# CurML: A Curriculum Machine Learning Library

Yuwei Zhou
zhou-yw21@mails.tsinghua.edu.cn
Tsinghua University

Hong Chen
h-chen20@mails.tsinghua.edu.cn
Tsinghua University

Zirui Pan
panzr20@mails.tsinghua.edu.cn
Tsinghua University

Chuanhao Yan
yanch21@mails.tsinghua.edu.cn
Tsinghua University

Fanqi Lin
lfq20@mails.tsinghua.edu.cn
Tsinghua University

Xin Wang*
Wenwu Zhu*
xin_wang@tsinghua.edu.cn
wwzhu@tsinghua.edu.cn
Tsinghua University

## ABSTRACT

Curriculum learning (CL) is a machine learning paradigm gradually learning from easy to hard, which is inspired by human curricula. As an easy-to-use and general training strategy, CL has been widely applied to various multimedia tasks covering images, texts, audios, videos, etc. The effectiveness of CL has recently facilitated an increasing number of new CL algorithms. However, there has been no open-source library for curriculum learning, making it hard to reproduce, evaluate and compare the numerous CL algorithms on fair benchmarks and settings. To ease and promote future research on CL, we develop CurML, the first Curriculum Machine Learning library to integrate existing CL algorithms into a unified framework. It is convenient to use and flexible to customize by calling the provided five APIs, which are designed for easily plugging into a general training process and conducting the data-oriented, model-oriented and loss-oriented curricula. Furthermore, we present empirical results obtained by CurML to demonstrate the advantages of our library. The code is available online at https://github.com/THUMNLab/CurML.

## CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**.

## KEYWORDS

Curriculum Learning, Machine Learning, Training Strategy

**ACM Reference Format:**
Yuwei Zhou, Hong Chen, Zirui Pan, Chuanhao Yan, Fanqi Lin, Xin Wang, and Wenwu Zhu. 2022. CurML: A Curriculum Machine Learning Library. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), Oct. 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3503161.3548549

## 1 INTRODUCTION

Curriculum (machine) learning (CL) has continuously gained attentions since its first advent [2]. CL borrows the idea of human learning curricula from easy content to hard content, forming a general training strategy for various machine learning models and applications [34]. Given that CL can help models to generalize better and converge faster [2, 11, 36], researchers have proposed numerous CL algorithms and shown their effectiveness in a wide range of multimedia tasks including visual question answering [24], video captioning [18], image classification [4, 5, 9, 29, 38, 39, 41, 44], semantic segmentation [7, 10], object detection [5, 7], speech recognition [1, 3], neural machine translation [12, 19, 20, 26, 33, 40, 42, 45], as well as different settings including unsupervised [37], semi-supervised [4, 10, 41], out-of-distribution [39] etc.

However, as the applications and scenarios of CL become increasingly diverse and wide, it is more difficult for the researchers to make comparisons among different CL algorithms and choose the most proper CL algorithm for their tasks. Therefore, relevant researchers have surveyed existing CL algorithms, summarized a general framework of CL design, and classified them based on their key components, i.e., a difficulty measurer to determine learning difficulty and a training scheduler to arrange the increase of learning difficulty [34]. Current surveys on CL [30, 34] analyze CL algorithms from the theoretical perspective, but do not provide their implementations and performances. As such, it still requires a lot of effort for researchers to empirically compare the CL algorithms through experimental results.

For fair comparisons and evaluations, an open-source library with established benchmarks and state-of-the-art CL algorithms is of great necessity. However, to the best of our knowledge, there are no public libraries that integrate existing CL algorithms. Each public implementation for CL only contains one specific CL algorithm, which is provided by the authors. Therefore, there is no surprise that separately running these implementations fails to reach fair comparisons because existing CL works usually have different tasks, datasets, backbone architectures, etc.

To fill this gap, we develop Curriculum Machine Learning (CurML), the first public open-source library for CL. We implement most of the existing CL algorithms through a unified and extensive framework, which is illustrated in Figure 1. From the outer view, the main part of the framework is a CL trainer whose inputs are data, model and predefined hyperparameters, and outputs are trained models and performances. From the inner view, we wrap each CL algorithm as a class with five APIs, *data_prepare*, *model_prepare*, *data_curriculum*, *model_curriculum* and *loss_curriculum*, all of which are designed to plug into a general model trainer, e.g., a trainer for an image classifier or language model, and implement the required

*Corresponding authors.



