Revisiting Fairness in Multitask Learning: A Performance-Driven Approach for Variance Reduction

Supplementary Material

6. Theoretical Analysis

6.1. Proof of Proposition 1.

Proposition 1. *Let Δm^{max} be the maximum value of $\{\Delta m_i\}_{i=1}^k$, and Δm^{min} be the minimum value. Define $s = \min\left(\frac{1}{\underline{\omega}}, \overline{\omega}\right)$. Then, for $\tau > \frac{\Delta m^{max} - \Delta m^{min}}{\log s}$, Property 3 is satisfied.*

Proof. Define ω_{min} and ω_{max} as the minimum and maximum value of $\boldsymbol{\omega}$ respectively, since ω_i is positively correlated with Δm_i, we have

$$\omega_{max} = \frac{k \cdot \exp(\Delta m^{max}/\tau)}{\sum_{j=1}^k \exp(\Delta m_j/\tau)}, \omega_{min} = \frac{k \cdot \exp(\Delta m^{min}/\tau)}{\sum_{j=1}^k \exp(\Delta m_j/\tau)}.$$

We notice that

$$\frac{\omega_{max}}{\omega_{min}} = \frac{\exp(\Delta m^{max}/\tau)}{\exp(\Delta m^{min}/\tau)} = \exp(\frac{\Delta m^{max} - \Delta m^{min}}{\tau}) \stackrel{\text{def}}{=} s.$$

By incorporating property 2, we have

$$k = \sum_{i=1}^k \omega_i \leq \sum_{i=1}^k \omega_{max} \leq \sum_{i=1}^k s\, \omega_{min} = ks\, \omega_{min},$$

showing that $\omega_{min} \geq \frac{1}{s}$. In the same way, we have $\omega_{max} \leq s$. Thus by setting $s = \min(\frac{1}{\underline{\omega}}, \overline{\omega})$, we can derive that

$$\omega_i \in [\frac{1}{s}, s] \subseteq [\underline{\omega}, \overline{\omega}].$$

Notice that $\frac{\Delta m^{\max} - \Delta m^{\min}}{\tau} = \log s$, then for $\tau > \frac{\Delta m^{\max} - \Delta m^{\min}}{\log s}$, Property 3 is satisfied. In practice, we predefine a threshold τ^*, and let $\tau = \max(\frac{\Delta m^{\max} - \Delta m^{\min}}{\log s}, \tau^*)$ to further guarantee the smoothness and contraints. □

6.2. Proof of Proposition 2.

Proposition 2. *For $\boldsymbol{\omega}$ satisfying the three properties above, we have the following approximation:*

$$\mathrm{Var}[\Delta m_i] \approx \frac{\tau^2}{k} \boldsymbol{\omega}^\top \boldsymbol{\omega} - \tau^2.$$

Proof. Following the notation in the main paper, let $\mathbb{E}[\Delta m_i] = \frac{1}{k}\sum_{i=1}^k \Delta m_i$ and $\mathrm{Var}[\Delta m_i] = \sigma^2$. Define $\{\epsilon_i\}_{i=1}^k$ such that $\Delta m_i = \mathbb{E}[\Delta m_i] + \epsilon_i$, thus $\mathbb{E}[\epsilon_i] = 0$. We know that $\omega_i = \frac{k \cdot \exp(\Delta m_i/\tau)}{\sum_{j=1}^k \exp(\Delta m_j/\tau)}$. For the numerator,

$$\begin{aligned} k \cdot \exp(\Delta m_i/\tau) &= k \cdot \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau} + \frac{\epsilon_i}{\tau}\right) \\ &= k \cdot \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau}\right) \cdot \exp\left(\frac{\epsilon_i}{\tau}\right) \\ &\approx k \cdot \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau}\right)\left(1 + \frac{\epsilon_i}{\tau}\right), \end{aligned}$$

since τ is generally large enough such that $\frac{\epsilon_i}{\tau}$ is pretty small. Similarly, for the denominator, we have

$$\begin{aligned} \sum_{j=1}^k \exp(\Delta m_j/\tau) &\approx \sum_{j=1}^k \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau}\right)\left(1 + \frac{\epsilon_j}{\tau}\right) \\ &= \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau}\right)\left(k + \sum_{j=1}^k \frac{\epsilon_j}{\tau}\right) \\ &\approx k \cdot \exp\left(\frac{\mathbb{E}[\Delta m_i]}{\tau}\right). \end{aligned}$$

