In this document, we compare `LibMTL` and the recent popular implementations (i.e., CAGrad[1] and Nash-MTL[2]). There are something required special attention as follows.

- We comment out the code about reproducibility (`https://github.com/median-research-group/LibMTL/blob/main/LibMTL/utils.py#L18-L20`) for faster running speed;
- Each experiment is repeated over three random seeds and the average value is reported.

Table 1: Hyperparameters Configuration.

	Configuration
common	`GPU`: NVIDIA GeForce RTX 3090 `multi_input`: False; `aug`: True `train_bs`: 2; `test_bs`: 2; epochs: 200 `optim`: Adam; `lr`: 0.0001; `weight_decay`: 0.0 `scheduler`: step; `step_size`: 100; `gamma`: 0.5
MGDA	`rep_grad`: False; `mgda_gn`: none
DWA	`T`: 2
GradDrop	`leak`: 0.0
CAGrad	`calpha`: 0.5; `rescale`: 1
Nash-MTL	`update_weights_every`: 1 `optim_niter`: 20; `max_norm`: 1.0

Table 2: Performance on the *NYUv2* dataset with 3 tasks on **SegNet+MTAN** architecture.

		Segmentation		**Depth**		**Normal**				
						Angle Distance		**Within t°**		
		mIoU↑	**PAcc↑**	**AErr↓**	**RErr↓**	**Mean↓**	**MED↓**	**11.25↑**	**22.5↑**	**30↑**
EW	[4, 6]	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08
	`LibMTL`	40.89	66.14	0.5524	0.2347	27.27	22.41	24.38	50.18	63.36
DWA [5]	[6]	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39
	`LibMTL`	40.50	65.65	0.5358	0.2222	27.58	22.93	23.30	49.16	62.57
UW [2]	[6]	36.87	63.17	0.5446	0.2260	27.04	22.61	23.54	49.05	63.65
	`LibMTL`	39.34	64.88	0.5294	0.2242	26.47	21.30	25.86	52.40	65.47
MGDA [7]	[4, 6]	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36
	`LibMTL`	29.91	60.06	0.5901	0.2432	24.55	18.63	30.49	58.02	70.14
PCGrad [8]	[4, 6]	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14
	`LibMTL`	40.61	65.89	0.5416	0.2287	26.97	22.05	24.68	50.90	64.05
GradDrop [1]	[4, 6]	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87
	`LibMTL`	40.00	65.61	0.5886	0.2517	28.05	23.54	22.81	48.01	61.33
CAGrad [4]	[4, 6]	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58
	`LibMTL`	41.27	66.70	0.5409	0.2356	25.35	19.81	28.44	55.47	68.05
Nash-MTL [6]	[6]	40.13	65.93	0.5261	0.2171	25.26	20.08	28.40	55.47	68.15
	`LibMTL`	40.66	66.25	0.5339	0.2266	25.11	19.59	28.70	55.97	68.52
RLW [3]	[6]	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62
	`LibMTL`	38.82	64.45	0.5718	0.2366	28.09	23.65	22.54	47.76	61.27

[1] `https://github.com/Cranial-XIX/CAGrad`

[2] `https://github.com/AvivNavon/nash-mtl`

References

[1] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Neural Information Processing Systems*, 2020.

[2] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[3] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

[4] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Neural Information Processing Systems*, 2021.

[5] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[6] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, 2022.

[7] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Neural Information Processing Systems*, 2018.

[8] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Neural Information Processing Systems*, 2020.