*MM '22, October 10–14, 2022, Lisboa, Portugal*

ACM ISBN 978-1-4503-9203-7/22/10.
https://doi.org/10.1145/3503161.3548549

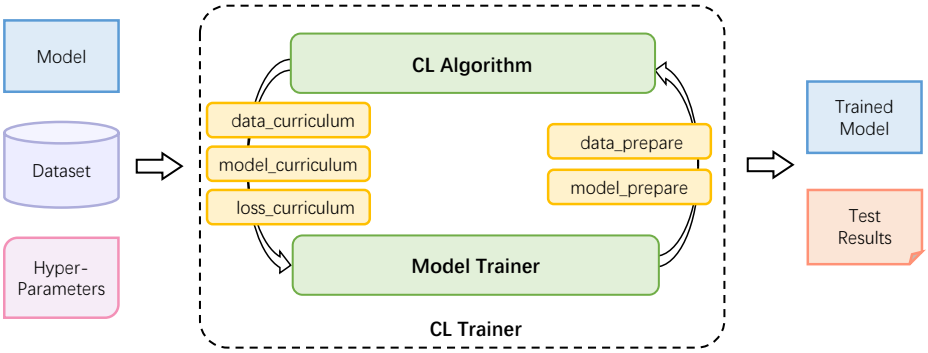


**Figure 1: The overall framework of CurML library.**

curriculum manner. The former two APIs are used to obtain necessary information of data and model from the original training process, while the latter three are used to measure and schedule the learning difficulty, completing the functions of the aforementioned difficulty measurer and training scheduler. This design of splitting the CL trainer and CL algorithm aims to satisfy the demands for different customization levels. The CL trainer aims at handling the task predefined by our CurML library, while the CL algorithm class is more suitable for user-defined tasks. To summarize, our library has the following advantageous features:

- **Convenient to run**: Even researchers new to CL can implement a CL algorithm with our library within less than five lines of codes. Experts can conduct quick CL experiments as their baselines.
- **Flexible to combine and extend**: Users can combine the current CL algorithms together by calling APIs from multiple CL algorithm classes. They can also extend existing CL methods flexibly by overwriting the corresponding APIs.
- **Easy to apply and plug in**: Researchers or scholars in other research fields can easily plug the state-of-the-art CL algorithms into their machine learning tasks with our provided APIs.

## 2 CURRICULUM MACHINE LEARNING

In this section, we describe our CurML library in detail. This library follows the object-oriented programming paradigm to avoid redundant duplication of implementation through composition and inheritance among classes. Its main body is a **CL Trainer** class, composed of a **CL Algorithm** class and a **Model Trainer** class, which interact with each other through five key **APIs** of the CL Algorithm class to implement the curriculum machine learning.

### 2.1 CL Trainer

The CL Trainer is a highly encapsulated class, which applies CL algorithms to the original machine learning process. It is designed for ease of use within less than five lines of code. Users only need to give predefined inputs, including dataset, model and hyperparameters, and then it can output the trained model and test results after a period of time of training and evaluation. For those users who do not care about the details of the CL implementations, they can easily run these algorithms and obtain the results they want by calling this trainer and its corresponding member functions.

Inside the CL Trainer, there are two member classes. One is a Model Trainer class which trains a model under the guidance of CL. And the other is a CL Algorithm class which schedules the difficulty of the training process. These two classes are described in the next two subsections.

### 2.2 Model Trainer

This trainer deals with the training and evaluation process given datasets and tasks. For instance, the model of the trainer can be a image classifier if the dataset is CIFAR-10 [16] or ImageNet [8], or a language model if Penn Treebank (PTB) [21] or WikiText [23].

More importantly, the trainer is specifically designed to receive functions as input variables for ease of CL implementation. As is shown in Figure 2, it mainly calls the input functions in two stages. The first stage is when the trainer has initiated its dataset and model, and it can provide the data or model required by the CL algorithm by passing function parameters. The second stage is when the trainer is training its model in every epoch, and it can achieve guidance from the CL algorithm.

