Curriculum Learning:
Theories, Approaches, Applications, Tools, and Future Directions in the Era of LLM

Xin Wang, Yuwei Zhou, Hong Chen, Wenwu Zhu

Media and Network Lab, Tsinghua University

Presenters

Xin Wang
Tsinghua University

Yuwei Zhou
Tsinghua University

Hong Chen
Tsinghua University

Wenwu Zhu
Tsinghua University

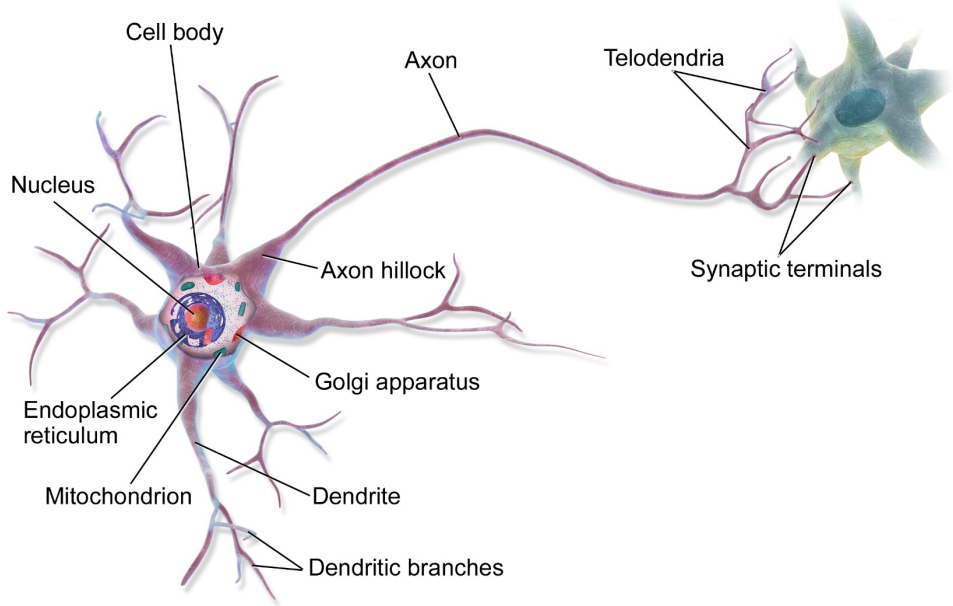
Outline

01	**Introduction**	Xin Wang
02	**Theories**	Xin Wang
03	**Approaches**	Yuwei Zhou
04	**Applications**	Hong Chen
05	**Tools**	Hong Chen
06	**Future Directions**	Xin Wang

Introduction

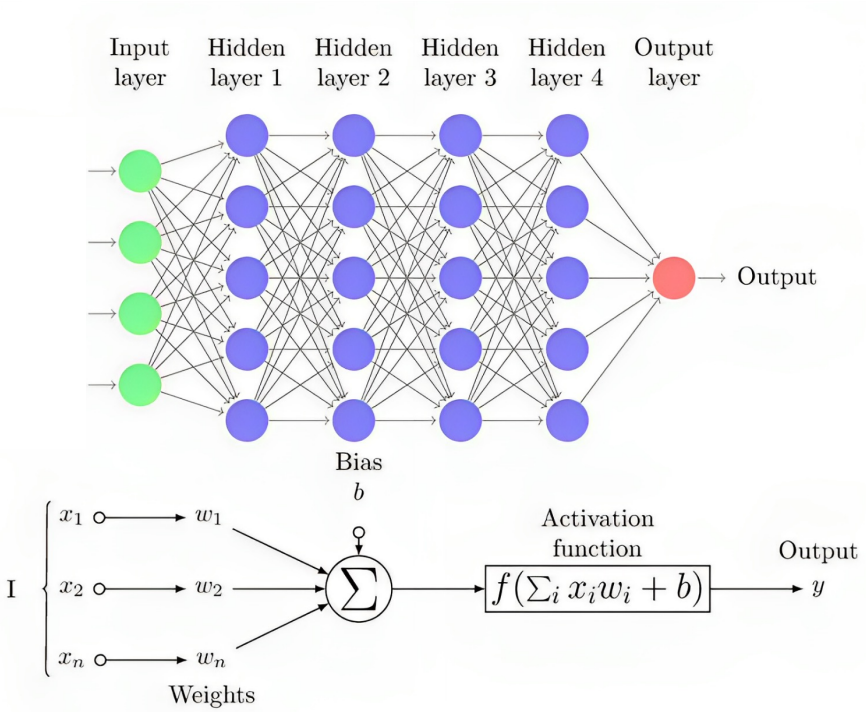
Human Learning and Machine Learning

- Human Learning has inspired various machine learning algorithms.



Neuron

Inspire



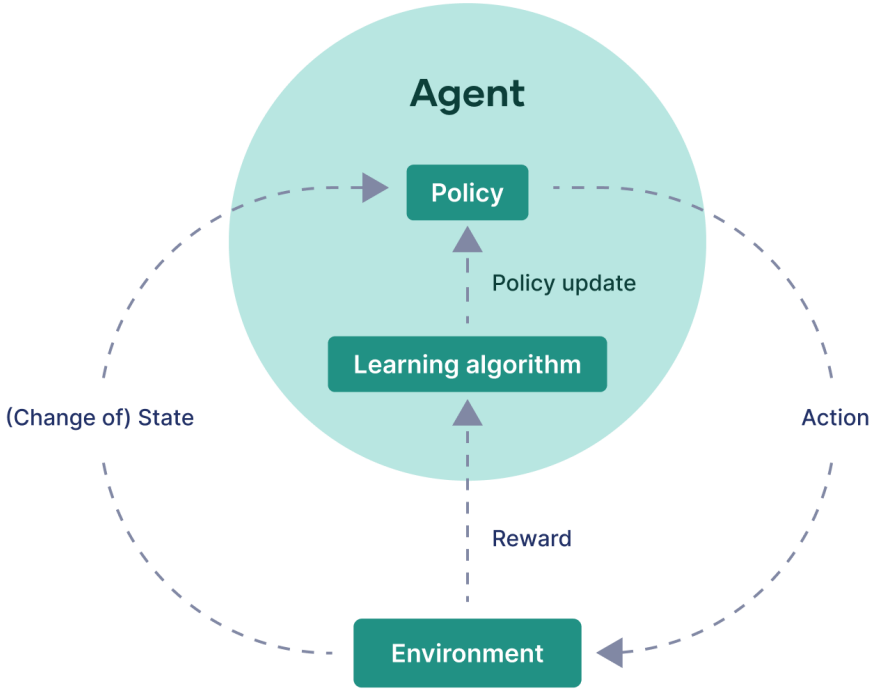
Multi-Layer Perception

Human Learning and Machine Learning

- Human Learning has inspired various machine learning algorithms.

Reward and Punishment

Inspire



Reinforcement Learning

Human's Curriculum Learning

Mathematics Curricula

$1 + 1 = 2$

$2 \times 3 = 6$

Arithmetic

$f(x) = 3x + 4$

$f^{-1}(y) = \sin y$

Function

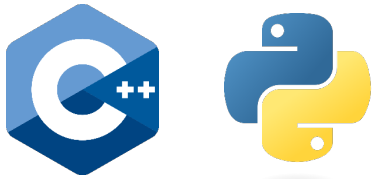
$$\lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

Limit

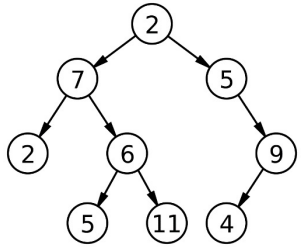
$$\int_{\pi}^{\pi} \cos x \, dx$$

Calculus

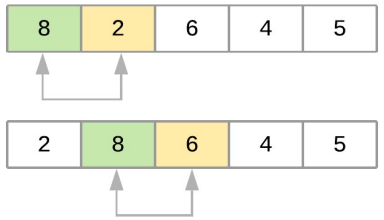
Computer Curricula



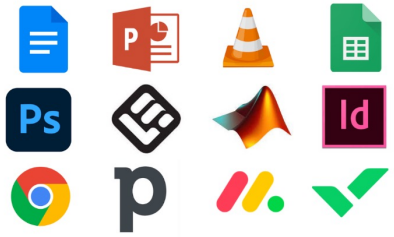
Programming



Data Structure



Algorithm



Software

Machine's Curriculum Learning

Human can benefit from curriculum.

Can machine benefit from curriculum?

Machine's Curriculum Learning

- Can machine benefit from curriculum?
- Our answer is:

YES

- ~~Otherwise there wouldn't be this tutorial.~~

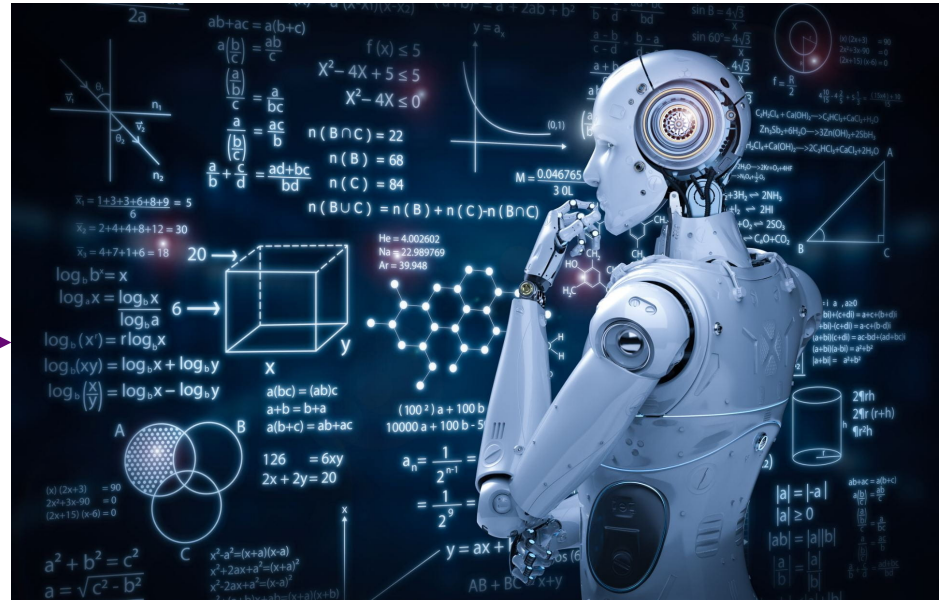
Machine's Curriculum Learning

- We well know curriculum in human learning, but what is curriculum in machine learning?

Human can learn from curriculum.

Insp

?



What is curriculum for machine?

Machine's Curriculum Learning

- Let us revisit the definition of **machine learning** from Mitchell:

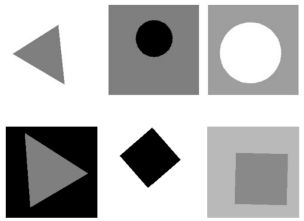
> *A model M is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*
>
> ——Machine Learning, Tom Mitchell, McGraw Hill, 1997.

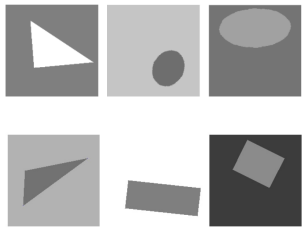
Model | **Data** | **Task**

Machine's Curriculum Learning

Data Curricula

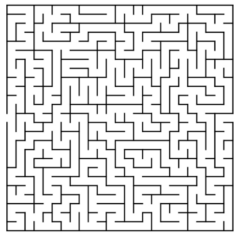
Single → Multiple



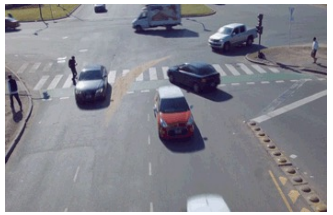


Regular → Irregular

Task Curricula

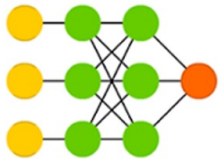


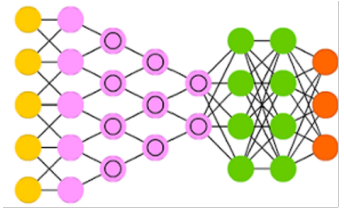
Simple → Complex



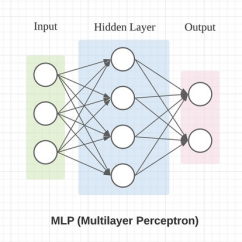
Static → Dynamic

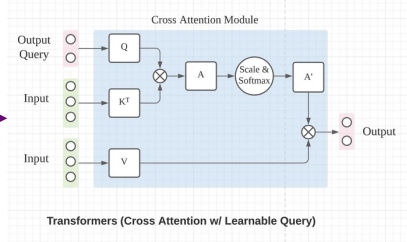
Model Curricula





Shallow → Deep

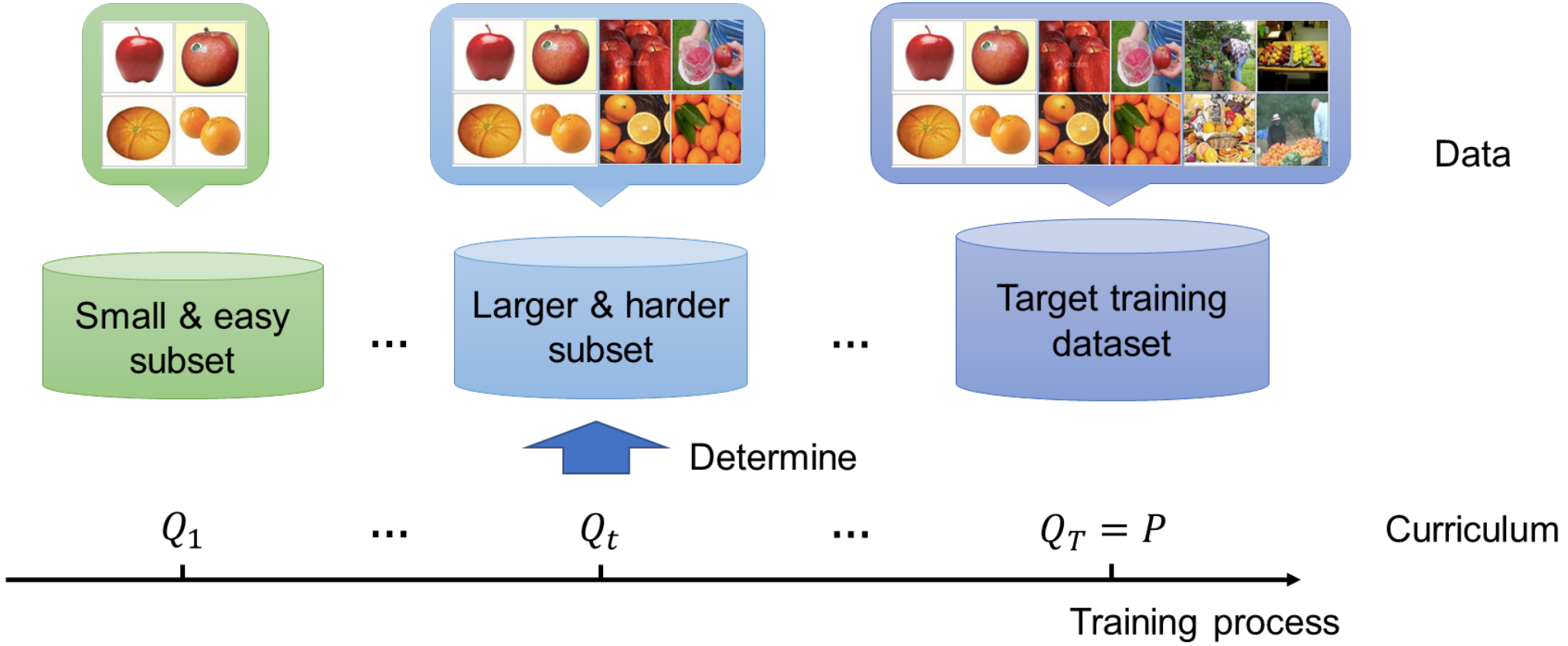




Basic → Advanced

Idea of Curriculum Learning

- Taking recognition as an example to explain the idea of curriculum in machine learning: **let the machine learn in a meaningful order.**



Outline

- In this tutorial, we hope to provide an overall picture of curriculum learning by answering the following questions:
  - **Theories**: What is the definition of curriculum learning, and why can it be effective?
  - **Approaches**: How to design a curriculum, and how does the state-of-the-art approaches design curriculum?
  - **Applications:** What applications benefit from curriculum learning?
  - **Tools**: What tools are available for ease of use and understanding CL?
- Lastly, we will summarize and discuss the future direction of curriculum learning.

Theories

Definition

Description of Curriculum Learning

- Recall the idea of curriculum learning: **let the machine learn in a meaningful order.**

The examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones.

——Curriculum Learning, Yoshua Bengio, et al., ICML, 2009.

Definition

Formulation of Curriculum Learning

- To formulate the description of "**let the machine learn in a meaningful order**", three definitions from narrow to board are developed:
  - Original Curriculum Learning
  - Data-Level Generalized Curriculum Learning
  - Generalized Curriculum Learning

Definition	Easy-to-Hard	Data-Level	Stage-Training
Original	√	√	√
Data-Level Generalized		√	√
Generalized			√

Definition

Original Curriculum Learning

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},$$

where z is a data instance and $\mathcal{D}$ is the training set.

- The following three conditions are satisfied:
  - Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$. (1)
  - The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$. (2)
  - $Q_T(z) = P(z)$. (3)

Definition

Original Curriculum Learning

- A curriculum is a sequence of training criteria over T training steps:

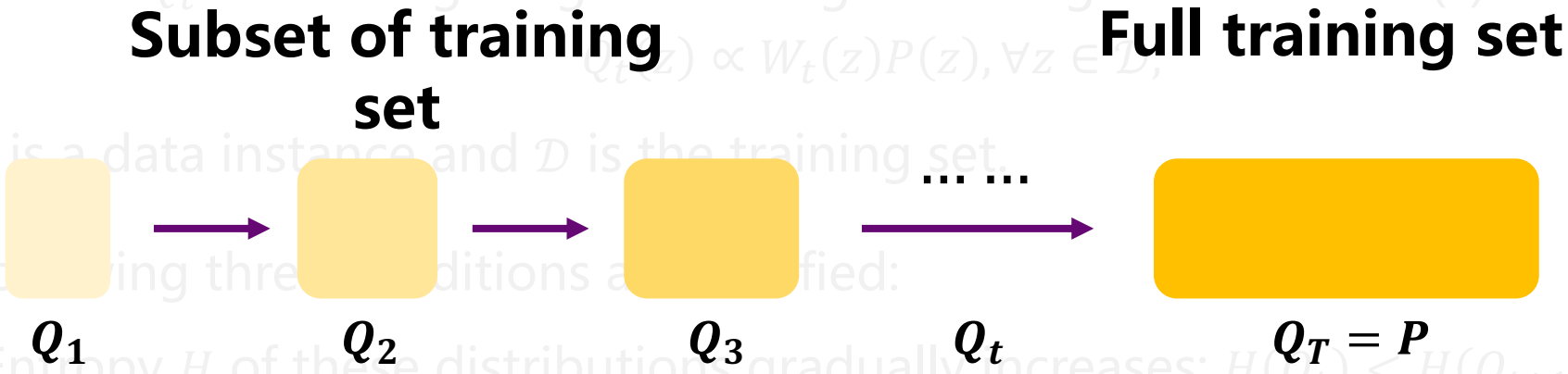
$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},$$

where z is a data instance and $\mathcal{D}$ is the training set.