Thus, $\omega_i \approx \frac{k\left(1+\frac{\epsilon_i}{\tau}\right)}{k} = 1 + \frac{\epsilon_i}{\tau}$. Therefore, we can deduce

$$\begin{aligned} \mathbb{E}[\omega_i] &\approx 1 + \frac{\mathbb{E}[\epsilon_i]}{\tau} = 1, \\ \mathbb{E}[\omega_i^2] &\approx 1 + \frac{2\mathbb{E}[\epsilon_i]}{\tau} + \frac{\mathbb{E}[\epsilon_i^2]}{\tau^2} = 1 + \frac{\mathbb{E}[\epsilon_i^2]}{\tau^2}. \end{aligned}$$

In fact, only $\mathbb{E}[\omega_i^2]$ is approximated, as $\boldsymbol{\omega}$ always satisfies $\mathbf{1}^\top \boldsymbol{\omega} = k$, which implies that $\mathbb{E}[\omega_i] = 1$. On the other hand,

$$\begin{aligned} \sigma^2 &= \mathrm{Var}[\Delta m_i] = \mathbb{E}[\Delta m_i^2] - (\mathbb{E}[\Delta m_i])^2 \\ &= \frac{\sum_{i=1}^k (\mathbb{E}[\Delta m_i] + \epsilon_i)^2}{k} - \mathbb{E}[\Delta m_i]^2 \\ &= \frac{k\mathbb{E}[\Delta m_i]^2 + 2\mathbb{E}[\Delta m_i]\sum_{i=1}^k \epsilon_i + \sum_{i=1}^k \epsilon_i^2}{k} - \mathbb{E}[\Delta m_i]^2 \\ &= \frac{\sum_{i=1}^k \epsilon_i^2}{k}, \end{aligned}$$

which implies

$$\sum_{i=1}^k \epsilon_i^2 = k\sigma^2 \Rightarrow \mathbb{E}[\epsilon_i^2] = \sigma^2.$$

Then we can derive that

$$\mathrm{Var}[\omega_i] = \mathbb{E}[\omega_i^2] - (\mathbb{E}[\omega_i])^2 \approx 1 + \frac{\sigma^2}{\tau^2} - 1 = \frac{\sigma^2}{\tau^2} = \frac{\mathrm{Var}[\Delta m_i]}{\tau^2}.$$

From another perspective,

$$\text{Var}[\omega_i] = \mathbb{E}[\omega_i^2] - (\mathbb{E}[\omega_i])^2 = \frac{\boldsymbol{\omega}^\top \boldsymbol{\omega}}{k} - 1,$$

which gives the final conclusion

$$\text{Var}[\Delta m_i] \approx \frac{\tau^2}{k}\boldsymbol{\omega}^\top\boldsymbol{\omega} - \tau^2.$$

□

6.3. Proof of Theorem 1.

Theorem 1. *Suppose Assumptions 1 and 2 hold. We set the stepsize* $\eta_t = \frac{\sum_i \sqrt{\omega_{i,t}/\alpha_{i,t}}}{kL\sum_i \sqrt{\omega_{i,t}\alpha_{i,t}}}$. *Then, the sequence* $\{\theta_t\}_{t=1}^\infty$ *has a subsequence that converges to a Pareto stationary point* θ^*.

Proof. Since $g_i^\top d = \sqrt{\frac{\omega_i}{\alpha_i}}$ and $d = \sum_{i=1}^k \alpha_i g_i$, we have $\|d\|^2 = \sum_i \alpha_i g_i^\top d = \sum_i \sqrt{\omega_i \alpha_i}$. Given that each loss function $\ell_i(\theta)$ is L-smooth, we have

$$\begin{aligned}
\ell_i(\theta_{t+1}) &\le \ell_i(\theta_t) - \eta_t g_{i,t}^\top d_t + \frac{L}{2}\|\eta_t d_t\|^2 \\
&= \ell_i(\theta_t) - \eta_t\sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}} + \frac{L}{2}\eta_t^2\|d_t\|^2 \\
&= \ell_i(\theta_t) - \eta_t\sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}} + \frac{L\eta_t^2}{2}(\sum_{j=1}^k \sqrt{\omega_{j,t}\alpha_{j,t}}).
\end{aligned}$$

Set the learning rate $\eta_t = \frac{\sum_i \sqrt{\omega_{i,t}/\alpha_{i,t}}}{kL\sum_i \sqrt{\omega_{i,t}\alpha_{i,t}}}$. Consider the averaged loss function $\mathcal{L}(\theta) = \frac{1}{k}\sum_i \ell_i(\theta)$, we have

$$\begin{aligned}
\mathcal{L}(\theta_{t+1}) &\le \mathcal{L}(\theta_t) - \eta_t\frac{1}{k}\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}} + \frac{L\eta_t^2}{2}(\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}}) \\
&= \mathcal{L}(\theta_t) - L\eta_t^2(\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}}) + \frac{L\eta_t^2}{2}(\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}}) \\
&= \mathcal{L}(\theta_t) - \frac{L\eta_t^2}{2}(\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}}).
\end{aligned}$$