### 2.3 CL Algorithm

Each CL algorithm included in our library is encapsulated into a class, whose member functions are designed to achieve the original training context and provide a new training schedule where training difficulty varies with epochs increasing. The CL algorithms implemented in our library are listed as follows:

- **Baby Step**[2, 31]: It is a predefined CL that sorts data by difficulty based on the predetermined criteria and continues adding harder data to the training set every predetermined epoch interval.
- **Lambda Step** [13]: It is similar to Baby Step, but the difference is its way to add harder data. It features a pacing function $\lambda(t)$ mapping epoch $t$ to the proportion $\lambda$ of added data to total data.
- **Self-Paced Learning** [17]: It is a representative automatic CL, which measures data difficulty by loss instead of predefined criteria, making it a flexible CL. It inspires lots of later works.
- **Transfer Teacher** [36]: It is like Self-Paced Learning, but the difference is that data difficulty is decided by a pre-trained teacher.
- **RL Teacher** [22]: It is similar to Transfer Teacher, but the teacher model in this method can receive feedback from the training model and finetune itself.

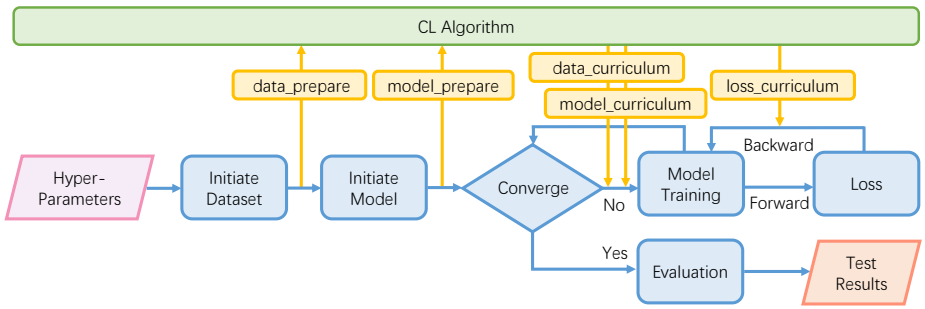


**Figure 2: A general workflow of curriculum machine learning.**

- **Minimax CL** [43]: It is inspired by Self-Paced Learning and adds a submodular function $F(A)$ to loss. By optimizing a minimax target, it prefers diverse data but limits the upper bound of loss.
- **Screener Net** [14]: It belongs to RL Teacher and designs a loss function for the teacher model based on the training loss.
- **Meta Reweight** [25]: It utilizes meta-learning to optimize the trainable weight of each data.
- **Meta Weight Net** [28]: It uses meta-learning like Meta Reweight, but its weight optimization is based on a small MW-Net.
- **Data Parameters** [27]: It assigns trainable parameters to data instances and classes and optimizes the parameters to serve as the difficulty of data and classes.
- **Local to Global** [6]: It continues adding classes and each data belong to the added classes to the training set.
- **Differentiable Data Selection** [35]: It is another CL with meta-learning. It samples training data with a scorer net, which can be updated based on the differentiable rewards on validation set.
- **Dynamic Instance Hardness** [44]: It defines three types of instantaneous hardness and measures data difficulty through an exponential average of the hardness.
- **Superloss** [5]: It proposes a task-agnostic confidence-aware loss plugged on top of the original training loss. The confidence actually represents data weights.
- **Curriculum by Smoothing** [29]: It is different from most CL algorithms in that it schedules models instead of data in a curriculum manner. It adds a Gaussian kernel after each convolutional layer and continues decreasing the variance of the kernel.
- **Coarse to Fine** [32]: It is similar to Local to Global, but it pre-clusters all classes and continues adding clusters and all classes belong to the added clusters.
- **Adaptive CL** [15]: It belongs to Transfer Teacher, but measures difficulty by both previous scores and the scores from the teacher.

To summarize, the above CL methods realize their own curriculum scheme in different ways. Therefore, we dedicatedly design the following five APIs to adapt these algorithms to the CL Algorithm class, so that they can be easily plugged into any training process.

## 2.4 APIs

The five APIs are the key part of the CL algorithm implementation, which can be grouped into two categories: one for data and model preparation, and the other for curriculum schedule with respect to data, model and loss.

**data_prepare**. This API will feed datasets from Model Trainer to CL Algorithm. Most of the CL methods are proposed to schedule training data difficulty, so they need to have an overall grasp of the dataset for better sampling or assigning more proper weights to data instances. For example, Data Parameters [27] equips all data instances and classes in a dataset with learnable parameters, so it has to fetch in advance the number of instances and classes in the dataset for convenient initiation of the parameters. Dynamic Instance Hardness [44] selects data instances based on probabilities proportional to data hardness, so it also needs to acquire the original dataset to initiate the probabilities vector and select proper data from the dataset to train the model.