- The following three conditions are satisfied:
  - Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$. (1)
  - The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$. (2)
  - $Q_T(z) = P(z)$. (3)



Definition

Data-level Generalized Curriculum

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},$$

where z is a data instance and $\mathcal{D}$ is the training set.

- ~~The following three conditions are satisfied:~~
  - ~~Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$.~~ (1)
  - ~~The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$.~~ (2)
  - ~~$Q_T(z) = P(z)$.~~ (3)

Definition

Data-level Generalized Curriculum

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},$$

where z is a data instance and $\mathcal{D}$ is the training set.

- The following three conditions are satisfied:
  - ◆ Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$. (1)
  - ◆ The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$. (2)
  - ◆ $Q_T(z) = P(z)$. (3)

Reweighted training set

Definition

Generalized Curriculum Learning

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

~~Each criterion Q_t is a reweighting of the target training distribution $P(z)$:~~

$$\cancel{Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},}$$

~~where z is a data instance and $\mathcal{D}$ is the training set.~~

- ~~The following three conditions are satisfied:~~
  - ~~Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$.~~ ~~(1)~~
  - ~~The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$.~~ ~~(2)~~
  - ~~$Q_T(z) = P(z)$.~~ ~~(3)~~

Definition

Generalized Curriculum Learning

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z) P(z), \forall z \in \mathcal{D},$$

where z is a data instance and $\mathcal{D}$ is the training set.

- The following three conditions are satisfied:
  - ◆ Entropy H of these distributions gradually increases: $H(Q_t) < H(Q_{t+1})$. (1)
  - ◆ The weight W for any instance increases: $W_t(z) \leq W_{t+1}(z), \forall z \in \mathcal{D}$. (2)
  - ◆ $Q_T(z) = P(z)$. (3)

Stage 1	Stage 2	Stage 3	… …	Stage T
Q_1	Q_2	Q_3	Q_t	Q_T

Effectiveness of Curriculum Learning

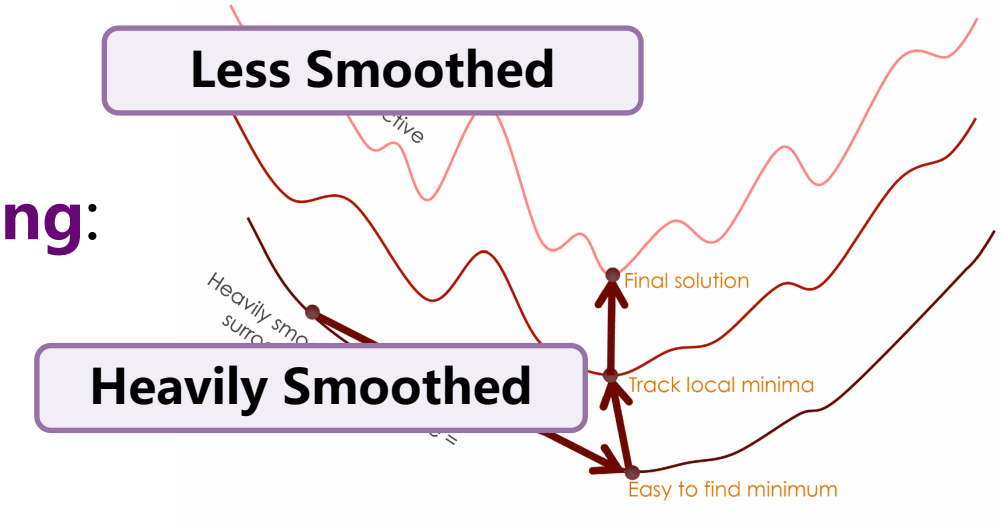
- After introducing the definition of curriculum learning in detail, we are now ready to answer this question:

How does curriculum learning benefit machine learning?

- Based on existing theoretical analysis, we summarize the answer as:
  - ◆Curriculum learning enables **better performance.**
  - ◆Curriculum learning enables **faster convergence.**

Effectiveness: Better Performance

- Curriculum learning is a particular **continuation method**:
  - First optimize a smooth version of problem to reveal the global picture,
  - Gradually consider less smoothed objective, until the target objective.
- Curriculum learning act similarly to **unsupervised pre-training**:
  - As a way to find better local minima,
  - As a regularizer.
- Curriculum learning is like **annealing**:
  - Tracking the local minima,
  - Easy to find global minimum.

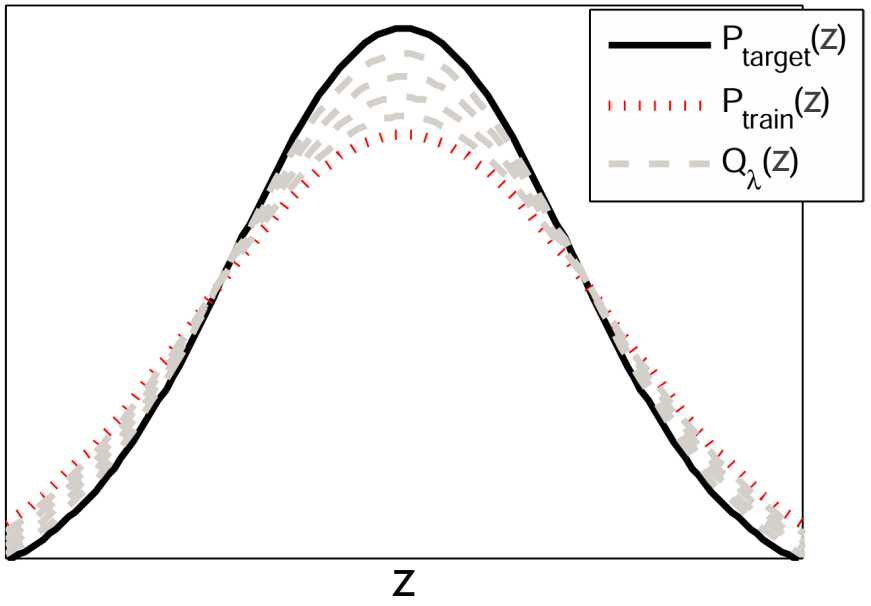


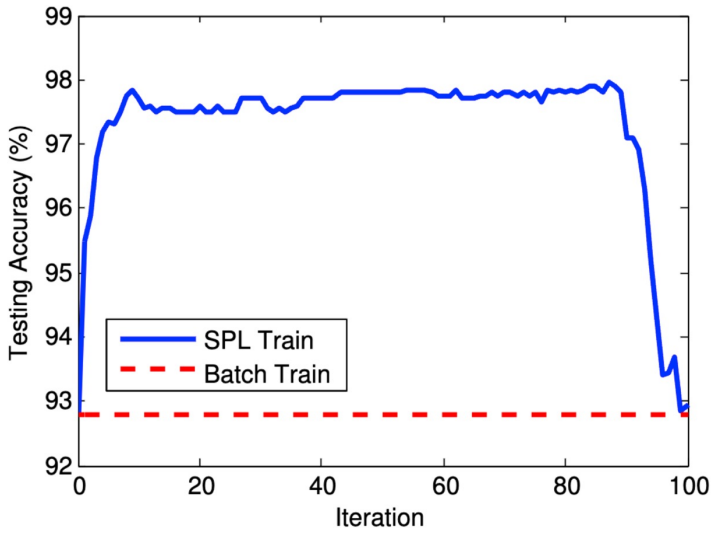
Curriculum Learning. ICML 2009.
Evolving Culture Versus Local Minima. Growing Adaptive Machines 2014.

Effectiveness: Better Performance

Curriculum as Continuation Method

- Curriculum changes the optimization landscape.
  - The distribution of training and testing can be different in big/noise data.
  - A sequence of pace distributions $Q(z)$ varying from $P_{train}(z)$ to $P_{target}(z)$.





Effectiveness: Better Performance

Curriculum as Unsupervised Pre-training

- **Example**: the task of k-parities ($d = 8,\ k = 6,\ y = sum(x[0{:}6])\ mod\ 2$)

This condition is unknown to the model !

Why this is easy?

Intuitively:

1. The 6th digit matters.
2. The 8th digit does not matter.
3. The 4th digit matters.

Easy Distribution

(1 1 1 1 1 1 0 1 , 0)

(1 1 1 1 1 0 1 1 , 1)

(1 1 1 1 1 1 1 1 , 0)

(1 1 1 1 1 1 1 0 , 0)

(1 1 1 0 1 1 1 1 , 1)

$$x \sim Rad\left(1 - \frac{1}{8}\right)^{\otimes 8}$$

$$i.e., P(x = 0) = \frac{1}{8}, P(x = 1) = \frac{7}{8}$$

Hard Distribution

(1 1 1 0 0 0 0 1 , 1)

(0 0 1 0 1 0 1 1 , 0)

(1 1 1 0 0 1 0 0 , 0)

(1 0 0 1 1 0 1 0 , 1)

(1 1 0 1 0 0 1 0 , 1)

$$x \sim Rad\left(1 - \frac{1}{2}\right)^{\otimes 8}$$

$$i.e., P(x = 0) = \frac{1}{2}, P(x = 1) = \frac{1}{2}$$

Why this is hard?

Intuitively:

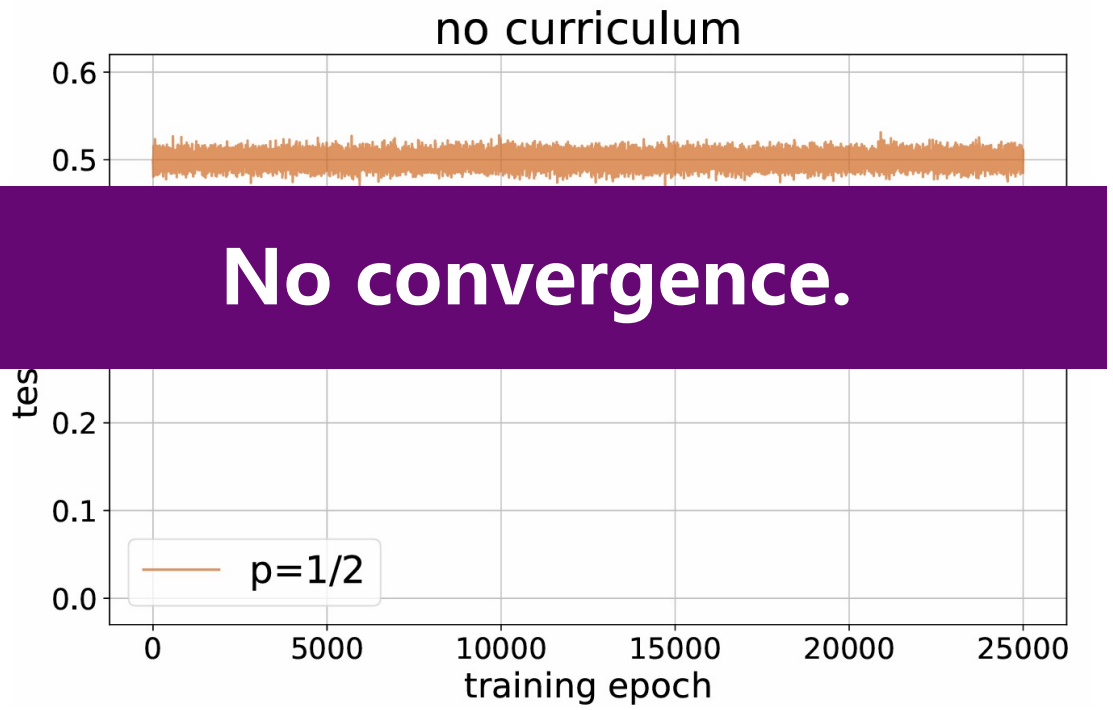
It is hard to say which digit matters to the answer.

A Mathematical Model for Curriculum Learning for Parities. ICML 2023.

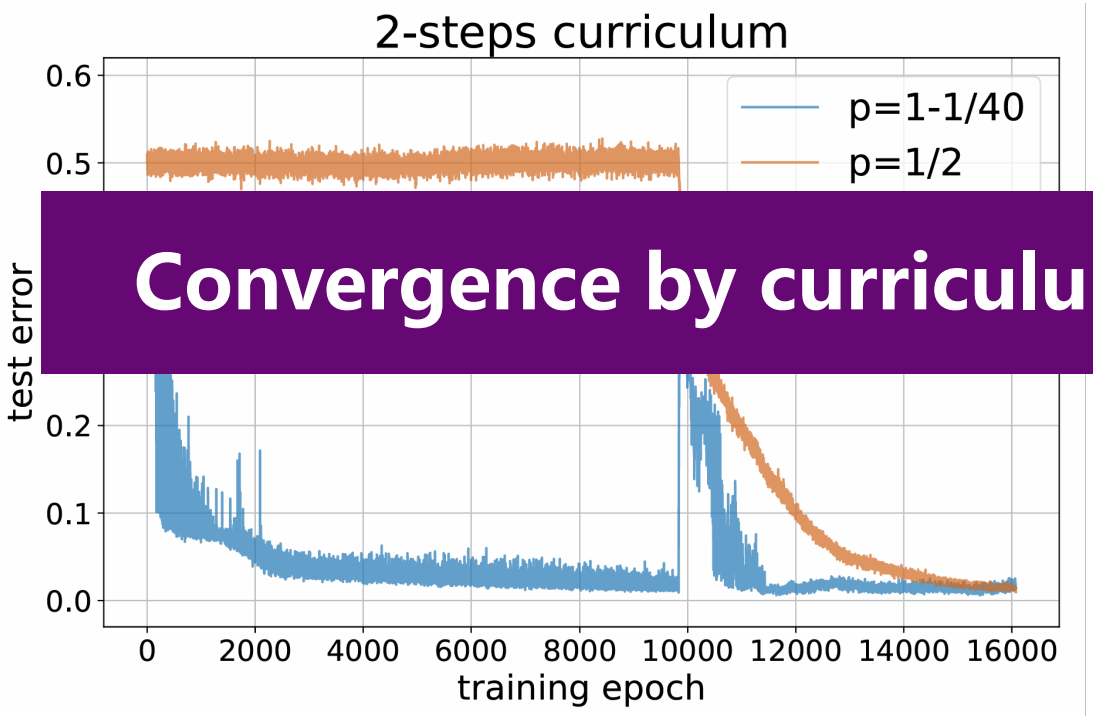
Effectiveness: Better Performance

Curriculum as Unsupervised Pre-training

- Curriculum enables to learn difficult distribution by learning on simple distribution first.



No convergence.



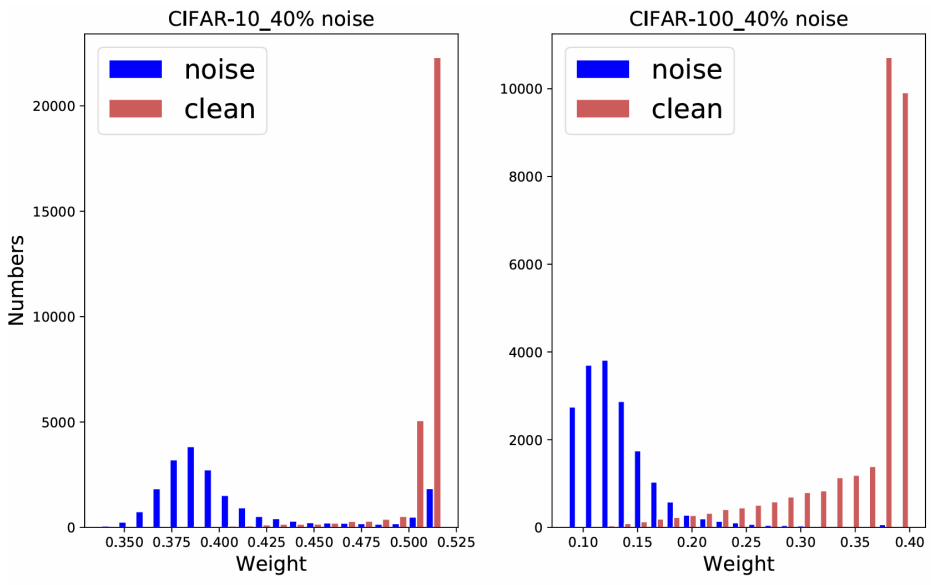
Convergence by curriculum.

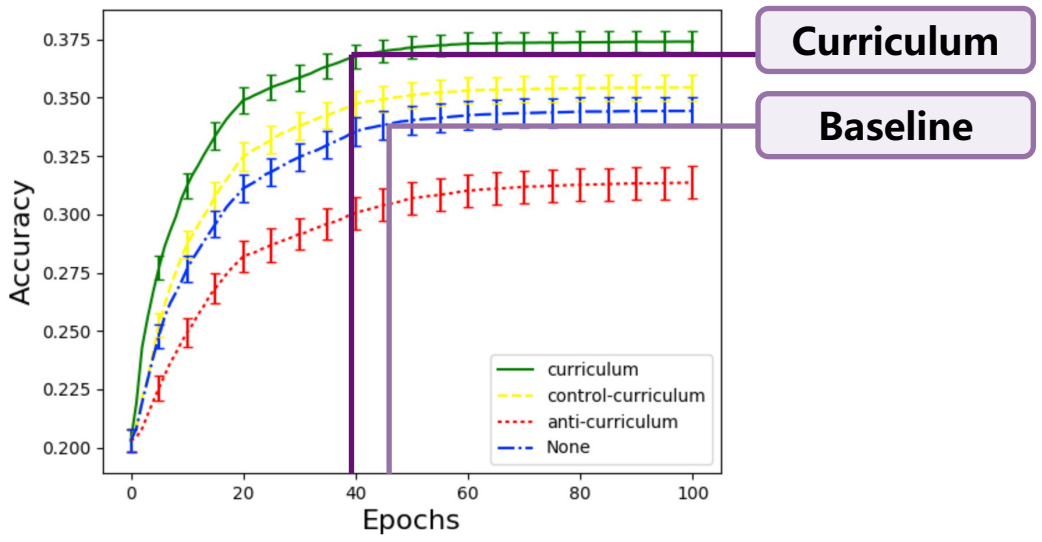
A Mathematical Model for Curriculum Learning for Parities. ICML 2023.

