathcal{D}_t; \boldsymbol{\theta}, \boldsymbol{\psi}_t)}$; (iii) RLW (Lin et al., 2022), in which the task weights are sampled from the standard normal distribution; (iv) *loss balancing* methods including UW (Kendall et al., 2018), DWA (Liu et al., 2019a), IMTL-L (Liu et al., 2021b), and IGBv2 (Dai et al., 2023); (v) *gradient balancing* methods including MGDA (Sener & Koltun, 2018), GradNorm (Chen et al., 2018b), PCGrad (Yu et al., 2020), GradDrop

Table 1
Performance on *NYUv2* with 3 tasks. ↑ (↓) means the higher (lower) the result, the better the performance. The best and second best results are marked in **bold** and underline, respectively .

	Segmentation		Depth Estimation		Surface Normal Prediction					Δ_p↑
	mIoU↑	PAcc↑	AErr↓	RErr↓	Angle Distance		Within $t°$			
					Mean↓	MED↓	**11.25**↑	22.5↑	30↑	
STL	53.50	75.39	0.3926	0.1605	21.99	**15.16**	**39.04**	**65.00**	75.16	0.00
EW	53.93	75.53	0.3825	0.1577	23.57	17.01	35.04	60.99	72.05	$-1.78_{\pm 0.45}$
GLS	54.59	**76.06**	0.3785	**0.1555**	22.71	16.07	36.89	63.11	73.81	$+0.30_{\pm 0.30}$
RLW	54.04	75.58	0.3827	0.1588	23.07	16.49	36.12	62.08	72.94	$-1.10_{\pm 0.40}$
UW	54.29	75.64	0.3815	0.1583	23.48	16.92	35.26	61.17	72.21	$-1.52_{\pm 0.39}$
DWA	54.06	75.64	0.3820	0.1564	23.70	17.11	34.90	60.74	71.81	$-1.71_{\pm 0.25}$
IMTL-L	53.89	75.54	0.3834	0.1591	23.54	16.98	35.09	61.06	72.12	$-1.92_{\pm 0.25}$
IGBv2	**54.61**	76.00	0.3817	0.1576	22.68	15.98	37.14	63.25	73.87	$+0.05_{\pm 0.29}$
MGDA	53.52	74.76	0.3852	0.1566	22.74	16.00	37.12	63.22	73.84	$-0.64_{\pm 0.25}$
GradNorm	53.91	75.38	0.3842	0.1571	23.17	16.62	35.80	61.90	72.84	$-1.24_{\pm 0.15}$
PCGrad	53.94	75.62	0.3804	0.1578	23.52	16.93	35.19	61.17	72.19	$-1.57_{\pm 0.44}$
GradDrop	53.73	75.54	0.3837	0.1580	23.54	16.96	35.17	61.06	72.07	$-1.85_{\pm 0.39}$
GradVac	54.21	75.67	0.3859	0.1583	23.58	16.91	35.34	61.15	72.10	$-1.75_{\pm 0.39}$
IMTL-G	53.01	75.04	0.3888	0.1603	23.08	16.43	36.24	62.23	73.06	$-1.89_{\pm 0.54}$
CAGrad	53.97	75.54	0.3885	0.1588	22.47	15.71	37.77	63.82	74.30	$-0.27_{\pm 0.35}$
MTAdam	52.67	74.86	0.3873	0.1583	23.26	16.55	36.00	61.92	72.74	$-1.97_{\pm 0.23}$
Nash-MTL	53.41	74.95	0.3867	0.1612	22.57	15.94	37.30	63.40	74.09	$-1.01_{\pm 0.13}$
MetaBalance	53.92	75.57	0.3901	0.1594	22.85	16.16	36.72	62.91	73.62	$-1.06_{\pm 0.17}$
MoCo	52.25	74.56	0.3920	0.1622	22.82	16.24	36.58	62.72	73.49	$-2.25_{\pm 0.51}$
Aligned-MTL	52.94	75.00	0.3884	0.1570	22.65	16.07	36.88	63.18	73.94	$-0.98_{\pm 0.56}$
IMTL	53.63	75.44	0.3868	0.1592	22.58	15.85	37.44	63.52	74.09	$-0.57_{\pm 0.24}$
DB-MTL (**ours**)	53.92	75.60	**0.3768**	0.1557	**21.97**	15.37	38.43	64.81	**75.24**	$+\mathbf{1.15}_{\pm 0.16}$

(Chen et al., 2020), GradVac (Wang et al., 2021), IMTL-G (Liu et al., 2021b), CAGrad (Liu et al., 2021a), MTAdam (Malkiel & Wolf, 2021), Nash-MTL (Navon et al., 2022), MetaBalance (He et al., 2022), MoCo (Fernando et al., 2023), and Aligned-MTL (Senushkin et al., 2023); and (vi) *hybrid balancing* method IMTL (Liu et al., 2021b). For comparison, we also include the *single-task learning* (STL) baseline, which learns each task separately.

All methods are implemented based on the open-source `LibMTL` library (Lin & Zhang, 2023). For all MTL methods, the hard-parameter sharing (*HPS*) pattern (Caruana, 1993) is used, which consists of a task-sharing feature encoder and T task-specific heads. For the proposed DB-MTL, following MoCo (Fernando et al., 2023), we perform grid search for β over $\{0.1, 0.5, 0.9, \frac{0.1}{k^{0.5}}, \frac{0.5}{k^{0.5}}, \frac{0.9}{k^{0.5}}\}$ for each dataset, where k is the number of iterations.

Implementation Details. Following RLW (Lin et al., 2022), we use the *DeepLabV3+* network (Chen et al., 2018a), which contains a *ResNet-50* network with dilated convolutions pre-trained on the *ImageNet* dataset (Deng et al., 2009) as the shared encoder and the *Atrous Spatial Pyramid Pooling* (Chen et al., 2018a) module as task-specific head. We train the model for 200 epochs by using the Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-4} and weight decay 10^{-5}. The learning rate is halved to 5×10^{-5} after 100 epochs. The cross-entropy loss $\ell_{seg} = -\frac{1}{N\times H\times W}\sum_{n=1}^{N}\sum_{i=1}^{H\times W}\sum_{c=1}^{C} y_{n,i,c}\log(\hat{y}_{n,i,c})$, L_1 loss $\ell_{depth} = \frac{1}{N\times H\times W}\sum_{n=1}^{N}\sum_{i=1}^{H\times W}|d_{n,i}-\hat{d}_{n,i}|$, and cosine loss $\ell_{normal} = \frac{1}{N\times H\times W}\sum_{n=1}^{N}\sum_{i=1}^{H\times W}(1-\frac{\mathbf{n}_{n,i}\cdot\hat{\mathbf{n}}_{n,i}}{||\mathbf{n}_{n,i}||\cdot||\hat{\mathbf{n}}_{n,i}||})$ are used as the loss functions of the semantic segmentation, depth estimation, and surface normal prediction tasks, respectively, where N is the batch size, H and W are the height and width of the image, $y_{n,i,c}$ and $\hat{y}_{n,i,c}$ are the ground truth label and predicted probability for pixel i in image n and class c, $d_{n,i}$ and $\hat{d}_{n,i}$ are the ground truth and predicted depth values for pixel i in image n, and $\mathbf{n}_{n,i}$ and $\hat{\mathbf{n}}_{n,i}$ are the ground truth and predicted normal vectors for pixel i in image n. For *NYUv2*, the images are resized to 288×384, and the batch size is 8. For *Cityscapes*, the images are resized to 128×256, and the batch size is 64. Each experiment is repeated three times.