**model_prepare**. This API is similar to *data_prepare*, but it passes variables related to the training model instead, such as model architecture, parameter optimizer and learning rate scheduler. Some CL methods involve these variables, so they need to record their initial state before the training process. For instance, in Local to Global [6], there is a step to restore the current learning rate to the original one before adding data from new clusters, so it at least has to acquire and preserve the initial learning rate.

These two APIs above are called in the initiation and preparation stage of the training process. They are similar to each other and help the preparation of CL algorithms at data and model level respectively.

**data_curriculum**. This API aims to measure data difficulty and decide which data instance should be sampled according to the training epoch, in order to conduct the data-oriented curriculum learning process. Since most CL algorithms are at data level, this API is critical for them in the library. For instance, Baby Step [31] and Lambda Step [13] add data to the training set from easy to hard in a predefined order. Local to Global [6] and Coarse to Fine [32] add data from class and cluster level respectively. Dynamic Instance Hardness [44] samples data based on dynamic data hardness. All of them sample part of the data instead of all in the early stage of training, so this API is mainly designed for such data selection.

**model_curriculum**. This API is like *data_curriculum*, but it updates the model instead of data as the training epoch increases. This kind of CL approaches which schedules learning difficulty by adjusting model is generally rare, but our library involves one, Curriculum by Smoothing [29]. It gradually increases the amount of high-frequency information available by adding a Gaussian kernel whose variance continues decreasing after each convolutional layer

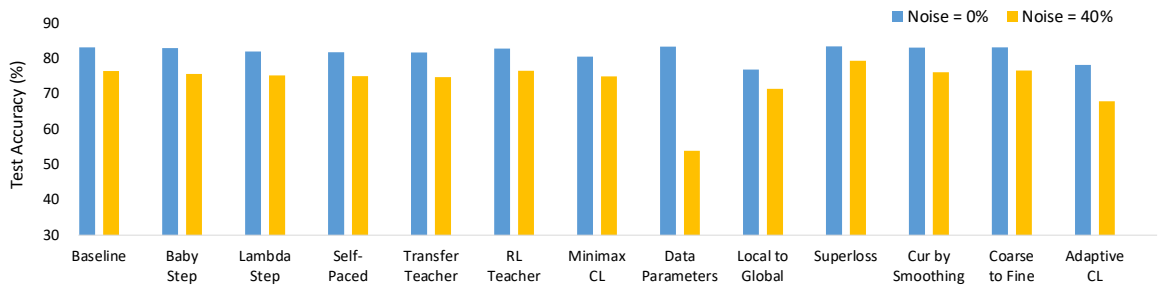


**Figure 3: Test Accuracy of part of the CL algorithms on CIFAR-10 with/without noise.**

of the model. Therefore, we design this API to implement this CL algorithm and look forward to future CL works at model level.

**loss_curriculum**. This API aims to find the best loss weights sequence and optimize reweighted loss, which is a common practice in CL. From the perspective of loss, adjusting loss can affect the gradients backward and further make different influences on learnable parameters of the model. For instance, Superloss [5] is a confidence-aware loss function on top of any existing task loss. Self-Paced Learning [17] and Minimax CL [43] add a self-paced regularizer to the original loss. The key insight of the three works is to downweight or even discard the hard data with high loss early in training. On the other hand, reweighting loss equals reweighting data, which can be regarded as a soft sampling of data in that a weight value close to one means selecting data but close to zero means discarding. Therefore, this API can also be regarded as a complement to *data_curriculum* to implement curriculum process at data level. For instance, Data Parameters [27], Screener Net [14], Meta Reweight [25], Meta Weight Net [28], Adaptive CL [15], etc., will assign each data a learnable weight, which participate in training by multiplying with the original loss. To summarize, whether these algorithms aim at loss or data, this API is necessary to handle the original loss and map it to curriculum loss with a well-designed function or weights vector.

These three APIs above are called in every epoch of the training process. For every predefined interval of epochs, CL Algorithm measures the current difficulty and schedules the training process through these three APIs, which are plugged into Model Trainer.

## 3 EMPIRICAL RESULTS

In this section, we present some empirical results to show the usage of our library. It is worth noting that our main target is to reproduce, evaluate and compare existing CL algorithms instead of proposing a new CL algorithm or achieving state-of-the-art performances.

