A Survey on Curriculum Learning

Xin Wang, *Member, IEEE*, Yudong Chen, and Wenwu Zhu, *Fellow, IEEE*

Abstract—Curriculum learning (CL) is a training strategy that trains a machine learning model from easier data to harder data, which imitates the meaningful learning order in human curricula. As an easy-to-use plug-in, the CL strategy has demonstrated its power in improving the generalization capacity and convergence rate of various models in a wide range of scenarios such as computer vision and natural language processing etc. In this survey article, we comprehensively review CL from various aspects including motivations, definitions, theories, and applications. We discuss works on curriculum learning within a general CL framework, elaborating on how to design a manually predefined curriculum or an automatic curriculum. In particular, we summarize existing CL designs based on the general framework of *Difficulty Measurer + Training Scheduler* and further categorize the methodologies for automatic CL into four groups, i.e., Self-paced Learning, Transfer Teacher, RL Teacher, and Other Automatic CL. We also analyze principles to select different CL designs that may benefit practical applications. Finally, we present our insights on the relationships connecting CL and other machine learning concepts including transfer learning, meta-learning, continual learning and active learning, etc., then point out challenges in CL as well as potential future research directions deserving further investigations.

Index Terms—Curriculum Learning, Machine Learning, Training Strategy, Example Reweighting, Self-Paced Learning.

1 Introduction

Human learning has inspired various algorithm designs throughout the development of machine learning. As an outstanding feature of human learning, curriculum, or learning in a meaningful order, has been exploited and transferred to machine learning, which forms the subdiscipline named *curriculum learning (CL)*. In essence, human education is highly organized as curricula, by "starting small" and gradually presenting more complex concepts. For example, to learn calculus at college, a student should first learn basic arithmetic at primary school, abstract function at middle school, and then derived function at high school. However, in traditional machine learning algorithms, all the training examples are randomly presented to the model, ignoring the various complexities of data samples and the learning status of the current model. Therefore, an intuitive question is: "*could the curriculum-like training strategy ever benefit machine learning?*" According to the extensive experiments from early work [6], [54], [131] to recent efforts [17], [29], [33], [86] in various applications of machine learning, we may summarize the answer as: "yes, but not always." As we will demonstrate in this survey, the power of introducing curriculum into machine learning depends on how we design the curriculum for specific applications and datasets.

The original concept of CL is first proposed by Bengio et al. [6]. In short, curriculum learning means "training from easier data to harder data". More specifically, the basic idea is to "start small" [15], train the machine learning model with easier data subsets (or easier subtasks), and then gradually increase the difficulty level of data (or subtasks) until the whole training dataset (or the target task(s)). An illustration of CL is demonstrated in Fig. 1, where we take the image classification task as an example. Initially, CL trains the model on a small subset of "easy" images, i.e., the images of apples and oranges are clear, typical, and easily recognizable. With the progress of model training, CL adds more "harder" images (i.e., harder to recognize) to the current subset, which is akin to the increasing difficulty of learning materials in human curricula. Finally, CL leverages the whole training dataset for training.

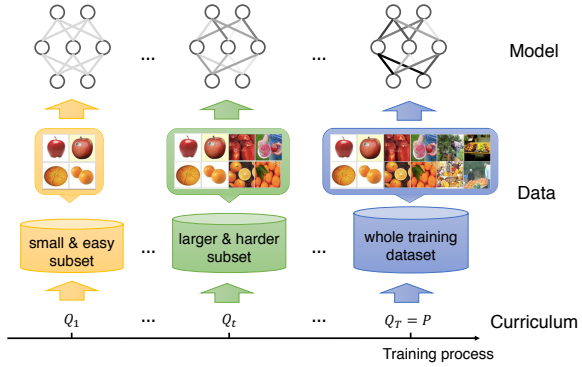

Fig. 1. Illustration of the Curriculum Learning (CL) concept (The fruit images are from [106]). CL is a training strategy for machine learning that trains from easier data to harder data, imitating human curricula. Specifically, CL initially trains the model on a small and easy subset. With the progress of the training, CL gradually introduces more harder examples into the subset, and finally trains the model on the whole training dataset. This CL strategy can improve both model performance and convergence rate, compared with direct training on the whole training dataset. Q_t here stands for a reweighting of the training data distribution P at the t-th training epoch (See details in Sec. 2).

As the idea of CL serves as a general training strategy beyond specific machine learning tasks, scholars have been exploiting its power in considerably wide application scopes, including supervised learning tasks within computer vision (CV) [31], [40], natural language processing (NLP) [86], [112], healthcare prediction [14], etc., various reinforcement learning (RL) tasks [20], [77], [93] as well as other applications such as graph learning [25], [88] and

• *Xin Wang, Yudong Chen, Wenwu Zhu are with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. E-mail: xin_wang@tsinghua.edu.cn, cyd18@mails.tsinghua.edu.cn, wwzhu@tsinghua.edu.cn. Corresponding Author: Wenwu Zhu.* *This work is supported by the National Key Research and Development Program of China (No.2020AAA0106300, 2020AAA0107800, 2018AAA0102000).*

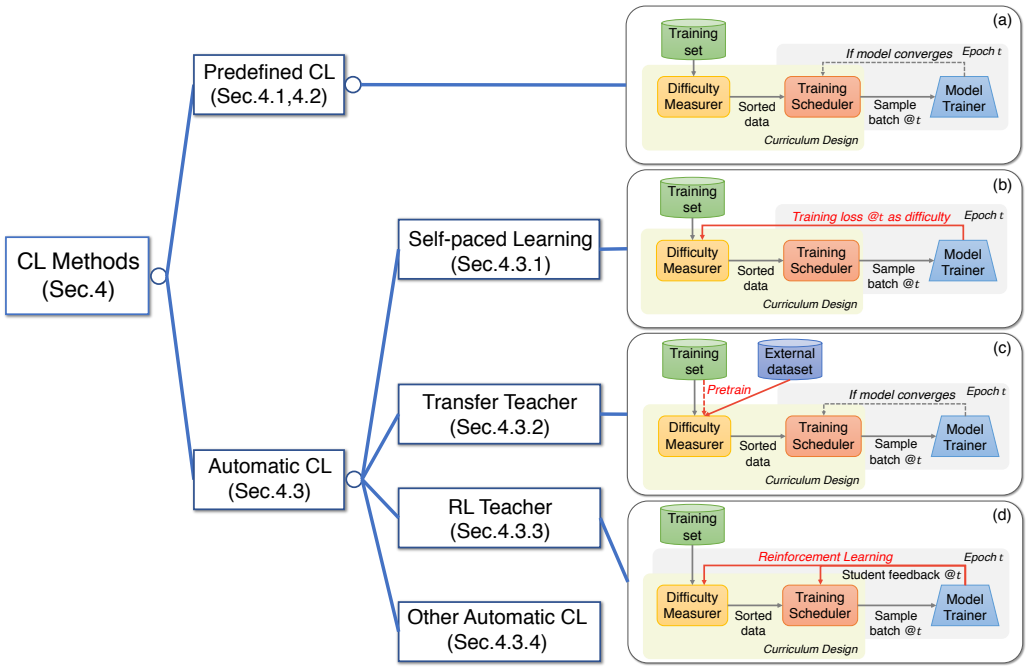

Fig. 2. A categorization of CL methods and the corresponding illustrations. We divide the existing methods into predefined CL and automatic CL, the latter of which including Self-paced Learning, Transfer Teacher, RL Teacher and Other Automatic CL. As shown in the illustrations, most CL methods comply with the general framework of *Difficulty Measurer + Training Scheduler* in Sec. 4.

neural architecture search (NAS) [32]. The advantages of applying CL training strategies to miscellaneous real-world scenarios can be mainly summarized as *improving the model performance on target tasks* and *accelerating the training process*, which cover the two most significant requirements in major machine learning research. For example, in [86], CL helps the neural machine translation model reduce training time by up to 70% and improves the performance by up to 2.2 BLEU points, compared to plain training without curricula. In [41], CL brings a relative 45.8% MAP boost from normal batch training with an obvious faster convergence in multimedia event detection task. In [20], CL enables the RL agents to solve hard goal-oriented problems that they cannot solve without curricula. Apart from the above two main advantages, CL is also easy-to-use, since it is a flexible plug-and-play submodule independent of the original training algorithms in most CL literature. However, to the best of our knowledge, little effort has been made to systematically summarize the methodologies and applications of CL.

In this paper, we fill this gap by comprehensively reviewing CL and summarizing its methodologies. To be more specific, we hope to provide the readers with an overall picture of CL, which includes comprehensible and elaborate answers to the following questions: (i) What is the definition of CL (Sec. 2)? (ii) Why is CL effective, and why should researchers use CL (Sec. 3)? (iii) How to design a curriculum (Sec. 4)? We conclude the paper with a comparison of "easier first" and "harder first" training strategies and discussion on the relationship between CL and other machine learning concepts in Sec. 5. We also summarize several open questions and future directions for CL to inspire future researchers in Sec. 6.

2 Definition of CL

History context. Empirical evidence supporting the meaningfulness of taking curricula in human and animal learning has been early provided in behavior and cognitive science literature. Skinner [84], [105] provides the earliest behavior evidence on the importance of *shaping*, i.e., another name for CL in animal training context. Cognitive evidence is then provided in human size constancy learning [116] and language learning [79]. The idea of introducing a curriculum into the training strategy of machine learning algorithms can be traced back to Selfridge et al.'s work [99]. The authors proposed to train a cart pole controller, a classic problem in robotics, first on long and light poles and then gradually on shorter and heavier poles. Later related work [95], [98] in RL and robotics domains also discussed how to organize the presenting order of tasks from easy to hard. The first attempt of the curriculum-like idea on supervised learning is made by Elman [15] in the NLP task of grammar learning with recurrent networks. The author highlighted the importance of "starting small": restricting the range of data exposed to neural networks during initial training. This strategy is also revisited in [94] and [52], the latter of which provides evidence for faster convergence.

Based on all these previous works, the concept of CL was first proposed by Bengio et al. [6] with experiments on supervised visual and language learning tasks, exploring when and why a curriculum could benefit machine learning. The original definition of CL by Bengio et al. [6] is as follows.

Definition 1: Original Curriculum Learning [6]. A curriculum is a sequence of training criteria over T training steps: $\mathcal{C} = \langle Q_1, \ldots, Q_t, \ldots, Q_T \rangle$. Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z)P(z) \quad \forall \text{example } z \in \text{training set } D, \tag{1}$$

such that the following three conditions are satisfied:

- 1) The entropy of distributions gradually increases, i.e., $H(Q_t) < H(Q_{t+1})$.
- 2) The weight for any example increases, i.e., $W_t(z) \leq W_{t+1}(z) \quad \forall z \in D$.
- 3) $Q_T(z) = P(z)$.

Curriculum learning is the training strategy that trains a machine learning model with a curriculum.

In Definition 1, Condition (1) means the diversity and information of the training set should gradually increase, i.e., the reweighting of examples in later steps increases the probability of sampling slightly more difficult examples. Condition (2) means to gradually add (in binary or soft manner) more training examples, so the size of the training set increases. Condition (3) means finally, the reweighting of all examples is uniform and we train on the target training set.

Most of the CL methods discussed in this paper (especially those in Sec. 4.2, 4.3.1, and 4.3.2) meet Definition 1, illustrated in Fig 1. As shown in the figure, the CL strategy determines the training data subset of each training step, such that the size and overall difficulty of the subsets are gradually increasing throughout the training process.

Since the concept of CL was formally proposed, the academic community follows and further extends the definition of CL. Within the spirit of "training from easier data (tasks) to harder data (tasks)", i.e., fixing Condition (1) in Definition 1, Condition (2) and (3) can be relaxed to enable more flexible CL strategies. For example, in [29], [83], [131] of multi-task setting and most CL for RL settings [19], Condition (2) and (3) are relaxed since at each step the model is trained on only one task. However, the diversity or difficulty of the current task/goal gradually increases, which guides the model to boost the performance on the target task(s). The CL methods based on One-Pass scheduler [112], [114], [117] discussed in Sec. 4.2.2 also relaxes Condition (2) and (3) as they train the model from easier subsets to harder subsets. Moreover, other works also extend Definition 1 by adding more conditions of data characteristics for different application purposes. For instance, Jiang et al. [41] propose to train "from easy & diverse to hard" to avoid overfitting to the same sample group in multi-group event detection tasks. Wang et al. [121] train the model "from easy & imbalanced to hard & balanced" data to alleviate the severe class imbalance in human attribute analysis.

At a more abstract level, a curriculum can be seen as a sequence of (binary) *instance selection* [80] or (soft) *example reweighting* along the training process to achieve faster convergence or better generalization, which is beyond the "easy to hard" or "starting small" principles. This perspective inspires the academic community to bring more connotations to CL definition with new methodologies, which can be summarized as follows.

Definition 2: Data-level Generalized Curriculum Learning. Discarding all the three conditions in Definition 1, a curriculum is a sequence of reweighting of target training distribution over T training steps. Curriculum learning is the strategy that trains a model with such a curriculum.

Most CL methods in Sec. 4.3.3 and Sec. 4.3.4 could learn to automatically and dynamically select the most suitable examples or tasks (with adjustable loss weights) for each current training step and thus meet Definition 2. Interestingly, in some of the works, the best curriculum found by the algorithm is the opposite of traditional CL, i.e., "hard to easy" [17], [118] or "starting big" (from full dataset to informative subset) [118], [119], [140]. There is also a line of research named hard example mining (HEM) [45], [101] selecting the most difficult examples in each training batch. HEM actually falls in Definition 2 and is explored in some CL literature [39], [145]. A discussion on this seemingly paradoxical phenomenon will be made in Sec. 5.1.

To even further broaden the scope of CL, some scholars jump from data level to criteria level, to regard a curriculum as a sequence of *training criteria* during the training process. This further generalizes the CL definition:

Definition 3: Generalized Curriculum Learning. Discarding the definition of Q_t (Eq. 1) and its three conditions in Definition 1, a curriculum is a sequence of training criteria over T training steps. Each criterion Q_t includes the design for all the elements in training a machine learning model, e.g., data/tasks, model capacity, learning objective, etc. Curriculum learning is the strategy that trains a model with such a curriculum.

Examples for training criteria in Definition 3 include, but are not limited to, loss function [97], [124], supervision generation [34], [133], model capacity [46], [75], [104], input scheme [4], and hypothesis space [32]. Note that the criteria in such a generalized curriculum in Definition 3 usually change progressively, analogous to the gradual curriculum in human education. For example, in Curriculum Dropout [75], the algorithm gradually reduces the ratio of active units in dropout operation from 1 to a predefined $\theta_0 \in (0, 1)$ to achieve adaptive regularization during training. In Curriculum NAS [32], the algorithm starts from a small search space and gradually incorporates the learned knowledge to guide the search in larger spaces, which significantly improves the search efficiency and also finds better neural architectures. These works broaden the extension of CL and exploit the potentialities of the human curriculum idea for machine learning at a higher level, leaving room for imagination for future work.

3 Analysis on Effectiveness of CL and Suitable Application Scenes

Before applying CL to their studies, researchers might be curious about a fundamental question: why on earth does this human-curriculum-like training strategy work? To explain why CL could lead to generalization improvement and convergence speedup, scholars have provided hypotheses and proofs from different perspectives. Basically, existing analyses uncover the essence of CL from the perspectives of *optimization problem* and *data distribution*, based on which we can further summarize the two main motivations for applying CL: *to guide* and *to denoise*.