Performance Evaluation. Following DWA (Liu et al., 2019a) and RLW (Lin et al., 2022), we use (i) the mean intersection over union (mIoU) and class-wise pixel accuracy (PAcc) for semantic segmentation; (ii) relative error (RErr) and absolute error (AErr) for depth estimation; (iii) mean and median angle errors, and percentage of normals within $t°$ (where $t = 11.25, 22.5, 30$) for surface normal prediction. Following (Lin et al., 2022; Maninis et al., 2019; Vandenhende et al., 2021), we report the relative performance improvement of an MTL method $\mathcal{A}$ over STL, averaged over all the metrics above, i.e.,

$$\Delta_p(\mathcal{A}) = \frac{1}{T}\sum_{t=1}^{T}\Delta_{p,t}(\mathcal{A}), \tag{3}$$

where T is the number of tasks and

$$\Delta_{p,t}(\mathcal{A}) = 100\,\%\times\frac{1}{N_t}\sum_{i=1}^{N_t}(-1)^{s_{t,i}}\frac{M_{t,i}^{\mathcal{A}}-M_{t,i}^{\mathrm{STL}}}{M_{t,i}^{\mathrm{STL}}}, \tag{4}$$

where N_t is the number of metrics for task t, $M_{t,i}^{\mathcal{A}}$ is the ith metric value of method $\mathcal{A}$ on task t, and $s_{t,i}$ is 0 if a larger value indicates better performance for the ith metric on task t, and 1 otherwise.

Performance Results. Table 1 shows the results on *NYUv2*. As can be seen, the proposed DB-MTL performs the best in terms of average Δ_p. Note that most of the MTL baselines perform better than STL on semantic segmentation and depth estimation, but have a large drop on the surface normal prediction task, suffering from the task balancing problem. Only the proposed DB-MTL has comparable performance with STL on the surface normal prediction task and maintains superiority on the other tasks. Table 2 shows the results on *Cityscapes*. As can be seen, DB-MTL again achieves the best in terms of average Δ_p. Note that all MTL baselines perform worse than STL in terms of average Δ_p and only the proposed DB-MTL outperforms STL on all tasks.

4.2. Evaluation on molecular property prediction

Dataset. Following Nash-MTL (Navon et al., 2022), we use the *QM9* (Ramakrishnan et al., 2014) dataset, which is for molecular property

Table 2
Performance on *Cityscapes* with 2 tasks. ↑ (↓) indicates that the higher (lower) the result, the better the performance. The best and second best results are highlighted in **bold** and underline, respectively.

	Segmentation		Depth Estimation		Δ_p↑
	mIoU↑	PAcc↑	AErr↓	RErr↓	
STL	69.06	91.54	0.01282	43.53	0.00
EW	68.93	91.58	0.01315	45.90	$-2.05_{\pm 0.56}$
GLS	68.69	91.45	0.01280	44.13	$-0.39_{\pm 1.06}$
RLW	69.03	91.57	0.01343	44.77	$-1.91_{\pm 0.21}$
UW	69.03	91.61	0.01338	45.89	$-2.45_{\pm 0.68}$
DWA	68.97	91.58	0.01350	45.10	$-2.24_{\pm 0.28}$
IMTL-L	68.98	91.59	0.01340	45.32	$-2.15_{\pm 0.88}$
IGBv2	68.44	91.31	0.01290	45.03	$-1.31_{\pm 0.61}$
MGDA	69.05	91.53	0.01280	44.07	$-0.19_{\pm 0.30}$
GradNorm	68.97	91.60	0.01320	44.88	$-1.55_{\pm 0.70}$
PCGrad	68.95	91.58	0.01342	45.54	$-2.36_{\pm 1.17}$
GradDrop	68.85	91.54	0.01354	44.49	$-2.02_{\pm 0.74}$
GradVac	68.98	91.58	0.01322	46.43	$-2.45_{\pm 0.54}$
IMTL-G	69.04	91.54	0.01280	44.30	$-0.46_{\pm 0.67}$
CAGrad	68.95	91.60	0.01281	45.04	$-0.87_{\pm 0.88}$
MTAdam	68.43	91.26	0.01340	45.62	$-2.74_{\pm 0.20}$
Nash-MTL	68.88	91.52	**0.01265**	45.92	$-1.11_{\pm 0.21}$
MetaBalance	69.02	91.56	0.01270	45.91	$-1.18_{\pm 0.58}$
MoCo	**69.62**	**91.76**	0.01360	45.50	$-2.40_{\pm 1.50}$
Aligned-MTL	69.00	91.59	0.01270	44.54	$-0.43_{\pm 0.44}$
IMTL	69.07	91.55	0.01280	44.06	$-0.32_{\pm 0.10}$
DB-MTL (**ours**)	69.17	91.56	0.01280	**43.46**	$+\mathbf{0.20}_{\pm 0.40}$

Table 3
Performance (MAE) on *QM9* with 11 tasks. ↑ (↓) indicates that the higher (lower) the result, the better the performance. The best and second best results are highlighted in **bold** and underline, respectively.

	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	Δ_p↑
STL	**0.062**	**0.192**	**58.82**	**51.95**	**0.529**	4.52	63.69	60.83	68.33	60.31	**0.069**	**0.00**
EW	0.096	0.286	67.46	82.80	4.655	12.4	128.3	128.8	129.2	125.6	0.116	$-146.3_{\pm 7.86}$
GLS	0.332	0.340	143.1	131.5	1.023	**4.45**	**53.35**	**53.79**	**53.78**	**53.34**	0.111	$-81.16_{\pm 15.5}$
RLW	0.112	0.331	74.59	90.48	6.015	15.6	156.0	156.8	157.3	151.6	0.133	$-200.9_{\pm 13.4}$
UW	0.336	0.382	155.1	144.3	0.965	4.58	61.41	61.79	61.83	61.40	0.116	$-92.35_{\pm 13.9}$
DWA	0.103	0.311	71.55	87.21	4.954	13.1	134.9	135.8	136.3	132.0	0.121	$-160.9_{\pm 16.7}$
IMTL-L	0.277	0.355	150.1	135.2	0.946	4.46	58.08	58.43	58.46	58.06	0.110	$-77.06_{\pm 11.1}$
IGBv2	0.235	0.377	132.3	139.9	2.214	5.90	64.55	65.06	65.12	64.28	0.121	$-99.86_{\pm 10.4}$
MGDA	0.181	0.325	118.6	92.45	2.411	5.55	103.7	104.2	104.4	103.7	0.110	$-103.0_{\pm 8.62}$
GradNorm	0.114	0.341	67.17	84.66	7.079	14.6	173.2	173.8	174.4	168.9	0.147	$-227.5_{\pm 1.85}$
PCGrad	0.104	0.293	75.29	88.99	3.695	8.67	115.6	116.0	116.2	113.8	0.109	$-117.8_{\pm 3.97}$
GradDrop	0.114	0.349	75.94	94.62	5.315	15.8	155.2	156.1	156.6	151.9	0.136	$-191.4_{\pm 9.62}$
GradVac	0.100	0.299	68.94	84.14	4.833	12.5	127.3	127.8	128.1	124.7	0.117	$-150.7_{\pm 7.41}$
IMTL-G	0.670	0.978	220.7	249.7	19.48	55.6	1109	1117	1123	1043	0.392	$-1250_{\pm 90.9}$
CAGrad	0.107	0.296	75.43	88.59	2.944	6.12	93.09	93.68	93.85	92.32	0.106	$-87.25_{\pm 1.51}$
MTAdam	0.593	1.352	232.3	419.0	24.31	69.7	1060	1067	1070	1007	0.627	$-1403_{\pm 203}$
Nash-MTL	0.115	0.263	85.54	86.62	2.549	5.85	83.49	83.88	84.05	82.96	0.097	$-73.92_{\pm 2.12}$
MetaBalance	0.090	0.277	70.50	78.43	4.192	11.2	113.7	114.2	114.5	111.7	0.110	$-125.1_{\pm 7.98}$
MoCo	0.489	1.096	189.5	247.3	34.33	64.5	754.6	760.1	761.6	720.3	0.522	$-1314_{\pm 65.2}$
Aligned-MTL	0.123	0.295	98.07	94.56	2.397	5.90	86.42	87.42	87.19	86.75	0.106	$-80.58_{\pm 4.18}$
IMTL	0.138	0.344	106.1	102.9	2.595	7.84	102.5	103.0	103.2	100.8	0.110	$-104.3_{\pm 11.7}$
DB-MTL (**ours**)	0.112	0.264	89.26	86.59	2.429	5.41	60.33	60.78	60.80	60.59	0.098	$-58.10_{\pm 3.89}$

