MaxGNR: A Dynamic Weight Strategy via Maximizing Gradient-to-Noise Ratio for Multi-Task Learning

Caoyun Fan[1,†], Wenqing Chen[2,†], Jidong Tian[1], Yitian Li[1], Hao He[1,*], and Yaohui Jin[1]

[1] Shanghai Jiao Tong University, Shanghai, China
`{fcy3649, frank92, yitian_li, hehao, jinyh}@sjtu.edu.cn`
[2] Sun Yat-sen University, Guangzhou, China
`chenwq95@mail.sysu.edu.cn`

Abstract. When modeling related tasks in computer vision, Multi-Task Learning (MTL) can outperform Single-Task Learning (STL) due to its ability to capture intrinsic relatedness among tasks. However, MTL may encounter the insufficient training problem, i.e., some tasks in MTL may encounter non-optimal situation compared with STL. A series of studies point out that too much gradient noise would lead to performance degradation in STL, however, in the MTL scenario, Inter-Task Gradient Noise (ITGN) is an additional source of gradient noise for each task, which can also affect the optimization process. In this paper, we point out ITGN as a key factor leading to the insufficient training problem. We define the Gradient-to-Noise Ratio (GNR) to measure the relative magnitude of gradient noise and design the MaxGNR algorithm to alleviate the ITGN interference of each task by maximizing the GNR of each task. We carefully evaluate our MaxGNR algorithm on two standard image MTL datasets: NYUv2 and Cityscapes. The results show that our algorithm outperforms the baselines under identical experimental conditions.

Keywords: Multi-Task Learning · Gradient Noise · Weight Strategy.

1 Introduction

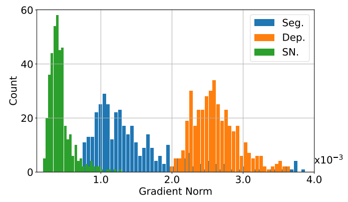
Deep learning [10] has achieved significant success in Multi-Task Learning (MTL) in the field of computer vision. The multi-task model captures the intrinsic correlation among tasks and also allows multiple inferences in one single forward pass, so MTL could achieve the unity of high performance and efficiency in the optimization process [24]. However, MTL may suffer from the insufficient training problem [16], which means some tasks in MTL are not optimal compared with the single-task solution.

Most deep learning models are trained to be optimum by the Stochastic Gradient Descent (SGD) technique [1]: models are optimized by the loss gradient

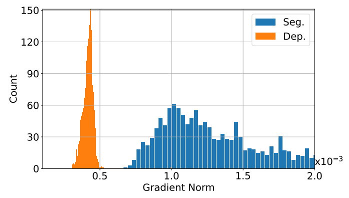
[*] Corresponding author.
[†] These authors contributed equally.

Table 1. Percentage change in performance[1] of MTL relative to STL. NYUv2 and CityScapes are two one-to-many image datasets, where **Seg.**, **Dep.**, and **SN.** denote Segmentation task, Depth task, and Surface Normal task, respectively.

Task	**NYUv2**			**CityScapes**	
	Seg.	**Dep.**	**SN.**	**Seg.**	**Dep.**
performance	8.3% ↑	6.9% ↑	9.7% ↓	1.1% ↑	13.2% ↓



(a) NYUv2



(b) CityScapes

Fig. 1. Gradient Norm distribution on NYUv2 and CityScapes.

based on a mini-batch randomly selected from all data. The gradient obtained by the SGD algorithm has noise [2,26], and many studies [14,13] point out that the gradient noise magnitude could affect the SGD generalization in Single-Task Learning (STL). This viewpoint gives us a key insight: in the MTL scenario, due to the joint learning of multiple tasks, the Inter-Task Gradient Noise (ITGN) could also interfere with the optimization of specific tasks, which may be the cause of the insufficient training problem.

Following this insight, we analyze the performance of each task (Tab. 1) as well as gradient norm distribution (Fig. 1) on two image MTL datasets. We find that the variance of the gradient norm varies widely across tasks, which implies that different tasks typically have widely varying gradient noise. At the same time, small gradient tasks tend to suffer from performance deterioration in MTL, which shows the relative disadvantage of small gradient tasks in MTL. In this paper, we demonstrate that Inter-Task Gradient Noise (ITGN) is one of the key optimization challenges in MTL (Section 3). In fact, the gradient in MTL optimization contains the noises of all tasks, so ITGN is an additional source of gradient noise compared to STL. As a result, some tasks (especially those with small gradients) may suffer from performance deterioration due to the effect of ITGN because the model has difficulty converging to the optimal position based on the gradient when the gradient noise is too high.