3.1 Theoretical Analysis on CL

To begin with, from the perspective of **optimization problem**, Bengio et al. [6] initially point out that CL can be seen as a particular *continuation method*. Intuitively, continuation methods [2] are optimization strategies for non-convex criteria which first optimize a smoother (and also easier) version of the problem to reveal the "global picture", and then

gradually consider less smoothing versions, until the target objective of interest. This strategy also shares the same spirit with *simulated annealing*. As illustrated in Fig 3, continuation methods provide a sequence of optimization objectives, starting with a heavily smoothed objective for which it is easy to find a global minimum, and tracking the local minima throughout the training. In this way, continuation methods *guide* the training towards better regions in parameter space, i.e., as shown in Fig 3, the local minima learned from easier objectives have better generalization ability and are more likely to approximate global minima. Moreover, from the view of transfer learning, this continuation strategy can also be regarded as a sequence of unsupervised pre-training [6]: training on the preceding objectives could act as a pre-training process which both helps optimization and provides regularization on succeeding objectives.

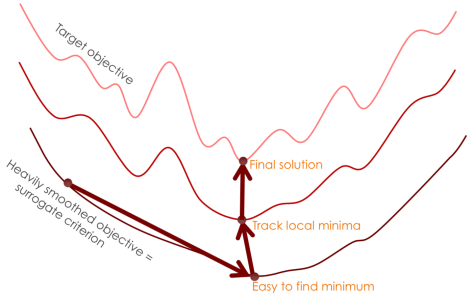

Fig. 3. Illustration of the continuation method from [5], which is the essence of the CL [6]. It starts from optimizing a heavily smoothed version of the objective, and gradually moves to the target objective. Tracking the local minima throughout the training guides the model towards better parameter space and makes it more generalizable.

Additionally, recent studies provide more theoretical evidence for the convergence speedup in CL from the optimization perspective. Weinshall et al. [123] prove a theorem

On the other hand, researchers also analyze the CL mechanism from the perspective of **data distribution**. In the era of deep learning, large-scale data sources are required for training, which are collected and annotated by company users, the web, and crowd-sourcing systems. This big data collection brings noisy data that is less cognizable or wrongly annotated. In the CL setting, the noisy data corresponds to harder examples in the datasets while the cleaner data form the easier part. Since CL strategy encourages training more on the easier data, an intuitive hypothesis is that CL learner wastes less time with the harder and noisy examples to achieve faster training [6]. This hypothesis reveals the *denoising* efficacy of CL on noisy data.

To have a closer look at this denoising mechanism, Gong et al. [27] provide a theory based on the assumption that there exists deviation between training and testing distributions caused by noisy/wrongly-annotated training data. Intuitively, training and target/testing distributions share a common high-confidence annotated region with large density, which corresponds to the easier examples in CL. Therefore, to start training from easier examples by CL strategy actually simulates learning from this high-confidence common region (as an approximation to the target distribution), which guides the learning towards the expected target while reduces the negative impacts from low-confidence noisy examples. This data distribution perspective of CL is illustrated in Fig 4. The common density peak (at the center of the x-axis) of training and target distributions $P_{\text{train}}(x)$ and $P_{\text{target}}(x)$ in the left part refers to the common high-confidence area, while the heavy tail of $P_{\text{train}}(x)$ demonstrates the relatively more noisy data in training distribution. The right part illustrates the sequence of weight functions in CL, which initially assigns small values to the noisy tails and much larger values in the common easy area, and gradually moves to equal weights for all examples. Based on the above analysis, the authors formulate $P_{\text{target}}(x)$ as the weighted expression of $P_{\text{train}}(x)$. A follow-up theory clarifies that CL essentially minimizes an upper bound of the expected risk under target distribution, and this bound shows that we could approach the task of minimizing the expected risk on $P_{\text{target}}(x)$ by taking the core idea of CL: gradually taking relatively easy examples according to the curriculum and minimizing the empirical risk on these examples.

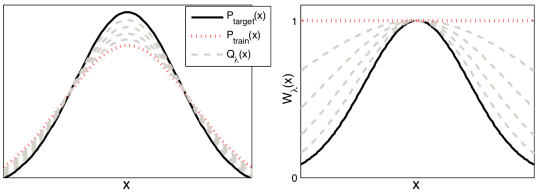

Fig. 4. Illustration of the CL from the data distribution perspective [27]. The left part demonstrates the data distribution shifts from the easy subset (the solid curve, which is assumed to approximate the testing distribution $P_{\text{target}}(x)$ well) to the full training set $P_{\text{train}}(x)$ (the red dashed curve). The right part shows the corresponding weighting scheme to enable this distribution shift. The center peak of curves refers to the high-confidence clean data, while the tails refer to the noisy data in the distributions. As shown in the left part, $P_{\text{target}}(x)$ is cleaner than $P_{\text{train}}(x)$.

3.2 Suitable Application Scenes of CL

Based on the above analysis on why CL is effective, we can categorize the motivations for applying CL into two groups: **to guide**, regularizing the training towards better regions in parameter space (with steeper gradients) as from the perspective of the optimization problem, and **to denoise**, focusing on high-confidence easier area to alleviate the interference of noisy data as from the perspective of data distribution. Not surprisingly, most of the existing application scenes of CL can be classified into these two groups, as demonstrated in Table 1.

The application scenes based on the "to guide" motivation often involve difficult target tasks where direct training on these tasks results in poor performance or slow convergence. CL strategies are adopted to guide the training from easier tasks or smoother versions of objectives to the target tasks. For instance, in sparse-reward RL, direct training on the final tasks rarely gets any positive rewards, which hinders agent learning. Therefore, researchers propose to take the CL strategy and manually [72] or automatically [20] design a sequence of auxiliary (sub)tasks/goals from easy to hard to guide the training. In multi-task learning, learning all the tasks simultaneously or in random order often leads to unsatisfactory performance. To yield performance gains, CL strategies are adopted to automatically choose the easier tasks which are more related to the previous one [83] or can bring more learning progress to the model training [29],

TABLE 1
Suitable Application Scenes of CL.

Motivation	Effect	Scene	Examples
To guide	make training possible / better and faster	the target task is hard or has a different distribution	sparse reward RL, multi-task learning, GAN training, NAS; domain adaption, imbalanced classification
To denoise	make training faster, more robust and generalizable	tasks with noisy, uneven quality, heterogeneous data (often large-scale, cheaply collected)	weakly-supervised or unsupervised learning, NLP tasks (neural machine translation, natural language understanding, etc.)

[72]. Other examples include CL for training GANs [22], [46], [106] and NAS [32].

Besides, the "to guide" application scenes also include the tasks where the target distribution is quite different from the training distribution, and a good curriculum helps to guide the training for adaption to the target distribution. A representative scene is domain adaption, which aims at improving prediction performance on unlabeled target domain data by knowledge transfer from richly annotated source domain data with a distribution drift. Recent studies [103], [139] propose to train from more in-domain data (similar to target domain) to less in-domain data, guiding the model to adapt to the target domain while adequately exploiting the source domain data. Note that CL for domain adaption is also related to the "to denoise" motivation, if we regard the less in-domain data as a kind of noisy data. Another example is imbalanced classification problems, where the training distribution on different classes is extremely imbalanced. Different studies adopt various curricula either beginning from balanced subset to more imbalanced full dataset [39] or from easy and imbalanced subset to harder and more balanced subset [121] to improve the generalization capacity of the classifier.

On the other hand, the application scenes based on the "to denoise" motivation often have a noisy or heterogeneous training dataset, and CL strategies could help denoise, making the training faster, more robust, and more generalizable. A popular application of CL with this motivation is neural machine translation (NMT), whose dataset is highly heterogeneous in quality, difficulty, and noise [53]. This is because the translation of a sentence could be long and short with different vocabulary and grammar structures, and different annotators always provide translations of different qualities Moreover, the training of NMT models (e.g., RNNs) is often time-consuming. Therefore, CL is naturally suitable for NMT tasks to denoise during training and to achieve both performance boost and faster convergence. Similarly, CL is also adopted in other NLP tasks with noisy or heterogeneous data, including natural language understanding [126], relation extraction [37], reading comprehension [112], etc. Moreover, CL is also effective in weakly-supervised CV tasks [31], [64].

From the perspective of supervision in training, CL can help supervised, weakly-supervised, and unsupervised learning by guiding or denoising. Specifically, CL helps supervised setting mainly by guiding when (i) the task is hard [20], [83], (ii) parts of the training data are difficult to learn [6], [41], (iii) the target distribution heavily shifts from training distribution [103], [121]. Weakly-supervised setting includes three typical types [147], all of which are enhanced by CL denoising. (i) For inaccurate supervision, i.e., the training set is noisy and usually collected from the web, CL helps to denoise, enabling the model to focus on a cleaner subset to avoid bad local minimum [31], [64], [86], [103]. (ii) For incomplete supervision, i.e., the semi-supervised setting where some training data are unlabeled, CL helps to distinguish the easier (more confident) unlabeled examples and add them to the training set earlier or with higher weights, denoising the pseudo labels with low confidence (harder unlabeled data) [24], [25], [114], [136]. (iii) For inexact supervision, i.e., only coarse-grained labels are given, CL helps to gradually integrate confident fine-grained pseudo labels into training while denoising the noisy ones, usually under a multi-instance learning framework [34], [111], [134], [135]. Finally, CL can also help unsupervised setting, e.g., clustering [22], [127], feature selection [142], domain adaption [11], etc. The mechanism in most work is similar to the semi-supervised setting, i.e., denoising the noisy pseudo labels [11], [22], [142]. The function to guide is also explored in [127]. With carefully designed CL, [133] even learns deep saliency network without human annotation by progressively synthesizing supervision masks.

4 CL Design: A General Framework

Since we have understood why CL is effective and why researchers apply CL to different scenes, a natural and important question should be: how to design an appropriate curriculum for a specific learning task? In this section, we provide a general framework of "Difficulty Measurer + Training Scheduler" (Sec. 4.1), which unifies most of CL methodologies. Based on this framework, we categorize the existing CL methods into predefined CL (Sec. 4.2) and automatic CL (Sec. 4.3) and introduce the representative designs in each category. Fig. 2 illustrates the typology of CL methods introduced in this section.

4.1 The General Framework of Difficulty Measurer + Training Scheduler

Recall that the core definition of CL (Definition 1) lies in the strategy of "training from easier data to harder data". In essence, to design such a curriculum, we need to decide two things: 1) What kind of training data is supposed to be easier than other data? 2) When should we present more harder data for training, and how much more? Issue 1) can be abstracted to a **Difficulty Measurer**, which decides the relative "easiness" of each data example. Issue 2) can be abstracted to a **Training Scheduler**, which decides the sequence of data subsets throughout the training process based on the judgment from the Difficulty Measurer.

Therefore, a general framework for curriculum design consists of these two core components: Difficulty Measurer + Training Scheduler, which is illustrated in Fig 2(a). To begin with, all the training examples are sorted by the Difficulty Measurer from the easiest to the hardest and passed to the Training Scheduler. Then, at each training epoch t, the Training Scheduler samples a batch of training

data from the relatively easier examples and sends it to Model Trainer for training. With the progress of training epochs, Training Scheduler will decide when to sample from more harder data, (usually) until uniform sampling from the whole training set. This schedule sometimes also depends on the training loss feedback from the Model Trainer (the dashed arrow in Fig 2(a)), e.g., Training Scheduler presenting more harder data when the current model converges. Note that in [33], the authors conclude the two core components as *scoring function* and *pacing function*, which share the same spirit with Difficulty Measurer and Training Scheduler, respectively, while the latter names adopted in this paper are chosen to be more abstract and clearer.

Let us take the experiment in Fig 1 as an instantiation example for our CL framework. Difficulty Measurer is the human annotations deciding that some fruit images in the dataset are easier than other images based on recognizability and complexity. Training Scheduler can be, for example, a linear scheduler (see Sec. 4.2.2) that starts with 40% of easiest examples in each class, and increases this proportion by 5% each epoch until 100%. In this way, an effective curriculum is designed by instantiating the general CL framework according to the specific image classification task.

According to our framework, we could also clarify the scopes of predefined CL and automatic CL in the next two sections. Specifically, when both the Difficulty Measurer and Training Scheduler are designed by human prior knowledge with no data-driven algorithms involved, we call the CL method **predefined CL**. If any (or both) of the two components are learned by data-driven models or algorithms, then we denote the CL method as **automatic CL**.

4.2 Predefined CL

In this section, we discuss the common types of manually predefined Difficulty Measurers (Sec. 4.2.1) and Training Schedulers (Sec. 4.2.2) under our CL framework, and conclude the main limitations of predefined CL (Sec. 4.2.3).

4.2.1 Common Types of Predefined Difficulty Measurer

Researchers have manually designed various Difficulty Measurers mainly based on the data characteristics of specific tasks. We summarize common types of Difficulty Measurers in Table 2. Most of the predefined Difficulty Measurers are designed for image and text data in various CV and NLP scenarios, while other data types include audio data, programs, tabular data, etc. Interestingly, we find that except for some domain knowledge-based measurement (marked as "Domain"), most of the predefined Difficulty Measurers are designed from the angles of complexity, diversity, and noise estimation, which are separate but also correlated.

Firstly, *complexity* stands for the structural complexity of a particular data example, such that examples with higher complexity have more dimensions and are thus harder to be captured by models. For instance, sentence length, the most popular Difficulty Measurer in NLP tasks [86], [107], [112], intuitively expresses the complexity of a sentence/paragraph. Therefore, longer sentences are often supposed as harder training data. Other examples include the number of objects in images in the task of semantic segmentation [122]; the number of coordinating conjunctions (e.g., "and", "or") [50] or phrases (e.g., prepositional phrases) [113]; the parse tree depth [113] that measures the sentence complexity in the view of grammar; and the nesting of operations in program text [131] that measures the complexity of the instruction set in program execution tasks.

TABLE 2
Common types of predefined Difficulty Measurer. The "+" in $\propto$Easy means the higher the measured value, the easier the data example, and the "-" has the opposite meaning.

Difficulty Measurer*	Angle	Data Type	$\propto$Easy
Sentence length [86], [107]	Complexity	Text	-
Number of objects [122]	Complexity	Images	-
# conj. [50], #phrases [113]	Complexity	Text	-
Parse tree depth [113]	Complexity	Text	-
Nesting of operations [131]	Complexity	Programs	-
Shape variability [6]	Diversity	Images	-
Word rarity [50], [86]	Diversity	Text	-
POS entropy [113]	Diversity	Text	-
Mahalanobis distance [14]	Diversity	Tabular	-
Cluster density [11], [31]	Noise	Images	+
Data source [10]	Noise	Images	/
SNR / SND [7], [89]	Noise	Audio	-
Grammaticality [66]	Domain	Text	+
Prototypicality [113]	Domain	Text	+
Medical based [44]	Domain	X-ray film	/
Retrieval based [18], [82]	Domain	Retrieval	/
Intensity [30] / Severity [111]	Intensity	Images	+
Image difficulty score [106], [114]	Annotation	Images	-
Norm of word vector [68]	Multiple	Text	-

* Abbreviations: POS = Part Of Speech, SNR = Signal to Noise Ratio, SND = Signal to Noise Distortion, Domain = Domain knowledge, # conj. = number of coordinating conjunctions.

