Towards Consistent Multi-Task Learning: Unlocking the Potential of Task-Specific Parameters

Supplementary Material

6. Experimental Details

To ensure the reliability of the experimental results and alleviate the influence of random errors, the results presented in the main text follow the methodology of prior works [3, 22, 29], where each experiment is run three times with different random seeds, and the average value is reported. Table 5, 7 and 8 below present the specific performance of ConsMTL across different benchmarks, including the corresponding standard errors.

Fig. 5 shows the cosine similarity matrices of gradients for the shared representation z at different training stages, including the first 60 epochs and the 300th epoch at the end of training, for (1) the conventional MTL method LS (linear scalarization), which optimizes $\mathcal{L} = \sum_{i=1}^{k} \ell_i$; (2) upper-level optimization only, which is discussed in Section 3.2; and (3) introducing lower-level optimization for task-specific parameters, i.e., ConsMTL. In the early stages of training, different tasks are more easily aligned for collaborative optimization, and the introduction of lower-level optimization significantly enhances the cosine similarity of gradients from different tasks. This effectively mitigates gradient conflicts, allowing the shared parameters to learn a more unified representation. Towards the end of training, the gradient similarity stabilizes or even decreases, with the gradient directions becoming linearly correlated along the Pareto frontier, consistent with the observation in [16]. At Epoch 300, when the training has largely converged, ConsMTL still maintains a certain level of gradient similarity, indicating that task-specific parameters are better equipped to capture task-specific information, which aligns with the fundamental design goal of MTL. This fully demonstrates the potential of our method and its contribution to the MTL community.

Table 5. 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 ↓		
ConsMTL (mean)	75.57	93.32	0.0131	26.41	-0.59	-1.42
ConsMTL (stderr)	±0.66	±0.10	±0.0002	±0.25	±0.28	±0.37

7. Detailed Computational Overhead

To analyze the additional computational overhead introduced by lower-level optimization, we conducted experiments on NYUv2, QM9, and CelebA, reporting the per-epoch computational time. The results, shown in Table 6, demonstrate that lower-level optimization introduces only marginal overhead. All experiments are performed on a single NVIDIA 4090 GPU.

Table 6. Detailed per-epoch computational overhead on different datasets. "ULO only" means using only upper-level optimization.

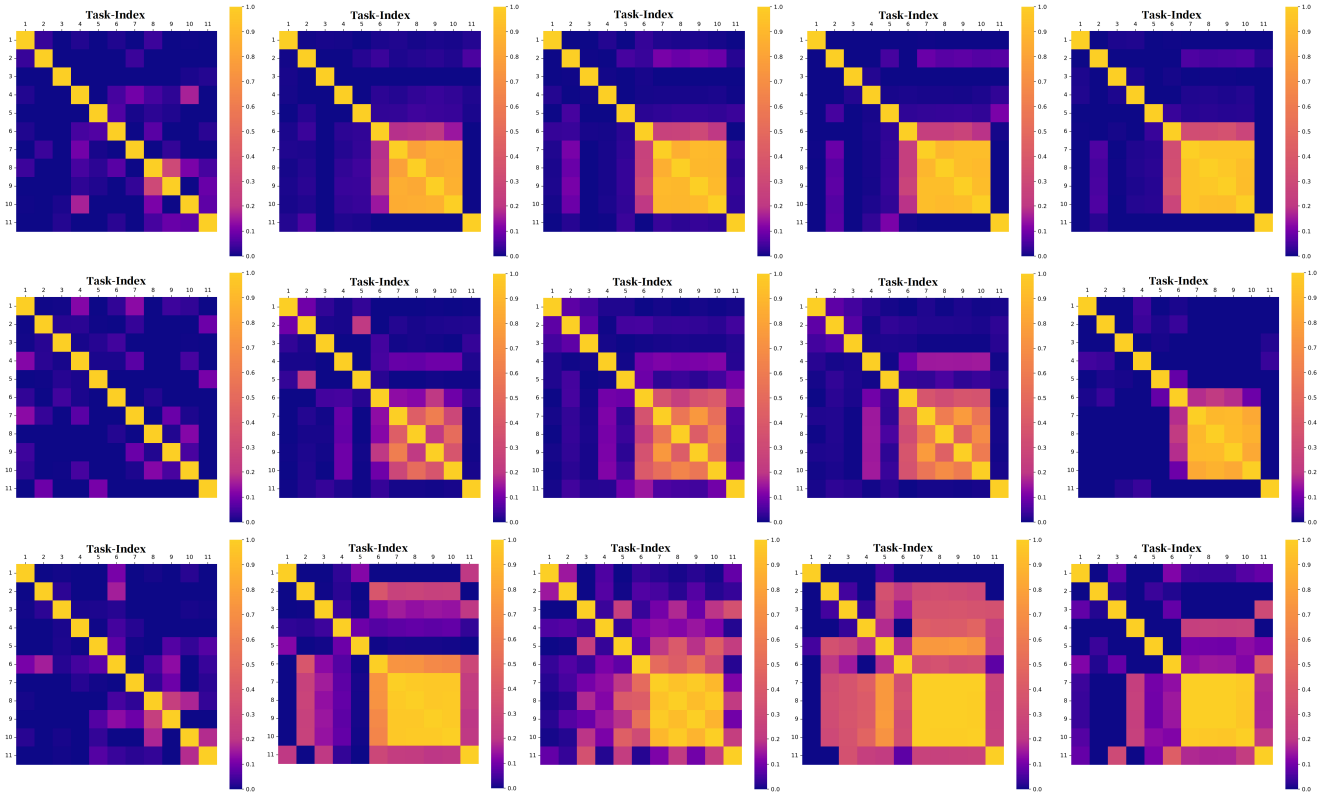
Method	NYUv2	QM9	CelebA
ULO only	5.7min	4.5min	15.9min
ConsMTL	6.4min	5.6min	17.8min

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	
ConsMTL (mean)	40.33	65.32	0.5491	0.2151	24.35	18.80	31.06	58.28	70.31	-6.72
ConsMTL (stderr)	±0.15	±0.20	±0.007	±0.0015	±0.034	±0.18	±0.09	±0.23	±0.19	±0.21

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 ↓											
ConsMTL (mean)	0.115	0.202	82.69	67.58	1.61	3.33	48.84	49.04	49.07	49.63	0.077	23.2
ConsMTL (stderr)	±0.0013	±0.0051	±0.69	±2.55	±0.049	±0.072	±1.17	±1.13	±1.09	±1.16	±0.0005	±2.03



(a) Epoch=0 (a) Epoch=20 (a) Epoch=40 (a) Epoch=60 (a) Epoch=300

Figure 5. The cosine similarity matrix of gradients for shared representation $\boldsymbol{z}$ across different tasks during training on QM9 (11-task). Negative values are clipped to 0. The top row represents the conventional MTL method LS (linear scalarization). The middle row represents the case using only upper-level optimization, which can be considered as a naive gradient-based method. The bottom row introduces lower-level optimization for task-specific parameters, i.e., ConsMTL.