To alleviate the ITGN effect in MTL, we design a dynamic weight strategy called MaxGNR. Specifically, we realize that the magnitude of gradients and noise may vary greatly from task to task, and to quantitatively describe the ef-

[1] Compared STL and MTL at the same settings. Details in Section 5.

fect of noise on SGD optimization, we define the Gradient-to-Noise Ratio (GNR) to measure the relative magnitude of gradient noise. Further, we explain that maximizing the GNR of each task in the MTL scenario is a reasonable method to mitigate ITGN interference (Section 4.1). Then, we propose the momentum method to approximate the theoretical GNR to a computable expression (Section 4.2), therefore, we can dynamically select the weights that maximize GNR at each iteration (Section 4.3). As a result, the ITGN interference is reduced through the MaxGNR algorithm, and the insufficient training problem could be mitigated.

Furthermore, we extensively examine our MaxGNR algorithm on two standard image MTL datasets: NYUv2 [22] and CityScapes [7], and the experimental results reveal that the MaxGNR algorithm obtain superior performance than other baselines under the same premise of other settings. Our contributions are:

– We analyze the effect of gradient noise on MTL and connect the insufficient training problem with ITGN interference.
– We define the Gradient-to-Noise Ratio (GNR) to measure the relative magnitude of gradient noise in MTL, and quantitatively describe the ITGN interference by GNR.
– We propose the MaxGNR algorithm, a dynamic weight strategy to alleviate ITGN interference and experiments demonstrate the algorithm's validity.

2 Related Work

2.1 Multi-Task Learning

There are mainly two aspects in the field of MTL: network architecture improvement [31,25] and optimization strategy development, aiming at feature extraction and balance of performance in MTL, respectively. There have been lots of studies on multi-task architecture improvements, some works focus on the encoder structure [19,20,9,17] and others focus on the decoder part [27,31,25]. In this paper, we focus on optimization strategy development. A major challenge of MTL is how to balance the performance of tasks in joint learning [24]. To solve this challenge, researchers have proposed many algorithms, here we divide these algorithms into two categories: *coarse-grained* and *fine-grained*. The coarse-grained algorithm is to design optimization strategy according to the metrics of prediction, Uncertainty [6] utilizes homeostatic uncertainty to balance each loss; Dynamic Weight Averaging [17] tries to balance the decline rate of different task losses; Dynamic Task Prioritization [11] focuses on the balance of key performance indicators for multiple tasks; GLS [5] uses the geometric mean of task-specific losses as the target loss. The fine-grained algorithm is based on the gradient of model parameters, which can more accurately reflect the updating direction in each iteration. In recent years, such algorithms have been developed rapidly. GradNorm [4] stimulates the task-specific gradients to be of similar magnitude; MGDA [21] regards MTL as multi-objective optimization; PCGrad [28] cut the conflict gradient of different tasks. However, although gradient noise is an important problem in STL, the effect of gradient noise in MTL has never been considered.

2.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the standard methods [1] to optimize machine learning models. It is originally proposed to make up for the computational bottleneck of gradient descent (GD). Some studies focus on the generation capability of SGD. [32] reports SGD outperforms GD, [13,14] deduce theoretically that small-batch SGD generalizes better than large-batch SGD. The gradient obtained by the SGD algorithm has un-biased gradient noise [18]. To suppress the optimization oscillation caused by gradient noise, [23] proposes to introduce the momentum method in the optimization process, and many optimizers absorb the idea of momentum method, such as Adagrad [8], Adadelta [29], Adam [15].

3 Effect of Gradient Noise

In this section, we analyze the source of gradient noise from the perspective of SGD (Section 3.1), and analyze the effect of gradient noise on STL (Section 3.2) and MTL (Section 3.3).

3.1 Preliminaries of SGD

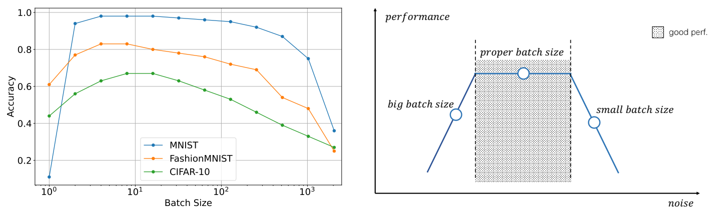
Formally, the design of the machine learning algorithm is based on data $D = \{(X_i, Y_i)\}$, the hypothesis function F, and loss function l. Algorithms are designed to search the optimal parameter $\{F_\theta | \theta \in \Theta \subset \mathbb{R}^d\}$ to get the lowest expected risk $\mathcal{R}(\theta)$ under the loss function l, where θ is the parameter of the hypothesis function and d is the dimension of the parameter. The expected risk $\mathcal{R}(\theta)$ cannot be obtained, so stochastic Gradient Descent algorithm (SGD) [1] selects a mini-batch $S = \{(X_i, Y_i)\}_{|S|}$ independent and identically distributed (i.i.d.) from the data D, and the expected risk $\mathcal{R}(\theta)$ can be estimated by the empirical risk $\widehat{\mathcal{R}}(\theta)$ calculated by mini-batch. In this paper, $\nabla_\theta \mathcal{R}(\theta)$ and $\nabla_\theta \widehat{\mathcal{R}}(\theta)$ are denoted as $g(\theta)$ and $\widehat{g_S}(\theta)$, and $l(F_\theta(X_i), Y_i)$ is expressed as $l_i(\theta)$.