Secondly, the angle of *diversity* here stands for the distributional diversity of a group of data (e.g., regular or irregular shapes [6]) or the elements (e.g., words) of a data point (e.g., sentence). A larger value of diversity means the data is more various, including more (rare) types/styles of data or elements, and is thus more difficult for model learning. For example, a sentence with more rare words is usually considered harder to learning [86]. A popular measure of diversity is information entropy, which is exploited both in text data as the Part-Of-Speech (POS) entropy [113] and in tabular data as the Mahalanobis distance of feature vectors [14]. Intuitively, both high complexity and high diversity bring more degrees of freedom to the data, which needs a model with larger capacity and bigger effort of training.

Larger diversity sometimes also makes the data noisier. Therefore, another angle is *noise estimation*, which estimates the noise level of data examples and defines cleaner data as easier. A quite intuitive method is taken in [10] to judge the noise level by the source of image data on the web: images retrieved by a search engine like Google are supposed to be cleaner, and images posted on photo-sharing website like Flickr are more realistic and noisier. In [31], the authors map images to vectors by CNNs and suppose that cleaner images often appear similar, and thus have larger values of local density. Therefore, examples with lower local density are supposed to be noisier and harder to predict. Moreover, the Signal to Noise Ratio/Distortion (SNR/SND) [7], [89] is widely adopted to estimate the noise in audio data.

Other interesting Difficulty Measurers include signal intensity [30], [111] and human-annotation-based Image Difficulty Scores [106], [114], both designed for image data. Signal intensity can be regarded as a measurement for the informativeness of data features. For example, in the task of facial expression recognition [30], more intense/exaggerated

faces are supposed to be easier data than poker faces. In the task of thoracic disease diagnosis [111], more severe symptoms provide more information and are easier to recognize. Moreover, Image Difficulty Score [114] is proposed to measure the difficulty of an image by collecting the response times of human annotators in the following protocol: (i) ask the annotator "Is there an {object class} (e.g., elephant) in the next image?" and (ii) record the time spent by the annotator to answer "Yes" or "No" and use this response time to estimate Image Difficulty Score: intuitively, longer response time corresponds to harder image example. After collecting the annotation, the authors train a regression model to map the CNN features of new images to the difficulty score.

4.2.2 Common Types of Predefined Training Scheduler

While predefined Difficulty Measurers vary among different data types and tasks, the existing predefined Training Schedulers are usually data/task agnostic, i.e., the majority of CL literature in various scenarios leverages similar types of Training Schedulers. Generally, Training Schedulers can be divided into *discrete* and *continuous* schedulers. The difference is: discrete schedulers adjust the training data subset after every fixed number (> 1) of epochs or convergence on the current data subset, while continuous schedulers adjust the training data subset at every epoch.

Discrete schedulers are widely adopted owing to their simplicity and effectiveness. The most popular discrete scheduler is named as *Baby Step* [6], [107] (Algorithm 1), which first distributes the sorted data into buckets (or shards/bins) from easy to hard and starts training with the easiest bucket. After a fixed number of training epochs or convergence, the next bucket is merged into the training subset. Finally, after all the buckets are merged and used, the whole training process either stops or further continues several extra epochs. Note that at each epoch, the scheduler usually shuffles both the current buckets and the data in each bucket and then sample mini-batches for training (instead of using all data at once).

Algorithm 1 The Baby Step Training Scheduler [12].

Input: $\mathcal{D}$: training dataset; $\mathcal{C}$: the Difficulty Measurer;
Output: M^*: the optimal model.
1: $\mathcal{D}' = \text{sort}(\mathcal{D}, \mathcal{C})$;
2: $\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^k\} = \mathcal{D}'$ where $\mathcal{C}(d_a) < \mathcal{C}(d_b),\ d_a \in \mathcal{D}^i, d_b \in \mathcal{D}^j, \forall i < j$;
3: $\mathcal{D}^{train} = \emptyset$;
4: **for** $s = 1 \cdots k$ **do**
5: $\quad \mathcal{D}^{train} = \mathcal{D}^{train} \cup \mathcal{D}^s$;
6: $\quad$ **while** not converged for p epochs **do**
7: $\quad\quad$ train$(M, \mathcal{D}^{train})$;
8: $\quad$ **end while**
9: **end for**

Another discrete scheduler called *One-Pass* [6] takes a similar strategy of data bucketing from easy to hard and starting training from the easiest bucket. However, when updating, One-Pass scheduler discards the current bucket and switches to the next harder bucket. One-Pass is less used than Baby Step in CL literature (see [112], [114], [117], [131] for One-Pass examples), probably due to the lower performance in many tasks. Intuitive reasons might include: 1) The complexity/diversity of the training data is gradually increasing in Baby Step scheduler, which helps improve generalization capacity; 2) The One-Pass scheduler is like training on a sequence of independent tasks as in continual learning [13], which faces the problem of catastrophic forgetting even though the early tasks are easier. The two schedulers are compared on LSTMs in [12].

Other discrete schedulers are also based on data bucketing but take different sampling strategies. For example, in [50], the authors modify the Baby Step to unevenly divide the examples into buckets such that easier buckets have more data examples, which is natural to reach in the case of machine translation corpora. Then they sample examples without replacement from the easiest bucket only until there remain the same number of examples as in the second most easy bucket. Afterward, they uniformly sample from the first two buckets until the size is the same as that of the third bucket. In an empirical study of CL on NMT tasks [138], the authors also test other extensions of Baby Step, including 1) "boost": to copy the hardest bucket for further training; 2) "reduce and add-back": to gradually remove one easiest bucket from training set once all buckets have been used, and then add them back and repeat the removing until convergence; 3) "no-shuffle": to discard inter-bucket shuffling and always present from easier to harder buckets to the model. A conclusion is, including Baby Step, no single scheduler consistently outperforms others.

Continuous schedulers, on the other hand, can be mostly regarded as a function $\lambda(t)$ to map training epoch number t to a scalar $\lambda \in (0, 1]$, which means λ proportion of easiest training examples are available at the t-th epoch. According to the Definition 1 in Sec. 2, this function $\lambda(t)$ must be monotone and non-decreasing, starting at $\lambda(0) > 0$ and ending at $\lambda(T) = 1$ This function is also called *pacing function* [33] or *competence function* [86] in literature.

Existing $\lambda(t)$ functions are various, while researchers could design new functions for their specific tasks. The most intuitive function is the *linear function*, where λ_0 is the initial proportion of available easiest examples, and T_{grow} denotes the epoch when the function reaches 1 for the first time.

$$\lambda_{\text{linear}}(t) = \min\left(1, \lambda_0 + \frac{1-\lambda_0}{T_{\text{grow}}} \cdot t\right) \tag{2}$$

Root function is later proposed in [86] according to the observation that in linear function, the newly added examples are less likely to be sampled as the training data subset grows in size. Therefore, to give the model sufficient time to learn the newly added examples, the authors reduce the number of newly added examples as training progresses by defining the rate of adding examples to be inversely proportional to the size of the current training subset: $\frac{d\lambda(t)}{dt} = \frac{P}{d\lambda(t)}$, where $P \geq 0$ is a constant. Then we get:

$$\lambda_{\text{root}}(t) = \min\left(1, \sqrt{\frac{1-\lambda_0^2}{T_{\text{grow}}} \cdot t + \lambda_0^2}\right). \tag{3}$$

To make the curve even sharper, a more general form *root-p function* is also considered as follows, where $p \geq 1$:

$$\lambda_{\text{root-p}}(t) = \min\left(1, \sqrt[p]{\frac{1-\lambda_0^p}{T_{\text{grow}}} \cdot t + \lambda_0^p}\right). \tag{4}$$

Interestingly, in [82] the authors oppositely propose to give easier examples more training time, by taking the following *geometric progression function*:

$$\lambda_{\text{geom}}(t) = \min\left(1, 2^{\left(\frac{\log_2 1 - \log_2 \lambda_0}{T_{\text{grow}}} \cdot t + \log_2 \lambda_0\right)}\right). \tag{5}$$

The above continuous scheduler functions are illustrated in Figure 5. Note that training without CL ("baseline") and Baby Step are also regarded as special cases of continuous schedulers. The experiments in [86] and [82] on NLP tasks show that the root-p function ($p \geq 2$) is the most beneficial predefined Training Scheduler for CL, though the relative improvement to other schedulers is not drastic.

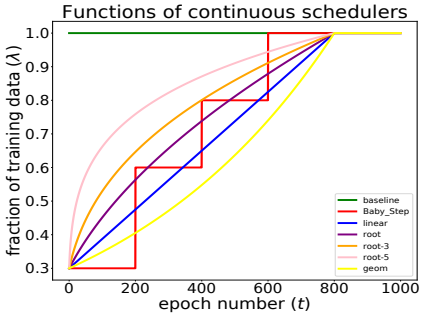

Fig. 5. Visualization of common continuous schedulers. The horizontal axis t stands for the training epoch number, and the vertical axis λ is the corresponding proportion of the easiest training data subset. Baseline is without curriculum and involves the whole training set from the beginning. The Baby Step scheduler is also visualized for comparison.

Moreover, there is also a special group of continuous schedulers which do not follow the original definition of CL but perform as a sequence of data selection as in Definition 2. We name these schedulers as *distribution shift*, which start training on an initial distribution and gradually move to a target distribution. For example, in [66], all the examples are divided into 2 groups: Common (lower quality and simpler) and Target (higher quality and more complex). The sampling weights are initially distributed on the Common and gradually shifted to the Target. In [39], to alleviate extreme data imbalance in the lung nodule detection task, the scheduler starts sampling purely from images with nodules to learn to represent nodules, and then gradually decreases the proportion of examples with nodules until the extremely imbalanced data distribution (rare nodule).

4.2.3 Limitations of predefined CL

Despite the simplicity and effectiveness of the predefined CL, there are some essential limitations as follows. (i) It is difficult to find the most suitable combination of Difficulty Measurer and Training Scheduler for a specific task and its dataset. There are no existing methodologies for selecting Difficulty Measurer and Training Scheduler other than exhaustive trials. (ii) Both the predefined Difficulty Measurers and Training Schedulers stay fixed during the training process, which is not flexible enough and to some extent ignores the feedback of the current model. (iii) Expert domain knowledge is often necessary for designing a predefined Difficulty Measurer. Moreover, when the dimension of example features is large, it is hard to predefine a computable Difficulty Measurer even by an expert. (iv) Easy examples for humans are not always easy for models, since the decision boundaries of models and humans are basically different [130]. (v) The best hyperparameters[1] of Training Scheduler are hard to find. Additionally, a basic problem in Baby Step scheduler is to decide the number of buckets and how to divide the buckets[2]. (vi) The performance of various predefined Training Schedulers is sensitive to the initial learning rate (in NMT task) [138].

These limitations of predefined CL have prevented CL from being explored in more various applications. A natural and critical question is: how can we design more automatic Difficulty Measurers and Training Schedulers, which are more data- and model-driven instead of human-driven, more dynamically adaptive to the current training, and need fewer or even no hyperparameters to fine-tune?

4.3 Automatic CL

In this section, we take a further step on the curriculum design by introducing automatic CL methods to break through the limits of predefined CL. A general comparison of predefined CL and automatic CL is presented in Table 3.

TABLE 3
Predefined CL v.s. automatic CL.

Issues	Predefined CL	Automatic CL
Applicability	Need expert domain knowledge	General, domain agnostic
Difficulty Measurer	Human defined, fixed	Model decided, dynamic
Training Scheduler	Ignore model feedback, fixed	Consider model feedback, dynamic

We summarize the four major methodologies for automatic CL. In predefined CL, the *teacher* designing the curriculum is a human expert, and the *student* getting trained by the curriculum is the machine learning model. To reduce the need for human teachers, the four methodologies take different ideas, which can be intuitively summarized as follows. (i) **Self-Paced Learning (SPL)** methods let the student himself act as the teacher and measure the difficulty of training examples according to its losses on them. This strategy is analogous to the self-study of human students: one decides his/her own learning pace based on his/her current status. (ii) **Transfer Teacher** methods invite a strong teacher model to act as the teacher and measure the difficulty of training examples according to the teacher's performance on them. The teacher model is pretrained and transfers its knowledge to measure example difficulty for student model training. (iii) **RL Teacher** methods adopt reinforcement learning (RL) models as the teacher to play dynamic data selection according to the feedback from the student. This strategy is the most ideal scene in human education, where the teacher and student improve together through benign interactions: the student makes the biggest progress based on the tailored learning materials selected by the teacher, while the teacher also effectively adjusts her teaching strategy to teach better. (iv) **Other Automatic CL** methods include various automatic CL strategies except for the above-mentioned. The works take different optimization techniques to automatically find the best curriculum

1. The hyperparameters include λ_0, T_{grow} and p (in root-p function) in continuous schedulers, and the number of steps, the number of epochs in each step in Baby Step based schedulers.

2. Division by thresholds on difficulty scores makes it hard to assign each bucket with roughly the same number of examples, while division by size may result in fluctuations in difficulty within a bucket or not enough difference between different buckets [138]. An alternative is the Jenks Natural Breaks classification algorithm, as adopted in [138].

for model training, including Bayesian Optimization, meta-learning, hypernetworks, etc. Taking Definition 2 or 3, the curriculum in these methods often refers to a sequence of loss weights or even loss functions on data batches.

The comparison of these automatic CL methodologies is in Table 4. Automatic CL is also broadly applied to Deep RL tasks, and we refer readers to the recent surveys [76], [87] for further reading. The automatic CL methods discussed in this section are mostly designed for (weakly- or un-) supervised learning settings, though some of them are also shown to be effective for RL tasks [48], [72].

4.3.1 Self-Paced Learning

Self-paced Learning (SPL) is a primary branch of CL that automates the Difficulty Measurer by taking the example-wise training loss of the current model as criteria. The concept of "self-paced learning" originates from human education, where the student can control the learning curriculum, including what to study, how to study, when to study, and how long to study [115]. Under machine learning settings, SPL refers in particular to a training strategy initially proposed by Kumar et al. [54], which trains the model at each iteration with the proportion of data with the lowest training losses. This proportion of easiest examples gradually grows to the whole training set, which essentially takes a predefined Training Scheduler in Sec. 4.2.2. Note that in the literature of SPL, CL and SPL are usually mentioned as two different strategies, where the CL actually refers to the predefined CL in Sec. 4.2. However, in this paper, SPL is regarded as a branch of automatic CL, since it shares the same spirit with CL and fits perfectly with our general CL framework, as shown in Fig 2(b). The most valuable advantages of SPL over predefined CL are mainly two-fold: 1) SPL is semi-automatic CL with a loss-based automatic Difficulty Measurer and dynamic curriculum, which makes it more flexible and adaptive for various tasks and data distributions. 2) SPL embeds the curriculum design into the learning objective of the original machine learning tasks, which makes it widely applicable as a plug-in tool.

a) The Original Version of SPL. The original SPL algorithm [54] is formally defined as follows. Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ denotes the training set, where x_i and y_i is the feature and label of example i, respectively. The model $f_{\boldsymbol{w}}$ with parameters $\boldsymbol{w}$ maps each x_i to the model prediction $f_{\boldsymbol{w}}(x_i)$, and gets a loss $l_i = L(f_{\boldsymbol{w}}(x_i), y_i)$, where L is the learning objective. The original goal is then to minimize the empirical loss on the whole training set:

$$\min_{\boldsymbol{w}} \mathbb{E}(\boldsymbol{w}; \lambda) \sum_{i=1}^{N} l_i + R(\boldsymbol{w}), \tag{6}$$

where $R(\boldsymbol{w})$ is a regularizer to encode prior knowledge on $\boldsymbol{w}$ to avoid overfitting[3]. SPL introduces example weight v_i into the above learning objective with an *SP-regularizer* $g(\boldsymbol{v}; \lambda)$, where $\boldsymbol{v} = [v_1, v_2, ..., v_N]^\top \in [0, 1]^N$ is a vector of weights, and λ is the *age parameter*, a hyperparameter which controls the learning pace (i.e., as Training Scheduler) and determines the proportion of the easiest selected examples at each training epoch. The new learning objective becomes:

$$\min_{\boldsymbol{w}; \boldsymbol{v} \in [0,1]^N} \mathbb{E}(\boldsymbol{w}, \boldsymbol{v}; \lambda) \sum_{i=1}^{N} v_i l_i + g(\boldsymbol{v}; \lambda). \tag{7}$$

In the original SPL, $g(\boldsymbol{v}; \lambda)$ is a negative l_1-norm:

$$g(\boldsymbol{v}; \lambda) = -\lambda \sum_{i=1}^{N} v_i. \tag{8}$$

The above learning objective is often optimized with the Alternative Optimization Strategy (AOS)[4]. Concretely, we alternatively optimize $\boldsymbol{w}$ and $\boldsymbol{v}$ while fix the other. With the fixed $\boldsymbol{w}^*$, we calculate the global optimum $\boldsymbol{v}^*$ by solving:

$$v_i^* = \arg \min_{v_i \in [0,1]} v_i l_i + g(v_i; \lambda), \quad i = 1, 2, \cdots, n \tag{9}$$

Then, with fixed $\boldsymbol{v}^*$, we learn the global optimum $\boldsymbol{w}^*$:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w}} \sum_{i=1}^{N} v_i^* l_i. \tag{10}$$

The two optimization steps are iteratively conducted, while the value of λ is gradually increased to add more harder examples. The overall algorithm is in Algorithm 2.