prediction with 11 tasks. Each task performs regression on one property. We use the same split as in Nash-MTL (Navon et al., 2022): 110,000 for training, 10,000 for validation, and 10,000 for testing.

Implementation Details. The experimental setups are the same with Nash-MTL (Navon et al., 2022). Specifically, a graph neural network (Gilmer et al., 2017) is used as the shared encoder, and a linear layer is used as the task-specific head. The targets of each task are normalized to have zero mean and unit standard deviation. The batch size and training epoch are set to 128 and 300, respectively. The Adam optimizer (Kingma & Ba, 2015) with the learning rate 0.001 is used for training, and the ReduceLROnPlateau scheduler (Paszke et al., 2019) is used to reduce the learning rate once Δ_p on the validation dataset stops improving. The mean squared error (MSE) $\ell_{mse} = \frac{1}{N}\sum_{n=1}^{N}(p_n - \hat{p}_n)^2$ is used as the loss function for each molecular property prediction task, where N is the batch size, p_n and $\hat{p}_n$ are the ground truth and predicted property values for sample n respectively. Mean absolute error (MAE) is used for performance evaluation. Each experiment is repeated three times.

Performance Results. Table 3 shows each task's testing MAE and overall performance Δ_p (Eq. (3)) on *QM9*, using the same set of baselines as in Section 4.1. Note that *QM9* is a challenging dataset in MTL and none of

Table 4
Classification accuracy (%) on *Office-31* and *Office-Home*. ↑ indicates that the higher the result, the better the performance. The best and second best results are highlighted in **bold** and underline, respectively. Results of MoCo are from Fernando et al. (2023).

	Office-31					*Office-Home*					
	Amazon	DSLR	Webcam	Avg↑	Δ_p↑	Artistic	Clipart	Product	Real	Avg↑	Δ_p↑
STL	**86.61**	95.63	96.85	93.03	0.00	65.59	**79.60**	**90.47**	80.00	78.91	0.00
EW	83.53	97.27	96.85	$92.55_{\pm 0.62}$	$-0.61_{\pm 0.67}$	65.34	78.04	89.80	79.50	$78.17_{\pm 0.37}$	$-0.92_{\pm 0.59}$
GLS	82.84	95.62	96.29	$91.59_{\pm 0.58}$	$-1.63_{\pm 0.61}$	64.51	76.85	89.83	79.56	$77.69_{\pm 0.27}$	$-1.58_{\pm 0.46}$
RLW	83.82	96.99	96.85	$92.55_{\pm 0.89}$	$-0.59_{\pm 0.95}$	64.96	78.19	89.48	**80.11**	$78.18_{\pm 0.12}$	$-0.92_{\pm 0.14}$
UW	83.82	97.27	96.67	$92.58_{\pm 0.84}$	$-0.56_{\pm 0.90}$	65.97	77.65	89.41	79.28	$78.08_{\pm 0.30}$	$-0.98_{\pm 0.46}$
DWA	83.87	96.99	96.48	$92.45_{\pm 0.56}$	$-0.70_{\pm 0.62}$	65.27	77.64	89.05	79.56	$77.88_{\pm 0.28}$	$-1.26_{\pm 0.49}$
IMTL-L	84.04	96.99	96.48	$92.50_{\pm 0.52}$	$-0.63_{\pm 0.58}$	65.90	77.28	89.37	79.38	$77.98_{\pm 0.38}$	$-1.10_{\pm 0.61}$
IGBv2	84.52	98.36	98.05	$93.64_{\pm 0.26}$	$+0.56_{\pm 0.25}$	65.59	77.57	89.79	78.73	$77.92_{\pm 0.21}$	$-1.21_{\pm 0.22}$
MGDA	85.47	95.90	97.03	$92.80_{\pm 0.14}$	$-0.27_{\pm 0.15}$	64.19	77.60	89.58	79.31	$77.67_{\pm 0.20}$	$-1.61_{\pm 0.34}$
GradNorm	83.58	97.26	96.85	$92.56_{\pm 0.87}$	$-0.59_{\pm 0.94}$	66.28	77.86	88.66	79.60	$78.10_{\pm 0.63}$	$-0.90_{\pm 0.93}$
PCGrad	83.59	96.99	96.85	$92.48_{\pm 0.53}$	$-0.68_{\pm 0.57}$	66.35	77.18	88.95	79.50	$77.99_{\pm 0.19}$	$-1.04_{\pm 0.32}$
GradDrop	84.33	96.99	96.30	$92.54_{\pm 0.42}$	$-0.59_{\pm 0.46}$	63.57	77.86	89.23	79.35	$77.50_{\pm 0.23}$	$-1.86_{\pm 0.24}$
GradVac	83.76	97.27	96.67	$92.57_{\pm 0.73}$	$-0.58_{\pm 0.78}$	65.21	77.43	89.23	78.95	$77.71_{\pm 0.19}$	$-1.49_{\pm 0.28}$
IMTL-G	83.41	96.72	96.48	$92.20_{\pm 0.89}$	$-0.97_{\pm 0.95}$	64.70	77.17	89.61	79.45	$77.98_{\pm 0.38}$	$-1.10_{\pm 0.61}$
CAGrad	83.65	95.63	96.85	$92.04_{\pm 0.79}$	$-1.14_{\pm 0.85}$	64.01	77.50	89.65	79.53	$77.73_{\pm 0.16}$	$-1.50_{\pm 0.29}$
MTAdam	85.52	95.62	96.29	$92.48_{\pm 0.87}$	$-0.60_{\pm 0.93}$	62.23	77.86	88.73	77.94	$76.69_{\pm 0.65}$	$-2.94_{\pm 0.85}$
Nash-MTL	85.01	97.54	97.41	$93.32_{\pm 0.82}$	$+0.24_{\pm 0.89}$	66.29	78.76	90.04	**80.11**	$78.80_{\pm 0.52}$	$-0.08_{\pm 0.69}$
MetaBalance	84.21	95.90	97.40	$92.50_{\pm 0.28}$	$-0.63_{\pm 0.30}$	64.01	77.50	89.72	79.24	$77.61_{\pm 0.42}$	$-1.70_{\pm 0.54}$
MoCo	84.33	97.54	98.33	93.39	–	63.38	79.41	90.25	78.70	77.93	–
Aligned-MTL	83.36	96.45	97.04	$92.28_{\pm 0.46}$	$-0.90_{\pm 0.48}$	64.33	76.96	89.87	79.93	$77.77_{\pm 0.70}$	$-1.50_{\pm 0.89}$
IMTL	83.70	96.44	96.29	$92.14_{\pm 0.85}$	$-1.02_{\pm 0.92}$	64.07	76.85	89.65	79.81	$77.59_{\pm 0.29}$	$-1.72_{\pm 0.45}$
DB-MTL (**ours**)	85.12	**98.63**	**98.51**	$\mathbf{94.09}_{\pm 0.19}$	$\mathbf{+1.05}_{\pm 0.20}$	**67.42**	77.89	90.43	80.07	$\mathbf{78.95}_{\pm 0.35}$	$\mathbf{+0.17}_{\pm 0.44}$