Effectiveness: Faster Convergence

Discarding Noisy Data

- Curriculum learning **wastes less time** with noisy and harder data instances.
  - The noisy data corresponds to harder data instances in the datasets.
  - A curriculum strategy tends to discard hard data instances.



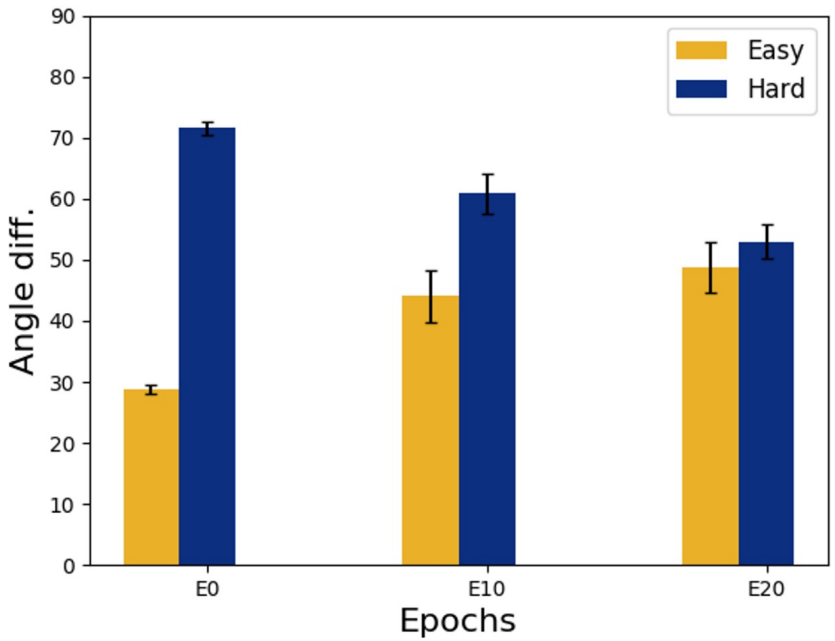


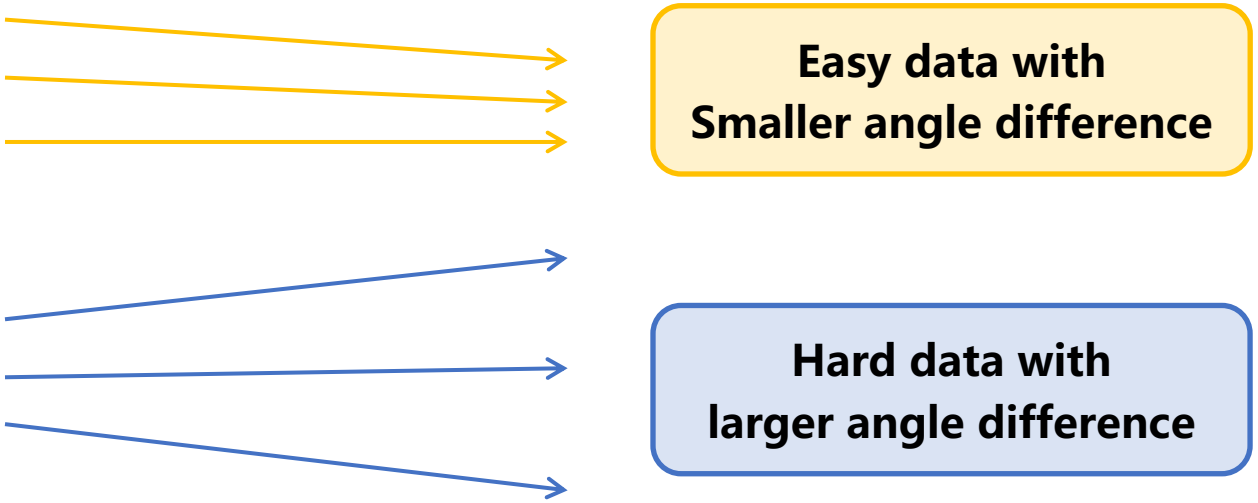
Curriculum Learning. ICML 2009.
Learning to Reweight Examples for Robust Deep Learning. ICML 2018.
Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. NeurIPS 2019.

Effectiveness: Faster Convergence

Steeper Optimization Landscape

- **The variance** in the gradient directions from easier data is significantly **smaller.**





Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks. ICML 2018.

Effectiveness: Faster Convergence

Steeper Optimization Landscape

- A ideal curriculum makes **the gradients overall steeper**.

Origin:

$$\mathcal{U}(\vartheta) = \hat{\mathbb{E}}[U_\vartheta] = \frac{1}{N}\sum_{i=1}^{N} U_\vartheta(X_i) \triangleq \frac{1}{N}\sum_{i=1}^{N} e^{-L_\vartheta(X_i)}$$

Curriculum:

$$\mathcal{U}_p(\vartheta) = \hat{\mathbb{E}}_{\boldsymbol{p}}[U_\vartheta] = \sum_{i=1}^{N} U_\vartheta(X_i)\underline{p(X_i)} = \sum_{i=1}^{N} e^{-L_\vartheta(X_i)}\underline{p_i}$$

Compare:

$$\begin{aligned}\mathcal{U}_p(\vartheta) &= \sum_{i=1}^{N}(U_\vartheta(X_i) - \hat{\mathbb{E}}[U_\vartheta])(p_i - \hat{\mathbb{E}}[p]) + N\hat{\mathbb{E}}[U_\vartheta]\hat{\mathbb{E}}[p] \\ &= \hat{\mathrm{Cov}}[U_\vartheta, p] + N\hat{\mathbb{E}}[U_\vartheta]\hat{\mathbb{E}}[p] = \mathcal{U}(\vartheta) + \underline{\hat{\mathrm{Cov}}[U_\vartheta, p]}\end{aligned}$$

Scoring Function (Reweighting)

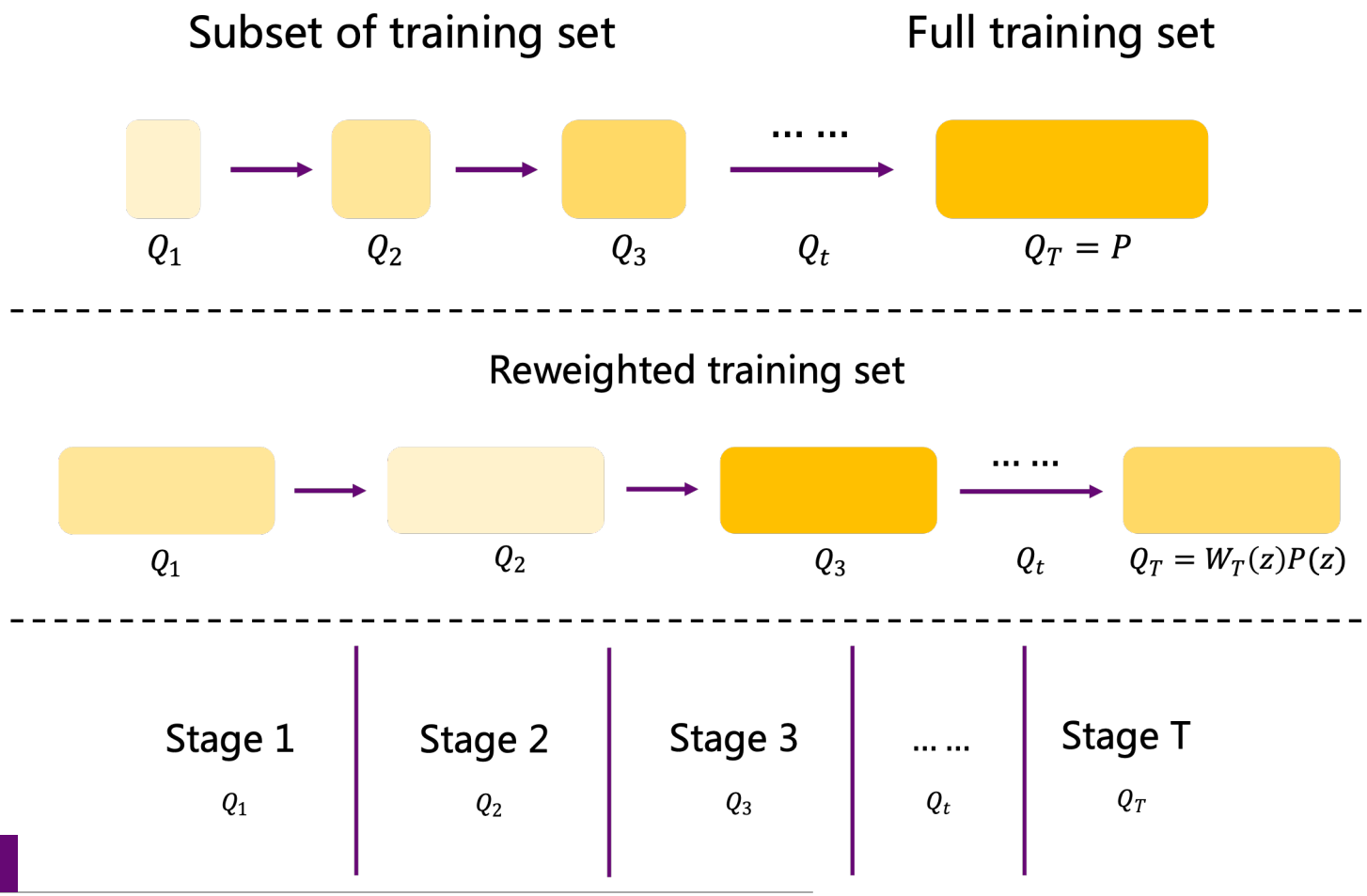
Steeper !

On The Power of Curriculum Learning in Training Deep Networks. ICML 2019.

Approaches

General Framework

- Recall 3 definition of curriculum learning:

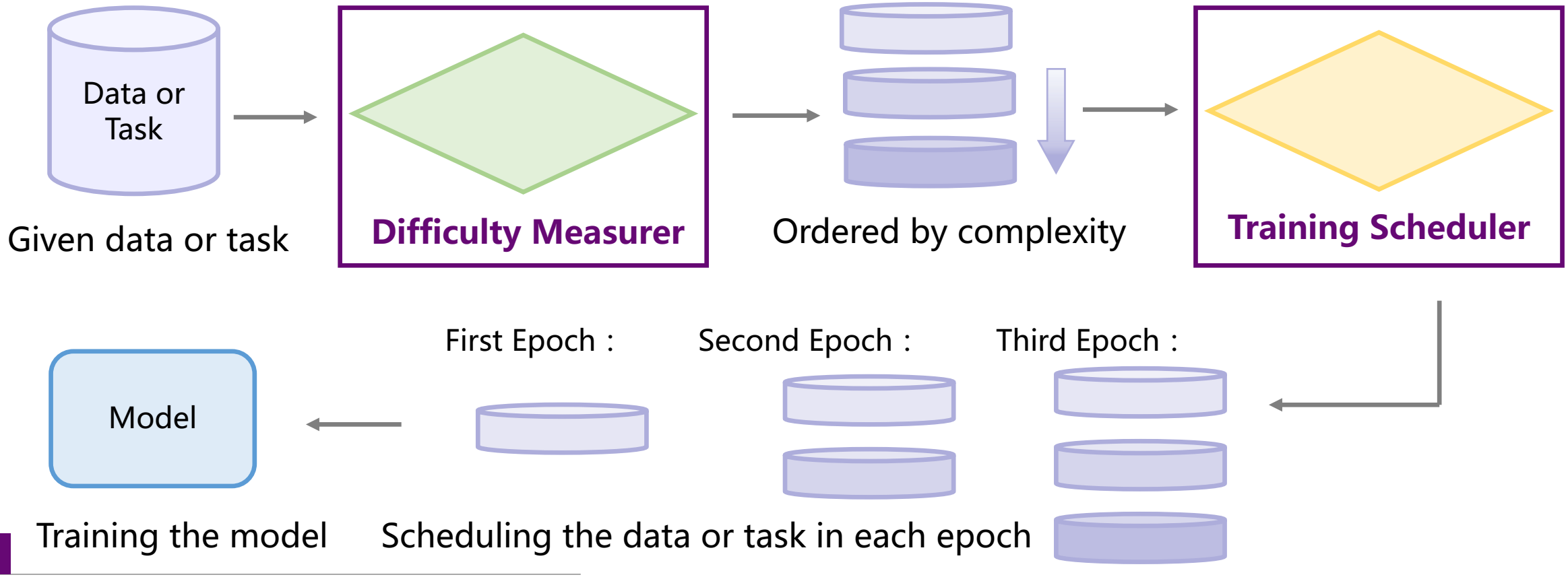


1. What is used for training at the beginning, and what is used later ?

2. When should we introduce more parts for training ?

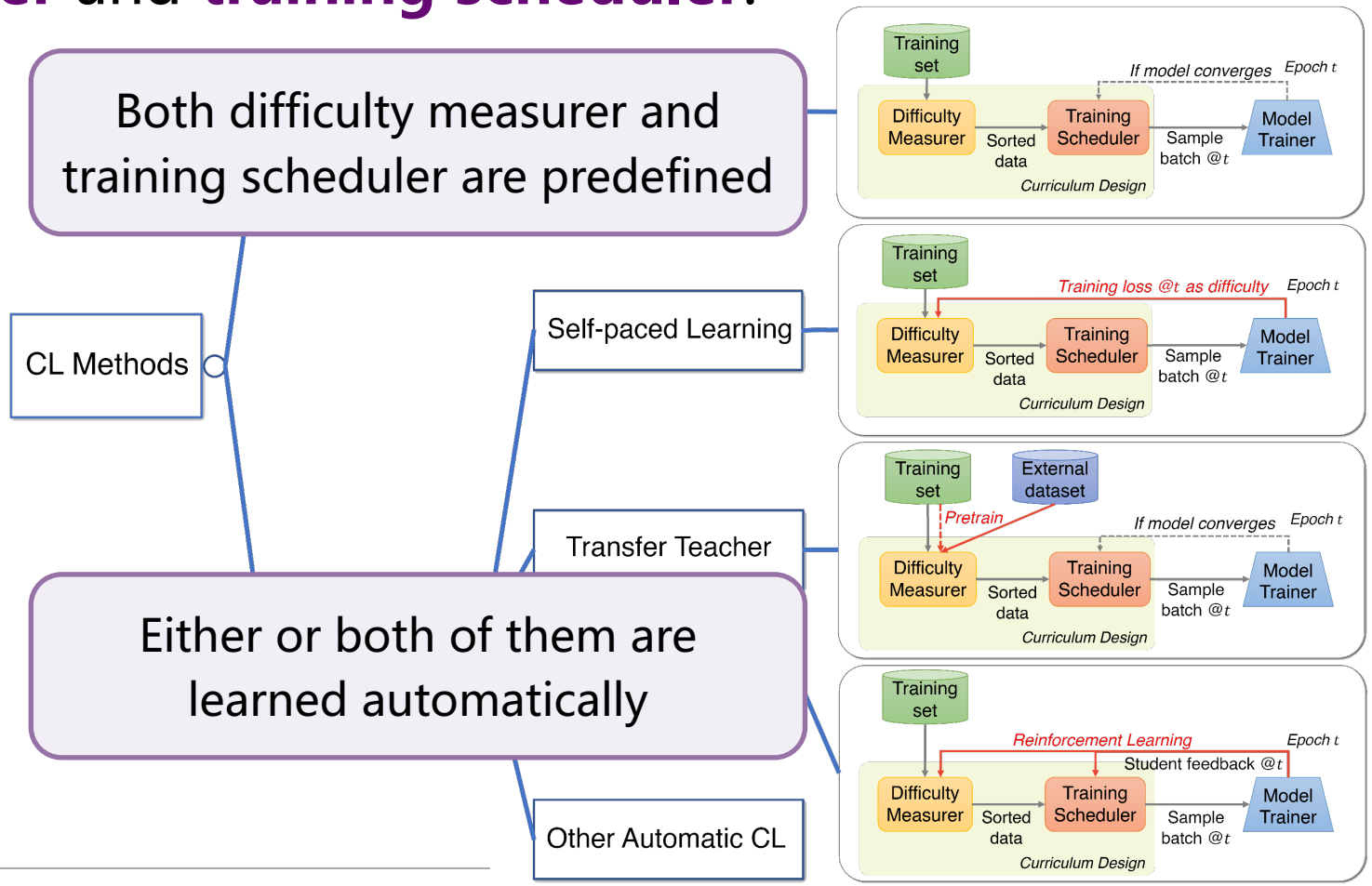
General Framework

- A general framework of **difficulty measurer** and **training scheduler** for the design of curriculum learning:

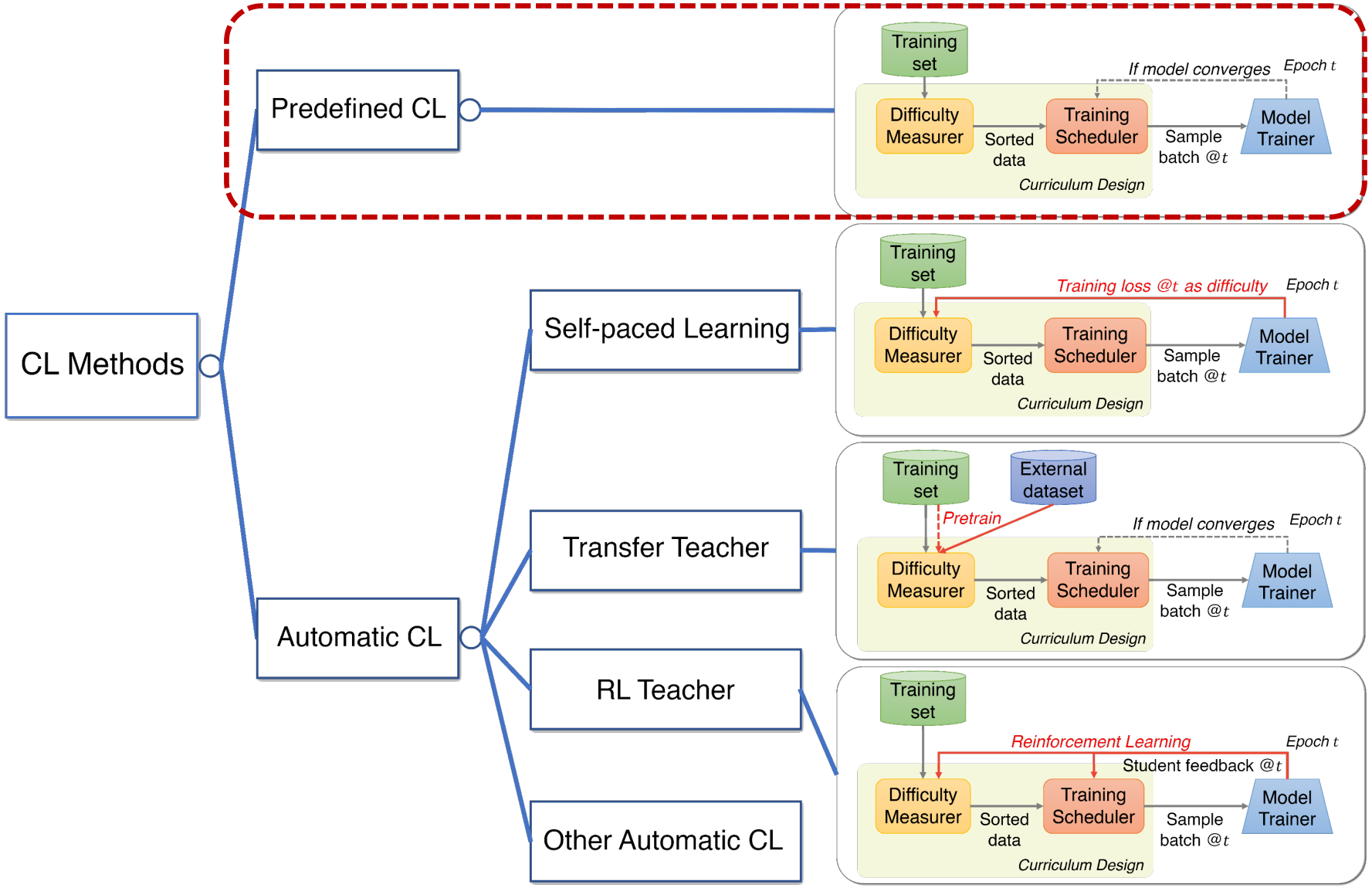


Taxonomy

- We categorize curriculum learning methods based on its **difficulty measurer** and **training scheduler**.