### 3.1 Experimental Setup

We mainly conduct experiments on image classification to verify our implementation of CL algorithms.

We use CIFAR-10 [16] downloaded from PyTorch[1] and split one-tenth of the data from the training set as a validation set, and the remaining as a new training set. For convenience, we take a simple CNN with four convolutional and batch normalization layers, two pooling layers and one linear layer as our backbone network. We adopt Pytorch-Cifar[2], a popular training framework, to train our model. To ensure a fair comparison, we basically maintain the same hyperparameters as the Pytorch-Cifar did. Furthermore, we also explore the performance of CL algorithms in the setting of noise by adding 40% label noise to the training set.

### 3.2 Main Results

The results are illustrated in Figure 3. It can be observed that in the clean dataset setting most of the CL algorithms show similar performances, except that Local to Global and Adaptive CL seems less effective. In the noisy setting, they tend to have different performances. For instance, RL Teacher performs well while Data Parameters shows its vulnerability to noise. Besides, in both settings, Superloss achieves consistently better results. To sum up, different CL algorithms adapt to different scenarios, so it is necessary to choose a proper CL algorithm and corresponding hyperparameters according to the target task and setting.

## 4 CONCLUSION AND FUTURE PLANS

In summary, we develop CurML, the first curriculum machine learning library to integrate existing CL algorithms into a unified framework. It is easy to use, flexible to customize, and enables users to reproduce, evaluate and compare the CL algorithms conveniently.

In the future, we plan to improve the current implementation and support more CL algorithms for various tasks and settings:

- To follow the latest works on CL, and compare with the official implementations in terms of performance and efficiency.
- To apply CL to more tasks and datasets, e.g., CIFAR-100 [16], ImageNet [8] in image classification, Penn Treebank (PTB) [21], WikiText [23] in language model, and other multimedia datasets.
- To support more settings and scenarios, e.g., out of distribution (OOD) and semi-supervision.
- To provide more detailed documentation.

## ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China No. 2020AAA0106300 and National Natural Science Foundation of China No. 62250008, No. 62102222.

[1] https://pytorch.org/

[2] https://github.com/kuangliu/pytorch-cifar

# REFERENCES


[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*. PMLR, 173–182.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML (ACM International Conference Proceeding Series, Vol. 382)*. ACM, 41–48.

[3] Stefan Braun, Daniel Neil, and Shih-Chii Liu. 2017. A curriculum learning method for improved noise robustness in automatic speech recognition. In *EUSIPCO*. IEEE, 548–552.

[4] Paola Cascante-Bonilla, Fuwen Tan, Yanjun Qi, and Vicente Ordonez. 2020. Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning. *arXiv preprint arXiv:2001.06001* (2020).

[5] Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. Superloss: A generic loss for robust curriculum learning. *Advances in Neural Information Processing Systems* 33 (2020), 4308–4319.

[6] Hao Cheng, Dongze Lian, Bowen Deng, Shenghua Gao, Tao Tan, and Yanlin Geng. 2019. Local to global learning: Gradually adding classes for training deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4748–4756.

[7] Dengxin Dai, Christos Sakaridis, Simon Hecker, and Luc Van Gool. 2020. Curriculum Model Adaptation with Synthetic and Real Data for Semantic Foggy Scene Understanding. *Int. J. Comput. Vis.* 128, 5 (2020), 1182–1204.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[9] Ürün Dogan, Aniket Anand Deshmukh, Marcin Machura, and Christian Igel. 2020. Label-Similarity Curriculum Learning. In *ECCV (29) (Lecture Notes in Computer Science, Vol. 12374)*. Springer, 174–190.

[10] Zhengyang Feng, Qianyu Zhou, Guangliang Cheng, Xin Tan, Jianping Shi, and Lizhuang Ma. 2020. Semi-supervised semantic segmentation via dynamic self-training and classbalanced curriculum. *arXiv preprint arXiv:2004.08514* 1, 2 (2020), 5.

[11] Tieliang Gong, Qian Zhao, Deyu Meng, and Zongben Xu. 2016. Why curriculum learning & self-paced learning work in big/noisy data: A theoretical perspective. *Big Data & Information Analytics* 1, 1 (2016), 111.

[12] Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Fine-Tuning by Curriculum Learning for Non-Autoregressive Neural Machine Translation. In *AAAI*. AAAI Press, 7839–7846.

[13] Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*. PMLR, 2535–2544.