Table 5
Effects of each component in DB-MTL on different datasets in terms of Δ_p (Eq. (3)).

loss-scale balancing	gradient-magnitude balancing	*NYUv2*	*Cityscapes*	*Office-31*	*Office-Home*	*QM9*
✗	✗	$-1.78_{\pm 0.45}$	$-2.05_{\pm 0.56}$	$-0.61_{\pm 0.67}$	$-0.92_{\pm 0.59}$	$-146.3_{\pm 7.86}$
✓	✗	$+0.06_{\pm 0.09}$	$-0.38_{\pm 0.39}$	$+0.93_{\pm 0.42}$	$-0.73_{\pm 0.95}$	$-74.40_{\pm 13.2}$
✗	✓	$+0.76_{\pm 0.25}$	$+0.12_{\pm 0.70}$	$+0.01_{\pm 0.39}$	$-0.78_{\pm 0.49}$	$-65.73_{\pm 2.86}$
✓	✓	$\mathbf{+1.15}_{\pm 0.16}$	$\mathbf{+0.20}_{\pm 0.40}$	$\mathbf{+1.05}_{\pm 0.20}$	$\mathbf{+0.17}_{\pm 0.44}$	$\mathbf{-58.10}_{\pm 3.89}$

the MTL methods performs better than STL, as also observed in previous works (Gasteiger et al., 2020; Navon et al., 2022). DB-MTL performs the best among all MTL methods and greatly improves over the second-best MTL method, Nash-MTL, in terms of average Δ_p.

4.3. Evaluation on image classification

Datasets. Following RLW (Lin et al., 2022) and MoCo (Fernando et al., 2023), two image classification datasets are used: *Office-31* (Saenko et al., 2010), which contains 4,110 images from three domains (tasks): Amazon, DSLR, and Webcam. Each task has 31 classes. *Office-Home* (Venkateswara et al., 2017), which contains 15,500 images from four domains (tasks): artistic images, clipart, product images, and real-world images. Each task has 65 object categories collected under office and home settings. We use the commonly-used data split as in RLW (Lin et al., 2022): 60 % for training, 20 % for validation, and 20 % for testing.

Implementation Details. Following RLW (Lin et al., 2022), a *ResNet-18* (He et al., 2016) pre-trained on the *ImageNet* dataset (Deng et al., 2009) is used as a shared encoder, and a linear layer is used as a task-specific head. We resize the input image to 224×224. The batch size and number of training epochs are set to 64 and 100, respectively. The Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-4} and weight decay 10^{-5} is used. For each image classification task, the cross-entropy loss $\ell_{cls} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(\hat{y}_{n,c})$ is used as the loss function, where N is the batch size, $y_{n,c}$ is the ground truth label and $\hat{y}_{n,c}$ is the predicted probability for sample n and class c. Classification accuracy is used for evaluation. Δ_p in Eq. (3) is used as the overall performance metrics. Each experiment is repeated three times.

Performance Results. Table 4 shows the results on *Office-31* and *Office-Home*, using the same set of baselines as in Section 4.1. On *Office-31*, DB-MTL achieves the top testing accuracy on the DSLR and Webcam tasks, and comparable performance on the Amazon task. On *Office-Home*, DB-MTL ranks top two on the Artistic, Product, and Real tasks. On both datasets, DB-MTL achieves the best average testing accuracy and Δ_p, showing its effectiveness and demonstrating that balancing both loss scale and gradient magnitude is effective.

4.4. Effectiveness of loss and gradient balancing components

Ablation Study. DB-MTL has two components: loss-scale balancing (i.e., logarithm transformation) in Section 3.1 and gradient-magnitude balancing in Section 3.2. In this experiment, we perform an ablation study on the effectiveness of each component. We consider the four combinations: (i) use neither loss-scale nor gradient-magnitude balancing (i.e., the EW baseline); (ii) use only loss-scale balancing; (iii) use only gradient-magnitude balancing; (iv) use both loss-scale and gradient-magnitude balancing (i.e., the proposed DB-MTL).

Table 5 shows the Δ_p's of the four combinations on five datasets (*NYUv2*, *Cityscapes*, *Office-31*, *Office-Home*, and *QM9*). As can be seen, on all datasets, both components are beneficial to DB-MTL and combining them achieves the best performance.

Effectiveness of Logarithm Transformation. The logarithm transformation can also be used with other gradient balancing methods. We integrate it into PCGrad (Yu et al., 2020), GradVac (Wang et al., 2021), IMTL-G (Liu et al., 2021b), CAGrad (Liu et al., 2021a), Nash-MTL (Navon et al., 2022), and Aligned-MTL (Senushkin et al., 2023). The experiment is

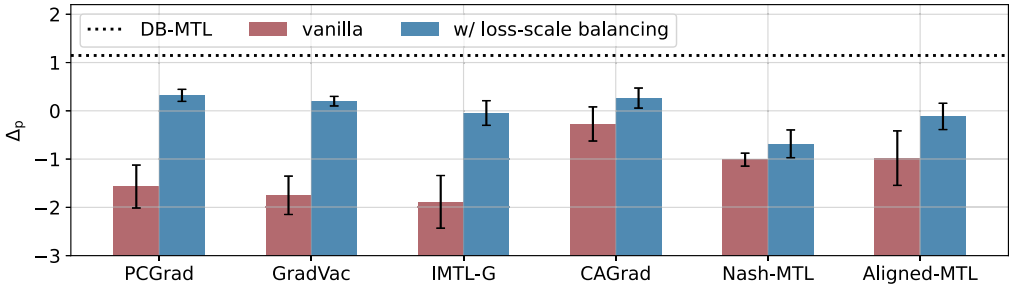


Fig. 1. Performance of existing gradient balancing methods with the loss-scale balancing method (i.e., logarithm transformation) on *NYUv2*. "vanilla" stands for the original method.

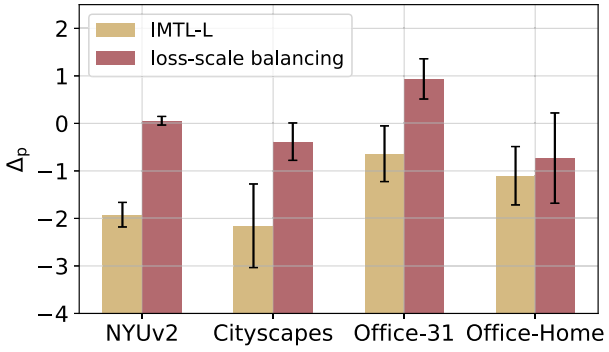


Fig. 2. Comparison of IMTL-L (Liu et al., 2021b) and the loss-scale balancing method on four datasets.