Algorithm 2 Self-Paced Learning

Input: $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$: training dataset; f: the machine learning model; T: the maximum number of iterations;
Output: $\boldsymbol{w}$: the optimal parameters of f.
1: Initialize $\boldsymbol{w}$, $\boldsymbol{v}$, $\lambda = \lambda_0$, $t = 0$.
2: **while** $t \neq T$ **do**
3: $t = t + 1$;
4: Update $\boldsymbol{v}^*$ by Eq. 9;
5: Update $\boldsymbol{w}^*$ by Eq. 10;
6: Update λ to a larger value; // to include harder data
7: **end while**

While the solution for Eq. 10 is provided by machine learning algorithms (e.g., gradient descent) for the original task, the solution for Eq. 9 is simple. In fact, since $g(\boldsymbol{v}; \lambda)$ in Eq. 8 is a convex function of $\boldsymbol{v}$, the global minimum can be easily derived by setting the partial derivative of $\mathbb{E}(\boldsymbol{w}, \boldsymbol{v}; \lambda)$ to v_i as zero. Considering $v_i \in [0, 1]$, we get the close-formed optimal solution for $\boldsymbol{v}^*$ with the fixed $\boldsymbol{w}^*$:

$$v_i^* = \begin{cases} 1, & l_i < \lambda \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

This solution can be intuitively explained: if an example has a training loss l_i less than the threshold λ, then it is regarded as an *easy* example for the current model, and should be selected at the current training epoch (i.e., $v_i^* = 1$). Otherwise, it is *hard* and should not be selected (i.e., $v_i^* = 0$). When the model becomes more mature, λ gets increased and more harder examples get involved in training.

Another remaining issue is how to adjust the threshold λ throughout the training. Initially, λ should be set as λ_0 to ensure that a small proportion of easy examples are selected. Later on, a simple method is to multiply or add a constant at each epoch, i.e., $\lambda_{t+1} = \eta \cdot \lambda_t$ $(\eta > 1)$ or $\lambda_{t+1} = \lambda_t + \mu$ $(\mu > 0)$, to gradually increase λ. Finally, λ becomes large enough so that all the examples are selected (i.e., $v_i^* = 1 \quad \forall i$). This strategy of adjusting λ is analogous to predefined continuous Training Scheduler. More methods for adjusting λ will be discussed in (e).

3. For brevity, we ignore $R(\boldsymbol{w})$ in the following discussion.

4. AOS is also called ASS (Alternative Search Strategy), ACS (Alternative Convex Search) [42], or CCM (Cyclic Coordinate Method) [40] in SPL literature.

TABLE 4
Comparison of the automatic CL methodologies, except "Other Automatic CL".

Issues	Self-Paced Learning	Transfer Teacher	RL Teacher
Characteristic	Student-driven difficulty	Teacher-driven difficulty	Teacher select data according to student feedback
Difficulty Measurer	Automatic	Automatic	Automatic
Training Scheduler	Predefined	Predefined	Automatic
Strength	Efficient, robust	Reliable difficulty	Flexible
Weakness	Fixed strategy	Extra pretraining	Costly (Deep RL)
CL Definition	Definition 1	Definition 1	Definition 2

b) Theories for SPL. Before we discuss variant SPL versions enhanced from different aspects, we briefly summarize existing theories on SPL. In short, sound theories have been established for the convergence, robustness, and essence of SPL to support its wide applications.

To begin with, the new learning objective Eq. 7 in SPL is equivalent to the following latent objective function:

$$\sum_{i=1}^{N} F_\lambda(l_i) = \sum_{i=1}^{N} \int_0^{l_i} v_i^*(\tau, \lambda) d\tau \tag{12}$$

where v_i^* is the solution in Eq. 9. Meng et al. [73] first prove that the AOS strategy in SPL intrinsically accords with the majorization minimization (MM) algorithm [56] on a minimization problem of the above latent SPL objective. Therefore, one could leverage theories of MM to provide analyses of the properties of SPL (e.g., convergence). Additionally, they find that this latent objective $\sum_{i=1}^{N} F_\lambda(l_i)$ is also closely related to the non-convex regularized penalty (NCRP), a well-known machine learning methodology with attractive properties in sparse estimation and robust learning, which provides evidence on the robustness of SPL. Based on this work, the authors further prove that the optimization of $\sum_{i=1}^{N} F_\lambda(l_i)$ converges to critical points of the original SPL problem under mild conditions [71].

Moreover, Liu et al. [67] establish a systematic framework for SPL under concave conjugacy theory, which completely tallies with the requirements of SPL models. Based on this framework, they provide a proof for the derived relationship among the SP-regularizer $g(\boldsymbol{v}; \lambda)$, latent objective $\sum_{i=1}^{N} F_\lambda(l_i)$, and the example weights $\boldsymbol{v}$. This result also inspires two general approaches for SPL designs.

c) Soft SP-regularizers. As a weighting strategy on the learning objective, the core design of SPL is the SP-regularizer $g(\boldsymbol{v}; \lambda)$, which directly determines the optimal weights $\boldsymbol{v}^*$ at each training epoch. Therefore, most of the existing improvements on SPL have been focused on SP-regularizers. Recall that in the original version of SPL, $g(\boldsymbol{v}; \lambda)$ leads to a hard/binary weighting on the examples (Eq. 11), assigning 1 to easy examples and 0 to hard examples. However, this style of *hard weights* tends to lose flexibility, since any two "easy" (or "hard") examples are unlikely to be strictly equally important and learnable [141]. Therefore, an intuitive choice is to design new SP-regularizers to result in *soft weights* $\boldsymbol{v}^*$. We call such a group of SP-regularizers *soft regularizers*.

A list of existing SP-regularizers $g(\boldsymbol{v}; \lambda)$ and the corresponding close-formed solutions of $\boldsymbol{v}^*$ is shown in Table 5. In addition, the l-v^* functions (i.e., the function of example weight v_i^* w.r.t. losses l_i) of these solutions are visualized in Fig 6. As in Fig 6, compared to the hard regularizer, the solutions of various soft regularizers assign soft weights to reflect example importance in finer granularity, which helps

TABLE 5
Common types of SP-regularizers $g(\boldsymbol{v}; \lambda)$ and the corresponding close-formed solutions $v^*(l; \lambda)$.

Regularizers	$g(\boldsymbol{v}; \lambda)$	$v_i^*(l_i; \lambda)$
Hard [54]	$-\lambda \sum_{i=1}^{N} v_i$	$\begin{cases} 1, & l_i < \lambda \\ 0, & \text{otherwise} \end{cases}$
Linear [40]	$\frac{1}{2}\lambda \sum_{i=1}^{N} (v_i^2 - 2v_i)$	$\begin{cases} 1 - l_i/\lambda, & l_i < \lambda \\ 0, & \text{otherwise} \end{cases}$
Logarithmic [40]	$\sum_{i=1}^{N} \left(\zeta v_i - \frac{\zeta^{v_i}}{\log \zeta}\right)$ $\zeta = 1 - \lambda,\ 0 < \lambda < 1$	$\begin{cases} \frac{\log(l_i + \zeta)}{\log \zeta}, & l_i < \lambda \\ 0, & \text{otherwise} \end{cases}$
Mixture [40]	$-\zeta \sum_{i=1}^{N} \log\left(v_i + \frac{\zeta}{\lambda_1}\right)$ $\zeta = \frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2},$ $\lambda_1 > \lambda_2 > 0$	$\begin{cases} 1, & l_i \le \lambda_2 \\ 0, & l_i \ge \lambda_1 \\ \zeta\left(\frac{1}{l_i} - \frac{1}{\lambda_1}\right), & \text{otherwise} \end{cases}$
Mixture2 [141]	$\frac{\gamma^2}{v_i + \frac{\gamma}{\lambda}}, \quad \gamma > 0$	$\begin{cases} 1, & l_i \le \left(\frac{\lambda\gamma}{\lambda + \gamma}\right)^2 \\ 0, & l_i \ge \lambda^2 \\ \gamma\left(\frac{1}{\sqrt{l_i}} - \frac{1}{\lambda}\right), & \text{otherwise} \end{cases}$
Logistic [127]	$\sum_{i=1}^{N} \ln(\mu_i)^{\mu_i}$ $+\ln(v_i)^{v_i} - \lambda v_i, \quad \lambda > 0,$ $\mu_i = 1 + e^{-\lambda} - v_i$	$\frac{1+e^{-\lambda}}{1+e^{l_i - \lambda}}$
Polynomial [26]	$\lambda\left(\frac{1}{t}\sum_{i=1}^{N}\sum_{i=1}^{N} v_i\right)$ $\lambda > 0,\ t \in \mathbb{N}^+$	$\begin{cases} \left(1 - \frac{l_i}{\lambda}\right)^{\frac{1}{t-1}}, & l_i < \lambda \\ 0, & \text{otherwise} \end{cases}$

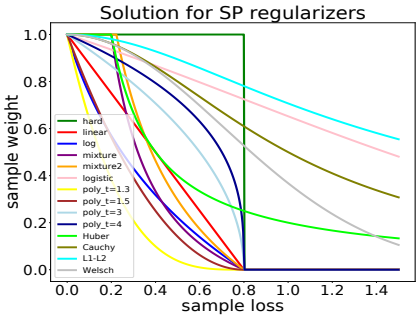

Fig. 6. Visualization of functions of best example weight v_i^* w.r.t. losses l_i (the l-v^* functions) of the SP-regularizers in Table 5. The age parameter λ (the threshold for non-zero weights) for many of the functions are set as 0.8. The Huber, Cauchy, L1-L2, and Welsch belong to the implicit SP-regularizers in [16], which are not presented in the table.

soft regularizers achieve better performance in various applications. However, one needs to choose suitable soft regularizers for specific scenarios. For example, the *logarithmic* is more prudent than the *linear*, while the *mixture* regularizers tolerate small losses, compared with other regularizers [40]. The *polynomial* regularizer extends the *linear* to arbitrary orders (when $t = 2$, it is identical to *linear*), and Li et al. [60] further propose to dynamically adjust the order t during training to improve flexibility.

To allow more possibility on SP-regularizer designs, a general and formal definition is taken as follows [40], [141]:

Definition 4: SP-regularizer. Suppose that v is a weight

variable, l is the loss, and λ is the age parameter. $g(\boldsymbol{v};\lambda)$ is called a self-paced regularizer, if:

1. $g(v;\lambda)$ is convex w.r.t. $v \in [0,1]$;

2. $v^*(l;\lambda)$ is monotonically decreasing w.r.t. l, and $\lim_{l\to 0} v^*(l,\lambda) = 1$, $\lim_{l\to\infty} v^*(l,\lambda) = 0$;

3. $v^*(l;\lambda)$ is monotonically increasing w.r.t. λ, and $\lim_{\lambda\to\infty} v^*(l,\lambda) \leq 1$, $\lim_{\lambda\to 0} v^*(l,\lambda) = 0$;

where $v^*(l;\lambda)$ is defined in Eq. 9.

It is not difficult to verify that all the regularizers in Table 5 conform to Definition 4. Based on this definition, Li et al. [58] propose a general framework for designing SP-regularizers, demonstrating that we can derive from any S-shaped $v^*(l;\lambda)$ which meets Conditions 2 and 3 to create new SP-regularizers. Essentially, this framework is equivalent to the theorem in [67].

While the SP-regularizers defined by Definition 4 have explicit form, Fan et al. [16] further introduce *implicit regularizers* into SPL (denoted as SPL-IR). Based on the convex conjugacy theory, a group of implicit SP-regularizers, whose analytic form can be even unknown, are deduced from some well-studied robust loss functions (e.g., Huber loss function), and the corresponding best weights $\boldsymbol{v}^*(l;\lambda)$ can be directly derived from these loss functions. The weights thus inherit the good robustness properties, which helps SPL-IR to outperform explicit SP-regularizers. The l-v^* functions of implicit regularizers derived from four types of robust loss functions, i.e., Huber, Cauchy, L1-L2, and Welsch loss functions, are visualized in Fig 6. [5]

d) Prior-embedded SPL. In SPL methods, given fixed SP-regularizers $g(\boldsymbol{v};\lambda)$, the example weights $\boldsymbol{v}^*$ are entirely determined by the example-wise losses and the age parameter λ. However, in some cases, we hope to introduce some *loss prior knowledge* into this learning scheme. For example, we may want to compulsively assign outliers with $v_i = 0$ to improve robustness, or assign pre-known high-quality examples with $v_i = 1$. Such prior knowledge is closely related to the predefined Difficulty Measurer in Sec. 4.2.1.

Fortunately, the AOS algorithm naturally decomposes SPL into two problems of optimizing $\boldsymbol{w}$ and $\boldsymbol{v}$, which makes it feasible to embed the loss prior knowledge into SPL by encoding it as a part of SP-regularizer or a constraint on $\boldsymbol{v}$. Four typical types of priors are summarized in [73] as follows: i) *Outlier prior*: Some outliers in the datasets show extremely large losses. ii) *Spatial/temporal smoothness prior*: Spatially or temporally adjacent examples tend to have similar losses. iii) *Sample importance order prior*: Some examples are pre-known to be more important than others. iv) *Diversity prior*: Important examples should be scattered across the data range to help learn global data knowledge.