The gradient $\widehat{g_S}(\theta)$ obtained by SGD is stochastic, so the gradient can be decomposed into the expected gradient and gradient noise. Many works [18,12] assume that the gradient noise belongs to the Gaussian class due to the classical central limit theorem [18]. In this paper, we follow this assumption in order to describe the effect of gradient noise on MTL quantitatively.

Assumption 1 *An individual data (X_i, Y_i) is selected independent and identically distributed (i.i.d.) from data D, and its gradient $\nabla_\theta l_i(\theta)$ obeys the Gaussian distribution:*

$$\nabla_\theta l_i(\theta) \sim \mathcal{N}(g(\theta), C) \tag{1}$$

where $g(\theta)$ is the expected gradient, C is the covariance matrix, and is approximately constant for θ, which is determined by data D and loss function l.



(a) Perf. vs. Batch Size (b) Perf. vs. Noise

Fig. 2. The effect of gradient noise magnitude on the performance of STL. The noise magnitude is controlled by the batch size. We conduct experiments on MNIST, FashionMNIST, and CIFAR10. As the batch size increases, performance first increases, then stabilizes, and finally decreases on all three datasets.

3.2 Gradient Noise in STL

Many studies [14,13] emphasize the magnitude of gradient noise could effect the generalization of SGD. In practice, noise magnitude can be easily adjusted by batch size $|S|$ [12]. According to Assumption 1, $\widehat{g_S}(\theta)$ can be simplified as:

$$\begin{gathered}\widehat{g_S}(\theta) = \frac{1}{|S|}\sum_{i=1}^{|S|}\nabla_\theta l_i(\theta) = g(\theta) + n_{g(\theta)} \\ \text{where} \quad n_{g(\theta)} \sim \mathcal{N}(0, \frac{C}{|S|})\end{gathered} \tag{2}$$

where $n_{g(\theta)}$ is the gradient noise. Because of the Positive Semi-definite of C, the noise magnitude can be measured as:

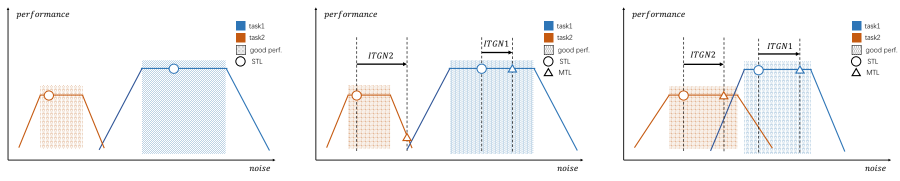
$$\mathbb{E}[\|n_{g(\theta)}\|^2] = tr(C)/\,|S| \tag{3}$$

By adjusting the batch size $|S|$, we evaluated the impact of noise magnitude on model performance. The result in Fig. 2(a) showed that the performance can be maintained only in the appropriate noise magnitude range. In fact, large noise would interfere with the convergence of the model, while the model is unable to escape the local optimum with small noise, so the effect of gradient noise on performance can be expressed as Fig. 2(b).

3.3 Inter-Task Gradient Noise

In the multi-task SGD process, multiple tasks have their own loss functions $\{l^k\}$, and Weighted Average Method (WAM) is a mainstream method to combine multiple loss functions. Specifically, WAM can be expressed as:

$$L = \sum_{k=1}^{n}\omega_k l^k(\theta) \quad s.t. \sum_{k=1}^{n}\omega_k = 1 \tag{4}$$



(a) Performance in STL (b) Performance in MTL (c) Design weights to maximize GNR

Fig. 3. Illustration of MaxGNR algorithm. In the STL scenario, reasonable settings (model, batch size, learning rate, etc.) can be selected to ensure that the gradient noise is in the appropriate range, as in Fig. 3(a). In the MTL scenario, ITGN is also a source of gradient noise that can affect the optimization of specific tasks. Therefore, ITGN may cause task-specific performance deterioration due to the large differences in ITGN, as is illustrated in Fig. 3(b). Our MaxGNR algorithm attempts to alleviate the ITGN interference as Fig. 3(c). According to Eq. 12, We can design appropriate weights to maximize GNR for the purpose of mitigating ITGN interference.

According to Assumption 1, the SGD gradient in MTL is expressed as follows:

$$\widehat{g_S}(\theta) = \sum_{k=1}^{n} \omega_k g^k(\theta) + \sum_{k=1}^{n} \omega_k n_{g^k(\theta)} \tag{5}$$

Based on Eq. 5, ITGN also contributes to task i's optimization in the MTL scenario, where ITGN_i is as follows:

$$\text{ITGN}_i = \sum_{k \neq i} \omega_k n_{g^k(\theta)} \sim \mathcal{N}(0, \frac{\sum_{k \neq i} \omega_k^2 C^k}{|S|}) \tag{6}$$

From Eq. 3, we can infer that $\{C^k\}$ and $|S|$ control the magnitude of $\{n_{g^k(\theta)}\}$, although multiple tasks share $|S|$, $\{C^k\}$ are task-specific and determined by the intrinsic characteristics of tasks, so $\{n_{g^k(\theta)}\}$ usually exist huge discrepancies, as shown in Fig. 1. Therefore, ITGN can cause performance deterioration in MTL (similar to the performance deterioration due to large noise in STL).