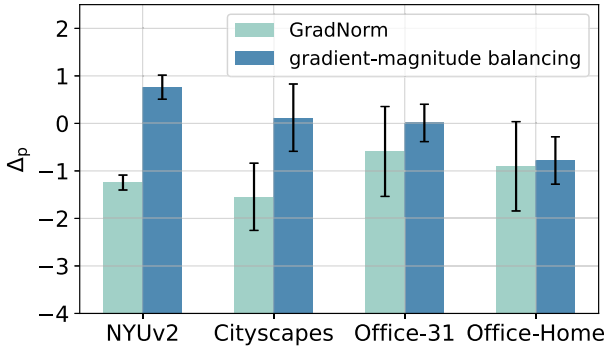
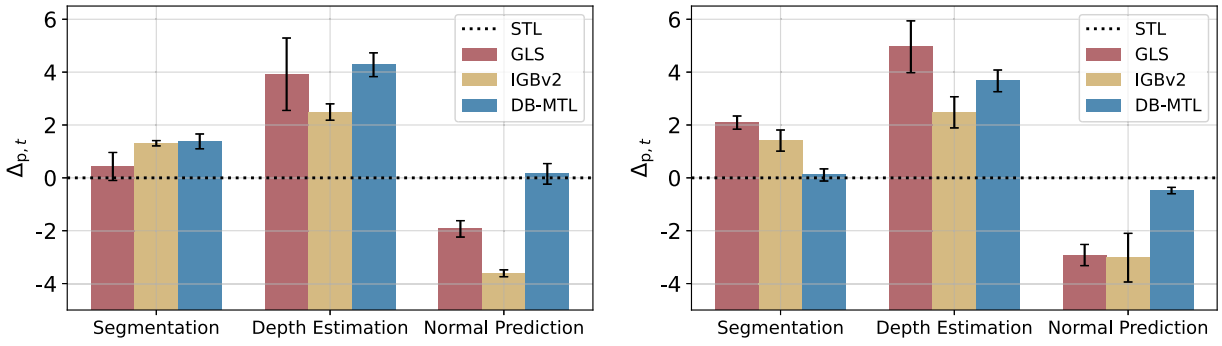


Fig. 3. Comparison of GradNorm (Chen et al., 2018b) and the gradient-magnitude balancing method on four datasets.



(a) *Cross-stitch.* (b) *MTAN.*

Fig. 4. Performance on *NYUv2* for *Cross-stitch* (Misra et al., 2016) and *MTAN* (Liu et al., 2019a) architectures.

performed on *NYUv2* using the setup in Section 4.1. Fig. 1 shows the Δ_p (Eq. (3)). As can be seen, logarithm transformation is consistently beneficial for these gradient balancing methods, showing the effectiveness of logarithm transformation. Moreover, DB-MTL still outperforms these gradient balancing baselines when they are combined with logarithm transformation, demonstrating the effectiveness of the proposed DB-MTL method.

Further to the discussion in Section 3.1, we compare the loss-scale balancing method (i.e., using logarithm transformation only) with IMTL-L (Liu et al., 2021b) on four datasets (*NYUv2*, *Cityscapes*, *Office-31*, and *Office-Home*). As can be seen from Fig. 2, the logarithm transformation consistently outperforms IMTL-L in terms of average Δ_p (Eq. (3)).

Effectiveness of Gradient-Magnitude Balancing. Further to the discussion in Section 3.2, we conduct a comparison between the proposed gradient-magnitude balancing method (i.e., DB-MTL without using logarithm transformation) and GradNorm (Chen et al., 2018b) on four datasets: *NYUv2*, *Cityscapes*, *Office-31*, and *Office-Home*. As can be seen from Fig. 3, the proposed method consistently achieves better performance than GradNorm in terms of average Δ_p on all datasets, demonstrating its effectiveness.

4.5. Sensitivity analysis

Effect of MTL Architecture. The proposed DB-MTL is agnostic to the choice of MTL architectures. In this section, we demonstrate this by evaluating DB-MTL on *NYUv2* using two more MTL architectures: *Cross-stitch* (Misra et al., 2016) and *MTAN* (Liu et al., 2019a). We compare with GLS (Chennupati et al., 2019) and IGBv2 (Dai et al., 2023), which perform well in Table 1. The implementation details are the same as in Section 4.1.

Fig. 4 shows each task's improvement performance $\Delta_{p,t}$. For *Cross-stitch* (Fig. 4(a)), DB-MTL performs the best on all tasks. For *MTAN* (Fig. 4(b)), all the MTL methods (GLS, IGBv2, and DB-MTL) perform better than STL on both semantic segmentation and depth estimation, but only DB-MTL achieves comparable performance as STL on the surface normal prediction task.

Table 6
Performance on the *NYUv2* dataset with *SegNet* network. ↑ (↓) indicates that the higher (lower) the result, the better the performance. The best and second best results are highlighted in **bold** and underline, respectively. Superscripts ♯, §, ‡, and ∗ denote the results are from Fernando et al. (2023), Liu et al. (2021a), Navon et al. (2022), Senushkin et al. (2023), respectively.

	Segmentation		Depth Estimation		Surface Normal Prediction					Δ_p↑
	mIoU↑	PAcc↑	AErr↓	RErr↓	Angle Distance		Within $t°$			
					Mean↓	MED↓	11.25↑	22.5↑	30↑	
STL§	38.30	63.76	0.6754	0.2780	25.01	19.21	30.14	57.20	69.15	0.00
EW§	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	+0.88
GLS	39.78	65.63	0.5318	0.2272	26.13	21.08	26.57	52.83	65.78	+5.15
RLW§	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62	−2.16
UW§	36.87	63.17	0.5446	0.2260	27.04	22.61	23.54	49.05	63.65	+0.91
DWA§	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39	+1.93
IMTL-L	39.78	65.27	0.5408	0.2347	26.26	20.99	26.42	53.03	65.94	+4.39
IGBv2	38.03	64.29	0.5489	0.2301	26.94	22.04	24.77	50.91	64.12	+2.11
MGDA§	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36	−1.66
GradNorm∗	20.09	52.06	0.7200	0.2800	**24.83**	**18.86**	**30.81**	**57.94**	**69.73**	−11.7
PCGrad§	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	+1.11
GradDrop§	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	+2.07
GradVac∗	37.53	64.35	0.5600	0.2400	27.66	23.38	22.83	48.66	62.21	−0.49
IMTL-G§	39.35	65.60	0.5426	0.2256	26.02	21.19	26.20	53.13	66.24	+4.77
CAGrad♯	39.18	64.97	0.5379	0.2229	25.42	20.47	27.37	54.73	67.73	+5.81
MTAdam	39.44	65.73	0.5326	0.2211	27.53	22.70	24.04	49.61	62.69	+3.21
Nash-MTL§	40.13	65.93	0.5261	0.2171	25.26	20.08	28.40	55.47	68.15	+7.65
MetaBalance	39.85	65.13	0.5445	0.2261	27.35	22.66	23.70	49.69	63.09	+2.67
MoCo‡	40.30	66.07	0.5575	**0.2135**	26.67	21.83	25.61	51.78	64.85	+4.85
Aligned-MTL∗	40.82	66.33	0.5300	0.2200	25.19	19.71	28.88	56.23	68.54	+8.16
IMTL	41.19	66.37	0.5323	0.2237	26.06	20.77	26.76	53.48	66.32	+6.45
DB-MTL (**ours**)	**41.42**	**66.45**	**0.5251**	0.2160	25.03	19.50	28.72	56.17	68.73	**+8.91**