Predefined Curriculum



Predefined Curriculum

Common Types of Predefined Difficulty Measurer

- Both the Difficulty Measurer and Training Scheduler are designed by **human prior knowledge** with no data-driven algorithms involved.
- In shape recognition:

BasicShapes: easy, at epoch 1~128

GeomShapes: hard, at epoch 129~256

- In digit summation:

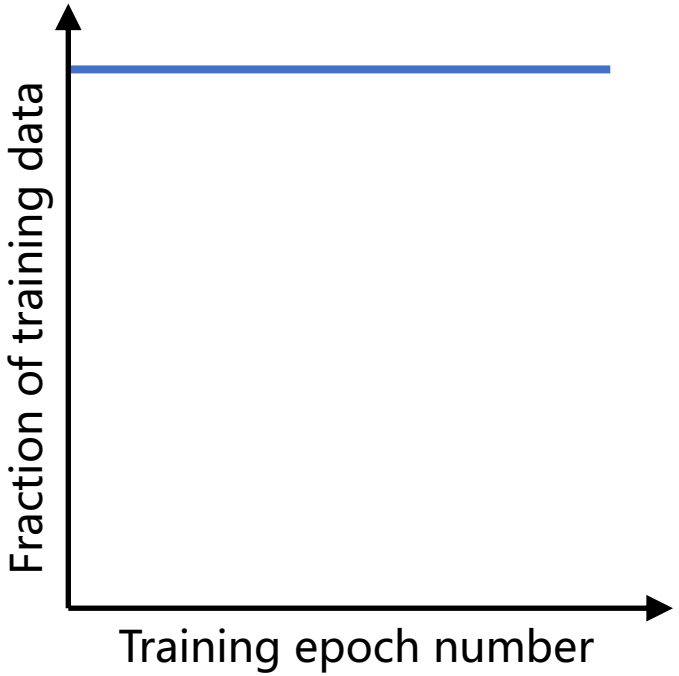
Input Sequence	Ground Truth
1	1
10	1
109	10
1091	11
10917	18
109173	21
1091735	26
10917356	32
109173567	39

Curriculum Learning. ICML 2009.
Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks. AAAI 2017.

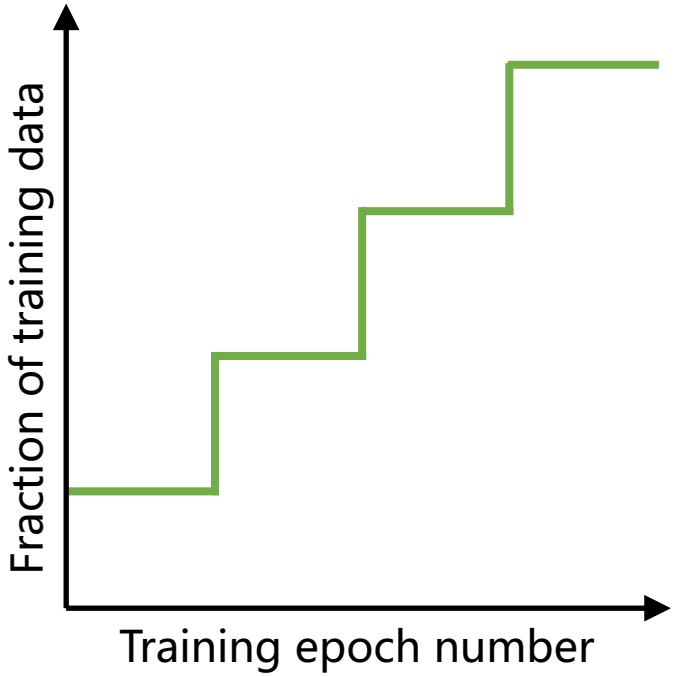
Predefined Curriculum

Common Types of Predefined Training Scheduler

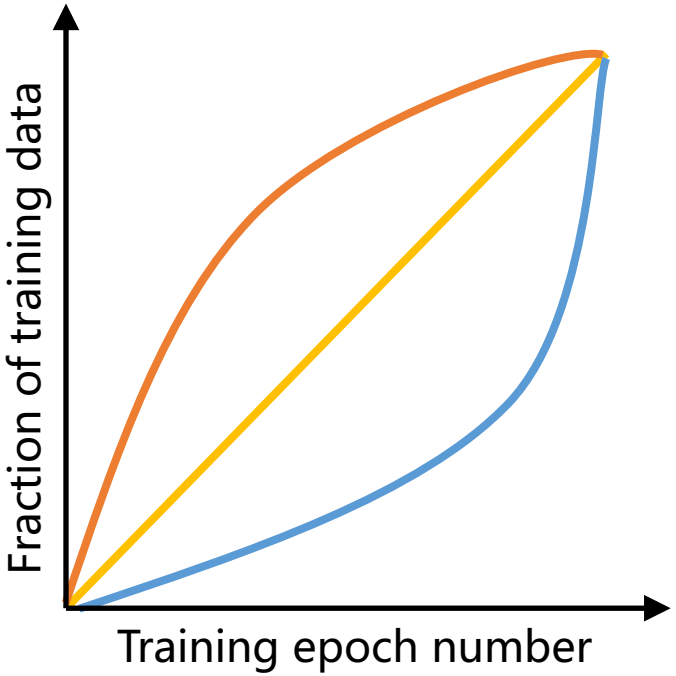
- No curriculum



- Discrete schedulers



- Continuous schedulers

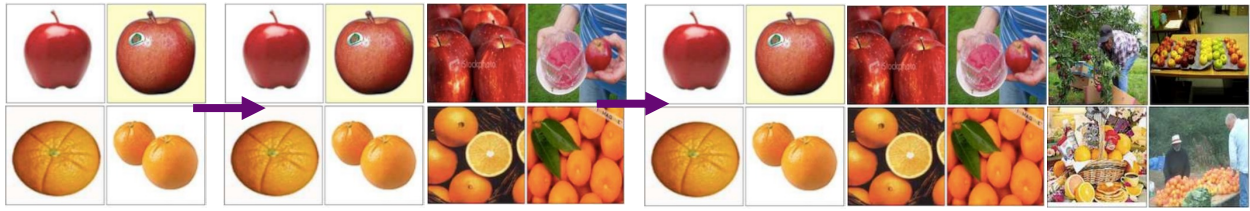


- The functions above are called **pacing functions** or **competence functions.**

Competence-based Curriculum Learning for Neural Machine Translation. NAACL 2019.
On The Power of Curriculum Learning in Training Deep Networks. ICML 2019.

Predefined Curriculum

Common Predefined Curriculum Learning Methods



Algorithm 1 One-Pass Curriculum

procedure OP-CURRICULUM(M,$\mathcal{D}$, $\mathcal{C}$)
  $\mathcal{D}'$ = sort($\mathcal{D}$, $\mathcal{C}$)
  $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^k\} = \mathcal{D}'$ where $\mathcal{C}(d_a) < \mathcal{C}(d_b)$ $d_a \in D^i$, $d_b \in D^j$, $\forall i < j$
  for $s = 1...k$ **do**
    while not converged for p epochs **do**
      train(M, $\mathcal{D}^s$)
    end while
  end for
end procedure

Algorithm 2 Baby Steps Curriculum

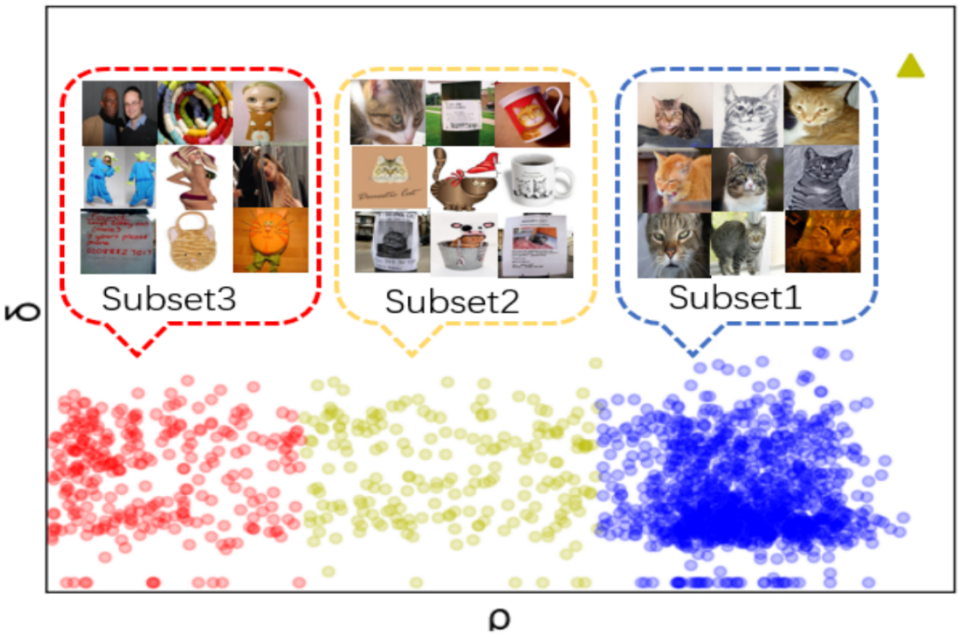
procedure BS-CURRICULUM(M,$\mathcal{D}$, $\mathcal{C}$)
  $\mathcal{D}'$ = sort($\mathcal{D}$, $\mathcal{C}$)
  $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^k\} = \mathcal{D}'$ where $\mathcal{C}(d_a) < \mathcal{C}(d_b)$ $d_a \in D^i$, $d_b \in D^j$, $\forall i < j$
  $\mathcal{D}^{train} = \emptyset$
  for $s = 1...k$ **do**
    $\mathcal{D}^{train} = \mathcal{D}^{train} \cup \mathcal{D}^s$
    while not converged for p epochs **do**
      train(M, $\mathcal{D}^{train}$)
    end while
  end for
end procedure

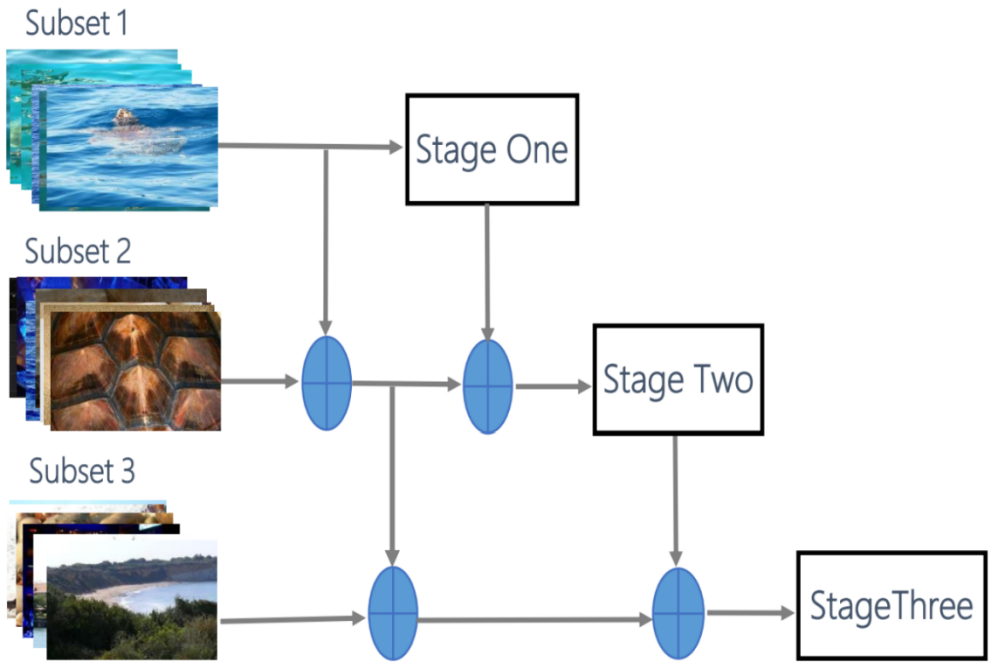
Difference !

There can be problems of catastrophic forgetting.

Predefined Curriculum

Example: Image Recognition



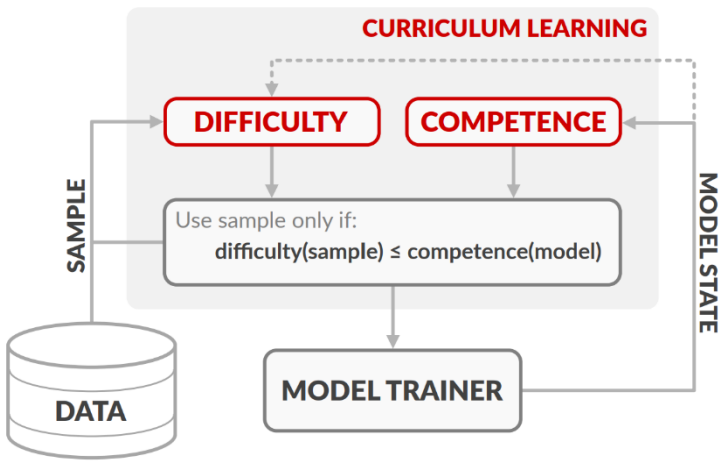


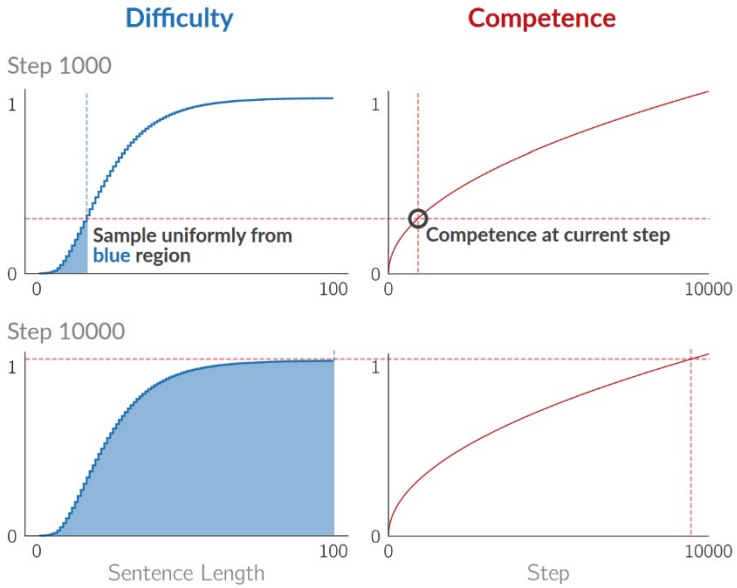
Motivation	Difficulty	Scheduler
Denoise: web images are noisy	High local density = easy, clean	3-stage, add new subset when converge

CurriculumNet : Weakly Supervised Learning from Large-Scale Web Images. ECCV 2018.

Predefined Curriculum

Example: Neural Machine Translation





Algorithm 1: Competence-based curriculum learning algorithm.

Input: Dataset, $\mathcal{D} = \{s_i\}_{i=1}^{M}$, consisting of M samples, model trainer, $\mathcal{T}$, that takes as input batches of training data to use at each step, difficulty scoring function, d, and competence function, c.

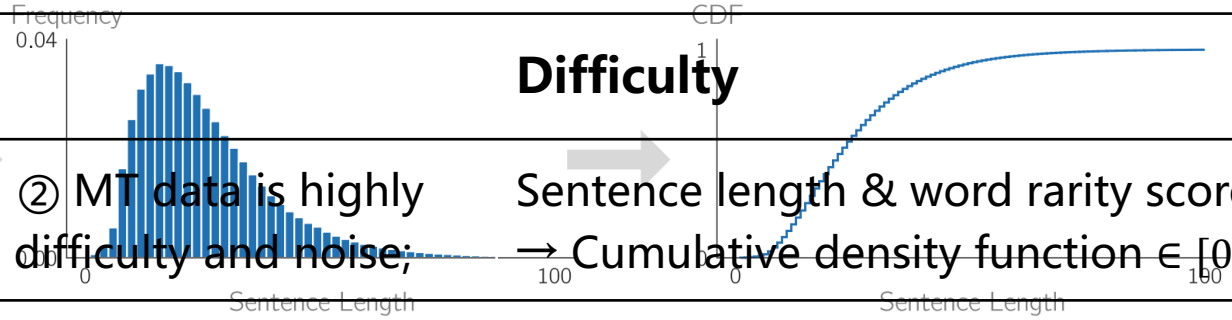
1 Compute the difficulty, $d(s_i)$, for each $s_i \in \mathcal{D}$.
2 Compute the cumulative density function (CDF) of the difficulty scores. This results in one difficulty CDF score per sample, $\bar{d}(s_i) \in [0, 1]$. Illustrated in Figure 2.
3 **for** *training step* $t = 1, \ldots$ **do**
4 Compute the model competence, $c(t)$.
5 Sample a data batch, B_t, uniformly from all $s_i \in \mathcal{D}$, such that $\bar{d}(s_i) \leq c(t)$. Illustrated in Figure 3.
6 Invoke the trainer, $\mathcal{T}$, using B_t as input.