We can observe that $\sum_{r=0}^t \frac{L\eta_r^2}{2}(\sum_{i=1}^k \sqrt{\omega_{i,r}\alpha_{i,r}}) \le \mathcal{L}(\theta_0) - \mathcal{L}(\theta_{t+1})$. Then, we get

$$\begin{aligned}
\sum_{r=0}^\infty \frac{L\eta_r^2}{2}(\sum_{i=1}^k \sqrt{\omega_{i,r}\alpha_{i,r}}) &= \frac{1}{2Lk^2}\sum_{r=0}^\infty \frac{\sum_{i=1}^k (\sqrt{\omega_{i,r}/\alpha_{i,r}})^2}{\sum_{i=1}^k \sqrt{\omega_{i,r}\alpha_{i,r}}} \\
&< \infty.
\end{aligned}$$

Then, it can be obtained that

$$\lim_{r\to\infty} \frac{\sum_{i=1}^k (\sqrt{\omega_{i,r}/\alpha_{i,r}})^2}{\sum_{i=1}^k \sqrt{\omega_{i,r}\alpha_{i,r}}} = 0. \tag{8}$$

From Eq. 7, we get

$$\left\|\sqrt{\frac{\boldsymbol{\omega_t}}{\boldsymbol{\alpha_t}}}\right\| \ge \sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t)\|\,\boldsymbol{\alpha_t}\|,$$

where $\sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t)$ is the smallest singular value of matrix $\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t$. Denote $\mathbf{1} = [1, \cdots, 1]^\top$ as the length-k vector whose elements are all 1. Note that we have

$$\left\|\sqrt{\frac{\boldsymbol{\omega}}{\boldsymbol{\alpha}}}\right\|^2 = \sum_{i=1}^k \frac{\omega_i}{\alpha_i} \le (\sum_{i=1}^k \sqrt{\frac{\omega_i}{\alpha_i}})^2 = \left\|\sqrt{\frac{\boldsymbol{\omega}}{\boldsymbol{\alpha}}}\right\|_1^2,$$

$$\|\boldsymbol{\alpha}\|_1 = \mathbf{1}^\top\boldsymbol{\alpha} \le \|1\| \cdot \|\boldsymbol{\alpha}\| = \sqrt{k}\|\boldsymbol{\alpha}\|.$$

Combine the above inequalities, we get

$$\begin{aligned}
\left\|\sqrt{\frac{\boldsymbol{\omega_t}}{\boldsymbol{\alpha_t}}}\right\|_1 \ge \left\|\sqrt{\frac{\boldsymbol{\omega_t}}{\boldsymbol{\alpha_t}}}\right\| &\ge \sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t)\|\boldsymbol{\alpha_t}\| \\
&\ge \frac{1}{\sqrt{k}}\sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t)\|\boldsymbol{\alpha_t}\|_1.
\end{aligned}$$

Then, we have

$$\frac{\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}}}{\sum_{i=1}^k \alpha_{i,t}} \ge \frac{1}{\sqrt{k}}\sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t). \tag{9}$$