4 Method

We describe the necessity of controlling the noise magnitude range to maintain excellent model performance in Section 3.2 as Fig. 3(a) and explain that ITGN could also interfere with the specific task's optimization in MTL in Section 3.3. In summary, the range of ITGN may vary greatly, so it can be inferred that ITGN is likely to cause performance deterioration on some tasks as Fig. 3(b). In this section, we propose the MaxGNR algorithm, a dynamic weight strategy to minimize the ITGN interference in MTL as Fig. 3(c).

Algorithm 1: MaxGNR algorithm

```
Input: data $\{X_i, Y_i^1, \dots, Y_i^n\}_{|D|}$, learning rate $\mu$.
Output: optimal network parameters $\theta$.
Initialize network parameters $\theta_0$ ;
for $t = 0$ to $T$ do
    select mini-batch data $S = \{X_i, Y_i^1, \dots, Y_i^n\}_{|S|}$;
    compute each empirical risk $\{\widehat{\mathcal{R}^k}(\theta_t)\}$;
    compute the gradient of empirical risk $\{\nabla\widehat{\mathcal{R}^k}(\theta_t)\}$;
    compute the momentum gradient $\{m_t^k\}$ and noise $\{n_t^k\}$ ;
    select $\{\omega_t^k\}$ by $\arg\max_{\{\omega_k\}} \left\{\min\{\mathrm{GNR}_k(m_t^k, \{n_t^k\})\}\right\}$ ;
    update network parameters $\theta_{t+1} = \theta_t - \mu\nabla_{\theta_t} \sum_{k=1}^n \omega_t^k \widehat{\mathcal{R}^k}(\theta_t)$ ;
end
```

Gradient-to-Noise Ratio (GNR) is the core concept of this paper, which is defined to measure the relative magnitude of gradient noise (Section 4.1), and it can be found that ITGN would reduce GNR compared to STL because ITGN is an additional gradient noise source. So it is a viable method to mitigate ITGN interference by maximizing GNR. To solve the dilemma that GNR is not computable, we propose to estimate $\{g^k(\theta)\}$ and $\left\{n_{g^k(\theta)}\right\}$ using the momentum method, and thus estimate a computable GNR (Section 4.2). Eventually, we can dynamically select the weights that maximize GNR in each iteration, thus achieving the purpose of alleviating ITGN interference (Section 4.3). The whole process of the MaxGNR algorithm is shown in Alg. 1.

4.1 Gradient-to-Noise Ratio

In order to measure the noise interference on stochastic gradient optimization, it is necessary to define an appropriate metric to measure the relative magnitude of gradient noise to the gradient.

Definition 1 *In the process of SGD, for a specific task, the expected gradient is $g(\theta)$, the random variable that could make $\widehat{g_S}(\theta)$ deviate from $g(\theta)$ is the gradient noise $n_{g(\theta)}$, we define **Gradient-to-Noise Ratio (GNR)** as:*

$$GNR = \frac{\|g(\theta)\|^2}{\mathbb{E}[\|n_{g(\theta)}\|^2]} \tag{7}$$

GNR reflects the relative magnitude of gradient noise to the gradient. The homogeneity ensures that GNR is independent of weight in STL, and according to Eq. 3, $\mathbb{E}[\|n_{g(\theta)}\|^2]$ is an approximate constant and easy to represent.

There is only one source of gradient noise in STL. According to definition 1, GNR_S is represented as:

$$\text{GNR}_S = \frac{\|g(\theta)\|^2}{\mathbb{E}[\|n_{g(\theta)}\|^2]} = \frac{\|g(\theta)\|^2}{tr(C)/\,|S|} \tag{8}$$

In the MTL scenario, due to the ITGN interference, task k's GNR is represented as:

$$\text{GNR}_M^k = \frac{\|\omega_k g^k(\theta)\|^2}{\mathbb{E}[\|\omega_k n_{g^k(\theta)} + \text{ITGN}_k\|^2]} \tag{9}$$

According to Eq. 3 and Eq. 5, $\mathbb{E}[\|\text{ITGN}_k\|^2]$ could be represented by $\{C^k\}$ and $|S|$, and due to the linear property of matrix's trace:

$$tr(m \cdot A + n \cdot B) = m \cdot tr(A) + n \cdot tr(B) \tag{10}$$

GNR_M^k is eventually simplified as:

$$\text{GNR}_M^k = \frac{\|\omega_k g^k(\theta)\|^2}{\sum_{k=1}^{n} \omega_k^2 tr(C^k)/\,|S|} \leq \text{GNR}_S^k \tag{11}$$