Output: Trained model.

Sentence	Length
Hey, what's up?	4
Barack Obama loves ...	13
My name is ...	6
What did she say?	123
⋮	

Motivation

① NMT training is slow; ② MT data is highly heterogenous in quality, difficulty and noise;



Difficulty

Sentence length & word rarity scores → Cumulative density function ∈ [0,1]

Sentence	Difficulty
Hey, what's up?	0.01
Barack Obama loves ...	0.15
My name is ...	0.03
What did she say?	0.91
⋮	

Scheduler

Model competence function $c(t) \in [0,1]$

Competence-based Curriculum Learning for Neural Machine Translation. NAACL 2019.

Predefined Curriculum

Example: Robotics and Reinforcement Learning:

We created a simple curriculum with 5 steps:

1. A single room with a target.
2. Two rooms separated by lava.
3. Two rooms separated by wall.
4. Three rooms separated by lava and wall, in random order.
5. Four rooms separated by lava and walls, in random order.

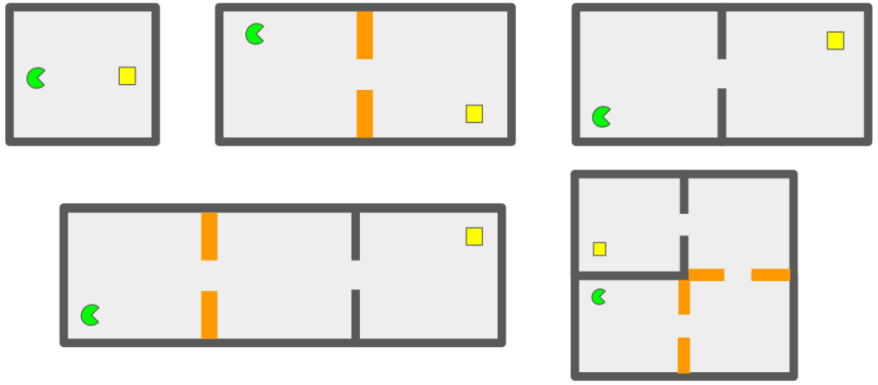


Figure 7: 5-step curriculum.

Motivation	Difficulty	Scheduler
Guide: target task is hard	Task difficulty	Move to new task when converge

Teacher-Student Curriculum Learning. NeurIPS 2017.

Predefined Curriculum

Example: k-Parities:

Definition 2.1 (r-steps curriculum learning (r-CL)). For a fixed $r \in \mathbb{N}$, let $T_1, ...T_r \in \mathbb{N}$ and $p_1, ..., p_r \in [0, 1]$. Denote by $\bar{p} := (p_1, ..., p_r)$ and $\bar{T} := (T_1, ..., T_{r-1})$. We say that a neural network $\mathrm{NN}(x; \theta^t)$ is trained by SGD with a r-CL$(\bar{T}, \bar{p})$ if θ^t follows the iterations in (1) with:

$$\begin{aligned}
\mathcal{D}^t &= \mathrm{Rad}(p_1), && 0 < t \le T_1, \\
\mathcal{D}^t &= \mathrm{Rad}(p_2), && T_1 < t \le T_2, \\
&\cdots \\
\mathcal{D}^t &= \mathrm{Rad}(p_r), && T_{r-1} < t \le T.
\end{aligned}$$

We say that r is the number of *curriculum steps*.

Easy Distribution		Hard Distribution
(<u>1111110</u>1,0)		(<u>1110000</u>1,1)
(<u>1111101</u>1,1)		(<u>0010101</u>1,0)
(<u>1111111</u>1,0)	→	(<u>1110010</u>0,0)
(<u>1111111</u>0,0)		(<u>1001101</u>0,1)
(<u>1110111</u>1,1)		(<u>1101001</u>0,1)
$x \sim Rad\left(1-\frac{1}{8}\right)^{\otimes 8}$		$x \sim Rad\left(1-\frac{1}{2}\right)^{\otimes 8}$
$i.e., P(x=0) = \frac{1}{8}$		$i.e., P(x=0) = \frac{1}{2}$

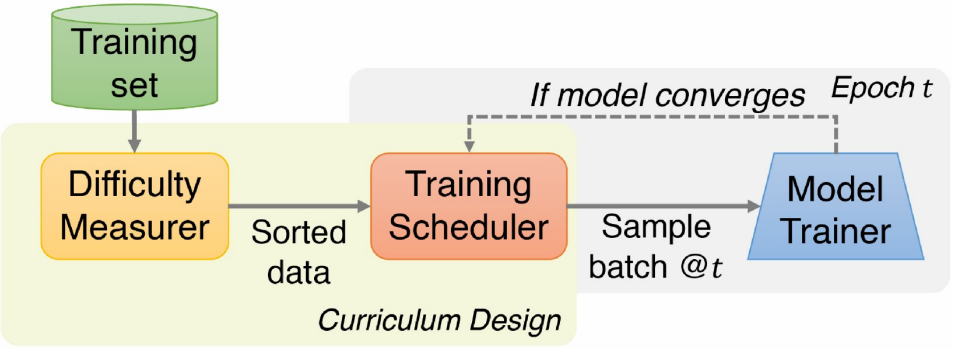
Motivation	Difficulty	Scheduler
Guide: certain distribution is unsolvable	Distribution difficulty	Move to new distribution when converge

A Mathematical Model for Curriculum Learning for Parities. ICML 2023.

Predefined Curriculum Learning

Summary

- Strengths:
  - Simple and effective.
  - Incorporate human prior knowledge.
- Limitations:
  - Not flexible for a specific task.
  - Ignore the model learning state.
  - Requiring expert knowledge and time.
  - Difficulty for human and machine is different.
  - Sensitive to the hyperparameters of a curriculum.

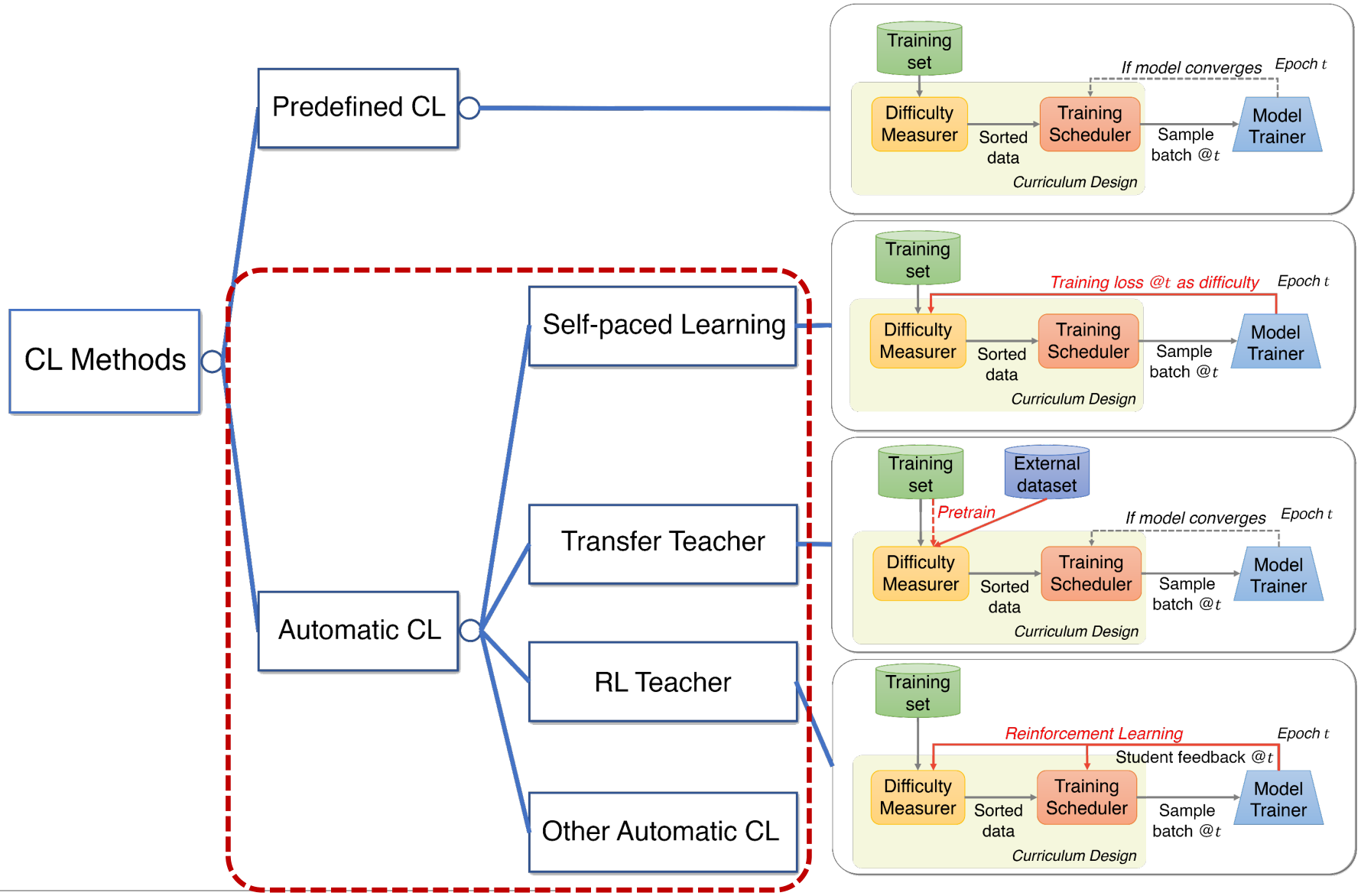


Algorithm 2 Baby Steps Curriculum

procedure BS-CURRICULUM(M,$\mathcal{D}$, $\mathcal{C}$)
  $\mathcal{D}'$ = sort($\mathcal{D}$, $\mathcal{C}$)
  $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^k\} = \mathcal{D}'$ where $\mathcal{C}(d_a) < \mathcal{C}(d_b)$ $d_a \in D^i$, $d_b \in D^j, \forall i < j$
  $\mathcal{D}^{train} = \emptyset$
  for $s = 1...k$ **do**
    $\mathcal{D}^{train} = \mathcal{D}^{train} \cup \mathcal{D}^s$
    while not converged for p epochs **do**
      train(M, $\mathcal{D}^{train}$)
    end while
  end for
end procedure

Too much heuristics, and not flexible.

Automatic Curriculum: Self-Paced Learning



Automatic Curriculum

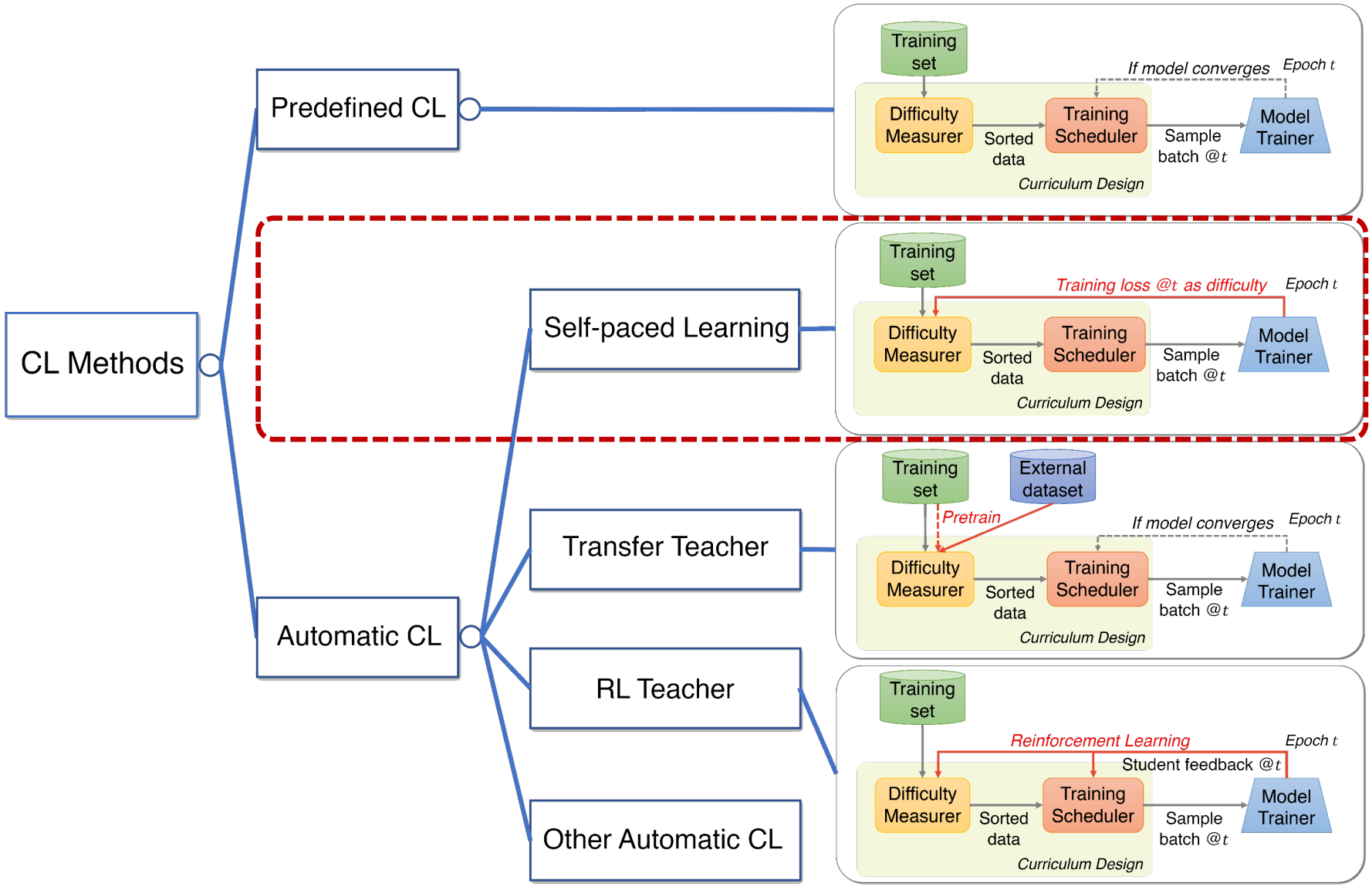
- Predefined Curriculum vs. Automatic Curriculum

	Predefined CL	**Automatic CL**
Applicability	Need domain knowledge	General
Difficulty	Human predefined, fixed	Model decided, dynamic
Scheduler	Ignore model feedback, fixed	Consider model feedback, dynamic

- Main Methodologies for Automatic Curriculum

Self-Paced Learning	**Transfer Teacher**	**RL Teacher**	**Others**
Student model tell difficulty by itself	Teacher model judge difficulty	Teacher judge according to student feedback	Curriculum on Optimization, Loss, Model, ……

Automatic Curriculum: Self-Paced Learning



Automatic Curriculum: Self-Paced Learning

Self-Paced Learning (SPL)

- Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ denote the training set, where x_i and y_i is the feature and label of data 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 $l_i = \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i)$, where $\mathcal{L}$ is the learning objective.
- In general machine learning:

$$\min_{\boldsymbol{w}} \mathbb{E}(\boldsymbol{w}) = \sum_{i=1}^{N} \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i).$$

- In self-paced learning:

$$\min_{\boldsymbol{w};\, \boldsymbol{v} \in [0,1]^N} \mathbb{E}(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + g(\boldsymbol{v}; \lambda),$$

where $\boldsymbol{v} = [v_1, v_2, \dots, v_N]^T \in [0,1]^N$ is a vector of data weights and λ is the age parameter, which works as the training scheduler to control the learning pace.

Automatic Curriculum: Self-Paced Learning

Self-Paced Learning (SPL)

- In self-paced learning:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}\in[0,1]^N} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + g(\boldsymbol{v}; \lambda).$$

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

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

- Therefore, the optimization target is:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}\in[0,1]^N} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda \sum_{i=1}^{N} v_i\,.$$

$$= \sum_{i=1}^{N} v_i (\mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda)\,.$$

Automatic Curriculum: Self-Paced Learning

Self-Paced Learning (SPL)

- To solve the learning objective:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}\in[0,1]^N} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i(\mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda)\,.$$

- Alternatively optimizing $\boldsymbol{w}$ and $\boldsymbol{v}$ while fixing the other:
  - With the fixed $\boldsymbol{v}^*$, $-\lambda v_i^*$ is a fixed item, so:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i)\,.$$

  - With the fixed $\boldsymbol{w}^*$, $\mathcal{L}(f_{\boldsymbol{w}^*}(x_i), y_i)$ can be replaced by l_i:

$$\boldsymbol{v}^* = \arg\min_{v_i\in[0,1]} \sum_{i=1}^{N} v_i(l_i - \lambda) = \begin{cases} 1, & l_i < \lambda \\ 0, & l_i \geq \lambda \end{cases}$$

Self-Paced Learning for Latent Variable Models. NeurIPS 2010.

Automatic Curriculum: Self-Paced Learning

Self-Paced Learning (SPL)

- **Notice** that the weight of easy data ($l_i < \lambda$) is 1, while that of hard data is 0:

$$\boldsymbol{v}^* = \arg\min_{v_i \in [0,1]} \sum_{i=1}^{N} v_i(l_i - \lambda) = \begin{cases} 1, & l_i < \lambda \\ 0, & l_i \geq \lambda \end{cases}$$

- It is consistent with the idea of curriculum learning: learning from easy ones. As the training proceeds, the λ can be adjusted to feed more difficult data to the model.
- How to adjust λ : commonly by controlling the ratio of selected data instances.
- Apart from the original self-paced learning, there are mainly **two types of variants**:
  - Based on different **regularizer** $g(\cdot)$, e.g., SPLD, MCL, Superloss, , SPL-IR……
  - Based on different **weight** v, e.g., SPCL, DIHCL, DCL, ScreenerNet, ……

Self-Paced Learning for Latent Variable Models. NeurIPS 2010.