A famous representative of Prior (iv) is SPL with diversity (SPLD) [41], which incorporates a negative $l_{2,1}$-norm into the hard SP-regularizer to avoid overfitting to a data subset while ignoring easy examples in other groups:

$$g(\boldsymbol{v};\lambda,\gamma) = -\lambda\sum_{i=1}^{N} v_i - \gamma\sum_{j=1}^{b}\|\boldsymbol{v}^{(j)}\|_2 \tag{13}$$

where $\gamma > 0$ is a balance factor between easiness and diversity, b is the number of groups (e.g., themes in the video event detection task) in the training set, and $\boldsymbol{v}^{(j)}$ is a vector of corresponding example weights v_i in group j. Since the $l_{2,1}$-norm is well-known to lead to group-wise sparse representation, its opposite term should then encourage diversity of non-zero v_i across groups. Alternatively, we can also adopt $-l_{0.5,1}$-norm [135], i.e., $-\sum_{j=1}^{b}\sqrt{\sum_{i=1}^{n_j} v_i^{(j)}}$, where n_j is the size of group j. This diversity term makes the whole $g(\boldsymbol{v};\lambda,\gamma)$ conform with Definition 4. While both $-l_{2,1}$-norm and $-l_{0.5,1}$-norm are based on the Group LASSO [129], Exclusive LASSO [51] can be also adopted [22], [35] by taking $-l_{1,2}$-norm to select confident samples from diverse groups or clusters.

For Prior (iii), a representative work is self-paced curriculum learning (SPCL) [42], which introduces a *curriculum region* Ψ with formal definition as a convex feasible region constraint on $\boldsymbol{v}$. SPCL combines the power of SPL and predefined CL, whose objective is as follows:

$$\min_{\boldsymbol{w};\boldsymbol{v}\in[0,1]^N} \mathbb{E}(\boldsymbol{w},\boldsymbol{v};\lambda,\Psi)\sum_{i=1}^{N} v_i l_i + g(\boldsymbol{v};\lambda). \quad \text{s.t.}\boldsymbol{v}\in\Psi \tag{14}$$

An example of Ψ is $\{\boldsymbol{v}|\boldsymbol{a}^\top\boldsymbol{v} \leq c\}$, where c is a constant and $\boldsymbol{a}$ is a N-dimensional vector derived from the total order relationship among the N examples[6]. Theoretical analysis on SPCL is provided in [67].

Another method for Prior (iii) is proposed in [134], which is helpful when the precise total order knowledge is hard to obtain. Similar to the $-l_{2,1}$-norm for Prior (iv), this method encodes the prior knowledge about image difficulty by adding a regularization term $h(\boldsymbol{v};\eta,\boldsymbol{p}) = -\eta\sum_{i=1}^{N} p_i v_i$ to the objective, where p_i indicates the priority values of each image. A larger p_i means the example i is easier and should be assigned larger weight v_i. To generate such p_i, all the Difficulty Measurers discussed in Sec. 4.2.1 can be adopted. Moreover, SPFTN [137] also jointly embeds prior (iii) and (iv) by the weighted sum of terms in [134] and [135].

Note that when the above kinds of convex constraint on $\boldsymbol{v}$ is applied, we could no longer use the close-formed solutions of $\boldsymbol{v}^*$ in Table 5. Instead, we can calculate $\boldsymbol{v}^*$ by applying gradient-based methods [42] or other off-the-shelf techniques like CVX toolbox [134] due to the convexity.

e) Other enhancements of SPL. Besides the various enhanced versions of SP-regularizers, there remain some other aspects to be carefully considered in SPL. A key element in SPL is the age parameter λ. As aforementioned, traditional SPL takes a naive strategy to add/multiply λ with a constant at each epoch. However, with the model making progress, the losses on all the examples are expected to become smaller and smaller, and thus an monotonic increasing threshold λ may add much more hard examples in the early epochs. For some SP-regularizers it would be more effective to gradually decrease the value of λ [16]. To design a better update strategy for λ, some works [60], [92] adopt a strategy analogous to Baby Step scheduler in Sec. 4.2.2. They predefine a sequence $\boldsymbol{N} = \{N_1, N_2, \cdots, N_T\}$ ($N_s < N_t$ for all $s < t$, $N_T = N$), where N_t is the number of selected examples in the t-th epoch. Then, the threshold of λ is dynamically updated to ensure exactly N_t examples are

5. For clearer comparison with other explicit l-v^* functions, we divide the weights v^* by 2 in Cauchy and Welsch. This linear scaling does not influence training if we accordingly amplify the learning rates in SGD.

6. $a_i < a_j$ for all example pairs (i,j) where example i should be learned earlier than example j.

TABLE 6
Representatives of Transfer Teacher. Diff. = different.

Representatives	Teacher model	Teacher pretraining dataset	Difficulty
Transfer learning [123]	Diff. structure with student	ImageNet	Loss
Bootstrapping [33]	Same structure as student	The training dataset	Loss
Cross Review [126]	Same structure as student	N training subset	Loss
Uncertainty [138], [146]	Language model	The training dataset	Cross entropy
Domain score [118], [139]	Language model	General- and in-domain datasets	Cross entropy difference
Noise score [118]	Same NMT models as student	Noisy and clean datasets	Cross entropy difference

assigned with non-zero weights v_i in the t-th epoch. Lin et al. [65] also propose to adjust λ as follows:

$$\lambda_t = \begin{cases} \lambda_0, & t = 0 \\ \lambda_{t-1} + \alpha \cdot \eta_t, & 1 \leq t \leq \tau \\ \lambda_{t-1}, & t > \tau, \end{cases} \tag{15}$$

where η_t is the model performance (e.g., accuracy) in the t-th epoch. When η_t is high, then λ will increase by a bigger step to add more harder examples, and vice versa. Recently, Shu et al. [102] further propose to leverage meta-learning paradigm to optimize λ based on a small and high-quality valid set, which entirely automates the update of λ.

In addition to λ, other hyperparameters, including initialization and stopping criteria, are also very difficult to determine and heavily influencing the SPL performance. What is more, each configuration of hyperparameters could only lead to a single solution, losing view for the entire solution spectrum [61]. To address these issues, Li et al. [26], [61] propose to discard the traditional AOS algorithm and reformulate the SPL problem as a multi-objective issue, which can obtain a set of solutions with different stopping criteria in a single run and improve the robustness of SPL even under bad initialization.

f) Applications of SPL. SPL has been widely applied to many practical problems, including CV tasks of visual category discovery [57], segmentation learning [55], [137], image classification [109], object detection [108], [134], reranking in multimedia retrieval [40], person ReID [143] etc., and traditional machine learning tasks of matrix factorization [141], feature selection [142], cross-modal matching [63], co-training [70], clustering [22], [127], [128], etc. As a primary branch of CL, SPL has the same application motivations as CL, i.e., to guide and to denoise (see Sec. 3.2). Besides, SPL is also effective for a group of applications where the algorithm needs to assign pseudo-labels by models, including reranking [40], co-saliency detection [135], and other weakly [34] or unsupervised learning tasks [22]. Additionally, some works also extend SPL by introducing group-wise weights to improve the performance on multiple data groups, e.g., multi-modal [24], multi-view [127], multi-instance [135], multi-label [58], multi-class [92], multi-task [59], etc. Finally, SPL is also combined with complementary data-selection-based training strategies like boosting [85] and active learning [65], [110] to benefit both schemes. Beyond the scope of SPL, the idea of "deciding learning materials by student" has also inspired self-paced-like designs in broader contexts, e.g., contextual RL [49], knowledge distillation [125], etc.

4.3.2 Transfer Teacher

SPL takes the current student model as an automatic Difficulty Measurer. However, this strategy has a risk of uncertainty at the beginning of training, when the student model is not mature enough (i.e., not sufficiently trained). This is analogous to human education: if a student understands little about the learning materials, it would be hard for him/her to measure the difficulty of the materials and find out the easy ones. Thus, a natural idea is to invite a mature teacher to help the student assess the materials and form an easy-to-hard curriculum. This idea leads to the CL approaches that we denote as Transfer Teacher. As illustrated in Fig 2(c), it is a semi-automatic CL method. Particularly, this method first pretrains a teacher model on the training dataset or an external dataset (e.g., ImageNet), and then transfers its knowledge to calculate the example-wise difficulty, based on which a predefined Training Scheduler can be applied to finish the CL design. Transfer Teacher reduces the burden of artificial Difficulty Measurer designs and thus could be helpful to the tasks where the example-wise easiness is hard to measure.

Some representatives of Transfer Teacher are presented in Table 6. The most general Transfer Teachers are the loss-based methods (the first three rows), which do not need any domain knowledge and are closely related to SPL. Concretely, these methods take the example-wise losses calculated by a teacher model as the example difficulty and assume that the lower the loss, the easier the example. The teacher model can either be different from the student model and have the greater model capacity (i.e., more complex) [123], or share the same structure with the student model [33], [126]. For instance, in [123], a strong teacher classifier pretrained on ImageNet is taken to transfer its knowledge to calculate the example-wise losses on the training dataset. The authors in [33] adopts a bootstrapping strategy, which uses a teacher classifier with the same network structure as the student classifier, and pretrains it on the training dataset. This pretrained teacher can be regarded as a mature version of the student to calculate loss-based difficulty. Note that the difference between bootstrapping and SPL is that the former's Difficulty Measurer is mature and fixed, while the latter's is the current student model which gradually grows up. Another example of loss-based Transfer Teacher is the Cross Review strategy [126], which alleviates the fluctuation of the difficulty measurement. Concretely, the authors uniformly divide the trainset into N shares and train one teacher on each share. Then for each example in the i-th share, they take the other $N-1$ teachers to calculate a loss-based difficulty score.

Moreover, in NLP literature, there exist some typical methods adopting a "teacher" model to measure example-wise difficulty for training data selection, which can be naturally incorporated into CL as Transfer Teacher. For example, some works [138], [146] leverage the following model-based data uncertainty $u^{data}(s) = -\frac{1}{|s|}\sum_{i=1}^{|s|} \log P(s_i|s_{<i})$ to measure sentence-wise difficulty in NMT tasks, where $P(s_i|s_{<i})$

TABLE 7
Representatives of RL Teacher. Acc. = accuracy, thres. = threshold.

Representatives	RL Algorithm	Reward/Student Feedback	Main Goal
AutoCL [29]	Multi-armed bandit	Loss/Complexity-driven learning progress	Efficiency
TSCL [72]	Non-stationary bandit	Absolute value of slope of learning curve	Efficiency
L2T [17]	REINFORCE	How fast the student achieve valid acc. thres.	Efficiency
RL-based CL [53]	Q-Learning	Log-likelihood on valid set	Performance
RCL [140]	Discriministic Actor-Critic	Perplexity difference on valid set	Performance

is the confidence of the pretrained language model (LM) for its prediction about the i-th word in sentence s, and $|s|$ is the length of s. The lower of this uncertainty score, the easier the sentence according to the teacher LM. Besides, Moore et al. [74] propose to use two LMs to measure how much a sentence s is related to a specific domain (e.g., news, talks, patents, etc.) and select domain sentences. This measurement of domain score is leveraged in [118], [139] as Transfer Teacher according to the specific scenarios (e.g., in-domain data can be seen as easier for domain adaption). Moreover, Wang et al. [118] also use two NMT models to measure the noise level of a sentence pair $\{x, y\}$. A lower noise level refers to cleaner and also easier data.

4.3.3 RL Teacher

The SPL and Transfer Teacher only automate the Difficulty Measurer and still use predefined Training Scheduler, and they only consider one side of the "curriculum" or teaching scenario: SPL takes the student feedback (i.e., losses) to adjust the curriculum, while Transfer Teacher leverages the teacher's knowledge to determine the order of presenting learning materials. A common sense in human education is that an ideal teaching strategy should involve both the teacher and the student, where the student could interactively provide feedback to the teacher, and the teacher could then adjust the teaching action accordingly. In this way, both the teacher and student will make progress together.

To this end, RL Teacher methods are proposed, which involve a student model and a reinforcement-learning-based teacher model. At each training epoch, the RL teacher will dynamically select examples/tasks for training according to the student feedback. Concretely, the data selection is taken as the *action* in the RL schemes, and the student feedback is taken as the *state* and *reward*. From the view of the general CL framework in Sec. 4.1, the RL Teacher sets the teacher model as both the Difficulty Measurer and Training Scheduler by dynamically considering the student feedback. The illustration of RL Teacher is shown in Fig 2(d). It is clear to see that, with this teacher-student interactive strategy, RL Teacher achieves the fully-automated CL design.

Some representatives of the RL Teacher are listed in Table 7. Both traditional RL and deep RL models are leveraged in these designs, where the deep RL models are stronger in performance but more time-consuming and harder to train. It is worth mentioning that RL Teacher methods make it possible to set different student feedback according to different goals, e.g., training efficiency or generalization performance, which brings great flexibility and applicability to various scenarios. Additionally, RL Teacher is typically suitable for multi-task learning, where the teacher model selects the most valuable tasks for the student training.

AutoCL [29] and TSCL [72] are two RL Teacher methods designed for multi-task settings, where the goal is to learn a student model that achieves high performance on all the tasks. In both works, bandit-based RL models are adopted as the teacher model, whose job is to receive the reward signals from the student model and select one training task for student learning in the next epoch. Specifically, the RL teachers learn the mapping from history reward sequences $\boldsymbol{r} = \{r_i\}_{i=1}^N$ (of different tasks) to the probability vector $\boldsymbol{\pi}$ of sampling the N training tasks. As both the works aim to design a CL algorithm to improve the training efficiency, various reward measurements are proposed. In AutoCL [29], the authors define a group of *learning progress* as the reward, which includes loss-driven and complexity-driven measurements. The intuition is, if a decrease in some loss or an increase in the student model's complexity is observed after training on the i-th task, then this task is helpful to the student model for making big progress and should be assigned larger sampling probability. On the other hand, in TSCL [72], the authors set the reward as the absolute value of the slope of the learning curve (the absolute difference between the performance scores of two successive epochs) on a specific task. This is an elegant design: when the slope is a large positive value, it means the student is making progress on this task; and when the slope is a large negative value, it implies that the student is forgetting this task. Both conditions should lead to a larger sampling probability on this task to achieve faster and more generalizable student training.

L2T (Learning to Teach) [17] adopts the REINFORCE algorithm as the RL teacher. Given a random mini-batch D_t in the t-th supervised training epoch, the goal of the teacher model is to dynamically determine which data examples are used and which are abandoned. To this end, the action $\boldsymbol{a}_t = \{a_t^{(m)}\}_{m=1}^M \in \{0, 1\}^M$ is a hard selection on each of the M examples in this mini-batch. The state $s_t = (D_t, f_t)$ is defined as the concatenation of various features of the current mini-batch D_t and the current state of student model f_t[7]. This design of state/observation is quite general and applicable to most learning scenarios. Moreover, aiming at fast convergence, the reward r_t is set as a terminal reward (i.e., $r_t = 0, \forall t < T$) to be related with how fast the student model learns. In particular, $r_T = -\log(i_\tau/T')$, where i_τ is the iteration number for the student model achieving an accuracy threshold $\tau \in [0, 1]$ on the valid set, and T' is a predefined maximum iteration number. With all the definition above, L2T trains the teacher model by maximizing the expected reward $J(\theta) = \mathbb{E}_{\phi_\theta(a|s)}[R(s, a)]$, where $R(s, a)$ is a state-action value function to estimate the reward, and ϕ_θ is the data selection policy parameterized by θ, which can be any binary classification model. Through this dynamic

7. For example, data features include the predefined Difficulty Measurer features in Table 2, and model features include iteration number, average historical training loss / validation accuracy, etc.

data selection by the teacher model, the student model is expected to converge faster to a better optima.