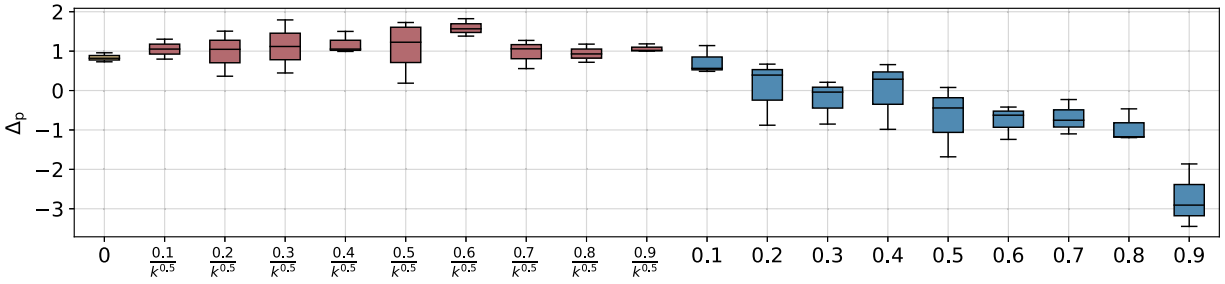


Fig. 5. Effect of EMA's Forgetting Rate β in Eq. (1) on the *Office-31* dataset. k denotes the number of iterations.

Effect of Backbone Network. We perform an experiment to evaluate DB-MTL on *NYUv2* with the *SegNet* network (Badrinarayanan et al., 2017) as the backbone. The implementation details are the same as in Section 4.1, except that the batch size is set to 2 and data augmentation is used (following CAGrad (Liu et al., 2021a)). As can be seen from Table 6, DB-MTL again achieves the best performance in terms of average Δ_p.

Effect of EMA's Forgetting Rate β in Eq. (1). As mentioned in Section 4.1, we perform grid search for β over $\{0.1, 0.5, 0.9, \frac{0.1}{k^{0.5}}, \frac{0.5}{k^{0.5}}, \frac{0.9}{k^{0.5}}\}$, where k is the number of iterations. In this experiment, we run DB-MTL on *Office-31* with $\beta \in \{0, 0.1, 0.2, \ldots, 0.9, \frac{0.1}{k^{0.5}}, \frac{0.2}{k^{0.5}}, \ldots, \frac{0.9}{k^{0.5}}\}$. The experimental setup is the same as in Section 4.3. As can be seen from Fig. 5, the average Δ_p of DB-MTL is insensitive over a large range of β ($\{\frac{0.1}{k^{0.5}}, \frac{0.2}{k^{0.5}}, \ldots, \frac{0.9}{k^{0.5}}\}$), and performs better than DB-MTL without EMA ($\beta = 0$).

Effect of α_k in Eq. (2). In this experiment, we use different settings of α_k in Eq. (2), namely, (i) constant; (ii) minimum of $\{\|\hat{\mathbf{g}}_{t,k}\|_2\}_{t=1}^T$; (iii) maximum of $\{\|\hat{\mathbf{g}}_{t,k}\|_2\}_{t=1}^T$; (iv) average of $\{\|\hat{\mathbf{g}}_{t,k}\|_2\}_{t=1}^T$; (v) median of $\{\|\hat{\mathbf{g}}_{t,k}\|_2\}_{t=1}^T$. Fig. 6 compares the results of these different DB-MTL variants on *NYUv2*. The experimental setup is the same as in Section 4.1. As can be seen, the maximum-norm strategy performs much better in terms of average Δ_p, and thus it is used.

4.6. Analysis of training efficiency

Fig. 7 shows the per-epoch running time of different MTL methods on *NYUv2* dataset. All methods are run for 100 epochs on a single NVIDIA GeForce RTX 3090 GPU and the average running time per epoch is reported. As can be seen, DB-MTL has a similar running time as gradient balancing methods (i.e., MGDA (Sener & Koltun, 2018), GradNorm (Chen et al., 2018b), PCGrad (Yu et al., 2020), GradVac (Wang et al., 2021), IMTL-G (Liu et al., 2021b), CAGrad (Liu et al., 2021a), MTAdam (Malkiel & Wolf, 2021), MetaBalance (He et al., 2022), MoCo (Fernando et al., 2023), and Aligned-MTL (Senushkin et al., 2023)) and IMTL (Liu et al., 2021b), but is larger than the loss balancing methods because each task's gradient is computed in every iteration (i.e., step 6 in Algorithm 1). This is a common disadvantage for gradient balancing methods (Chen et al., 2018b; He et al., 2022; Liu et al., 2021a,b; Malkiel & Wolf,

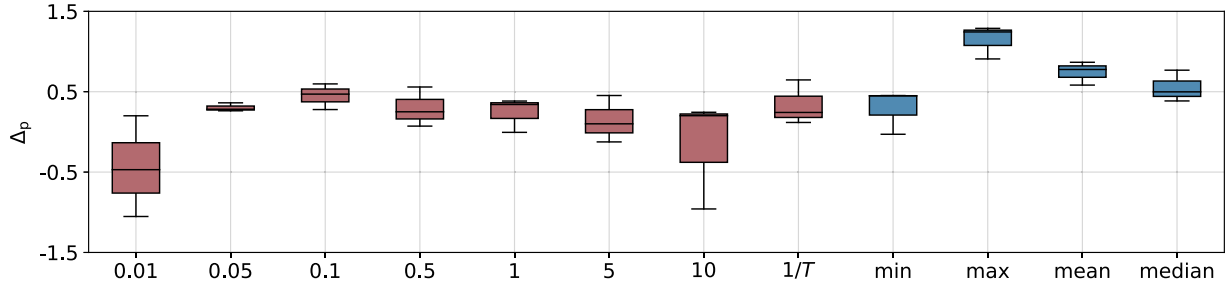


Fig. 6. Δ_p of different strategies for α_k in Eq. (2) on the *NYUv2* dataset. "min", "max", "mean", and "median" denote the minimum, maximum, average, and median of $\|\hat{\mathbf{g}}_{t,k}\|_2$ $(t = 1, \ldots, T)$, respectively. T is the number of tasks.