In general, hyperparameters with superior performance in STL would be selected for MTL, so we assume that GNR_S^k can be controlled as an appropriate GNR. However, ITGN interference reduces the GNR of each task, if $\text{ITGN}_k \gg n_{g^k(\theta)}$, GNR_M^k could be much smaller than GNR_S^k, which is likely to cause performance deterioration of task k. Therefore, selecting weights $\{\omega_k\}$ to maximize GNR_M^k is a feasible method to alleviate ITGN interference for task k. In the MTL scenario, to ensure that all tasks could alleviate ITGN interference as much as possible, we should maximize $\min\{\text{GNR}_M^k\}$ as:

$$\{\omega_k\} = \underset{\{\omega_k\}}{\arg\max} \left\{ \frac{\min\left\{\|\omega_k g^k(\theta)\|^2\right\}}{\sum_{k=1}^{n} \omega_k^2 tr(C^k)/\,|S|} \right\} \tag{12}$$

4.2 Estimation of Expected Gradient

However, GNR is a theoretical concept and it is not realistic to calculate $\left\{g^k(\theta_t)\right\}$ and $\left\{C^k\right\}$, therefore, we should design a method to estimate them. During the training process, we can only get the empirical gradient $\{\widehat{g_S}^k(\theta_t)\}$. In this paper, we propose that the momentum method can be used to estimate the expected gradient and gradient noise.

Momentum Method The momentum method is widely applied in various optimizers. AdaGrad, AdaDelta, Adam, and so on adopt the momentum method as a technique to stabilize gradient. Momentum is expressed as m, and we set $m_0 = \widehat{g_S}(\theta_0)$. The basic process of the momentum method is as follows:

$$\begin{aligned} m_t &= \gamma \cdot m_{t-1} + (1-\gamma) \cdot \widehat{g_S}(\theta_t) \\ \theta_t &= \theta_{t-1} - \mu \cdot m_t \end{aligned} \tag{13}$$

where γ is the decay rate, μ is the learning rate, both are hyperparameters. It is an option to estimate the expected gradient by momentum gradient, because the momentum gradient could reduce noise and stabilize gradient. According to Eq. 2 and Eq. 13, the momentum gradient can be derived as:

$$\begin{aligned} m_t^k &\approx (1-\gamma)\sum_{i=1}^{t}\gamma^{t-i}\cdot g^k(\theta_i) + n_{m_t^k} \\ \text{where}\quad n_{m_t^k} &\sim \sqrt{\frac{1-\gamma}{1+\gamma}}\mathcal{N}(0, \frac{C}{|S|}) \end{aligned} \tag{14}$$

By comparing Eq. 2 and Eq. 14, it can be found that the momentum method produces a contraction coefficient of $\sqrt{(1-\gamma)/(1+\gamma)}$ on noise magnitude, the larger γ is, the smaller the estimated noise magnitude is. At the same time, the cost is that the momentum gradient m_t^k is no longer the un-biased estimation of $g^k(\theta_t)$, the larger γ is, the greater the estimation error is. So γ becomes a trade-off factor of the expected gradient estimation, if the estimation error of $g^k(\theta_t)$ is too large, the momentum method is also invalid. Fortunately, parameter optimization is a long process, and the change of parameters in each iteration is small compared to the parameters themselves. Based on this fact, we make the following assumption:

Assumption 2 *the expected gradient changes slowly and the expected gradients in adjacent iterations are similar:*

$$g^k(\theta_n) \approx g^k(\theta_{n+\Delta n}),\ \textit{when } \Delta n \textit{ is small} \tag{15}$$

On the premise of Assumption 2, the estimation error of $g^k(\theta_t)$ in Eq. 14 can be reduced to a large extent, so it is reasonable to estimate $g^k(\theta_t)$ with $m^k(\theta_t)$.

4.3 Weight Selection

According to the estimation method proposed by Eq. 14, the empirical gradient of each task can be decomposed into momentum gradient $m^k(\theta)$ and gradient noise $n_{g^k}(\theta)$. Therefore, we can estimate $n_{g^k}(\theta)$:

$$n_{g^k}(\theta) = \widehat{g_S}^k(\theta) - m^k(\theta) \tag{16}$$

To calculate the GNR of each task, we can replace $g^k(\theta)$ with $m^k(\theta)$ according to Eq. 14 and Assumption 2, and although $\mathbb{E}[\|n_{g^k(\theta)}\|^2]$ is approximately determined, each gradient noise is a random variable, so it is reasonable to use $\|\sum_{k=1}^{n}\omega_k n_{g^k(\theta)}\|^2$ to replace $\sum_{k=1}^{n}\omega_k^2 tr(C^k)/|S|$. Therefore, the MaxGNR algorithm can ultimately expressed as:

$$\{\omega_k\} = \arg\max_{\{\omega_k\}}\left\{\frac{\min\left\{\|\omega_k m^k(\theta)\|^2\right\}}{\|\sum_{k=1}^{n}\omega_k n_{g^k(\theta)}\|^2}\right\} \tag{17}$$

Table 2. Experiment results of different algorithms for NYUv2, and we split the table into coarse-grained and fine-grained algorithms. The **bold** represents the top2 scores.