Beyond traditional RL algorithms, recent works also leverage deep RL models, e.g., Q-learning [53] and Deterministic Actor-Critic [140], to design RL Teacher methods for automatic data selection, sharing the same spirit with L2T. Both the two works focus on the NMT task, a typical application for CL discussed in Sec. 3.2. RL-based CL [53] first sorts the examples according to a predefined measurement and divide them into M bins of equal sizes, and then defines the action as selecting one bin for NMT training. The reward and state are related to the log-likelihood on the valid set and a prototype batch sampled from all bins, respectively. Moreover, in RCL [140], the state s is similarly defined as L2T, including feature embeddings from data and the student model. Given s, the actor network μ is optimized to select examples from a mini-batch (i.e. action $a = \mu(s)$) to form the training set at each epoch, such that the estimated reward $Q(s, a)$ by critic network Q is maximized. The critic network, on the other hand, is optimized to estimate the reward r more accurately, where r is defined as the performance improvement of the student model on the valid set after trained. Compared with traditional RL methods like REINFORCE, Actor-Critic is supposed to help reduce the update variance and accelerate convergence.

4.3.4 Other Automatic CL

Besides RL Teacher, there exist some other fully-automatic CL designs. Intuitively, these designs should require the generation of the curriculum to rely only on the dataset, the student model, and the goal of the task. According to the CL definition in Sec. 2, we can regard this curriculum as a sequence of training criteria or objectives. Thus, from the optimization perspective, at each training epoch, we hope to optimize the following mapping to improve performance: $\{$data, current state of student model, task goal$\}$ $\mapsto$ training objective. To this end, RL Teacher methods typically adopt an RL framework to learn the policy for training data selection. Additionally, more optimization methods, such as Bayesian Optimization (BO), Stochastic Gradient Descent (SGD), Meta-learning, and Hypernetwork, are also demonstrated to have great potential to learn this mapping. Note that these methods can also be regarded as a "teacher" searching for the best curriculum according to the student state/feedback. Since the methodologies and focuses of optimization are diverse in these works, we conclude them in this subsubsection as "Other Automatic CL" (Table 8).

Tsvetkov et al. [113] make one of the earliest attempts on automatic CL by leveraging BO to learn the best curricula for word representation learning. The curriculum here is determined by the scalar product of a learned weight vector $\boldsymbol{w}$ and an example-wise difficulty feature vector $\boldsymbol{x}$, according to which the examples are scored and sorted for later representation learning. While $\boldsymbol{x}$ is manually engineered, the weight vector $\boldsymbol{w}$ learned by BO provides the possibility for different curriculum according to different downstream tasks. Specifically, BO in this work is a sequential approach to performing a regression from $\boldsymbol{w}$ to the performance on the downstream task. At the t-th iteration, the algorithm first sort the examples by the $\boldsymbol{w}_t \cdot \boldsymbol{x}$, learn word representations V_t (i.e., student model) with this curriculum, and then train extrinsic models on downstream task and evaluate the performance $eval_t$. Finally, $eval_t$ is collected by BO algorithm to generate the $\boldsymbol{w}_{t+1}$. Through this process, BO learns to predict a better $\boldsymbol{w}$ and thus a better curriculum.

While SPL methods in Sec. 4.3.1 optimize the example-wise loss weights $\boldsymbol{v}$ by solving the new objective with manually designed SP-regularizers, existing works have made further effort to optimize $\boldsymbol{v}$ throughout training by different approaches. One idea is to predict the loss weight v_i of example $\{x_i, y_i\}$ by a teacher model, which is adopted in MentorNet [43] and ScreenerNet [48]. The MentorNet h is a teacher model with parameters Θ which maps the example-wise feature $z_i = \phi(x_i, y_i, \boldsymbol{w})$ to the corresponding loss weight v_i. Here, z_i includes the loss, loss difference to the moving average, label, and epoch percentage, and $\boldsymbol{w}$ denotes the parameters of the student model. Given fixed $\boldsymbol{w}$, the MentorNet is trained on a trusted small dataset $\mathcal{D}_{val}$ by SGD:

$$\Theta^* = \arg\min_{\Theta} \sum_{i \in \mathcal{D}_{val}} \mathrm{CE}(h(z_i; \Theta), v_i^*), \tag{16}$$

where v_i^* is manually annotated as 1 iff y_i is a correct label and 0 otherwise, and CE stands for cross-entropy. During the mini-batch training of the student model, the MentorNet is only updated a fixed number of times (with student fixed). Besides the data-driven curriculum learned on $\mathcal{D}_{val}$, we could also train the MentorNet to approximate a predefined curriculum, e.g., by setting v_i^* as the loss weights derived from some SPL objectives. The convergence and robustness of student learning are also theoretically proved.

Apart from teacher model, APL [136] also predicts the loss weights $\boldsymbol{v}$ in SPL by generative adversarial learning. Concretely, under semi-supervised setting, a pace-generator P outputting $\boldsymbol{v}$ is trained to discriminate annotated ($v_i = 1$) and predicted ($v_i = 0$) labels, and a task-predictor T predicting labels is alternatively trained with P to produce high-quality predictions. After the initial training on labeled data, the unlabeled data is then added to the training set with loss weights (or "pace") $\boldsymbol{v}$ given by P in each iteration. This APL paradigm is proven significantly more effective than SPL methods on the task of salient object detection with few labeled data. An analogous idea is adopted in [38] by assigning binary selection on unlabeled data based on pretrained discriminator on labeled data in semi-supervised semantic segmentation task.

Ren et al. [91] further propose a meta-learning [36] perspective for optimizing loss weights $\boldsymbol{v}$. Akin to MentorNet, a clean unbiased valid set is adopted to guide the meta-learning. Specifically, at the t-th epoch, they first locally update the student model (with parameters $\boldsymbol{w}_t$) by one gradient step on a training mini-batch $\mathcal{D}_{train}$, where the example weights v_i are perturbed by ϵ_i:

$$\hat{\boldsymbol{w}}_{t+1}(\epsilon) = \boldsymbol{w}_t - \alpha \nabla \sum_{i \in \mathcal{D}_{train}} \epsilon_i l_i(\boldsymbol{w}_t), \tag{17}$$

where $l_i(\boldsymbol{w})$ is the loss and α is the local learning rate. To estimate the best loss weights $\boldsymbol{v}$ according to the clean valid set, they take a meta-gradient step on a validation mini-batch $\mathcal{D}_{val}$ w.r.t. ϵ, and force the weights to be non-negative:

$$\tilde{v}_{i,t} = \max\left(0, -\eta \frac{\partial}{\partial \epsilon_j} \frac{1}{|\mathcal{D}_{val}|} \sum_{j \in \mathcal{D}_{val}} \hat{\boldsymbol{w}}_{t+1}(\epsilon)\right), \tag{18}$$

where η is the meta learning rate. The $\tilde{\boldsymbol{v}}_t$ is then normalized to obtain the final new weights $\boldsymbol{v}_t$. Finally, they meta up-

TABLE 8
Representatives of "Other Automatic CL" automatic CL methods.

Papers	What to Optimize	How to Optimize
Learning CL with BO [113]	Weights for difficulty dimensions	Bayesian Optimization
MentorNet [43], ScreenerNet [48]	Loss weights	SGD
APL [136]	Loss weights	Adversarial learning
Learning to reweight [91]	Loss weights	Meta-learning
L2T with dynamic loss function [124]	Loss function (as a linear model)	Hypernetwork
Data Parameters [97]	Class/Instance-wise loss function	Data Parameters

date the model parameters to $\boldsymbol{w}_{t+1}$ with the new objective weighted by $\boldsymbol{v}_t$, i.e., $\sum_{i\in\mathcal{D}_{train}} v_{i,t} l_i(\boldsymbol{w}_t)$. This meta-learning mechanism would lead the student model to converge to an appropriate distribution favored by the clean and balanced valid set and thus become more generalizable and robust.

Beyond loss weights, some other works [97], [124] also focus on learning dynamic loss function as a whole, which complies with the most general definition of CL in Sec. 2. As argued in L2T [17], while data selection is analogous to human teacher selecting teaching materials, designing good loss function corresponds to human teacher determining the examination criteria, which is another significant issue in a "curriculum". In [124], the scholars propose to leverage a two-layer perceptron as the teacher hypernetwork μ_Θ to predict the parameters of the loss function $l_\Phi(\hat{y}, y)$. In other words, the loss function is assumed to be itself a neural network with coefficients Φ, and at the t-th epoch, $\Phi_t = \mu_\Theta(s_t)$, where s_t is the state vector of the student model $f_{\boldsymbol{w}}$. Akin to MentorNet, the goal of the teacher model is to maximize the performance of induced student model on a valid set $\mathcal{D}_{val}$: $\Theta^* = \max_\Theta \mathcal{M}(f_{\boldsymbol{w}^*}, \mathcal{D}_{val})$, where $f_{\boldsymbol{w}^*} = \mathcal{F}(\mathcal{D}_{val}, \mu_\Theta)$ stands for the student model trained on the training set with the loss function predicted by μ_Θ, and $\mathcal{M}$ is the performance measurement on $\mathcal{D}_{val}$. Novel algorithms are also proposed to make this optimization of teacher hypernetwork possible.

4.4 How to Choose A Proper CL Method

Although we have reviewed the major ideas of different CL methodologies, how to choose them in real-world applications remains an important problem, which is rarely discussed in existing CL literature and there is no systematic conclusion. In this subsection, we make effort to summarize some empirical evidence and ideas on this topic.

Conclusions from empirical studies. Although such work is scarce, different CL methods are still compared and analyzed in a small number of works. Cirik et al. [12] compare different predefined schedulers on two sequence prediction tasks with LSTM models, showing that predefined CL benefits more when smaller models are applied and the size of the training set is limited. Zhang et al. [138] experiment on the combinations of various predefined Difficulty Measurer and various predefined Training Scheduler on neural machine translation task, reaching the result that predefined CL is highly sensitive to the choices of Difficulty Measurer and hyperparameters (i.e., learning rates). Hacohen et al. [33] compare SPL, anti-curriculum, and different Transfer Teacher methods with various Training Schedulers on image classification, demonstrating Transfer Teacher is the most robust, and the advantage of CL is more effective when the task is difficult.

Within each CL category, several empirical conclusions can be summarized as follows ($>$ means more effective), although most of them are not universal. (i) Predefined CL: for Training Scheduler, the continuous root-p function $>$ discrete Baby Step $>$ discrete One-Pass (see Fig. 5) [12], [82], [86]. (ii) SPL: for SP regularizers without embedded prior, implicit regularizers $>$ soft regularizers (e.g., *mixture*, *logarithmic*) $>$ hard regularizers (see Fig. 6) [16], [40]. (iii) SPL: if reliable prior knowledge or assumption is available, embedding it into the SPL objective always help [22], [41], [42], [134]. (iv) Fully Automatic CL (Sec. 4.3.3, 4.3.4): Many fully automatic CL methods are shown to be significantly more effective than SPL methods on weakly-supervised CV and NLP tasks [43], [91], [97], [120], [136], while MentorNet [43] is often selected as a baseline in these papers.

The best selection among different CL categories needs further empirical studies. However, qualitative comparison of different methodologies is provided in Table 3 and 4. A principle for selecting a proper CL category is to consider how much prior knowledge you know about your dataset and task goal. If sufficient expert domain knowledge is available, then predefined CL methods are more preferable to design a *knowledge-driven* curriculum specifically suitable to the exact scenario. On the other hand, if we have no prior assumptions on the data, then automatic CL methods are more preferable to learn a *data-driven* curriculum adaptive to the underlying dataset and task goal.

Hybrid CL. A further consideration of designing a CL framework is to adopt different CL methods jointly, making them complement each other. Generally, this hybrid CL can be designed by applying different CL methods on different evidence for curriculum or different levels of data. A typical example is the SPCL-like methods [42], [132], [134], [137] in Sec. 4.3.1 that embed the predefined sample-importance-order prior into the SPL objectives or SPL-like regimes, taking the advantages of both knowledge-driven predefined CL and data-driven SPL to enrich the curriculum from both sources of evidence, i.e., human and machine. Following this paradigm, an interesting idea for future researchers might be to embed human prior on sample importance into the fully data-driven CL methods in Sec. 4.3.3 and 4.3.4, which is being explored by frontier researchers [120]. On the other hand, we can also apply different CL to different levels of training data. For example, LFME [125] jointly adopts an SPL-like mechanism for expert selection (each expert is trained on a subset of training data) in knowledge distillation and a Transfer Teacher for instance selection in each subset.

Extra computational cost of CL. It is worth mentioning another concern of great practical significance: though seemingly effective and easy-to-use, how much does it cost to apply these CL methods, i.e., the extra computational cost

to the training? Before the analysis on the time complexity of CL, we remind readers that convergence speedup is one of the main advantages and motivations of CL, and many CL methods in different categories (e.g., [17], [41], [86]) can actually accelerate training. By reducing the number of iterations to convergence, the total cost of training is reduced despite additional computations for CL.

As additional computational complexity of CL is hardly discussed in the literature, we generally analyze it according to the taxonomy in Sec. 4. We assume there are n training examples to train M iterations. (i) Predefined CL methods in Sec. 4.2 calculate and then fix the curriculum before the training process starts. It often costs $O(n)$ (or $O(1)$ if human annotation is available) to calculate the difficulty of each sample and $O(n \log n)$ to sort the samples from easy to hard. During training, the scheduler calculates the difficulty threshold for batch sampling at each iteration, which costs $O(1)$ (for discrete schedulers) or $O(M)$ (for continuous schedulers, see Sec. 4.2.2). Thus, the overall complexity is $O(n \log n + M)$, which is the cheapest among all CL methods. (ii) SPL methods in Sec. 4.3.1 dynamically updates the sample weights $\boldsymbol{v} = \{v_i\}_1^n$ at each iteration, and thus the extra complexity is $O(Mn)$ or $O(Mnx)$ if close-formed solutions of $\boldsymbol{v}^*$ exists or not, where x is the computations of CVX toolbox for convex optimization on v_i. (iii) Transfer Teacher methods in Sec. 4.3.2 pretrain a teacher difficulty measurer before training, then it calculates a curriculum like predefined CL. So the overall complexity is $O(T + n \log n + M)$, where T is the cost of pretraining the teacher. (iv) RL Teacher methods in Sec. 4.3.3 dynamically learn the data weighting policy of the teacher and learn the student model at each iteration. The overall complexity is $O(RM + xMn)$, if R is the computations for one updating step of teacher, and x for predicting the weight for one example. R can be both small (bandit) and large (Deep RL).

In summary, from the theoretical perspective of time complexity, most CL methods in (i) to (iv) induce little or acceptable additional cost w.r.t. the cost of main training and are thus worth adoption according to their advantages. We have to admit that CL can also be expensive, e.g., Deep RL Teacher [53], [140]. Generating a task curriculum in RL setting often costs greater time than learning the tasks [76]. However, it is a trade-off between performance and efficiency, e.g., RL agents may fail to solve the target tasks without the expensive curriculum [20].

5 Discussions

5.1 Easier First v.s. Harder First

A fundamental question for the CL strategy (in Definition 1) is: does this "easy to hard" training strategy always help, given all of these works and theories? In some literature of CL, the answer to this question is "No". For example, Avramova [3] finds that convolutional neural networks derive most learning values from the hardest examples, and the damage of excluding those easiest examples is minor. Zhang et al. [138] also test a reverse version of CL (i.e., a copy of baseline CL reversing the difficulty ranking to "hard to easy", also called *anti-curriculum*), on NMT tasks, which shows that in some cases, anti-curriculum may even achieve the best performance among various Training Scheduler designs. Besides, Hacohen et al. [33] demonstrate that SPL will hurt the performance and significantly delay learning in their experiments. Other works [118], [144] also design "harder examples first" curricula.