Furthermore,

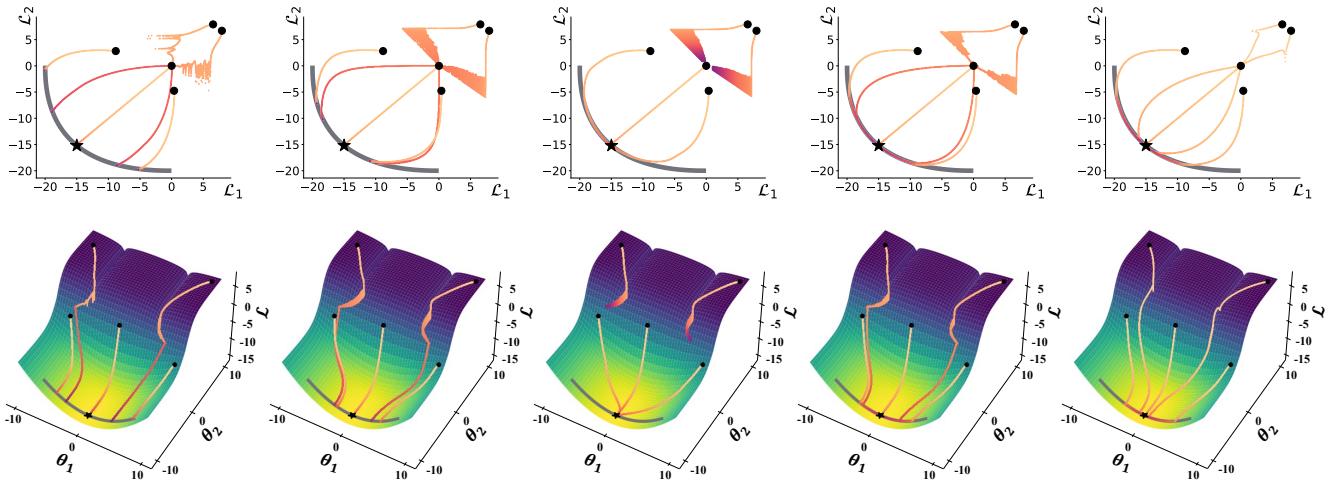
$$\begin{aligned}
\frac{\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}}}{\sum_{i=1}^k \alpha_{i,t}} &= \frac{(\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}})^2}{(\sum_{i=1}^k \alpha_{i,t}) \cdot (\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}})} \\
&= \frac{(\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}})^2}{\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}} + \sum_{i=1}^k \sum_{j=1,j\neq i}^k \alpha_{i,t}\sqrt{\frac{\omega_{j,t}}{\alpha_{j,t}}}} \\
&\le \frac{(\sum_{i=1}^k \sqrt{\frac{\omega_{i,t}}{\alpha_{i,t}}})^2}{\sum_{i=1}^k \sqrt{\omega_{i,t}\alpha_{i,t}}}.
\end{aligned} \tag{10}$$

For any fixed k, it can be concluded from Eq. 8, Eq. 9, and Eq. 10 that

$$\lim_{t\to\infty} \sigma_k(\boldsymbol{\mathcal{G}}_t^\top \boldsymbol{\mathcal{G}}_t) = 0.$$

Since the sequence $\mathcal{L}(\theta_t)$ is monotonically decreasing, we know the sequence θ_t is in the compact sublevel set $\{\theta|\mathcal{L}(\theta) \le \mathcal{L}(\theta_0)\}$. Then, there exists a subsequence θ_{t_j} that converges to $\theta^\star$ where we have $\sigma_k(\boldsymbol{\mathcal{G}}_\star^\top \boldsymbol{\mathcal{G}}_\star) = 0$ and $\boldsymbol{\mathcal{G}}_\star$ denotes the matrix of multiple gradients at $\theta^\star$. Therefore, the gradients at $\theta^\star$ are linearly dependent, and $\theta^\star$ is Pareto stationary.

□



(a) MGDA (b) PCGrad (c) CAGrad (d) Nash-MTL (e) PIVRG(ours)

Figure 2. Comparison of MTL approaches on a challenging synthetic two-task benchmark [22, 30]. We visualize optimization trajectories w.r.t. objectives value ($\mathcal{L}_1$ and $\mathcal{L}_2$, top row), and cumulative objective w.r.t. parameters (θ_1 and θ_2, bottom row). The starting points are indicated by black dots (●), and the Pareto front (see Definition 1) is represented by thick gray lines (▬).

7. Experimental Details

7.1. Toy Example

Following [30, 34], we employ a two-task toy example presented in [22]. The two tasks $\mathcal{L}_1(\theta)$ and $\mathcal{L}_2(\theta)$ are defined on $\theta = (\theta_1, \theta_2)^\top \in \mathbb{R}^2$,

$$\mathcal{L}_1(\theta) = f_1(\theta)g_1(\theta) + f_2(\theta)h_1(\theta)$$
$$\mathcal{L}_2(\theta) = f_1(\theta)g_2(\theta) + f_2(\theta)h_2(\theta),$$

where the functions are defined as follows:

$$f_1(\theta) = \max\big(\tanh(0.5\theta_2), 0\big)$$
$$f_2(\theta) = \max\big(\tanh(-0.5\theta_2), 0\big)$$
$$g_1(\theta) = \log\Big(\max\big(|0.5(-\theta_1 - 7) - \tanh(-\theta_2)|, 5e-6\big)\Big) + 6$$
$$g_2(\theta) = \log\Big(\max\big(|0.5(-\theta_1 + 3) - \tanh(-\theta_2) + 2|, 5e-6\big)\Big) + 6$$
$$h_1(\theta) = \big((-\theta_1 + 7)^2 + 0.1(-\theta_2 - 8)^2\big)/10 - 20$$
$$h_2(\theta) = \big((-\theta_1 - 7)^2 + 0.1(-\theta_2 - 8)^2\big)/10 - 20.$$

Following [4, 23, 30], we use five distinct starting points $\{(-8.5, 7.5), (0, 0), (9.0, 9.0), (-7.5, -0.5), (9.0, -1.0)\}$. The Adam optimizer is employed with a learning rate of 1×10^{-3}. The 2D and 3D optimization trajectories are shown in Fig. 2. On one hand, while other MTL methods (Fig. 2a to 2d) exhibit oscillations around local minima, leading to noisy optimization trajectories, our approach can swiftly escape these regions of local minima through guidance from the performance-informed weighting strategy. On the other hand, approaches designed to find a Pareto-stationary solution halt upon reaching the Pareto front (e.g. Fig. 2a and Fig. 2b), but PIVRG continues to transfer along the Pareto front and converges to a more balanced Pareto-optimal solution.