Self-Paced Learning: Variant based on $g(\cdot)$

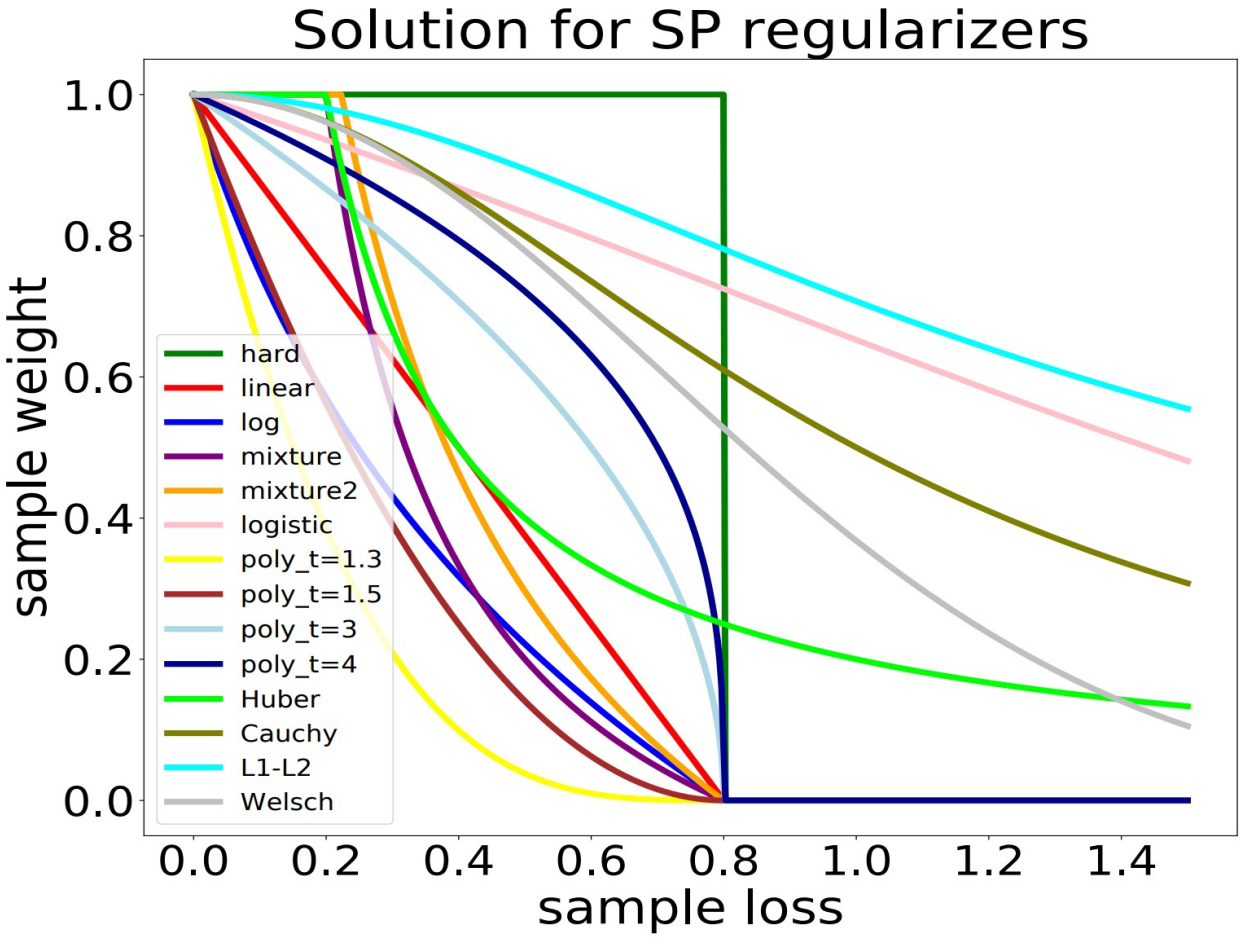
Common Self-Paced Regularizer

Regularizers	$g(\boldsymbol{v};\lambda)$	$\boldsymbol{v}^*(l_i,\lambda)$
Hard / Binary	$-\lambda\sum_{i=1}^{N} v_i$	$\begin{cases}1, & l_i<\lambda\\ 0, & otherwise\end{cases}$
Linear	$\frac{1}{2}\lambda\sum_{i=1}^{N} v_i^2-2v_i$	$\begin{cases}1-l_i/\lambda, & l_i<\lambda\\ 0, & otherwise\end{cases}$
Logarithmic	$\sum_{i=1}^{N}\left(\zeta v_i-\frac{\zeta^{v_i}}{\log\zeta}\right)$ $\zeta=1-\lambda, 0<\lambda<1$	$\begin{cases}\log(l_i+\zeta)/\log\zeta, & l_i<\lambda\\ 0, & otherwise\end{cases}$
Mixture	$-\zeta\sum_{i=1}^{N}\log\left(v_i+\frac{1}{\lambda_1}\zeta\right)$ $\zeta=\frac{\lambda_1\lambda_2}{\lambda_1-\lambda_2}, \lambda_1>\lambda_2>0$	$\begin{cases}1, & l_i\le\lambda_2\\ 0, & l_i\ge\lambda_1\\ \zeta(1\backslash l_i-1\backslash\lambda_1), & otherwise\end{cases}$
……	……	……

Easy Samples First: Self-paced Reranking for Zero-Example Multimedia Search. ACMMM 2014.

Self-Paced Learning: Variant based on $g(\cdot)$

Common Self-Paced Regularizers



Easy Samples First: Self-paced Reranking for Zero-Example Multimedia Search. ACMMM 2014.

Self-Paced Learning: Variant based on $g(\cdot)$

Self-Paced Learning with Diversity (SPLD)

- SPL prefers to select same-group samples as they are easy to current model.
- SPLD adds a diverse term, i.e., the negative $l_{2,1}$-norm, so that non-zero entries of $\boldsymbol{v}$ tend to be scattered across more groups:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}\in[0,1]^N} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda, \gamma) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda \sum_{i=1}^{N} v_i - \gamma \|\boldsymbol{v}\|_{2,1},$$

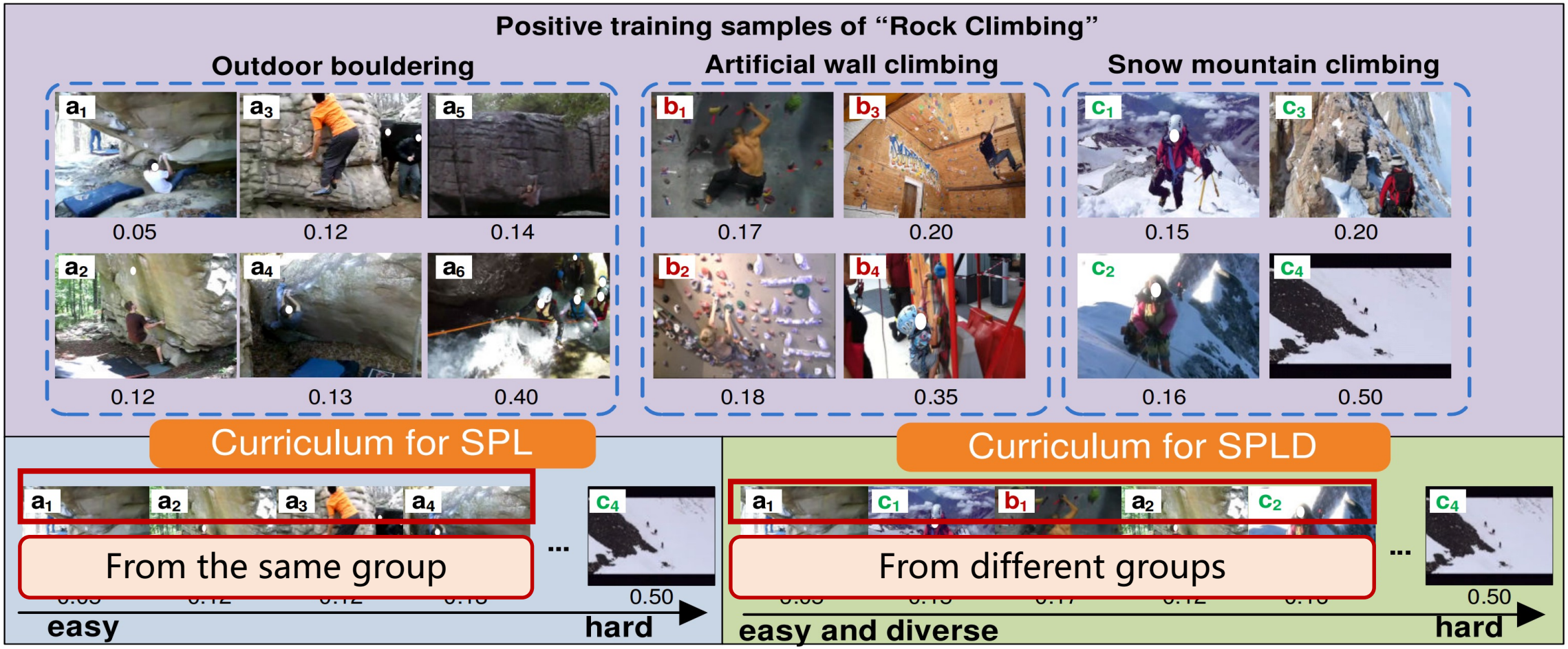
$$\|\boldsymbol{v}\|_{2,1} = \sum_{j=1}^{b} \left\|\boldsymbol{v}^{(j)}\right\|_2,$$

- where b is the number of the groups. The selected samples is diverse if they come from different groups.

Self-Paced Learning with Diversity. NeurIPS 2014.

Self-Paced Learning: Variant based on $g(\cdot)$

Self-Paced Learning with Diversity (SPLD)



Self-Paced Learning with Diversity. NeurIPS 2014.

Self-Paced Learning: Variant based on $g(\cdot)$

Minimax Curriculum Learning (MCL)

- SPLD:

$$\min_{\boldsymbol{w}} \boxed{\min_{\boldsymbol{v}\in[0,1]^n} \mathbb{E}(\boldsymbol{w}, \boldsymbol{v}; \lambda)} \text{ (Lower bound of loss)}$$

  - SPLD discards decision-boundary samples and selects well-learned samples.
  - SPLD selects outliers in late stages only because of diversity.
  - SPL / SPLD only minimizes a lower bound of the loss.

- MCL:

$$\min_{\boldsymbol{w}} \max_{A\subseteq\mathcal{D}} \mathbb{E}(\boldsymbol{w}, A) = \sum_{i=1}^{N} \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + \lambda F(A), \quad F: 2^{\mathcal{D}} \to \mathbb{R}_+$$

  - A is a subset, and $F(A)$ is a submodular function measuring its diversity
  - Inner max: prefer <u>harder and more diverse</u> samples
  - Outer min: minimize the <u>upper bound</u> of loss

  } better generalization and robustness // solve weakness (1)(3)

  - Scheduler: Increase subset A's size k and decrease λ with time // solve weakness (2)

Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. ICLR 2018.

Self-Paced Learning: Variant based on $g(\cdot)$

Minimax Curriculum Learning (MCL)

- MCL:

$$\min_{\boldsymbol{w}} \max_{A \subseteq \mathcal{D}} \mathbb{E}(\boldsymbol{w}, A) = \sum_{i=1}^{N} \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + \lambda F(A), \quad F: 2^{\mathcal{D}} \to \mathbb{R}_+$$

- In the experiments:

$$F(A) = \sum_{u \in \mathcal{U}} w_u \sqrt{c_u(A)}$$

$$c_u(A) = \sum_{a \in A} c_u(a)$$

- $F(A)$ is a simple "feature based" submodular function, where $\mathcal{U}$ is a set of features, $c_u(A)$ is a subset of clusters, and $c_u(a)$ is the nonnegative feature u of the centroid for cluster a, which can be interpreted as a nonnegative score for cluster a.

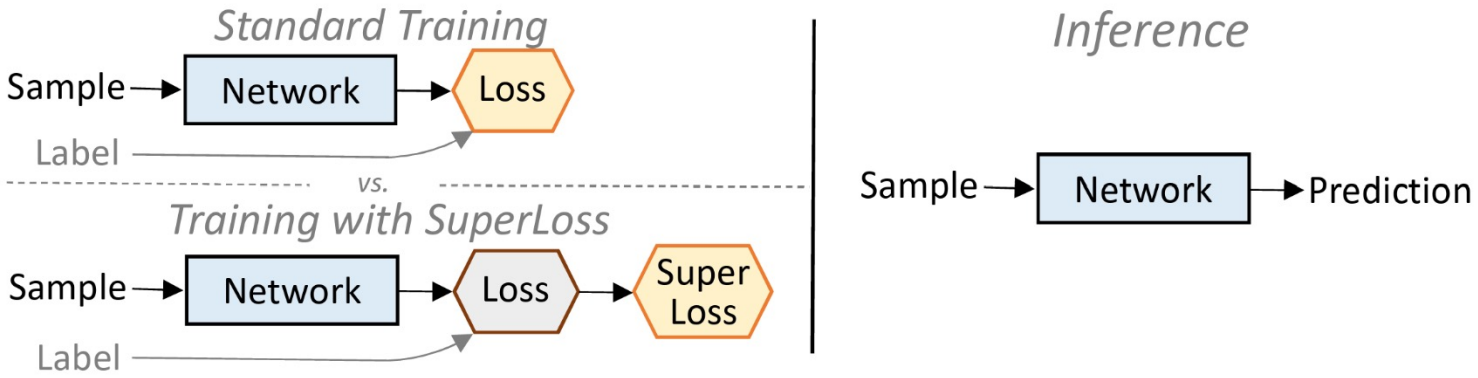
Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. ICLR 2018.

Self-Paced Learning: Variant based on $g(\cdot)$

Superloss

- Previous approaches are usually limited to a specific task and require extra data annotations, layers or parameters as well as a dedicated training procedure.
- Superloss proposes a generic loss function on top of any existing task loss:

$$L_\lambda(l_i, \sigma_i) = (l_i - \tau)\sigma_i + \lambda(\log \sigma_i)^2.$$



SuperLoss: A Generic Loss for Robust Curriculum Learning. NeurIPS 2020.

Self-Paced Learning: Variant based on $g(\cdot)$

Superloss

$$L_\lambda(l_i, \sigma_i) = (l_i - \tau)\sigma_i + \lambda(\log \sigma_i)^2.$$

- The equation seems not consistent with self-paced learning, but we can re-write it as:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda, \gamma) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda \sum_{i=1}^{N} v_i + \gamma(\log v_i)^2.$$

- The same color represents the same variable. The term $\gamma(\log v_i)^2$ can force the data weights v_i to approach 1.
- The solution is:

$$v_i = \begin{cases} e, & l_i - \lambda \leq -2\gamma/e \\ e^{-\mathcal{W}\left(\frac{l_i - \lambda}{2\gamma}\right)}, & l_i - \lambda > -2\gamma/e \end{cases}$$

- where $\mathcal{W}$ refers to Lambert W function.

SuperLoss: A Generic Loss for Robust Curriculum Learning. NeurIPS 2020.

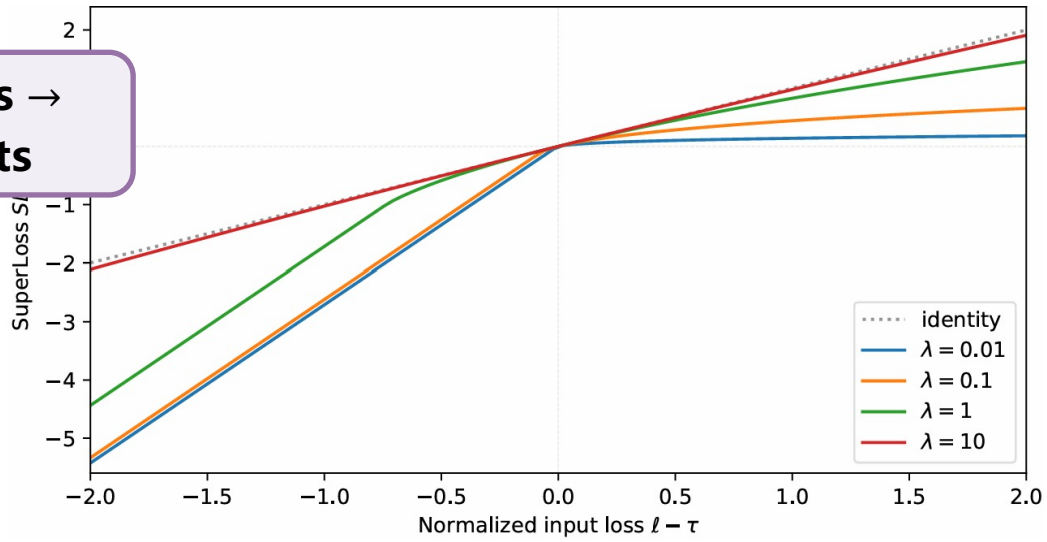
Self-Paced Learning: Variant based on $g(\cdot)$

Superloss

$$\min_{\boldsymbol{w};\,\boldsymbol{v}} \mathbb{E}\left(\boldsymbol{w}, \boldsymbol{v}; \lambda, \gamma\right) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) - \lambda \sum_{i=1}^{N} v_i + \gamma (\log v_i)^2.$$

$$v_i^* = \begin{cases} e, & l_i - \lambda \leq -2\gamma/e \\ e^{-\mathcal{W}\left(\frac{l_i - \lambda}{2\gamma}\right)}, & l_i - \lambda > -2\gamma/e \end{cases}$$

Steeper slopes → Large weights



Smoother slopes → Small weights

SuperLoss: A Generic Loss for Robust Curriculum Learning. NeurIPS 2020.

Self-Paced Learning: Variant based on $g(\cdot)$

Self-Paced Learning Implicit Regularizer (SPL-IR)

- SPL via **Explicit** Regularizers:

$$\min_{\boldsymbol{w};\,\boldsymbol{v}} \mathbb{E}\,(\boldsymbol{w},\boldsymbol{v};\lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + g(\boldsymbol{v};\lambda)$$

Known Regularizer g

Regularizers	$g(\boldsymbol{v};\lambda)$	
Hard [54]	$-\lambda \sum_{i=1}^{N} v_i$	
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]	$\frac{\gamma^2}{v_i + \frac{\gamma}{\lambda}}, \quad \gamma > 0$	$\begin{cases} 1, & l_i \le \left(\frac{\lambda\gamma}{\lambda+\gamma}\right)^2 \\ 0, & l_i \ge \lambda^2 \\ \gamma\left(\frac{1}{\sqrt{l_i}} - \frac{1}{\lambda}\right), & \text{otherwise} \end{cases}$

- SPL via **Implicit** Regularizers (SPL-IR):

$$\min_{\boldsymbol{w};\,\boldsymbol{v}} \mathbb{E}\,(\boldsymbol{w},\boldsymbol{v};\lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + \varphi(\boldsymbol{v};\lambda)$$

Unknown Regularizer φ

$$v_i^* = \sigma(\lambda, l_i) = \arg\min_{\boldsymbol{v}} \mathbb{E}(\boldsymbol{w},\boldsymbol{v};\lambda)$$

Known $\boldsymbol{v}^* = \boldsymbol{\sigma}$

	Huber	Cauchy	L1-L2	Welsch
Loss function $\phi(\lambda, t)$	$\begin{cases} t^2/2, & \lvert t\rvert \le \lambda \\ \lambda\lvert t\rvert - \frac{\lambda^2}{2}, & \lvert t\rvert > \lambda \end{cases}$	$\lambda^2 \log(1 + (t/\lambda)^2)$	$\sqrt{\lambda + t^2} - 1$	$\lambda^2(1 - \exp(-\frac{t^2}{\lambda^2}))$
Minimizer function $\sigma(\lambda, t)$	$\begin{cases} 1 & \lvert t\rvert \le \lambda \\ \lambda/\lvert t\rvert, & \lvert t\rvert > \lambda \end{cases}$	$2/(1 + (t/\lambda)^2)$	$1/\sqrt{\lambda + t^2}$	$2\exp(-\frac{t^2}{\lambda^2})$

Self-Paced Learning: An Implicit Regularization Perspective. AAAI 2017.