[14] Tae-Hoon Kim and Jonghyun Choi. 2018. Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904* (2018).

[15] Yajing Kong, Liu Liu, Jun Wang, and Dacheng Tao. 2021. Adaptive Curriculum Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5067–5076.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[17] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems* 23 (2010).

[18] Shanhao Li, Bang Yang, and Yuexian Zou. 2022. Adaptive Curriculum Learning for Video Captioning. *IEEE Access* 10 (2022), 31751–31759.

[19] Jinglin Liu, Yi Ren, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. Task-Level Curriculum Learning for Non-Autoregressive Neural Machine Translation. In *IJCAI*. ijcai.org, 3861–3867.

[20] Xuebo Liu, Houtim Lai, Derek F. Wong, and Lidia S. Chao. 2020. Norm-Based Curriculum Learning for Neural Machine Translation. In *ACL*. Association for Computational Linguistics, 427–436.

[21] Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Using Large Corpora* (1994), 273.

[22] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3732–3740.

[23] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *ICLR (Poster)*. OpenReview.net.

[24] Yonghua Pan, Zechao Li, Liyan Zhang, and Jinhui Tang. 2022. Causal Inference with Knowledge Distilling and Curriculum Learning for Unbiased VQA. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 18, 3 (2022), 1–23.

[25] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *International conference on machine learning*. PMLR, 4334–4343.

[26] Dana Ruiter, Josef van Genabith, and Cristina España-Bonet. 2020. Self-Induced Curriculum Learning in Self-Supervised Neural Machine Translation. In *EMNLP (1)*. Association for Computational Linguistics, 2560–2571.

[27] Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. 2019. Data parameters: A new family of parameters for learning a differentiable curriculum. *Advances in Neural Information Processing Systems* 32 (2019).

[28] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems* 32 (2019).

[29] Samarth Sinha, Animesh Garg, and Hugo Larochelle. 2020. Curriculum by smoothing. *Advances in Neural Information Processing Systems* 33 (2020), 21653–21664.

[30] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *International Journal of Computer Vision* (2022), 1–40.

[31] Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 751–759.

[32] Otilia Stretcu, Emmanouil Antonios Platanios, Tom M Mitchell, and Barnabás Póczos. 2021. Coarse-to-Fine Curriculum Learning. *arXiv preprint arXiv:2106.04072* (2021).

[33] Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. 2020. Learning a Multi-Domain Curriculum for Neural Machine Translation. In *ACL*. Association for Computational Linguistics, 7711–7723.

[34] Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[35] Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neubig. 2020. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*. PMLR, 9983–9995.

[36] Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*. PMLR, 5238–5246.

[37] Chang Xu, Dacheng Tao, and Chao Xu. 2015. Multi-view Self-Paced Learning for Clustering. In *IJCAI*. AAAI Press, 3974–3980.

[38] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Shrivastava. 2020. Curriculum Manager for Source Selection in Multi-source Domain Adaptation. In *ECCV (14) (Lecture Notes in Computer Science, Vol. 12359)*. Springer, 608–624.

[39] Qing Yu, Daiki Ikami, Go Irie, and Kiyoharu Aizawa. 2020. Multi-task Curriculum Framework for Open-Set Semi-supervised Learning. In *ECCV (12) (Lecture Notes in Computer Science, Vol. 12357)*. Springer, 438–454.

[40] Runzhe Zhan, Xuebo Liu, Derek F. Wong, and Lidia S. Chao. 2021. Meta-Curriculum Learning for Domain Adaptation in Neural Machine Translation. In *AAAI*. AAAI Press, 14310–14318.

[41] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling. In *NeurIPS*. 18408–18419.

[42] Mingjun Zhao, Haijiang Wu, Di Niu, and Xiaoli Wang. 2020. Reinforced Curriculum Learning on Pre-Trained Neural Machine Translation Models. In *AAAI*. AAAI Press, 9652–9659.

[43] Tianyi Zhou and Jeff A. Bilmes. 2018. Minimax Curriculum Learning: Machine Teaching with Desirable Difficulties and Scheduled Diversity. In *ICLR (Poster)*. OpenReview.net.

[44] Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. 2020. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems* 33 (2020), 8602–8613.

[45] Yikai Zhou, Baosong Yang, Derek F. Wong, Yu Wan, and Lidia S. Chao. 2020. Uncertainty-Aware Curriculum Learning for Neural Machine Translation. In *ACL*. Association for Computational Linguistics, 6934–6944.