7.2. Experimental Results with Standard Errors

We followed the experimental setup from [4, 23, 30], and the results for the baseline methods are taken from their original papers. PIVRG's results along with standard errors is presented in Table 6, 7 and 8.

Table 6. Results on CityScapes (2 tasks) and CelebA (40 tasks) datasets. Each experiment is repeated over 3 random seeds and the mean and stderr are reported.

Method	CityScapes					CelebA
	Segmentation		Depth		$\Delta m(\%) \downarrow$	$\Delta m(\%) \downarrow$
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓		
PIVRG (mean)	75.82	93.65	0.0126	27.87	-0.54	-0.96
PIVRG (stderr)	±0.05	±0.04	±0.0002	±0.24	±0.34	±0.34

7.3. Additional Results on Performance Variance

In Fig. 3 and Fig. 4, we show that both $\boldsymbol{\omega}^\top\boldsymbol{\omega}$ and $\text{Var}[\Delta m_i]$ decrease progressively throughout the optimization process, validating the effectiveness of our dynamic weights which serve as regularizers. In Table 10, 11 and 12, we compare the detailed performance drop Δm and performance variance $\text{Var}[\Delta m_i]$ with existing methods, the results show that PIVRG not only achieves SOTA performance on vari-

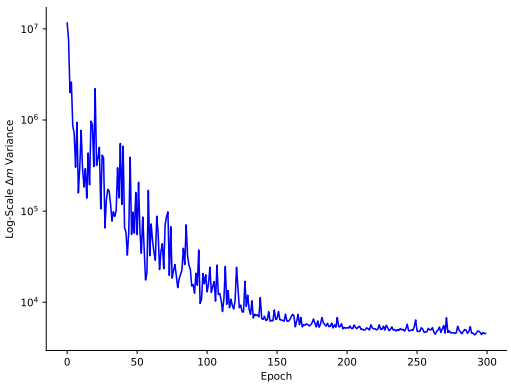


Figure 3. Performance Variance on QM9 dataset.

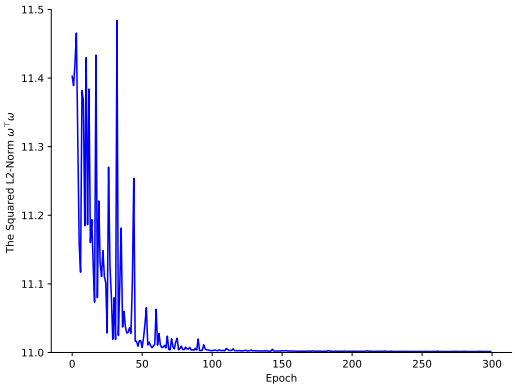


Figure 4. The squared L^2 norm of $\boldsymbol{\omega}$, i.e. $\boldsymbol{\omega}^\top\boldsymbol{\omega}$.

Table 7. Results on NYU-v2 dataset (3 tasks). Each experiment is repeated over 3 random seeds and the mean and stderr are reported.

Method	Segmentation		Depth		Surface Normal					$\Delta m(\%)\downarrow$
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Dist ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
PIVRG (mean)	39.90	65.74	0.5365	0.2243	24.30	18.80	30.95	58.26	70.38	-6.50
PIVRG (stderr)	±0.43	±0.21	±0.0007	±0.0014	±0.07	±0.09	±0.12	±0.18	±0.16	±0.24

Table 8. Results on QM-9 dataset (11 tasks). Each experiment is repeated over 3 random seeds and the mean and stderr are reported.

Method	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	$\Delta m(\%)\downarrow$
	MAE ↓											
PIVRG (mean)	0.125	0.226	94.80	81.98	1.41	3.87	57.79	57.90	58.09	57.86	0.085	33.6
PIVRG (stderr)	±0.0022	±0.0078	±2.829	±1.349	±0.0301	±0.0438	±0.68	±0.72	±0.70	±0.68	±0.0005	±2.31

Table 9. Detailed results on QM9 (11-task) dataset. Each experiment is repeated 3 times with different random seeds and the average is reported.

