Journal of Machine Learning Research 24 (2023) 1-7 Submitted 4/22; Revised 7/23; Published 7/23

LibMTL: A Python Library for Deep Multi-Task Learning

Baijiong Lin[*] BJ.LIN.EMAIL@GMAIL.COM
The Hong Kong University of Science and Technology (Guangzhou)

Yu Zhang[†] YU.ZHANG.UST@GMAIL.COM
Department of Computer Science and Engineering, Southern University of Science and Technology
Peng Cheng Laboratory

Editor: Alexandre Gramfort

Abstract

This paper presents `LibMTL`, an open-source `Python` library built on `PyTorch`, which provides a unified, comprehensive, reproducible, and extensible implementation framework for Multi-Task Learning (MTL). `LibMTL` considers different settings and approaches in MTL, and it supports a large number of state-of-the-art MTL methods, including 13 optimization strategies and 8 architectures. Moreover, the modular design in `LibMTL` makes it easy to use and well extensible, thus users can easily and fast develop new MTL methods, compare with existing MTL methods fairly, or apply MTL algorithms to real-world applications with the support of `LibMTL`. The source code and detailed documentations of `LibMTL` are available at `https://github.com/median-research-group/LibMTL` and `https://libmtl.readthedocs.io`, respectively.

Keywords: Multi-Task Learning, Python, PyTorch

1. Introduction

Multi-Task Learning (MTL) (Caruana, 1997; Zhang and Yang, 2022) is an important area in both machine learning and industrial communities. By learning several related tasks simultaneously, this learning paradigm could not only improve the generalization performance but also reduce the storage cost and inference time, thus it has been applied to many real-world scenarios such as autonomous driving, natural language processing, recommendation system, robotic control, bioinformation, and so on (Zhang and Yang, 2022). Although many State-Of-The-Art (SOTA) MTL models have been proposed recently, most of them are implemented in their respective frameworks with different experimental details or there is no public implementation. Therefore, it is not easy to extend existing MTL algorithms to real-world applications or make a fair comparison with them when designing new MTL models.

To remedy such situation, we develop a `Python` library for MTL called `LibMTL`, which has three key features. Firstly, `LibMTL` provides a unified code base to cover different MTL settings such as the single-input and multi-input problems. Hence, it allows a convenient, fair, and consistent comparison between different MTL algorithms in various application

∗. This work was done when the first author worked at Southern University of Science and Technology.
†. Corresponding author.

scenarios. Secondly, built on `PyTorch` (Paszke et al., 2019), `LibMTL` has supported lots of SOTA MTL models, especially deep MTL models, including 13 optimization strategies and 8 MTL architectures. Thirdly, `LibMTL` follows modular design principles, which allows users to flexibly and conveniently add customized components and make personalized modifications. Therefore, users can easily and fast develop new MTL models or apply existing MTL algorithms to their own application scenarios with the support of `LibMTL`.

2. Settings and Approaches in MTL

Suppose there are T tasks and each task t has its corresponding data set $\mathcal{D}_t = \{\mathbf{X}_t, \mathbf{Y}_t\}$. Let $f(\cdot; \theta, \psi_{1:T})$ denotes an MTL model with task-shared parameters θ and task-specific parameters $\psi_{1:T}$. MTL aims to train a model f on all data sets $\mathcal{D}_{1:T}$ and expects f to perform well on each task. There are usually two settings in MTL: *the single-input case* where each task has the same input data, i.e., $\mathbf{X}_m = \mathbf{X}_n$ for any $m \neq n$, and *the multi-input case* where each task has its own input data, i.e., $\mathbf{X}_m \neq \mathbf{X}_n$ for any $m \neq n$. Those two settings rely on concrete application scenarios and they are different in the training implementation.

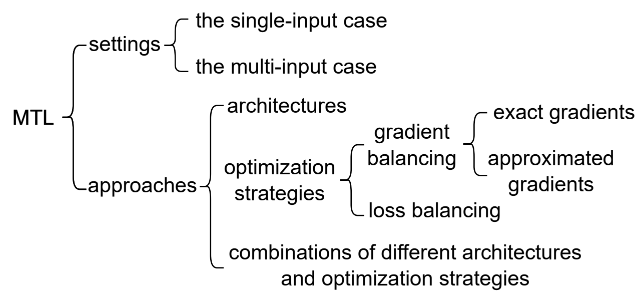

Figure 1: Categories of settings and approaches in MTL.