Algorithm	**Segmentation**		**Depth**		**Surface Normal**	
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Mean ↓	Median ↓
single task	26.15	53.27	0.6655	27.89	30.57	25.12
equal weights	28.32	55.60	0.6196	27.61	31.80	26.82
coarse-grained						
DWA	28.43	56.08	0.6075	25.08	31.98	26.89
Uncertainty	28.06	54.46	0.6059	25.69	31.49	26.32
fine-grained						
GradNorm	28.92	56.26	0.6057	**24.06**	31.66	26.49
MGDA	22.38	49.59	0.7414	27.51	30.50	**23.85**
PCGrad	28.72	56.17	0.6131	27.33	32.19	27.00
MaxGNR ($\gamma = 0.5$)	**29.76**	**57.64**	**0.5943**	**24.35**	**29.89**	**24.38**
MaxGNR ($\gamma = 0.9$)	**29.48**	**56.81**	**0.5904**	24.91	**30.43**	24.85

Because of the non-linearity of Eq. 17, a general analytical solution cannot be found. We use the steepest descent method [3] to obtain the optimal $\{\omega_k\}$ at each iteration. Compared with the number of parameters in the neural network, the number of parameters in the steepest descent method is the number of tasks, so the computation cost can be ignored. At the same time, the computation cost of the steepest descent method is not sensitive to the number of tasks, which is different from grid search.

5 Experiments

5.1 Experimental Settings

Datasets We focused on one-to-many predictions datasets $\{X_i, Y_i^1, \ldots, Y_i^n\}_{|D|}$ [30] in this paper. We evaluated the proposed MaxGNR algorithm on two image datasets: NYUv2 [22] and CityScapes [7]. NYUv2 is a challenging indoor scene dataset in various room types (bathrooms, living rooms, studies, etc.), and this dataset has three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. Although NYUv2 is a relatively small dataset (795 training, 654 test images), it contains both regression and classification tasks, making it a good candidate for testing the robustness of different types of loss functions. CityScapes is a high-resolution street-view images dataset, we chose two sub-tasks for our experiment: semantic segmentation and depth estimation. Compared to NYUv2, the street-view image contained in CityScapes is relatively simpler, because the viewpoints and lighting conditions are relatively fixed, and the appearance of each object class changes little in shape.

Table 3. Experiment results of different algorithms for CityScapes.

Algorithm	**Segmentation**		**Depth**	
	mIoU ↑	Pix Acc ↑	↑ Abs Err ↓	Rel Err ↓
single task	66.91	90.03	0.0153	35.85
equal weights	67.63	90.09	0.0163	57.13
GradNorm	67.20	90.76	0.0169	57.21
MGDA	66.83	90.88	0.0170	47.14
PGGrad	**67.75**	**91.01**	0.0164	54.95
MaxGNR ($\gamma = 0.5$)	**68.01**	**91.07**	**0.0162**	**44.11**
MaxGNR ($\gamma = 0.7$)	67.33	90.76	**0.0160**	**42.48**

Implementation Details We implemented our experiments based on the network architecture in [17]. The network architecture had an encoder-decoder structure that was homogeneous across all tasks, where the encoder extracted the representation and the decoder matched individual tasks. For the NYUv2 and CityScapes datasets, we set the batch size to 2 and 128, respectively, and the learning rate of the Adam optimizer to 1e-4. After training each model for 100/200 epochs, we selected the best model on the training set for testing. In our experiment, we set the learning rate of the steepest descent method to 1e-3, and $\{\omega_k\}$ updated 100 steps in each iteration.

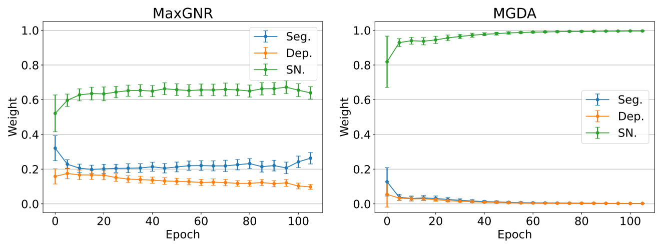
Evaluation Metrics To compare the baselines and our method, we chose two metrics for each task. Following the settings in [6,17,28], we chose mIoU and Pixel Accuracy for Segmentation task, Absolute and Relative Error for Depth task, and Mean and Median Angle Distance for Surface Normal task.