METHOD	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	MR↓	$\Delta m(\%)\downarrow$
	MAE ↓												
STL	0.067	0.181	60.57	53.91	0.502	4.53	58.8	64.2	63.8	66.2	0.072		
LS	0.106	0.325	**73.57**	89.67	5.19	14.06	143.4	144.2	144.6	140.3	0.128	9.09	177.6
SI	0.309	0.345	149.8	135.7	**1.00**	4.50	**55.3**	**55.75**	**55.82**	**55.27**	0.112	5.55	77.8
RLW	0.113	0.340	76.95	92.76	5.86	15.46	156.3	157.1	157.6	153.0	0.137	10.64	203.8
DWA	0.107	0.325	74.06	90.61	5.09	13.99	142.3	143.0	143.4	139.3	0.125	8.82	175.3
UW	0.386	0.425	166.2	155.8	1.06	4.99	66.4	66.78	66.80	66.24	0.122	7.27	108.0
MGDA	0.217	0.368	126.8	104.6	3.22	5.69	88.37	89.4	89.32	88.01	0.120	8.91	120.5
PCGRAD	0.106	0.293	75.85	88.33	3.94	9.15	116.36	116.8	117.2	114.5	0.110	7.27	125.7
CAGRAD	0.118	0.321	83.51	94.81	3.21	6.93	113.99	114.3	114.5	112.3	0.116	8.18	112.8
IMTL-G	0.136	0.287	98.31	93.96	1.75	5.69	101.4	102.4	102.0	100.1	0.096	7.18	77.2
NASH-MTL	**0.102**	0.248	82.95	**81.89**	2.42	5.38	74.5	75.02	75.10	74.16	0.093	4.36	62.0
FAMO	0.15	0.30	94.0	95.2	1.63	4.95	70.82	71.2	71.2	70.3	0.10	5.73	58.5
FAIRGRAD	0.117	0.253	87.57	84.00	2.15	5.07	70.89	71.17	71.21	70.88	0.095	4.73	57.9
PIVRG	0.125	**0.226**	94.80	81.98	1.41	**3.87**	57.79	57.90	58.09	57.86	**0.085**	**3.00**	**33.6**

Table 10. Comparison of Δm and performance variance for different methods on the NYUv2 dataset.

method	LS	SI	RLW [20]	DWA [26]	UW [18]
$\Delta m(\%)$	5.59	4.39	7.78	3.57	4.05
Var$[\Delta m_i]$	259.13	247.77	205.32	191.93	190.73

method	MGDA [33]	PCGrad [38]	GradDrop [10]	CAGrad [22]	IMTL-G [24]
$\Delta m(\%)$	1.38	3.97	3.58	0.20	-0.76
Var$[\Delta m_i]$	68.65	173.67	204.45	137.94	124.03

method	Moco [14]	Nash-MTL [30]	FAMO [23]	FairGrad [4]	**PIVRG (Ours)**
$\Delta m(\%)$	0.16	-4.04	-4.10	-4.66	**-6.50**
Var$[\Delta m_i]$	163.07	108.03	74.66	71.59	**52.21**

Table 11. Comparison of Δm and performance variance for different methods on the QM9 dataset.

method	LS	SI	RLW [20]	DWA [26]	UW [18]
$\Delta m(\%)$	177.6	77.8	203.8	175.3	108.0
Var$[\Delta m_i]$	59317.63	11807.60	77380.19	56660.16	18171.92

method	MGDA [33]	PCGrad [38]	CAGrad [22]	IMTL-G [24]	Nash-MTL [30]
$\Delta m(\%)$	120.5	125.7	112.8	77.2	62.0
Var$[\Delta m_i]$	20533.84	31570.73	18343.53	3309.90	10385.12

method	FAMO [23]	FairGrad [4]	**PIVRG (Ours)**
$\Delta m(\%)$	58.5	57.9	**33.6**
Var$[\Delta m_i]$	3963.84	7705.27	**3196.32**

Table 12. Comparison of Δm and performance variance for different methods on the CityScapes dataset.

method	LS	SI	RLW [20]	DWA [26]	UW [18]
$\Delta m(\%)$	22.60	14.11	24.38	21.45	5.89
Var$[\Delta m_i]$	803.24	133.23	879.98	630.71	21.32

method	MGDA [33]	PCGrad [38]	GradDrop [10]	CAGrad [22]	IMTL-G [24]
$\Delta m(\%)$	44.14	18.29	23.73	11.64	11.10
Var$[\Delta m_i]$	3588.05	466.50	871.26	220.86	261.86

method	MoCo [14]	Nash-MTL [30]	FAMO [23]	FairGrad [4]	**PIVRG (Ours)**
$\Delta m(\%)$	9.90	6.82	8.13	5.18	**-0.54**
Var$[\Delta m_i]$	126.75	128.77	73.43	53.26	**1.55**

ous benchmarks but also produces the lowest performance variance, indicating a fairer optimization.