Besides CL literature, hard example mining (HEM) [101] serves as another well-studied and popular data selection strategy, which is opposite to CL. Concretely, in each training batch, HEM selects the hardest examples for training (or assign them with higher weights), assuming that the harder examples are more informative. The difficulty in HEM is often defined according to the current model losses on examples [69], [101] or the gradient magnitude [1], [28]. Akin to CL, HEM also has various applications, and the famous boosting algorithm [21] in ensemble learning also takes the same strategy by upweighting the wrongly-classified examples.

So which strategy should we apply to our own scenario, "easier first" as CL or "harder first" as HEM? It remains an unsolved problem to be carefully considered. Theoretically, under different settings, both CL and HEM strategies can benefit the learning as long as the "curriculum" is positively correlated with the optimal utility[8] [33]. However, this criterion is very hard to verify. More intuitively, Chang et al. [8] point out that CL is more suitable for the scenarios with more noisy labels or outliers to improve the model robustness and convergence rate, while HEM is more beneficial for cleaner datasets and leads to faster and more stable SGD. One should also note that if the target task is very difficult, CL will be more preferable to HEM, since CL is able to result in a more effective training process through the easier/smoother versions.

An alternative is to combine the two strategies together with a trade-off policy. For example, Pi et al. [85] embed the self-paced regularizers into the objective of boosting algorithm, which simultaneously enhances the learning effectiveness (by boosting) and robustness (by SPL). Besides, Chang et al. [8] propose to select the most uncertain examples according to the prediction history, which is consistent with the variance reduction strategies in active learning [100]. The uncertain examples are predicted both incorrectly and correctly in history and are thus neither too easy (always correct) nor too difficult (always incorrect). It is worth mentioning that the fully automatic CL methods (e.g., RL Teacher in Sec. 4.3.3) would also be an ideal choice when it is hard to choose between "easier first" CL and "harder first" HEM.

From a higher perspective, both the original CL (Definition 1) and HEM belong to the instance selection or example reweighting, defined as data-level generalized CL (Definition 2) in Sec. 2. As argued in [91], one crucial advantage of reweighting examples is robustness against training set biases. The biases include class imbalance and label noise, both of which have been studied as typical problems of machine learning with various practical methods (e.g., [9], [47] for the former and [23], [62], [78], [90] for the latter). By reweighting examples, HEM prioritizes higher-loss examples which more likely belong to minority classes, and thus alleviates class imbalance bias. On the other hand,

8. The optimal utility is $\sum_{i \in \mathcal{D}} e^{-l_i(\theta^*)}$, where $\mathcal{D}$ is the training dataset, $l_i(\theta^*)$ is the loss on the i-th example calculated by the optimal model θ^*.

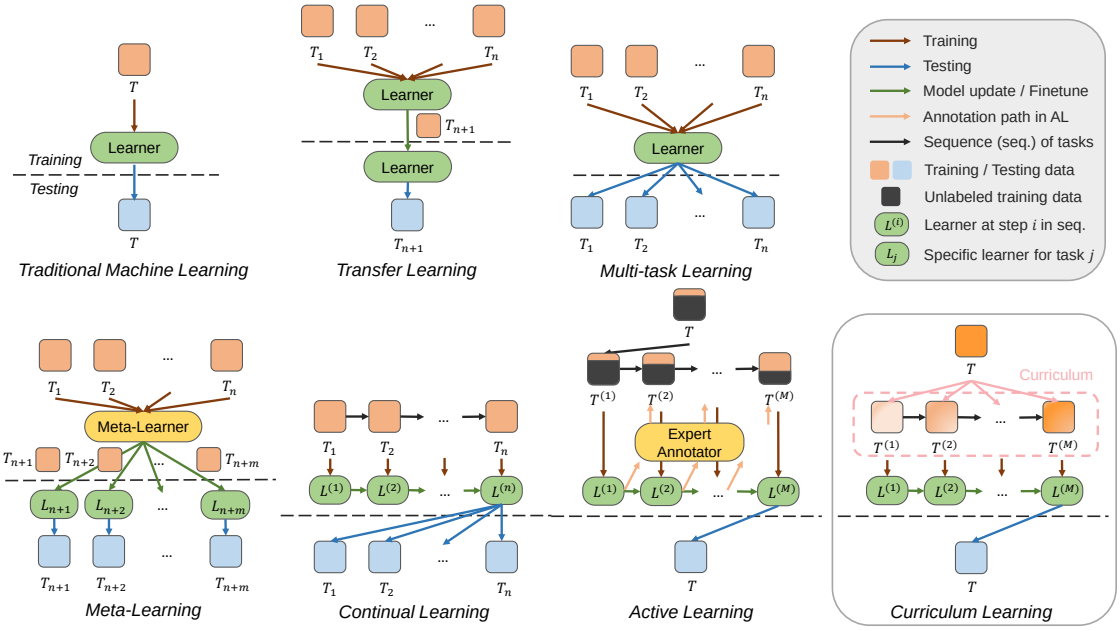

Fig. 7. Illustration of different machine learning paradigms from the perspective of data distribution. Different paradigms aim to solve different distribution discrepancies among training and testing data, while we see similar mechanisms among some of them, which help us understand their connections and may potentially inspire new methodologies. For curriculum learning, we illustrate Definition 2, and the curriculum can be both predefined and automatically learned. Note that T_j stands for different tasks, while $T^{(i)}$ is the modified distribution at the i-th step in training.

CL favors lower-loss examples which are more likely to be clean data, and thus reduces the label noise bias. When assumptions on the training set biases are uncertain, many fully automatic CL methods are designed to reweight the examples to achieve a certain goal of learning, e.g., training efficiency [29], [72], valid set accuracy [53], [91], [120], etc.

5.2 Relationship between CL and Other Concepts

From the perspective of data distribution, different machine learning paradigms focus on different settings on data distribution discrepancy, which is illustrated in Fig. 7. For example, transfer learning [81] aims at alleviating the discrepancy between source tasks $\{T_i\}_{i=1}^{n}$ and target task by transferring through model parameters of the learner. Meta-learning [36] mitigates the discrepancy between multiple source tasks $\{T_i\}_{i=1}^{n}$ and target tasks $\{T_i\}_{i=n+1}^{n+m}$ by learning common meta-knowledge on learning algorithms across tasks. Continual learning [13] eases the discrepancy among an online sequence of tasks by updating one learner to defy forgetting. From this view, Data-level Generalized Curriculum Learning (Definition 2) smooths the discrepancy between the testing distribution and training distribution by a sequence of reweighting, which results in a gradual optimization process towards the target.

With Fig. 7, we can see the differences and connections between CL and other concepts, which may inspire new ideas. (i) **CL v.s. Transfer Learning (TL):** as pointed out by Bengio et al. [6], CL can be seen as a special form of TL where the initial tasks are used to guide the learner so that it will perform better on the final task. Thus, CL is naturally suitable for TL settings like domain adaption [103], [139]. The green arrows also show that CL is a sequence of TL throughout the curriculum. (ii) **CL v.s. Multi-task Learning (MTL):** we can regard the T in CL as a distribution of tasks and the n tasks in MTL are sampled from this distribution. CL then provides a sequence of task distributions to guide MTL, which is empirically proven helpful [29], [59], [83], [96]. (iii) **CL v.s. Meta-Learning (ML):** although ML and CL seem quite different in Fig. 7, we argue that ML is highly related to automatic CL (AutoCL). In fact, the teaching policy (i.e., curriculum) in AutoCL can be regarded as the meta-knowledge in ML to optimize the student's progress [36], from which view AutoCL is a specific form of CL. In essence, ML is about learning to learn and AutoCL is about learning to teach [17], i.e., they both aim to optimize the hyperparameters of algorithms from different views of students and teachers. Therefore, it is no wonder that ML is shown effective for AutoCL designs [91], [102], [120], and shall inspire more AutoCL ideas. We also advocate the integration of ML and AutoCL to enable fully automatic machine learning and teaching. (iv) **CL v.s. Continual Learning (ContL):** although both of them involve a sequence of tasks, the settings are quite different. Specifically, with a different distribution, the tasks $\{T_i\}_{i=1}^{n}$ in ContL are predefined and fixed. While in CL, derived from the same distribution T, the distributions $\{T^{(i)}\}_{i=1}^{M}$ in M steps can be flexibly adjusted by the curriculum. However, we argue that within each task in ContL, CL methods may help to improve robustness and defy forgetting by the transfer between preceding tasks and the current task. (v) **CL v.s. Active Learning (AL)**: AL [100] is the most analogous paradigm to CL in Fig. 7, both of which involves dynamic data selection. In AL, an active learner achieves great performance with fewer labeled data via generating queries to ask an expert to annotate several unlabeled instances for further training. The goals of CL and AL are different: the former improves performance and accelerates convergence in supervised, weakly-supervised, and unsupervised settings, while the latter is designed for

label-saving training in the semi-supervised setting. However, the criteria for data selection can somehow be shared among CL and AL, and recent works [65], [110] have made efforts to combine SPL with AL to utilize the complementariness between the criteria.

6 Future Directions of CL

We conclude this paper with some ongoing or future directions of CL, which are worthy of discussion:

Evaluation benchmarks. Although various CL methods have been proposed and demonstrated effective, few works have made efforts on evaluating them with general benchmarks. In existing literature, the datasets and metrics are diverse in different applications. For instance, the CIFAR datasets with different label corruption settings are widely used to evaluate CL methods on image classification with accuracy metric [43], [97], [120] , and the WMT datasets are widely chosen to evaluate CL methods for neural machine translation with BLEU metric [53], [68], [86]. However, it is challenging to design a unified dataset with unified metrics to evaluate and compare the CL algorithms. Such a benchmark may incorporate datasets for different applications (e.g., CV, NLP, recommendation, etc.) with different noise levels (e.g., clean, weakly-supervised, etc.). Accordingly, evaluation metrics on the relative performance boost, convergence speedup, additional computational cost, etc., should also be carefully designed. The challenges are threefold: (i) Dataset construction: the data of different applications have different levels of sparsity, heterogeneity, noisiness, etc. (ii) Metric design: different applications naturally need different metrics, and their urgency of requirements for convergence speed is also different. (iii) Ground-truth curriculum: most CL literature does not provide an oracle curriculum to evaluate whether the algorithm-based curriculum is reasonable. Therefore, it would be interesting to design such an ideal curriculum in the benchmark to compare CL methods more intuitively.

More advanced theories. Existing theoretical analyses in Sec. 3.1 provide different angles for understanding CL. Nevertheless, more theories are still required to help us reveal why typical CL (Definition 2 in Sec. 2) is effective. For example, if the dataset has no noise, are there any bounds for the effectiveness of CL? What is the actual effect of each condition in Definition 2, i.e., increasing dataset size/variance and increasing difficulty? Besides, the fully automatic CL methods in Sec. 4.3.3 and 4.3.4 also need more theoretical guarantees on their effectiveness. Moreover, a remaining fundamental question is to theoretically reveal the relations between the data distribution, task objective, and the best training strategy among "easier first" (CL), "harder first" (HEM), and other strategies. Theories on this topic shall provide the basis for the application of CL in a specific task.

More CL algorithms and various applications. Automatic CL (Sec. 4.3) provides the potential application values for CL in wider research areas and has become a cutting-edge direction. Therefore, one promising direction is to design more automatic CL methodologies with different optimizations (e.g., bandit algorithms, meta-learning, hyper-parameter optimization, etc.) and different objectives (e.g., data selection/reweighting, finding the best loss function or hypothesis space, etc.). Moreover, as shown in [65], [85], [110], CL methods can be incorporated with other strategies like boosting and AL to achieve improvement. In addition to methodologies, more efforts should be made to explore the power of CL in more various applications, including both cutting-edge research areas (e.g., meta-learning, continual learning, NAS, graph neural network, self-supervised learning, etc.) and traditional machine learning topics (e.g., clustering, regression, etc.). Although the directions mentioned above may adopt Definition 3 of CL as a sequence of training criteria in Sec. 2, the spirit of imitating the human curriculum shall drive more breakthroughs in the machine learning community.

References

[1] G. Alain, *et al.* Variance reduction in sgd by distributed importance sampling. *arXiv preprint*, 2015.
[2] E. L Allgower, *et al. Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012.
[3] V. Avramova. Curriculum learning with deep convolutional neural networks, 2015.
[4] S. Bengio, *et al.* Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 1171–1179, 2015.
[5] Y. Bengio. Evolving culture versus local minima. In *Growing Adaptive Machines*, 109–138. Springer, 2014.
[6] Y. Bengio, *et al.* Curriculum learning. In *ICML*, 41–48, 2009.
[7] S. Braun, *et al.* A curriculum learning method for improved noise robustness in automatic speech recognition. In *EUSIPCO*, 548–552. IEEE, 2017.
[8] H. Chang, *et al.* Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NeurIPS*, 1002–1012, 2017.
[9] N. Chawla, *et al.* Smote: synthetic minority over-sampling technique. *JAIR*, 16:321–357, 2002.
[10] X. Chen, *et al.* Webly supervised learning of convolutional networks. In *ICCV*, 1431–1439, 2015.
[11] J. Choi, *et al.* Pseudo-labeling curriculum for unsupervised domain adaptation. *arXiv preprint*, 2019.
[12] V. Cirik, *et al.* Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint*, 2016.
[13] M. Delange, *et al.* A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, 2021.
[14] R. El-Bouri, *et al.* Student-teacher curriculum learning via reinforcement learning: Predicting hospital inpatient admission location. *arXiv preprint*, 2020.
[15] J. L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.
[16] Y. Fan, *et al.* Self-paced learning: an implicit regularization perspective. *arXiv preprint*, 2016.
[17] Y. Fan, *et al.* Learning to teach. *ICLR*, 2018.
[18] N. Ferro, *et al.* Continuation methods and curriculum learning for learning to rank. In *CIKM*, 1523–1526, 2018.
[19] C. Florensa, *et al.* Automatic goal generation for reinforcement learning agents. In *ICML*, 1515–1528, 2018.
[20] C. Florensa, *et al.* Reverse curriculum generation for reinforcement learning. In *CoRL*, 2017.
[21] Y. Freund, *et al.* Experiments with a new boosting algorithm. In *ICML*, volume 96, 148–156. Citeseer, 1996.
[22] K. Ghasedi, *et al.* Balanced self-paced learning for generative adversarial clustering network. In *CVPR*, 4391–4400, 2019.
[23] J. Goldberger, *et al.* Training deep neural-networks using a noise adaptation layer. 2016.
[24] C. Gong, *et al.* Multi-modal curriculum learning for semi-supervised image classification. *TIP*, 25(7):3249–3260, 2016.
[25] C. Gong, *et al.* Multi-modal curriculum learning over graphs. *TIST*, 10:1 – 25, 2019.
[26] M. Gong, *et al.* Decomposition-based evolutionary multiobjective optimization to self-paced learning. *TEVC*, 23(2):288–302, 2018.
[27] T. Gong, *et al.* Why curriculum learning & self-paced learning work in big/noisy data: A theoretical perspective. *Big Data & Information Analytics*, 1(1):111, 2016.

[28] S. Gopal. Adaptive sampling for sgd by exploiting side information. In *ICML*, 364–372, 2016.

[29] A. Graves, *et al.* Automated curriculum learning for neural networks. *ICML*, 2017.