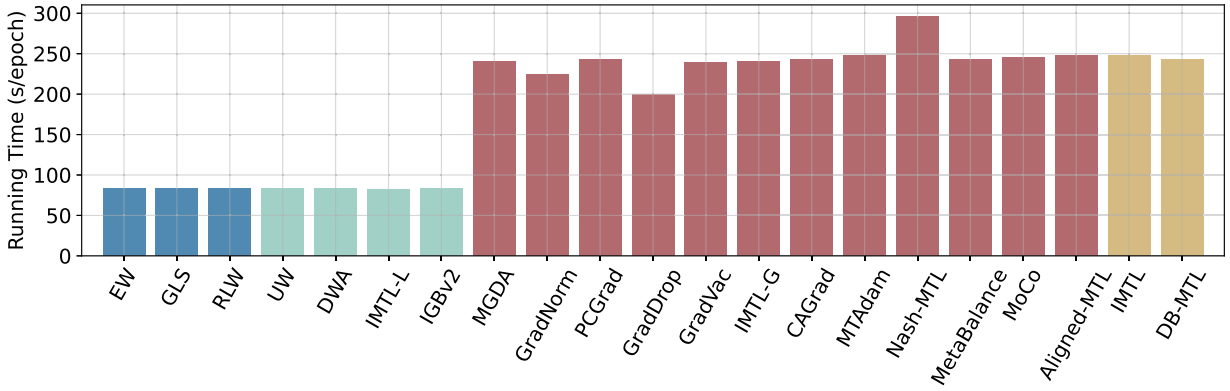
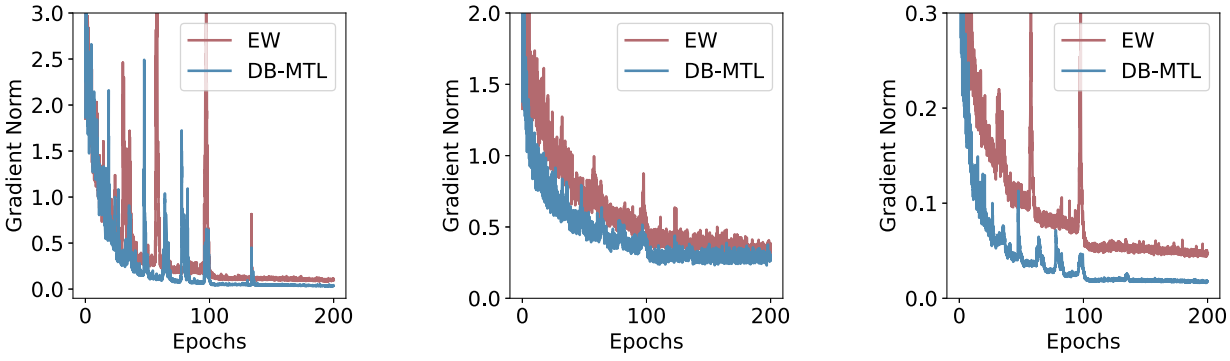
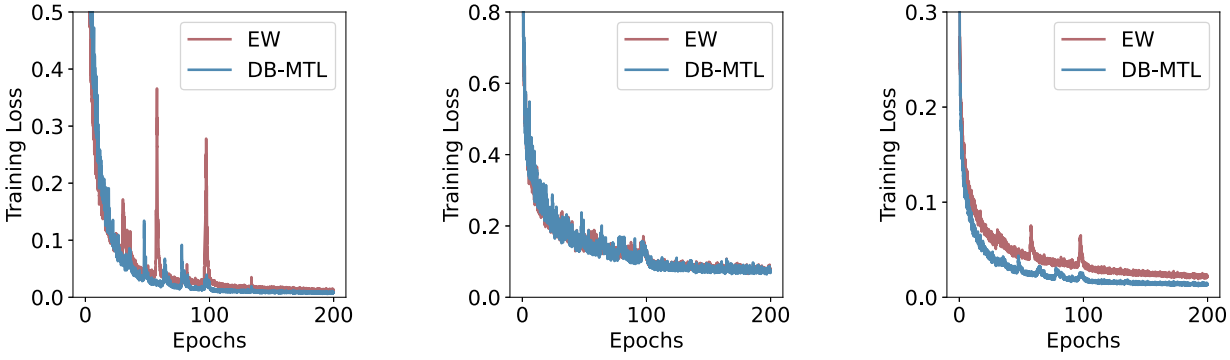


Fig. 7. The running time per epoch averaged 100 repetitions of different methods on *NYUv2* dataset. Cyan, red, yellow, and blue denote loss balancing methods, gradient balancing methods, hybrid balancing methods, and others, respectively.



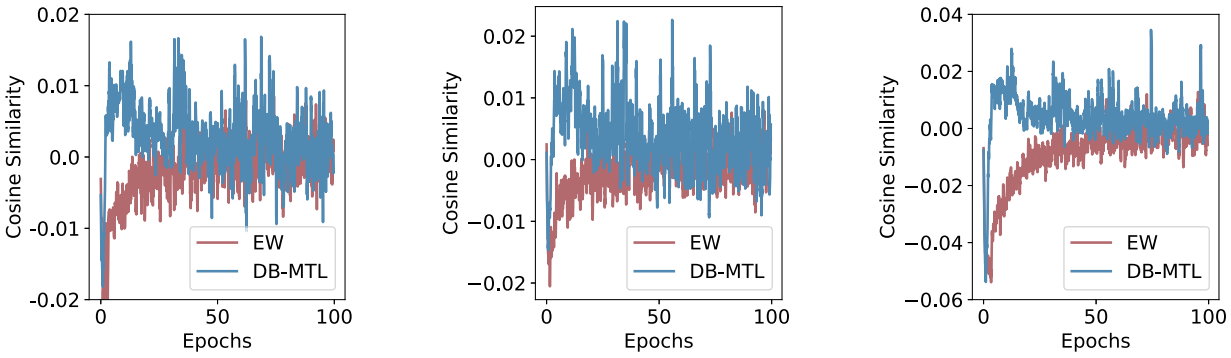
(a) *Segmentation task.* (b) *Depth estimation task.* (c) *Normal prediction task.*

Fig. 8. Gradient norm curves of EW and DB-MTL on the *NYUv2* dataset.



(a) *Segmentation task.* (b) *Depth estimation task.* (c) *Normal prediction task.*

Fig. 9. Training loss curves of EW and DB-MTL on the *NYUv2* dataset.



(a) *Amazon vs. DSLR.* (b) *Amazon vs. Webcam.* (c) *DSLR vs. Webcam.*

Fig. 10. Gradient cosine similarity of EW and DB-MTL on the *Office-31* dataset.

2021; Navon et al., 2022; Sener & Koltun, 2018; Senushkin et al., 2023; Wang et al., 2021; Yu et al., 2020). Although DB-MTL is slower than loss balancing methods, it achieves better performance, as shown in Tables 1–4, and 6.

4.7. Analysis of training stability

Figs. 8 and 9 compare the gradient norms $\|\nabla_{\theta_k} \ell_t(\mathcal{B}_{t,k}; \theta_k, \psi_{t,k})\|_2$ and training losses of EW and DB-MTL on the *NYUv2* dataset. As can be seen, for each task, the training loss of DB-MTL decreases smoothly and finally converges, and the gradient norm of DB-MTL is much more lower than EW. This indicates the logarithm transformation and maximum-norm strategy do not affect training stability.

4.8. Analysis of gradient conflict and task imbalance

Fig. 10 shows the gradient cosine similarity of EW and DB-MTL on the *Office-31* dataset, measuring the gradient conflict and task imbalance (Yu et al., 2020). As can be seen, comapred to EW, the cosine similarity of DB-MTL increases faster and then keeps postive during the training process, indicating that DB-MTL can reduce the gradient conflict and improve the task balance.

5. Conclusion

In this paper, we alleviate the task-balancing problem in MTL by presenting Dual-Balancing Multi-Task Learning (DB-MTL), a novel approach that performs both loss-scale balancing (which makes all task losses have a similar scale via the logarithm transformation) and gradient-magnitude balancing (which rescales task gradients to comparable magnitudes using the maximum gradient norm). Extensive experiments on a number of benchmark datasets demonstrate that DB-MTL outperforms the current state-of-the-art. Moreover, the logarithm transformation can also benefit existing gradient balancing methods. For future work, we will extend our approach to incorporate gradient variance in addition to magnitudes for more refined task weighting, and develop theoretical analysis to provide convergence guarantees and optimality conditions for our method.