There are two main lines of research for MTL. The first line is to design the *architecture* in deep neural networks for MTL and it directly determines which parameters are shared and how to share. The second line is to design the *optimization strategy* for MTL. Since how to balance multiple training losses in MTL directly affects the update of the task-shared parameters θ, several methods are proposed to balance the losses or gradients of all the tasks in different ways, which are called loss balancing methods and gradient balancing methods, respectively. Moreover, gradient balancing methods need to calculate the gradients of the task-shared parameters θ for every task, which may be computationally intensive when the number of shared parameters or tasks is large. Thus, Sener and Koltun (2018) propose to use gradients of feature representations to approximate the exact gradients of shared parameters, which significantly reduces the computational cost and is followed by other gradient balancing methods such as GradDrop (Chen et al., 2020) and IMTL (Liu et al., 2021b). Obviously, those two ways to calculate gradients are different in implementations. Noticeably, those two lines of research are almost orthogonal to each other as the optimization methods are mainly related to the objective function, while the design of the architecture is to learn relationships between tasks. Thus, optimization strategies can be seamlessly combined with architectures to further improve the performance of MTL.

To summarize, as shown in Figure 1, MTL has two settings and its learning approaches can be divided into three categories.

3. The `LibMTL` Library

In this section, we introduce the `LibMTL` library, which provides a unified and easy-to-use framework for MTL as mentioned in Section 2. In Section 3.1, we introduce MTL methods implemented in `LibMTL`, which enables consistent and reproducible comparisons between different MTL algorithms. In Section 3.2, we present the modular design in `LibMTL`, which allows flexible and extensible customization for new MTL methods or potential MTL applications. In Section 3.3, we compare different MTL models on a benchmark data set based on the `LibMTL` library. In Section 3.4, we show that `LibMTL` is more comprehensive and up-to-date than existing MTL libraries.

3.1 Supported MTL Methods

Currently, `LibMTL` supports 13 optimization strategies, namely, Equal Weighting (**EW**), Gradient Normalization (**GradNorm**) (Chen et al., 2018), Uncertainty Weights (**UW**) (Kendall et al., 2018), **MGDA** (Sener and Koltun, 2018), Dynamic Weight Average (**DWA**) (Liu et al., 2019), Geometric Loss Strategy (**GLS**) (Chennupati et al., 2019), Projecting Conflicting Gradient (**PCGrad**) (Yu et al., 2020), Gradient sign Dropout (**GradDrop**) (Chen et al., 2020), Impartial Multi-Task Learning (**IMTL**) (Liu et al., 2021b), Gradient Vaccine (**GradVac**) (Wang et al., 2021), Conflict-Averse Gradient descent (**CAGrad**) (Liu et al., 2021a), **Nash-MTL** (Navon et al., 2022), and Random Weighting (**RW**) (Lin et al., 2022). Moreover, it supports 8 MTL architectures, i.e., Hard Parameter Sharing (**HPS**) (Caruana, 1993), **Cross-stitch** Networks (Misra et al., 2016), Multi-gate Mixture-of-Experts (**MMoE**) (Ma et al., 2018), Multi-Task Attention Network (**MTAN**) (Liu et al., 2019), Customized Gate Control (**CGC**) (Tang et al., 2020), Progressive Layered Extraction (**PLE**) (Tang et al., 2020), Learning to Branch (**LTB**) (Guo et al., 2020), and **DSelect-k** (Hazimeh et al., 2021). Besides, `LibMTL` supports combinations of each optimization strategy and each architecture.

3.2 The Modular Design of `LibMTL`

Figure 2 shows the overall framework of `LibMTL`, which is divided into different functional modules to allow users to flexibly and conveniently add customized designs or modifications in any module.