[30] L. Gui, *et al.* Curriculum learning for facial expression recognition. In *FG 2017*, 505–511. IEEE, 2017.

[31] S. Guo, *et al.* Curriculumnet: Weakly supervised learning from large-scale web images. In *ECCV*, 135–150, 2018.

[32] Y. Guo, *et al.* Breaking the curse of space explosion: Towards efficient nas with curriculum search. In *ICML*, 2020.

[33] G. Hacohen, *et al.* On the power of curriculum learning in training deep networks. *ICML*, 2019.

[34] J. Han, *et al.* Weakly-supervised learning of category-specific 3d object shapes. *TPAMI*, 2019.

[35] L. Han, *et al.* Self-paced mixture of regressions. In *IJCAI*, 2017.

[36] T. Hospedales, *et al.* Meta-learning in neural networks: A survey. *arXiv preprint*, 2020.

[37] Y. Huang, *et al.* Self-attention enhanced cnns and collaborative curriculum learning for distantly supervised relation extraction. In *EMNLP-IJCNLP*, 389–398, 2019.

[38] W. Hung, *et al.* Adversarial learning for semi-supervised semantic segmentation. In *BMVC*, 2018.

[39] A. Jesson, *et al.* Cased: curriculum adaptive sampling for extreme data imbalance. In *MICCAI*, 639–646. Springer, 2017.

[40] L Jiang, *et al.* Easy samples first: Self-paced reranking for zero-example multimedia search. In *MM*, 547–556, 2014.

[41] L Jiang, *et al.* Self-paced learning with diversity. In *NeurIPS*, 2078–2086, 2014.

[42] L Jiang, *et al.* Self-paced curriculum learning. In *AAAI*, volume 2, page 6, 2015.

[43] L Jiang, *et al.* Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2304–2313, 2018.

[44] A. Jiménez-Sánchez, *et al.* Medical-based deep curriculum learning for improved fracture classification. In *MICCAI*, 694–702. Springer, 2019.

[45] S. Jin, *et al.* Unsupervised hard example mining from videos for improved object detection. In *ECCV*, 307–324, 2018.

[46] T. Karras, *et al.* Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2017.

[47] S. Khan, *et al.* Cost-sensitive learning of deep feature representations from imbalanced data. *TNNLS*, 29(8):3573–3587, 2017.

[48] T. Kim, *et al.* Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint*, 2018.

[49] P. Klink, *et al.* Self-paced contextual reinforcement learning. In *CoRL*, 2019.

[50] T. Kocmi, *et al.* Curriculum learning and minibatch bucketing in neural machine translation. In *RANLP*, 2017.

[51] D. Kong, *et al.* Exclusive feature learning on arbitrary structures via l_1,2-norm. In *NeurIPS*, 1655–1663, 2014.

[52] K. A Krueger, *et al.* Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

[53] G. Kumar, *et al.* Reinforcement learning based curriculum optimization for neural machine translation. In *NAACL-HLT*, 2019.

[54] M. Kumar, *et al.* Self-paced learning for latent variable models. In *NeurIPS*, 1189–1197, 2010.

[55] M. Kumar, *et al.* Learning specific-class segmentation from diverse data. In *ICCV*, 1800–1807. IEEE, 2011.

[56] K. Lange, *et al.* Optimization transfer using surrogate objective functions. *JCGS*, 9(1):1–20, 2000.

[57] Y. Lee, *et al.* Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 1721–1728. IEEE, 2011.

[58] C. Li, *et al.* A self-paced regularization framework for multilabel learning. *TNNLS*, 29(6):2660–2666, 2017.

[59] C. Li, *et al.* Self-paced multi-task learning. In *AAAI*, 2016.

[60] H. Li, *et al.* Self-paced convolutional neural networks. In *IJCAI*, 2110–2116, 2017.

[61] H. Li, *et al.* Multi-objective self-paced learning. In *AAAI*, 1802–1808, 2016.

[62] Y. Li, *et al.* Learning from noisy labels with distillation. In *ICCV*, 1910–1918, 2017.

[63] J. Liang, *et al.* Self-paced cross-modal subspace matching. In *SIGIR* , 569–578, 2016.

[64] J. Liang, *et al.* Learning to detect concepts from webly-labeled video data. In *IJCAI*, 1746–1752, 2016.

[65] L. Lin, *et al.* Active self-paced learning for cost-effective and progressive face identification. *TPAMI*, 40(1):7–19, 2017.

[66] C. Liu, *et al.* Curriculum learning for natural answer generation. In *IJCAI*, 2018.

[67] S. Liu, *et al.* Understanding self-paced learning under concave conjugacy theory. *arXiv preprint*, 2018.

[68] X. Liu, *et al.* Norm-based curriculum learning for neural machine translation. *ACL*, 2020.

[69] I. Loshchilov, *et al.* Online batch selection for faster training of neural networks. *arXiv preprint*, 2015.

[70] F. Ma, *et al.* Self-paced co-training. In *ICML*, 2017.

[71] Z. Ma, *et al.* On convergence properties of implicit self-paced objective. *Information Sciences*, 462:132–140, 2018.

[72] T. Matiisen, *et al.* Teacher-student curriculum learning. *TNNLS*, 2019.

[73] D. Meng, *et al.* A theoretical understanding of self-paced learning. *Information Sciences*, 414:319–328, 2017.

[74] R. Moore, *et al.* Intelligent selection of language model training data. In *ACL*, 2010.

[75] P. Morerio, *et al.* Curriculum dropout. In *ICCV*, 3544–3552, 2017.

[76] S. Narvekar, *et al.* Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint*, 2020.

[77] S. Narvekar, *et al.* Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, 2536–2542, 2017.

[78] N. Natarajan, *et al.* Learning with noisy labels. In *NIPS*, volume 26, 1196–1204, 2013.

[79] E. Newport. Maturational constraints on language learning. *Cognitive science*, 14(1):11–28, 1990.

[80] J. Olvera-López, *et al.* A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, 2010.

[81] S. Pan, *et al.* A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2009.

[82] G. Penha, *et al.* Curriculum learning strategies for ir: An empirical study on conversation response ranking. *arXiv preprint*, 2019.

[83] A. Pentina, *et al.* Curriculum learning of multiple tasks. In *CVPR*, 5492–5500, 2015.

[84] G. Peterson. A day of great illumination: Bf skinner's discovery of shaping. *JEAB*, 82(3):317–328, 2004.

[85] T. Pi, *et al.* Self-paced boost learning for classification. In *IJCAI*, 1932–1938, 2016.

[86] E. Platanios, *et al.* Competence-based curriculum learning for neural machine translation. In *NAACL-HLT*, 2019.

[87] R. Portelas, *et al.* Automatic curriculum learning for deep rl: A short survey. *arXiv preprint*, 2020.

[88] M. Qu, *et al.* Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In *WSDM*, 468–476, 2018.

[89] S. Ranjan, *et al.* Curriculum learning based approaches for noise robust speaker recognition. *TASLP*, 26(1):197–210, 2017.

[90] S. Reed, *et al.* Training deep neural networks on noisy labels with bootstrapping. In *ICLR*, 2014.

[91] M. Ren, *et al.* Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[92] Y. Ren, *et al.* Robust softmax regression for multi-class classification with self-paced learning. In *IJCAI*, 2641–2647, 2017.

[93] Z. Ren, *et al.* Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *TNNLS*, 29(6):2216–2226, 2018.

[94] D. Rohde, *et al.* Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109, 1999.

[95] T. Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE TRA*, 10(3):323–333, 1994.

[96] N. Sarafianos, *et al.* Curriculum learning for multi-task classification of visual attributes. In *ICCVW*, 2608–2615, 2017.

[97] S. Saxena, *et al.* Data parameters: A new family of parameters for learning a differentiable curriculum. In *NeurIPS*, 11095–11105, 2019.

[98] J. Schmidhuber. Curious model-building control systems. In *IJCNN*, 1458–1463, 1991.

[99] O. Selfridge, *et al.* Training and tracking in robotics. In *IJCAI*, 670–672, 1985.

[100] B. Settles. Active learning literature survey. Technical report, UW-Madison Department of CS, 2009.

[101] A. Shrivastava, *et al.* Training region-based object detectors with online hard example mining. In *CVPR*, 761–769, 2016.

[102] J. Shu, *et al.* Meta self-paced learning. *SCIENTIA SINICA Informationis*, 50(6):781–793, 2020.
[103] Y. Shu, *et al.* Transferable curriculum for weakly-supervised domain adaptation. In *AAAI*, volume 33, 4951–4958, 2019.
[104] S. Sinha, *et al.* Curriculum by smoothing. In *NeurIPS*, 2020.
[105] B. Skinner. Reinforcement today. *American Psychologist*, 13(3):94, 1958.
[106] P. Soviany, *et al.* Image difficulty curriculum for generative adversarial networks (cugan). In *WCACV*, 3463–3472, 2020.
[107] V. Spitkovsky, *et al.* From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *NAACL-HLT*, 751–759, 2010.
[108] K. Tang, *et al.* Shifting weights: Adapting object detectors from image to video. In *NeurIPS*, 638–646, 2012.
[109] Y. Tang, *et al.* Self-paced dictionary learning for image classification. In *MM*, 833–836, 2012.
[110] Y. Tang, *et al.* Self-paced active learning: Query the right thing at the right time. In *AAAI*, volume 33, 5117–5124, 2019.
[111] Y. Tang, *et al.* Attention-guided curriculum learning for weakly supervised classification and localization of thoracic diseases on chest radiographs. In *MLMI*, 249–258. Springer, 2018.
[112] Y. Tay, *et al.* Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *ACL*, 2019.
[113] Y. Tsvetkov, *et al.* Learning the curriculum with bayesian optimization for task-specific word representation learning. In *ACL*, 2016.
[114] R. Tudor Ionescu, *et al.* How hard can it be? estimating the difficulty of visual search in an image. In *CVPR*, 2157–2166, 2016.
[115] J. Tullis, *et al.* On the effectiveness of self-paced learning. *JML*, 64(2):109–118, 2011.
[116] G. Turkewitz, *et al.* Limitations on input as a basis for neural organization and perceptual development: A preliminary theoretical statement. *ISDP*, 15(4):357–368, 1982.
[117] C. Wang, *et al.* Curriculum pre-training for end-to-end speech translation. In *ACL*, 2020.
[118] W. Wang, *et al.* Dynamically composing domain-data selection with clean-data selection by" co-curricular learning" for neural machine translation. In *ACL*, 2019.
[119] W. Wang, *et al.* Learning a multi-domain curriculum for neural machine translation. In *ACL*, 7711–7723, 2020.
[120] X. Wang, *et al.* Optimizing data usage via differentiable rewards. In *ICML*, 9983–9995. PMLR, 2020.
[121] Y. Wang, *et al.* Dynamic curriculum learning for imbalanced data classification. In *ICCV*, 5017–5026, 2019.
[122] Y. Wei, *et al.* Stc: A simple to complex framework for weakly-supervised semantic segmentation. *TPAMI*, 39(11):2314–2320, 2016.
[123] D. Weinshall, *et al.* Curriculum learning by transfer learning: Theory and experiments with deep networks. In *ICML*, 2018.
[124] L. Wu, *et al.* Learning to teach with dynamic loss functions. In *NeurIPS*, 6466–6477, 2018.
[125] L. Xiang, *et al.* Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *ECCV*, 2020.
[126] B. Xu, *et al.* Curriculum learning for natural language understanding. In *ACL*, 6095–6104, 2020.
[127] C. Xu, *et al.* Multi-view self-paced learning for clustering. In *IJCAI*, 2015.
[128] H. Yu, *et al.* Self-paced learning for k-means clustering algorithm. *PRL*, 132:69–75, 2020.
[129] M. Yuan, *et al.* Model selection and estimation in regression with grouped variables. *JRSS: Series B*, 68(1):49–67, 2006.
[130] X. Yuan, *et al.* Adversarial examples: Attacks and defenses for deep learning. *TNNLS*, 30(9):2805–2824, 2019.
[131] W. Zaremba, *et al.* Learning to execute. *arXiv preprint*, 2014.
[132] D. Zhang, *et al.* Learning object detectors with semi-annotated weak labels. *IEEE TCSVT*, 29:3622–3635, 2019.
[133] D. Zhang, *et al.* Synthesizing supervision for learning deep saliency network without human annotation. *TPAMI*, 42:1755–1769, 2020.
[134] D. Zhang, *et al.* Leveraging prior-knowledge for weakly supervised object detection under a collaborative self-paced curriculum learning framework. *IJCV*, 127(4):363–380, 2019.
[135] D. Zhang, *et al.* A self-paced multiple-instance learning framework for co-saliency detection. In *ICCV*, 594–602, 2015.
[136] D. Zhang, *et al.* Few-cost salient object detection with adversarial-paced learning. In *NeurIPS*, 2020.
[137] D. Zhang, *et al.* Spftn: A self-paced fine-tuning network for segmenting objects in weakly labelled videos. In *CVPR*, 4429–4437, 2017.
[138] X. Zhang, *et al.* An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint*, 2018.
[139] X. Zhang, *et al.* Curriculum learning for domain adaptation in neural machine translation. *arXiv preprint*, 2019.
[140] M. Zhao, *et al.* Reinforced curriculum learning on pre-trained neural machine translation models. In *AAAI*, 2020.
[141] Q. Zhao, *et al.* Self-paced learning for matrix factorization. In *AAAI*, volume 3, page 4, 2015.
[142] W. Zheng, *et al.* Unsupervised feature selection by self-paced learning regularization. *PRL*, 132:4–11, 2020.
[143] S. Zhou, *et al.* Deep self-paced learning for person re-identification. *Pattern Recognition*, 76:739–751, 2018.
[144] T. Zhou, *et al.* Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*, 2018.
[145] T. Zhou, *et al.* Curriculum learning by dynamic instance hardness. In *NeurIPS*, 2020.
[146] Y. Zhou, *et al.* Uncertainty-aware curriculum learning for neural machine translation. In *ACL*, 6934–6944, 2020.
[147] Z. Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

Xin Wang is currently an Assistant Professor at the Department of Computer Science and Technology, Tsinghua University. He got both of his Ph.D. and B.E degrees in Computer Science and Technology from Zhejiang University, China. He also holds a Ph.D. degree in Computing Science from Simon Fraser University, Canada. His research interests include relational media big data analysis, multimedia intelligence and recommendation in social media. He has published several high-quality research papers in top conferences including ICML, KDD, WWW, SIGIR ACM Multimedia etc. He is the recipient of 2017 China Postdoctoral innovative talents supporting program. He receives the ACM China Rising Star Award in 2020.

Yudong Chen is a graduate student at the Department of Computer Science and Technology, Tsinghua University. His research interests include machine learning, data mining, and multimedia analysis.

Wenwu Zhu is currently a Professor and the Vice Chair of the Department of Computer Science and Technology at Tsinghua University. His research interests are in the area of data-driven multimedia networking and Cross-media big data computing. He has published over 350 referred papers and is the inventor or co-inventor of over 50 patents. He received eight Best Paper Awards, including ACM Multimedia 2012 and IEEE Transactions on Circuits and Systems for Video Technology in 2001 and 2019.

He served as EiC for IEEE Transactions on Multimedia from 2017-2019. He served in the steering committee for IEEE Transactions on Multimedia (2015-2016) and IEEE Transactions on Mobile Computing (2007-2010), respectively.He is an AAAS Fellow, IEEE Fellow, SPIE Fellow, and a member of The Academy of Europe (Academia Europaea).