8. Comparison With Other Methods

In this section, we present a concise overview of representative loss-based and gradient-based approaches used in multitask or multiobjective optimization, and provide a brief analysis of the characteristics of each method.

8.1. Loss-Based Methods

Linear scalarization (LS). LS aims to directly optimize the average of all task losses. The optimization objective for LS is given by

$$\mathcal{L}(\theta) = \min_{\theta} \frac{1}{k} \sum_{i=1}^{k} \ell_i(\theta),$$

where $\ell_i(\theta)$ represents the loss for task i. LS focuses on minimizing the overall average loss, treating each task equally without considering individual task difficulties or imbalances.

Scale-Invariant (SI). The SI method aims to optimize the logarithmic mean of all task losses. The optimization objective for SI is given by

$$\min_{\theta} \frac{1}{k} \sum_{i=1}^{k} \log(\ell_i(\theta)),$$

where $\ell_i(\theta)$ represents the loss for task i. The advantage of SI is that it is invariant to any scalar multiplication of task losses, allowing it to handle varying loss scales effectively.

Dynamic Weight Average (DWA) [26]. It is a heuristic for adjusting task weights based on rates of loss changes. The optimization objective is a weighted sum of all task losses, where the weights are λ_i:

$$\min_{\theta} \sum_{i=1}^{k} \lambda_i \ell_i(\theta).$$

Similar to PIVRG, it also uses a softmax with temperature to determine the weights such that they sum to k. However, the softmax argument is $w_{i,t} = \ell_{i,t}/\ell_{i,t-1}$, which considers the relative change at the loss-level.

Random Loss Weighting (RLW) [20]. The optimization objective of RLW is also a weighted sum of all task losses, where the weights are λ_i:

$$\min_{\theta} \sum_{i=1}^{k} \lambda_i \ell_i(\theta).$$

Unlike previous methods, RLW simply samples from a normal distribution and applies softmax to obtain the weights. The authors found that even this simple modification leads to better performance. They argue that RLW provides a higher probability of escaping local minima compared to existing models with fixed task weights, resulting in improved generalization ability.

Fast Adaptive Multitask Optimization (FAMO) [23]. FAMO aims to decrease all task losses at an equal rate at each step as much as possible. The optimization objective is:

$$\max_{d \in \mathbb{R}^n} \min_{i \in [k]} \frac{\ell_{i,t} - \ell_{i,t+1}}{\eta_t \ell_{i,t}} - \frac{1}{2}\|d_t\|^2,$$

where η_t is the current step size. By amortizing over time, the authors propose a fast approximation to the solution, thus achieving highly competitive results while maintaining efficiency.

8.2. Gradient-Based Methods

Multiple Gradient Descent Algorithm (MGDA) [33]. The MGDA algorithm is one of the earliest gradient manipulation methods for multitask learning. In MGDA, the

per step update d_t is found by solving

$$\max_{d\in\mathbb{R}^n}\min_{i\in[k]} g_{i,t}^\top d - \frac{1}{2}\|d\|^2.$$

As a result, the solution d^* of MGDA optimizes the worst improvement across all tasks or equivalently seeks an equal descent across all task losses as much as possible. However, in practical applications, MGDA often encounters slow convergence due to the potential for d^* to be quite small. For instance, if one task has a very small loss scale, the advancement of other tasks becomes constrained by the progress made on this particular task.

Projecting Gradient Descent (PCGrad) [39]. PCGrad initializes $v_{\text{PC}}^i = g_{i,t}$, then for each task i, PCGrad loops over all task $j \neq i$:

$$v_{\text{PC}}^i \leftarrow v_{\text{PC}}^i - \frac{{v_{\text{PC}}^i}^\top g_{j,t}}{\|\ell_{j,t}\|^2} g_{j,t} \quad \text{if} \quad {v_{\text{PC}}^i}^\top g_{j,t} < 0.$$

In the end, PCGrad produces $d_t = \frac{1}{k}\sum_{i=1}^k v_{\text{PC}}^i$. Due to the construction, PCGrad will also help improve the "worst improvement" across all tasks since the "conflicts" have been removed. However, due to the stochastic iterative procedural of this algorithm, it is hard to understand PCGrad from a first principle approach.