5.2 Baselines

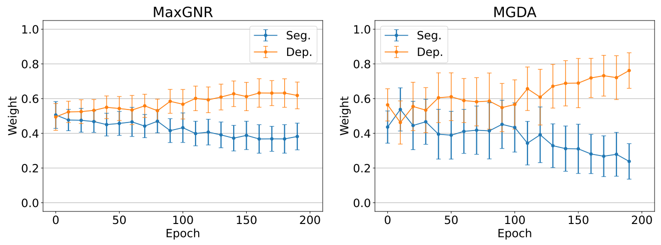
In addition to equal weights and single-task models, the baselines can be divided into two types: coarse-grained algorithms and fine-grained algorithms. Coarse-grained algorithms included DWA [17] and Uncertainty [6], fine-grained algorithms included GradNorm [4], MGDA [21] and PCGrad [28].

5.3 Main Results

Results on NYUv2 was shown in Tab. 2. Our MaxGNR algorithm outperformed the baselines in almost all metrics. When specific tasks were analyzed, most approaches outperformed the single-task model in the Depth task, and most baselines performed marginally better than the single-task model in the Segmentation task, while, in the Surface Normal task, almost all baselines suffered from performance deterioration. It was important to note that MGDA,



(a) NYUv2



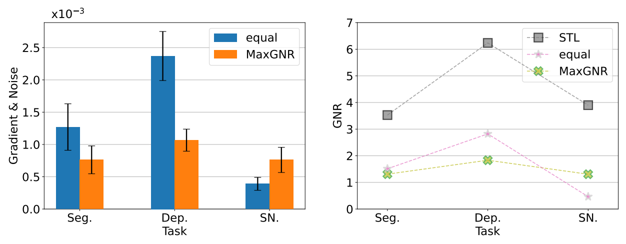
(b) CityScapes

Fig. 4. Dynamic change of weights and the training process.

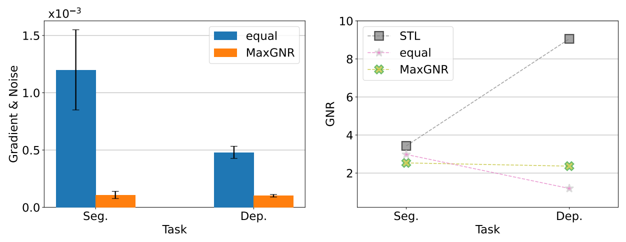
which outperformed the single-task model in the Surface Normal task, cannot balance the optimization performance on all tasks, resulting in MGDA's bad performance in the other two tasks. Our MaxGNR algorithm not only outperformed the single-task model and baselines in the Segmentation and Depth tasks, but it also performed well in the Surface Normal task and overcame the insufficient training problem.
Results on CityScapes was shown in Tab. 3. In CityScapes, the performance of the single-task model was similar to all MTL algorithms on the Segmentation task (about 91%), while there was a serious performance deterioration in the Depth task. We analyzed the performance of baselines and our algorithm in two tasks. In the Segmentation task, our algorithm was not significantly better than other baselines, but in the Depth task, our algorithm improved the performance of the model and greatly narrowed the gap with the single-task model.

5.4 Dynamic Change of Weights

The dynamic weight strategy can assign appropriate weights to each task at different training stages. We compared the weights of the different stages obtained by MaxGNR and MGDA in Fig. 4. We found that the weight trends obtained by the two algorithms were similar, but MGDA's weights were more extreme compared to MaxGNR's weights. Specifically, in NYUv2, the weight assignment of MGDA led the model to focus only on the Surface Normal task, while MaxGNR's assignment was intuitively more balanced. The experimental results (MGDA in



(a) NYUv2



(b) CityScapes

Fig. 5. Gradient Norm Distribution and GNR of each task in NYUv2 and CityScapes.

Tab. 2) showed that extreme weight assignments may lead to insufficient training problems. The situation was similar in CityScapes. We believed that the weight assignment difference came from the object being balanced: MGDA attempted to balance each task's gradient, while MaxGNR attempted to balance each task's GNR.

6 Discussion

6.1 Gradient Noise and Performance

We fixed the best models under equal weights and MaxGNR settings and counted the distribution of the gradient norm for all data and calculated the current GNR of the models based on the gradient norm distribution, as shown in Fig. 5.

We found that in the equal weights model, the gradients and noises of each task differed significantly, and our algorithm clearly balanced the gradients and noises of each task (Fig. 5 Left). Compared with GNR$_S$, the GNR$_M$ of Surface Normal task in NYUv2 and Depth task in CityScapes decreased most significantly in the equal weights model, and the performances of these tasks also exhibited a huge decrease. Our MaxGNR algorithm balanced each GNR$_M$ (Fig. 5 Right), by slightly reducing the GNR$_M$ of some tasks (usually harmless), the tasks suffering from performance deterioration would mitigate the interference of ITGN, and gained performance improvements. The experimental results also showed that the model in the MaxGNR setting had significant performance improvements in the Surface Normal task in NYUv2 and the Depth task in

CityScapes. At the same time, the performance improvement of some tasks can help the model to get better representation, which can improve the performance of the model in general.