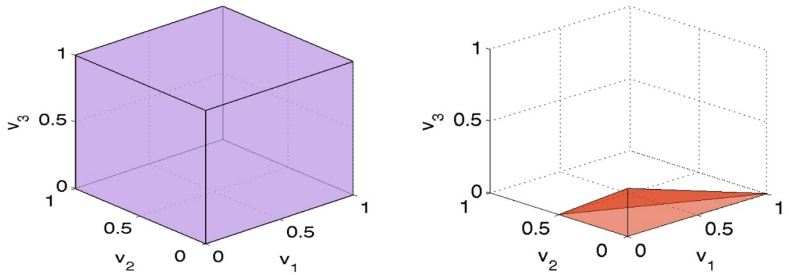
Self-Paced Learning: Variant based on v

Self-Paced Curriculum Learning (SPCL)

- SPL is unable to deal with prior knowledge, rendering it prone to overfitting.
- SPCL bridges the missing link between (predefined) CL and SPL:

$$\min_{\boldsymbol{w};\, \boldsymbol{v}\in[0,1]^N} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + g(\boldsymbol{v}; \lambda), s.t. \boldsymbol{v} \in \psi$$

  - ψ can be derived from predefined curriculum.
  - $g(\boldsymbol{v}; \lambda)$ generalizes the scheme of "self-paced functions".



(a) SPL (b) SPCL

(Predefined) CL	SPL	Proposed SPCL
Comparable to human learning	Instructor-driven	Student-driven	Instructor-student collaborative
Curriculum design	Prior knowledge	Learning objective	Learning objective + prior knowledge
Learning schemes	Multiple	Single	Multiple
Iterative training	Heuristic approach	Gradient-based	Gradient-based

Self-paced Curriculum Learning. AAAI 2015.

Self-Paced Learning: Variant based on v

Curriculum Learning by Dynamic Instance Hardness (DIHCL)

- Instantaneous hardness does not take the training history of each sample into account.
- DIHCL introduces "dynamic instance hardness (DIH)" as the **exponential moving average** of an instantaneous hardness measure of a sample over time:

$$r_{t+1}(i) = \begin{cases} \gamma \times a_t(i) + (1-\gamma) \times r_t(i) & if\ i \in S_t, \\ r_t(i) & otherwise, \end{cases}$$

- where $\gamma \in [0,1]$ is a discount factor, $S_t \subseteq \mathcal{D}$ is the subset selected in epoch t, $r_t(i)$ is the dynamic hardness, $a_t(i)$ is the instantaneous one and considered as the following ones:

$$a_t(i) \coloneqq \mathcal{L}\left(f_{\boldsymbol{w}_t}(x_i, y_i)\right)$$

$$a_t(i) \coloneqq \left|\mathcal{L}\left(f_{\boldsymbol{w}_t}(x_i, y_i)\right) - \mathcal{L}\left(f_{\boldsymbol{w}_{t-1}}(x_i, y_i)\right)\right|$$

$$a_t(i) \coloneqq \left|\mathbb{I}\left(\hat{y}_i^t = y_i\right) - \mathbb{I}\left(\hat{y}_i^{t-1} = y_i\right)\right|$$

Curriculum Learning by Dynamic Instance Hardness. NeurIPS 2020.

Self-Paced Learning: Variant based on v

Data parameters (DCL)

- The curriculum in the majority of early work is provided by a pre-determined heuristic.
- DCL equips each sample and class in a dataset with a learnable parameter:
  - For class parameters:

$$l_i = -\log(p^i_{y^i})$$

$$p^i_{y^i} = \frac{\exp\left(z^i_{y^i}/\sigma^{class}_{y^i}\right)}{\sum_j \exp\left(z^i_j/\sigma^{class}_{y^i}\right)}$$

  - For instance parameters:

$$p^i_{y^i} = \frac{\exp\left(z^i_{y^i}/\sigma^{inst}_i\right)}{\sum_j \exp\left(z^i_j/\sigma^{inst}_i\right)}$$

  - For data parameters:

$$\sigma^*_i = \sigma^{class}_{y^i} + \sigma^{inst}_i$$

Data Parameters: A New Family of Parameters for Learning a Differentiable Curriculum. NeurIPS 2019.

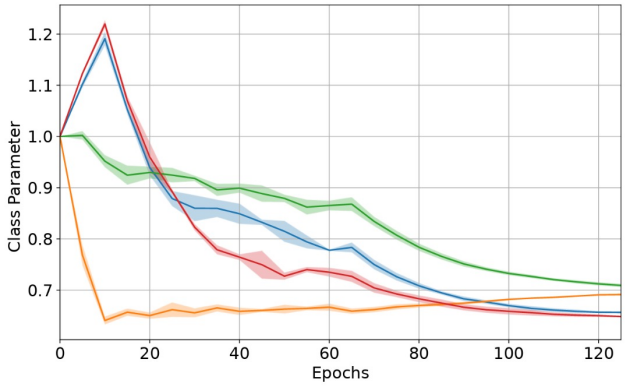
Self-Paced Learning: Variant based on v

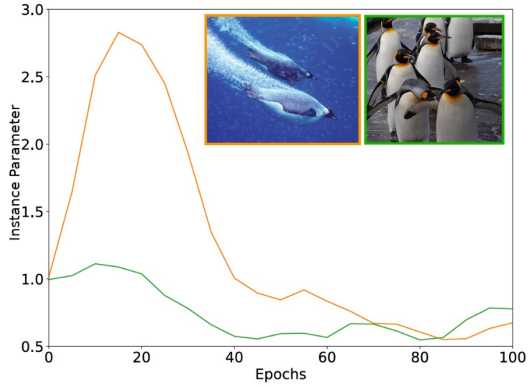
Data parameters (DCL)

- The gradients of the cross-entropy loss with respect to the data parameters are:

$$\frac{\partial l_i}{\partial \sigma_i^*} = \frac{\left(1 - p_{y_i}^i\right)}{\left(\sigma_i^*\right)^2}\left(z_{y_i}^i - \sum_{j \neq y_i} q_j^i z_j^i\right)$$

- The data parameter σ_i^* is updated with the negative of this gradient, where it will increase if the logit of the target class is less than the expected value of logits on non-target-classes (i.e., $z_{y_i}^i < \sum_{j \neq y_i} q_j^i z_j^i$) and vice-versa, consistent with the idea of curriculum.





Dataset	Model	Baseline	DCL	Class	Instance
CIFAR100	WRN-28-10	80.1 ± 0.2	$\mathbf{80.8 \pm 0.1}$	✓	✓
ImageNet	ResNet18	70.3 ± 0.1	$\mathbf{71.0 \pm 0.1}$	✓	✓
			70.8 ± 0.1	✓	✗
WebVision	ResNet18	66.3 ± 0.1	$\mathbf{67.5 \pm 0.1}$	✓	✓
			67.1 ± 0.1	✗	✓

Data Parameters: A New Family of Parameters for Learning a Differentiable Curriculum. NeurIPS 2019.

Self-Paced Learning: Variant based on v

ScreenerNet

- Selecting and ordering samples via hand-crafted rules can result in sampling bias.
- This work learns a weight for each sample by jointly training the ScreenerNet and the main network in an end-to-end self-paced fashion:

$$w_{\boldsymbol{x}} = \arg\min_{w_{\boldsymbol{x}}} \sum_{x \in X} \left((1 - w_{\boldsymbol{x}})^2 e_{\boldsymbol{x}} \right) + w_{\boldsymbol{x}}^2 \max(M - e_{\boldsymbol{x}}, 0) + \alpha \sum_{p \in W_S} \|p\|_1 ,$$

- Where $w_{\boldsymbol{x}} = \boldsymbol{S}(\boldsymbol{x})$, $e_{\boldsymbol{x}} = \mathcal{L}_F(F(x), t_x)$, $\boldsymbol{W}_S$ is parameters of the ScreenerNet $\boldsymbol{S}$, and α is a balancing hyper-parameter for the regularizer of the ScreenerNet. $\boldsymbol{X}$ is a set of training images and M is a margin hyper-parameter.

ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks. ArXiv.

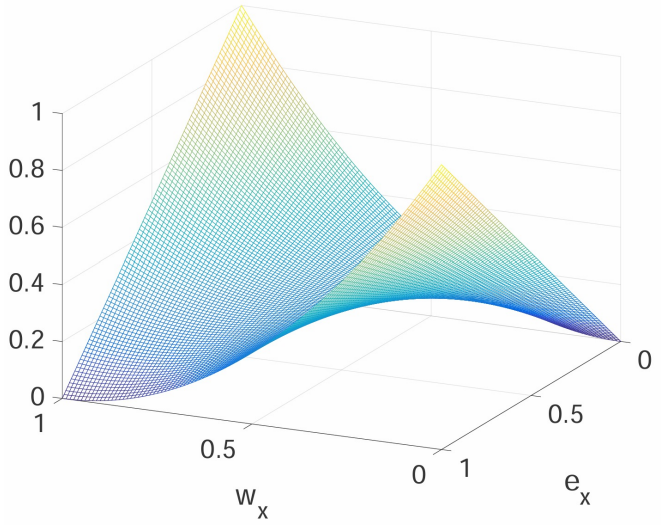
Self-Paced Learning: Variant based on v

ScreenerNet

$$w_{\boldsymbol{x}} = \arg\min_{w_{\boldsymbol{x}}} \sum_{x \in X} \left((1 - w_{\boldsymbol{x}})^2 e_{\boldsymbol{x}}\right) + w_{\boldsymbol{x}}^2 \max(M - e_{\boldsymbol{x}}, 0) + \alpha \sum_{p \in W_S} \|p\|_1 ,$$

- To keep it consistent with the previous method, we rewrite it:

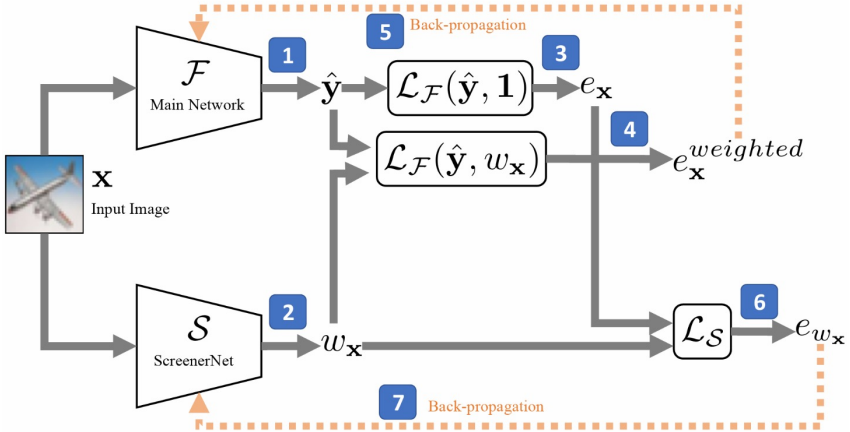
$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \sum_{i=1}^{N} \left((1 - v_i)^2 \mathcal{L}\big(h_{\boldsymbol{\Theta}}(x_i, y_i)\big) + v_i^2 \max\big(M - \mathcal{L}\big(h_{\boldsymbol{\Theta}}(x_i, y_i)\big), 0\big)\right),$$



- where M is a margin hyper-parameter, and h_Θ is the ScreenerNet h with parameters $\boldsymbol{\Theta}$.

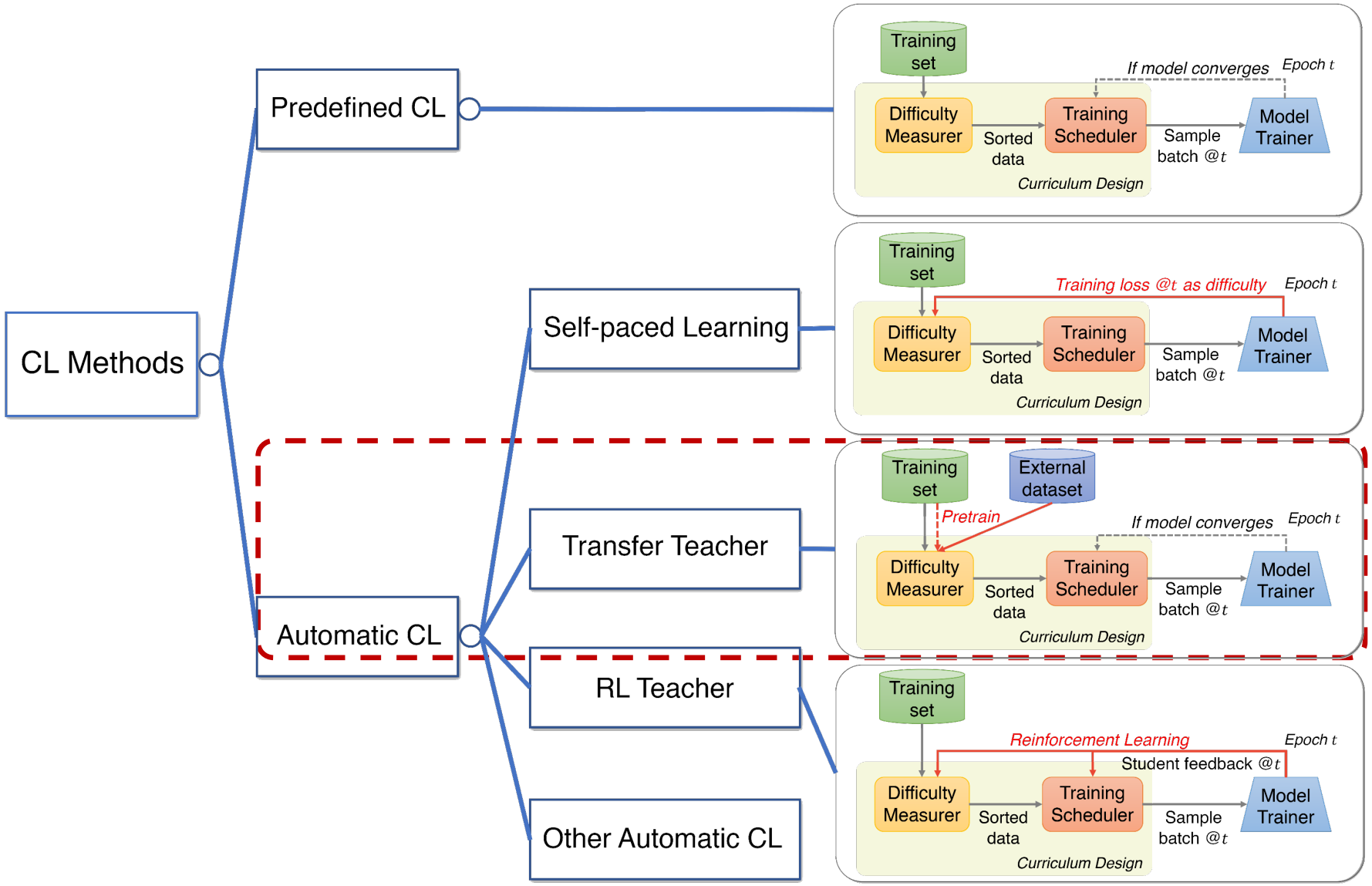
Algorithm 1: Training a network with ScreenerNet

Given: training samples $(\mathbf{X}, t_{\mathbf{X}})$, main network $\mathcal{F}$, ScreenerNet $\mathcal{S}$
Initialize $\mathcal{F}$, $\mathcal{S}$
for *each iteration of the training* **do**
  $\mathbf{x} \leftarrow$ Sample($\mathbf{X}$). /* sample a mini-batch from $\mathbf{X}$ */
  $w_{\mathbf{x}} \leftarrow \mathcal{S}(\mathbf{x})$. /* predict an weight of a sample */
  $\hat{\mathbf{y}} \leftarrow \mathcal{F}(\mathbf{x})$. /* prediction from the main network */
  $e_{\mathbf{x}}^{weighted} \leftarrow \mathcal{L}_{\mathcal{F}}(\hat{\mathbf{y}}, w_{\mathbf{x}})$. /* compute a weighted error */
  $e_{\mathbf{x}} \leftarrow \mathcal{L}_{\mathcal{F}}(\hat{\mathbf{y}}, \mathbf{1})$. /* compute an error of the sample */
  UpdateNetwork($\mathcal{F}$, $e_{\mathbf{x}}^{w}$). /* train main network */
  UpdateNetwork($\mathcal{S}$, $e_{\mathbf{x}}$). /* train ScreenerNet */
end



ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks. ArXiv.

Automatic Curriculum: Transfer Teacher



Automatic Curriculum: Transfer Teacher

- Self-Paced Learning (SPL):

$$\min_{\boldsymbol{w};\,\boldsymbol{v}} \mathbb{E}\,(\boldsymbol{w}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i) + g(\boldsymbol{v}; \lambda)$$

- Transfer Teacher Curriculum Learning (TTCL):

$$\min_{\boldsymbol{w}} \mathbb{E}\,(\boldsymbol{w}) = \sum_{i=1}^{N} v_i \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i)$$

$$v_i = h_{\boldsymbol{\Theta}}(x_i, y_i)$$

- **SPL** measures the data difficulty by the **student model** itself, while TTCL calculate the instance-wise difficulty with the help of teacher model, which was pretrained on the training or an external dataset.

Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks. ICML 2018.
On The Power of Curriculum Learning in Training Deep Networks. ICML 2019.