In `LibMTL`, each module has different functionalities. The `Dataloader` module is responsible for data pre-processing and loading. The `LibMTL.loss` module defines loss functions for each task. The `LibMTL.metrics` module defines evaluation metrics for all the tasks. The above three modules are highly dependent on the MTL problem under investigation. The `LibMTL.config` module is responsible for all the configuration parameters involved in the training process, such as the MTL setting (i.e., the multi-input case or not), possible hyper-parameters of optimization strategies and architectures, the training configuration (e.g., the batch size, the running epoch, the random seed, and the learning rate), and so on. This module adopts command-line arguments to enable users to set those configuration parameters conveniently. The `LibMTL.Trainer` module provides a unified framework for the training process under different MTL settings and for different MTL approaches as introduced in Section 2. The `LibMTL.utils` module implements useful functionalities for

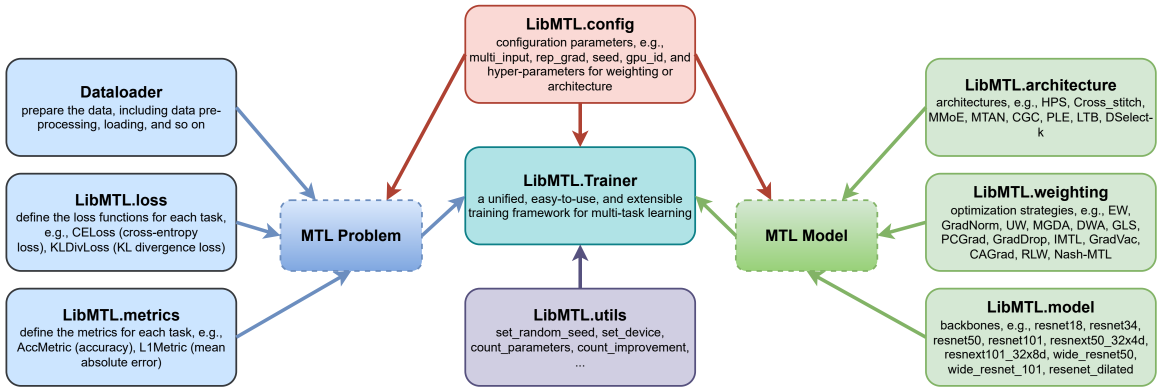

Figure 2: The overall framework of `LibMTL`.

the training process such as calculating the total number of parameters in an MTL model. The `LibMTL.architecture` and `LibMTL.weighting` modules contain the implementations of various architectures and optimization strategies, respectively, as introduced in Section 3.1. The `LibMTL.model` module includes some popular backbone networks (e.g., ResNet). The last three modules are highly related to MTL models.

Noticeably, such modular design makes `LibMTL` easy to use and well-extensible. For example, when applying to new applications, users only need to prepare the new dataloaders and select (or re-define) appropriate loss and metric functions, and they can use existing MTL methods implemented in `LibMTL`. Besides, for researchers to develop new MTL methods such as new architectures, they can easily implement their new method with the support of `LibMTL`, make a fair comparison with existing models, and combine the new architecture with modern optimization methods based on `LibMTL`.

3.3 Performance Comparison

In Table 1, we compare different MTL methods on the *NYUv2* data set (Silberman et al., 2012) and set up a benchmark for MTL. The *NYUv2* data set is an indoor scene understanding data set and has been used extensively in the MTL literature. It contains 3 tasks: semantic segmentation (denoted by Segmentation), depth estimation (denoted by Depth), and surface normal prediction (denoted by Normal). The implementation details and evaluation metrics are following Lin et al. (2022).

3.4 Comparison with Related Libraries

There are some libraries that have been developed for MTL recently. For example, RMTL (Cao et al., 2019) is implemented in `R` to support shallow MTL methods such as linear regularized methods. Another library, i.e., MTLV (Rahimi et al., 2021), only provides a limited number of MTL architectures for natural language processing. Compared with them, `LibMTL` is more comprehensive and up-to-date. Firstly, `LibMTL` covers more settings and approaches as introduced in Section 2, which means that `LibMTL` can be applied to more application scenarios. Secondly, `LibMTL` implements more SOTA MTL models, especially those based on deep neural networks.