Conflict-averse Gradient Descent (CAGrad) [22]. In CAGrad, d_t is found by solving

$$\max_{d\in\mathbb{R}^m}\min_{i\in[k]} g_{i,t}^\top d \quad \text{s.t.} \quad \|d - \nabla\ell_{0,t}\| \leq c\,\|\nabla\ell_{0,t}\|,$$

where $\ell_{0,t} = \frac{1}{k}\sum_{i=1}^k \ell_{i,t}$. CAGrad aims to determine an update d_t that maximizes the "worst improvement" while ensuring that the overall average loss decreases. By adjusting the hyperparameter c, CAGrad can replicate the behavior of MGDA when $c \to \infty$ and revert to the standard averaged gradient descent when $c \to 0$.

Impartial Multi-Task Learning (IMTL-G) [24]. IMTL-G finds d_t such that it shares the same cosine similarity with any task gradients:

$$\forall i \neq j, \quad d_t^\top \frac{g_{i,t}}{\|g_{i,t}\|} = d_t^\top \frac{g_{j,t}}{\|g_{j,t}\|},$$
$$\text{and} \quad d_t = \sum_{i=1}^k w_{i,t} g_{i,t}, \quad \text{for some} \quad w_t \in \mathbb{S}_k.$$

The constraint that $d_t = \sum_{i=1}^k w_{i,t} g_{i,t}$ is for preventing the problem from being under-determined. We can view IMTL-G as the equal angle descent, where the objective is to find d such that

$$\forall i \neq j, \qquad \cos(d, g_{i,t}) = \cos(d, g_{j,t}).$$

Nash-MTL [30]. Nash-MTL finds d_t by solving a bargaining game treating the local improvement of each task loss as the utility for each task:

$$\max_{d_t\in\mathbb{R}^n, \|d_t\|\leq\epsilon^2} \sum_{i=1}^k \log\left(g_{i,t}^\top d_t\right).$$

Note that the objective of Nash-MTL implicitly assumes that there exists d_t such that $\forall\ i,\ \ g_{i,t}^\top d_t > 0$, otherwise we reach the Pareto front. In our proposed PIVRG, we also adopt this assumption.

α-Fair Resource Allocation (FairGrad) [4]. FairGrad is inspired by fair resource allocation in communication networks. They treat the optimization in MTL as a resource allocation problem and apply the α-fairness framework:

$$U_\alpha(d) = \begin{cases} \sum_{i=1}^k \frac{u_i(d)^{1-\alpha}}{1-\alpha} & \text{if } \alpha > 0, \alpha \neq 1 \\ \sum_{i=1}^k \log(u_i(d)) & \text{if } \alpha = 1 \end{cases}$$

They also consider $g_i^\top d$ as the utility of task i. By introducing the α-fair framework, FairGrad achieves different types of fairness at the gradient level, yielding surprising results. It is noteworthy that most existing methods can also be categorized under the α-fair framework. For instance, LS is a special case when $\alpha = 0$, Nash-MTL corresponds to $\alpha = 1$, and MGDA is a special case as α approaches infinity. Similar to these methods, the basic optimization objective in (2) can also be viewed as a special case of α-fairness. However, our derivation is from the perspective of minimizing the average optimization steps for tasks, and this is not our main contribution.

8.3. Advantages of Our Method

Through the analysis of the aforementioned methods, we found that since loss-based methods cannot obtain the accurate gradient for each task, they primarily achieve fairness at the loss level through various scaling and weighted averaging of the loss. A major idea of gradient-based methods is to alleviate gradient conflict during the optimization process to achieve fairness at the gradient level. Additionally, some gradient-based methods use the first-order Taylor expansion to design utility functions, approximating the loss difference with $g_i^\top d$, thereby incorporating loss-level information.

However, only our proposed PIVRG considers utilizing the variance of performance drop as a fairness indicator to represent fairness in the optimization process of MTL. Extensive experiments demonstrate that PIVRG not only achieves state-of-the-art performance but also realizes further fair optimization, mitigating the common task imbalance phenomenon observed in previous methods. Integrating our dynamically designed weighting strategy based on performance-level information into existing methods can

significantly enhance their performance and reduce the variance of performance drop, achieving more equitable results. This further confirms the potential of our method and its contribution to the MTL community.

8.4. Discussion with Task Grouping Techniques

Task grouping [15, 36] provides another perspective for MTL. In our approach, we aim to use a single model that performs well across multiple tasks simultaneously. However, task similarity is not uniform across all tasks, and some tasks may be better suited to be trained together while others are not. This leads to a trade-off: by discarding tasks with lower similarity, tasks with higher similarity can be optimized more effectively. [15] can effectively determine which tasks should be trained together. However, after grouping, each task group requires a separate model for training, which incurs additional time and memory overhead. Notably, our method is orthogonal to task grouping techniques. After performing task grouping, our method can be applied to train the tasks within each group, potentially leading to better results.