6.2 The Paradox of Weight Design

In the field of weight design of MTL, researchers frequently constructed optimization algorithms depending on the difficulties of the tasks, however, the starting points of the design sometimes conflicted with each other. For example, Dynamic Task Prioritization (DTP) [11] allocated "difficult" tasks higher task-specific weights, while Uncertainty [6] assigned "easy" tasks higher task-specific weights. [24] gave a qualitative explanation for this conflict paradox: Uncertainty seemed to be more suitable for noisy labeled data, while DTP was more suitable for clean ground-truth annotations. The paradox can be described more clearly when we viewed this phenomenon from the perspective of MaxGNR algorithm. According to the Eq. 12, there are two factors that influence task's difficulty: small expected gradient $g(\theta)$ and large variance $\frac{C}{|S|}$.

If the model's upper limit is restricted by the small expected gradients of some tasks, the weights of these tasks should be adjusted to expand the expected gradients, similar to DTP. The noise part of Eq. 12 can be ignored as:

$$\{\omega_k\} = \underset{\{\omega_k\}}{\arg\max} \left\{ \min \left\{ \|\omega_k g^k(\theta)\|^2 \right\} \right\} \tag{18}$$

On the contrary, if the model's upper limit is restricted by the large variances of specific tasks, the weights of these tasks should be reduced, similar to Uncertainty, to reduce the impact on other small gradient tasks. So the gradient part of Eq. 12 can be ignored as:

$$\{\omega_k\} = \underset{\{\omega_k\}}{\arg\max} \left\{ \frac{1}{\sum_{k=1}^{n} \omega_k^2 tr(C^k)/|S|} \right\} \tag{19}$$

In conclusion, the MaxGNR algorithm not only considers the magnitude of the gradient but also considers the noise interference, which to some extent unifies these two kinds of algorithms with distinct starting points.

7 Conclusion

In this work, we attributed the insufficient training problem in MTL to the ITGN interference, and we proposed MaxGNR algorithm, a novel dynamic weight strategy to alleviate this interference. Experiments verified the effectiveness of our algorithm. Looking ahead, gradient noise in MTL is a new field, and we hope to explore the influence of gradient noise on more complex tasks. Besides, how to choose appropriate tasks for joint learning is an open question, and the GNR framework may be a possible research direction.

Acknowledgements This work was supported by the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and the Shanghai Science and Technology Innovation Action Plan (20511102600).

References

1. Bottou, L., et al.: Stochastic gradient learning in neural networks. NeurIPS (1991)
2. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. Siam Review (2018)
3. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2014)
4. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: ICML (2018)
5. Chennupati, S., Sistu, G., Yogamani, S., Rawashdeh, S.A.: Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In: CVPR Workshops (2019)
6. Cipolla, R., Gal, Y., Kendall, A.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: CVPR (2018)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
8. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. JMLR (2011)
9. Gao, Y., Ma, J., Zhao, M., Liu, W., Yuille, A.L.: Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In: CVPR (2019)
10. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive computation and machine learning, MIT Press (2016)
11. Guo, M., Haque, A., Huang, D.A., Yeung, S., Fei-Fei, L.: Dynamic task prioritization for multitask learning. In: ECCV (2018)
12. He, F., Liu, T., Tao, D.: Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In: NeurIPS (2019)
13. Hoffer, E., Hubara, I., Soudry, D.: Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In: NeurIPS (2017)
14. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: ICLR (2017)
15. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: ICLR (2015)
16. Liu, L., Li, Y., Kuang, Z., Xue, J.H., Chen, Y., Yang, W., Liao, Q., Zhang, W.: Towards impartial multi-task learning. In: ICLR (2021)
17. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: CVPR (2019)
18. Mandt, S., Hoffman, M.D., Blei, D.M.: Stochastic gradient descent as approximate bayesian inference. JMLR (2017)
19. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: CVPR (2016)

20. Ruder, S., Bingel, J., Augenstein, I., Søgaard, A.: Latent multi-task architecture learning. In: AAAI (2019)
21. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: NeurIPS (2018)
22. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
23. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: ICML (2013)
24. Vandenhende, S., Georgoulis, S., Gansbeke, W.V., Proesmans, M., Dai, D., Gool, L.V.: Multi-task learning for dense prediction tasks: A survey. TPAMI (2021)
25. Vandenhende, S., Georgoulis, S., Gool, L.V.: Mti-net: Multi-scale task interaction networks for multi-task learning. In: ECCV (2020)
26. Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., Zhu, Z.: On the noisy gradient descent that generalizes as sgd. In: ICML (2020)
27. Xu, D., Ouyang, W., Wang, X., Sebe, N.: Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In: CVPR (2018)
28. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. In: NeurIPS (2020)
29. Zeiler, M.D.: Adadelta: An adaptive learning rate method. arXiv preprint (2012)
30. Zhang, H., Xiao, L., Wang, Y., Jin, Y.: A generalized recurrent neural architecture for text classification with multi-task learning. In: IJCAI (2017)
31. Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., Yang, J.: Joint task-recursive learning for semantic segmentation and depth estimation. In: ECCV (2018)
32. Zhu, Z., Wu, J., Yu, B., Wu, L., Ma, J.: The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In: ICML (2019)