Methods	Segmentation		Depth		Normal				
					Angle Distance		Within t°		
	mIoU↑	**PAcc↑**	**AErr↓**	**RErr↓**	**Mean↓**	**MED↓**	**11.25↑**	**22.5↑**	**30↑**
Different optimization strategies on HPS architecture									
EW	53.93	75.53	0.3825	0.1577	23.57	17.01	35.04	60.99	72.05
GradNorm	53.91	75.38	0.3842	0.1571	23.17	16.62	35.80	61.90	72.84
UW	54.29	75.64	0.3815	0.1583	23.48	16.92	35.26	61.17	72.21
MGDA	53.52	74.76	0.3852	0.1566	22.74	16.00	37.12	63.22	73.84
DWA	54.06	75.64	0.3820	0.1564	23.70	17.11	34.90	60.74	71.81
GLS	54.59	76.06	0.3785	0.1555	22.71	16.07	36.89	63.11	73.81
PCGrad	53.94	75.62	0.3804	0.1578	23.52	16.93	35.19	61.17	72.19
GradDrop	53.73	75.54	0.3837	0.1580	23.54	16.96	35.17	61.06	72.07
IMTL	53.63	75.44	0.3868	0.1592	22.58	15.85	37.44	63.52	74.09
GradVac	54.21	75.67	0.3859	0.1583	23.58	16.91	35.34	61.15	72.10
CAGrad	53.97	75.54	0.3885	0.1588	22.47	15.71	37.77	63.82	74.30
Nash-MTL	53.41	74.95	0.3867	0.1612	22.57	15.94	37.30	63.40	74.09
RLW	54.04	75.58	0.3827	0.1588	23.07	16.49	36.12	62.08	72.94
Different architectures with EW strategy									
HPS	53.93	75.53	0.3825	0.1577	23.57	17.01	35.04	60.99	72.05
Cross-stitch	53.44	75.21	0.3818	0.1609	23.15	16.35	36.67	62.14	72.76
MMoE	53.14	75.07	0.3876	0.1613	23.02	16.36	36.45	62.40	73.17
MTAN	54.64	75.99	0.3771	0.1557	23.12	16.48	36.15	62.12	72.99
CGC	53.27	75.14	0.3914	0.1632	22.14	15.33	38.67	64.61	74.85
PLE	52.75	74.78	0.3943	0.1609	22.10	15.34	38.51	64.79	75.08
LTB	52.58	74.75	0.3828	0.1607	23.31	16.51	36.34	61.84	72.52
DSelect-k	53.75	75.44	0.3802	0.1569	23.18	16.44	36.29	62.14	72.85

Table 1: Performance comparison on the *NYUv2* data set with three tasks. Each experiment is repeated over 3 random seeds and the average performance is reported. ↑ (↓) indicates that the higher (lower) the value, the better the performance.

4. Conclusion

We present `LibMTL`, a comprehensive and extensible library for MTL. Built on `PyTorch`, it provides a unified training framework for different settings in MTL and possesses many SOTA MTL algorithms. In our future work, we will continuously maintain this library to incorporate newly proposed MTL models, update the documentation, add more applications from different areas, and provide more backbone models such as vision transformer (Dosovitskiy et al., 2021) and N-grams (Brown et al., 1992).

Acknowledgments

This work is supported by NSFC key grant 62136005 and NSFC general grant 62076118.

References

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based N-gram models of natural language. *Computational Linguistics*, 18 (4):467–480, 1992.

Han Cao, Jiayu Zhou, and Emanuel Schwarz. RMTL: An R library for multi-task learning. *Bioinformatics*, 35(10):1797–1798, 2019.

Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *International Conference on Machine Learning*, 1993.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018.

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.

Sumanth Chennupati, Ganesh Sistu, Senthil Kumar Yogamani, and Samir A. Rawashdeh. MultiNet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop on Autonomous Driving*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International Conference on Machine Learning*, 2020.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In *Neural Information Processing Systems*, 2021.

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.

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

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

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021b.

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.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, et al. PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

Fatemeh Rahimi, Evangelos E Milios, and Stan Matwin. MTLV: A library for building deep multi-task learning architectures. In *ACM Symposium on Document Engineering*, 2021.

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

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, 2012.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *ACM Conference on Recommender Systems*, 2020.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2021.

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.

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