CRediT authorship contribution statement

Baijiong Lin: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization; **Weisen Jiang:** Writing – review & editing, Conceptualization; **Feiyang Ye:** Writing – review & editing; **Yu Zhang:** Writing – review & editing; **Pengguang Chen:** Writing – review & editing; **Ying-Cong Chen:** Writing – review & editing, Funding acquisition, Supervision, Project administration; **Shu Liu:** Writing – review & editing; **Ivor W. Tsang:** Writing – review & editing; **James T. Kwok:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant no 92370204.

References

Badrinarayanan, 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.
Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge University Press.
Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *International conference on machine learning*.
Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.
Chen, L., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018a). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European conference on computer vision*.
Chen, S., Zhang, Y., & Yang, Q. (2024). Multi-task learning in natural language processing: an overview. *ACM Computing Surveys*, *56*(12), 1–32.
Chen, W., Zhang, X., Lin, B., Lin, X., Zhao, H., Zhang, Q., & Kwok, J. T. (2025). Gradient-based multi-objective deep learning: algorithms, theories, applications, and beyond. arXiv:2501.10945.
Chen, Z., Badrinarayanan, V., Lee, C.-Y., & Rabinovich, A. (2018b). GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*.
Chen, 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. In *Neural information processing systems*.
Chennupati, S., Sistu, G., Yogamani, S., & Samir, A. R. (2019). MultiNet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *IEEE conference on computer vision and pattern recognition workshops*.
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 *IEEE conference on computer vision and pattern recognition*.
Dai, Y., Fei, N., & Lu, Z. (2023). Improvable gap balancing for multi-task learning. In *Uncertainty in artificial intelligence*.
Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*.
Désidéri, J.-A. (2012). Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, *350*(5), 313–318.
Fernando, H. D., Shen, H., Liu, M., Chaudhury, S., Murugesan, K., & Chen, T. (2023). Mitigating gradient bias in multi-objective learning: a provably convergent approach. In *International conference on learning representations*.
Gasteiger, J., Groß, J., & Günnemann, S. (2020). Directional message passing for molecular graphs. In *International conference on learning representations*.
Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*.
Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., & Chi, E. (2021). Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In *Neural information processing systems*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*.

He, Y., Feng, X., Cheng, C., Ji, G., Guo, Y., & Caverlee, J. (2022). MetaBalance: Improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. In *ACM web conference*.

Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE conference on computer vision and pattern recognition*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Kurin, V., De Palma, A., Kostrikov, I., Whiteson, S., & Kumar, M. P. (2022). In defense of the unitary scalarization for deep multi-task learning. In *Neural information processing systems*.

Lin, B., Jiang, W., Chen, P., Liu, S., & Chen, Y.-C. (2025). MTMamba++: Enhancing multi-task dense scene understanding via mamba-based decoders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *47*(11), 10633–10645.

Lin, B., Jiang, W., Chen, P., Zhang, Y., Liu, S., & Chen, Y.-C. (2024). MTMamba: Enhancing multi-task dense scene understanding by mamba-based decoders. In *European conference on computer vision*.

Lin, B., Ye, F., Zhang, Y., & Tsang, I. (2022). Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*.

Lin, B., & Zhang, Y. (2023). LibMTL: A Python library for multi-task learning. *Journal of Machine Learning Research*, *24*(209), 1–7.

Liu, B., Liu, X., Jin, X., Stone, P., & Liu, Q. (2021a). Conflict-averse gradient descent for multi-task learning. In *Neural information processing systems*.

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., & Zhang, W. (2021b). Towards impartial multi-task learning. In *International conference on learning representations*.

Liu, P., Qiu, X., & Huang, X.-J. (2017). Adversarial multi-task learning for text classification. In *Annual meeting of the association for computational linguistics*.

Liu, S., James, S., Davison, A., & Johns, E. (2022). Auto-Lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research* .

Liu, S., Johns, E., & Davison, A. J. (2019a). End-to-end multi-task learning with attention. In *IEEE Conference on computer vision and pattern recognition*.

Liu, X., He, P., Chen, W., & Gao, J. (2019b). Multi-task deep neural networks for natural language understanding. In *Annual meeting of the association for computational linguistics*.

Luo, H., Hu, W., Wei, Y., He, J., & Yu, M. (2025). HirMTL: Hierarchical multi-task learning for dense scene understanding. *Neural Networks*, *181*, 106854.

Malkiel, I., & Wolf, L. (2021). MTAdam: Automatic balancing of multiple training loss terms. In *Conference on empirical methods in natural language processing*.

Maninis, K.-K., Radosavovic, I., & Kokkinos, I. (2019). Attentive single-tasking of multiple tasks. In *IEEE/CVF conference on computer vision and pattern recognition*.

Misra, I., Shrivastava, A., Gupta, A., & Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *IEEE Conference on computer vision and pattern recognition*.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., & Fetaya, E. (2022). Multi-task learning as a bargaining game. In *International conference on machine learning*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Neural information processing systems*.

Ramakrishnan, R., Dral, P. O., Rupp, M., & Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, *1*(1), 1–7.

Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010). Adapting visual category models to new domains. In *European conference on computer vision*.

Sener, O., & Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *Neural information processing systems*.

Senushkin, D., Patakin, N., Kuznetsov, A., & Konushin, A. (2023). Independent component alignment for multi-task learning. In *IEEE/CVF conference on computer vision and pattern recognition*.

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *European conference on computer vision*.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., & Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *International conference on machine learning*.

Sun, T., Shao, Y., Li, X., Liu, P., Yan, H., Qiu, X., & Huang, X. (2020). Learning sparse sharing architectures for multiple tasks. In *AAAI conference on artificial intelligence*.

Tang, H., Liu, J., Zhao, M., & Gong, X. (2020). Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *ACM conference on recommender systems*.

Tieleman, T., & Hinton, G. (2012). RMSProp: Neural networks for machine learning. Lecture 6.5.

Vandenhende, 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.

Venkateswara, H., Eusebio, J., Chakraborty, S., & Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *IEEE conference on computer vision and pattern recognition*.

Wang, Y., Lam, H. T., Wong, Y., Liu, Z., Zhao, X., Wang, Y., Chen, B., Guo, H., & Tang, R. (2023). Multi-Task Deep Recommender Systems: A Survey. arXiv:2302.03525.

Wang, Z., Tsvetkov, Y., Firat, O., & Cao, Y. (2021). Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International conference on learning representations*.

Xin, D., Ghorbani, B., Gilmer, J., Garg, A., & Firat, O. (2022). Do current multi-task optimization methods in deep learning even help? In *Neural information processing systems*.

Ye, F., Lin, B., Cao, X., Zhang, Y., & Tsang, I. (2024a). A first-order multi-gradient algorithm for multi-objective bi-level optimization. In *European conference on artificial intelligence*.

Ye, F., Lin, B., Yue, Z., Guo, P., Xiao, Q., & Zhang, Y. (2021). Multi-objective meta learning. In *Neural information processing systems*.

Ye, F., Lin, B., Yue, Z., Zhang, Y., & Tsang, I. (2024b). Multi-objective meta-learning. *Artificial Intelligence*, *335*, 104184.

Ye, H., & Xu, D. (2022). Inverted pyramid multi-task transformer for dense scene understanding. In *European conference on computer vision*.

Yi, Q., Wu, L., Tang, J., Zeng, Y., & Song, Z. (2025). Hybrid contrastive multi-scenario learning for multi-task sequential-dependence recommendation. *Neural Networks*, *183*, 106953.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., & Finn, C. (2020). Gradient surgery for multi-task learning. In *Neural information processing systems*.

Zeiler, M. D. (2012). AdaDelta: an adaptive learning rate method. arXiv:1212.5701.

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