Automatic Curriculum: Transfer Teacher

Common Transfer Teacher Methods:

- Just like SPL, TTCL can be implemented via hard selection or soft reweighting:

- Selection by Ranking

Algorithm 1 Curriculum learning method

Input: *pacing function* g_ϑ, *scoring function* f, data $\mathbb{X}$.
Output: sequence of mini-batches $\left[\mathbb{B}_1^{'}, ..., \mathbb{B}_M^{'}\right]$.
sort $\mathbb{X}$ according to f, in ascending order
$result \leftarrow []$
for all $i = 1, ..., M$ **do**
  $size \leftarrow g_\vartheta(i)$
  $\mathbb{X}_i^{'} \leftarrow \mathbb{X}[1, ..., size]$
  uniformly sample $\mathbb{B}_i^{'}$ from $\mathbb{X}^{'}$
  append $\mathbb{B}_i^{'}$ to $result$
end for
return $result$

- Reweighting by Scoring

Algorithm 1 Curriculum learning method

Input: ~~*pacing function* g_ϑ~~, *scoring function* f, data $\mathbb{X}$.
Output: sequence of mini-batches $\left[\mathbb{B}_1^{'}, ..., \mathbb{B}_M^{'}\right]$.
~~sort $\mathbb{X}$ according to f, in ascending order~~ $\boldsymbol{v} \leftarrow f(\mathbb{X})$
$result \leftarrow []$
for all $i = 1, ..., M$ **do**
  ~~$size \leftarrow g_\vartheta(i)$~~
  ~~$\mathbb{X}_i^{'} \leftarrow \mathbb{X}[1, ..., size]$~~ $\mathbb{X}' \leftarrow \{\mathbb{X}, \boldsymbol{v}\}$
  uniformly sample $\mathbb{B}_i^{'}$ from $\mathbb{X}^{'}$
  append $\mathbb{B}_i^{'}$ to $result$
end for
return $result$

Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks. ICML 2018.
On The Power of Curriculum Learning in Training Deep Networks. ICML 2019.

Automatic Curriculum: Transfer Teacher

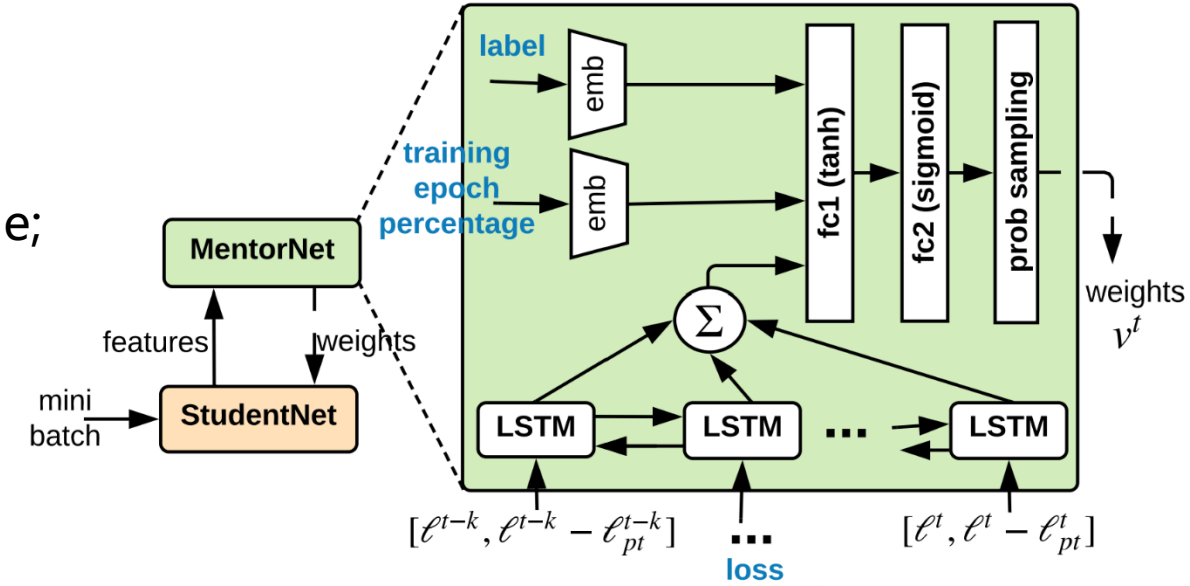
MentorNet

- Existing curriculum that is usually predefined by human experts.
- This work proposes to teach StudentNet by a MentorNet:

$$\min_{\boldsymbol{w},\boldsymbol{v}} \mathbb{E}(\boldsymbol{w}, \boldsymbol{\Theta}; \lambda) = \sum_{i=1}^{N} h(\boldsymbol{z}_i, \boldsymbol{\Theta}) \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i)$$

  - MentorNet h has parameters $\boldsymbol{\Theta}$;
  - $\boldsymbol{z}_i = \phi(x_i, y_i, \boldsymbol{w})$ is the feature of the i-th sample;
  - Given fixed $\boldsymbol{w}, \boldsymbol{\Theta}$ is learned on a trusted small dataset by minimizing cross-entropy with v_i^* ($v_i^* = 1$ iff y_i is a correct label):

$$\Theta^* = \underset{\Theta}{argmin} \sum_{i=1}^{M} CE(h(\boldsymbol{z}_i; \Theta), v_i^*)$$



l^t: loss at t, l_{pt}^t: moving average loss

Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. ICML 2018.

Automatic Curriculum: Transfer Teacher

Adaptive Curriculum Learning (Adaptive CL)

- A fixed curriculum ignores the fact that a proper curriculum should be adjusted according to the learning progress of every data instance.
- This work proposes to adapt the difficulty of examples given by the teacher model to the current state of the student model:
  - Objective function: The original loss function (like cross-entropy) and the KL divergence between the output of the pre-trained model h and the current model f:

$$\mathcal{L} = \mathcal{L}_{XE} + \lambda \mathcal{L}_{KL}$$

$$= -\sum_{i=1}^{N} y_i \log f_{\boldsymbol{w}}(x_i) - \sum_{i=1}^{N} h_{\boldsymbol{\Theta}}(x_i) \log f_{\boldsymbol{w}}(x_i)$$

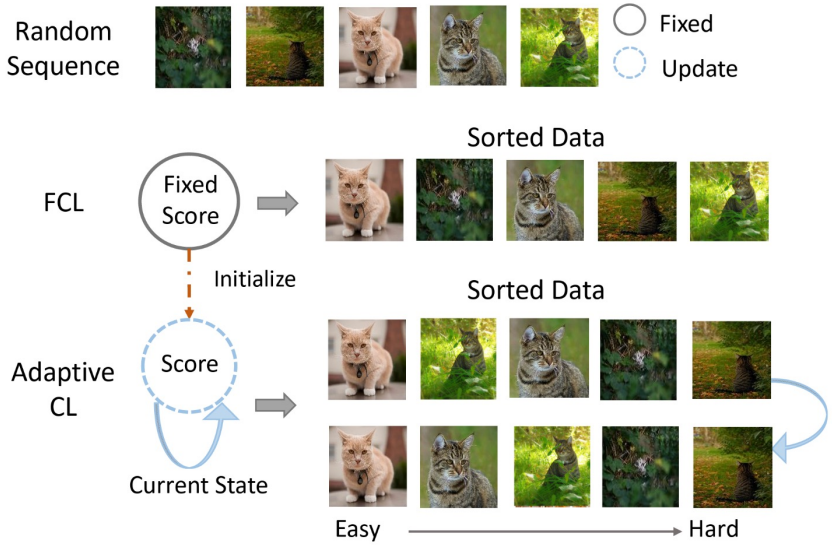
  - Pacing function: Exponential pacing function.

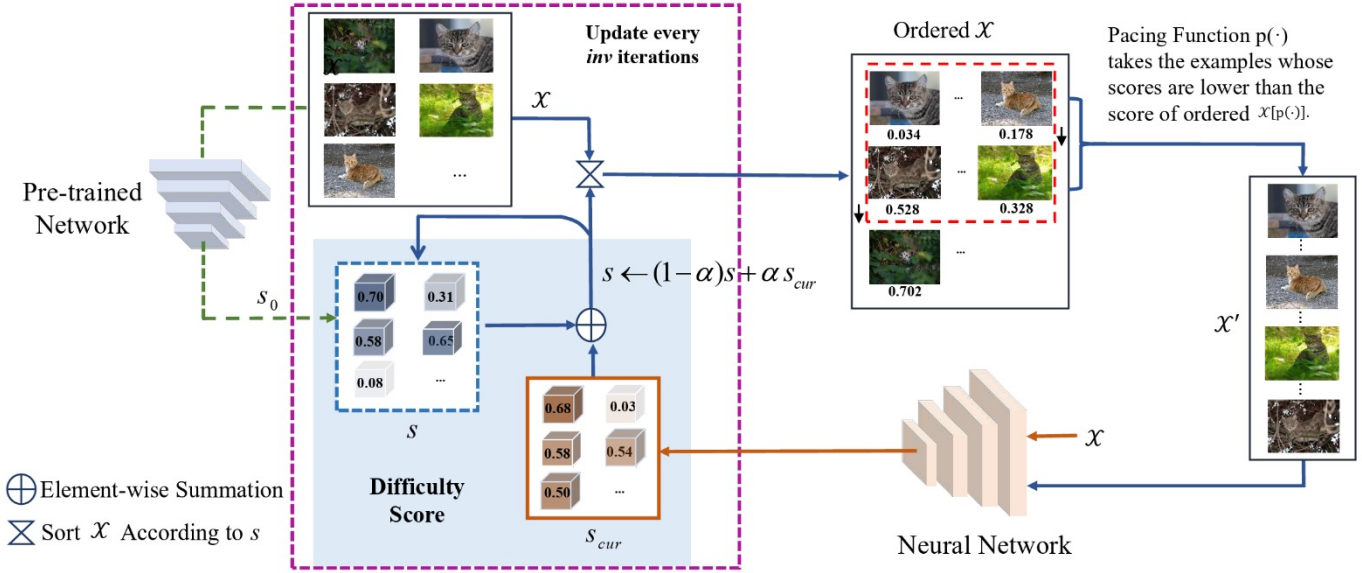
Adaptive Curriculum Learning. ICCV 2021.

Automatic Curriculum: Transfer Teacher

Adaptive Curriculum Learning (Adaptive CL)

- A fixed curriculum ignores the fact that a proper curriculum should be adjusted according to the learning progress of every data instance.
- This work proposes to adapt the difficulty of data given by the teacher model to the current state of the student model.





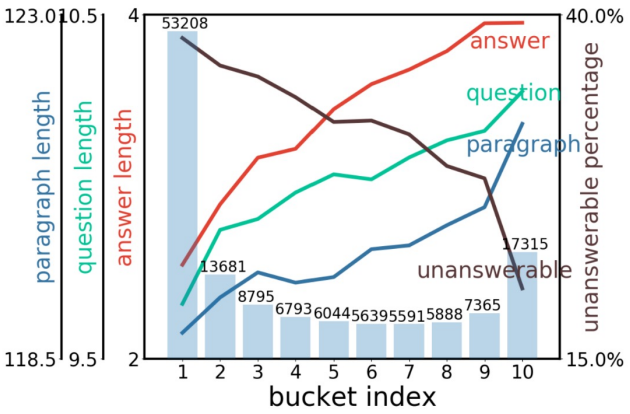
Adaptive Curriculum Learning. ICCV 2021.

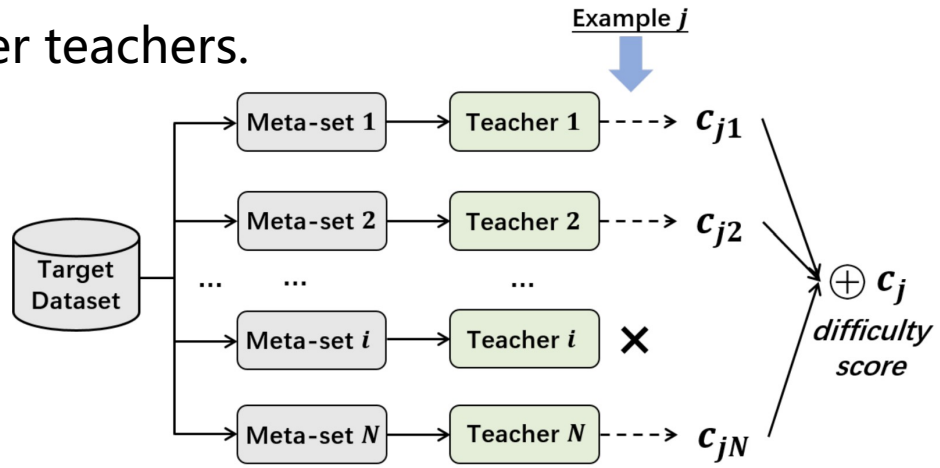
Automatic Curriculum: Transfer Teacher

Cross Review

- Even in the same task, the training data could vary significantly in difficulty levels.
- This work proposes to distinguish easy data from difficult data, and arrange a curriculum for language models by reviewing the trainset in a crossed way:
  - Uniformly divide trainset into N shares, which is called “Meta-set”.
  - Train N teachers based on each “Meta-set” for crosses reviewing.
  - Calculate difficulty score of each sample based on all other teachers.

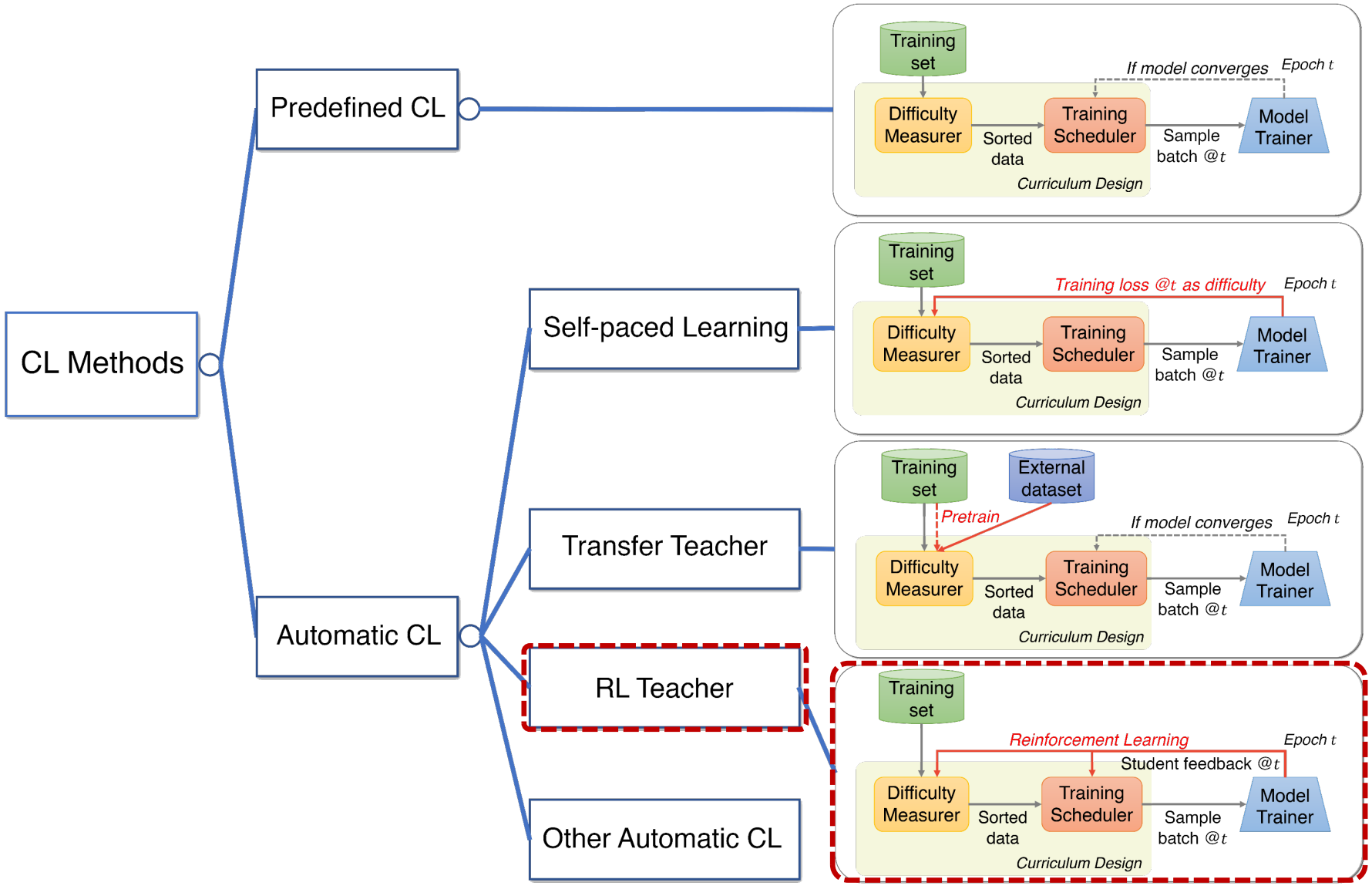
Easy cases:	
easy, comfortable	positive
most purely enjoyable	positive
most plain, unimaginative	negative
badly edited	negative
Hard cases:	
why didn’t Hollywood think of this sooner	positive
I simply can’t recommend it enough	positive
supposedly funny movie	negative
occasionally interesting	negative





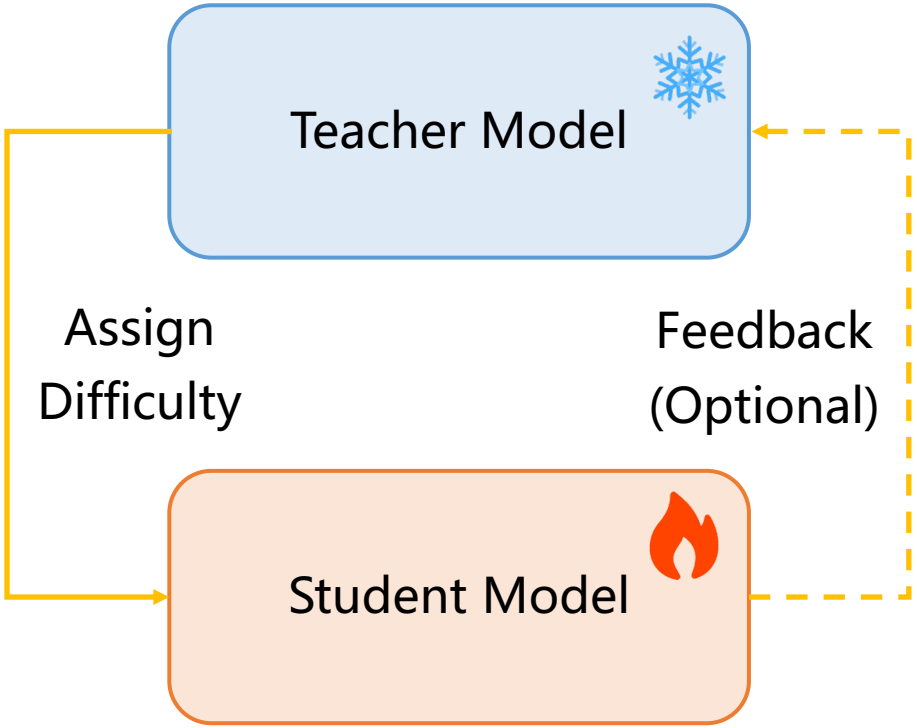
Curriculum Learning for Natural Language Understanding. ACL 2020.

Automatic Curriculum: RL Teacher

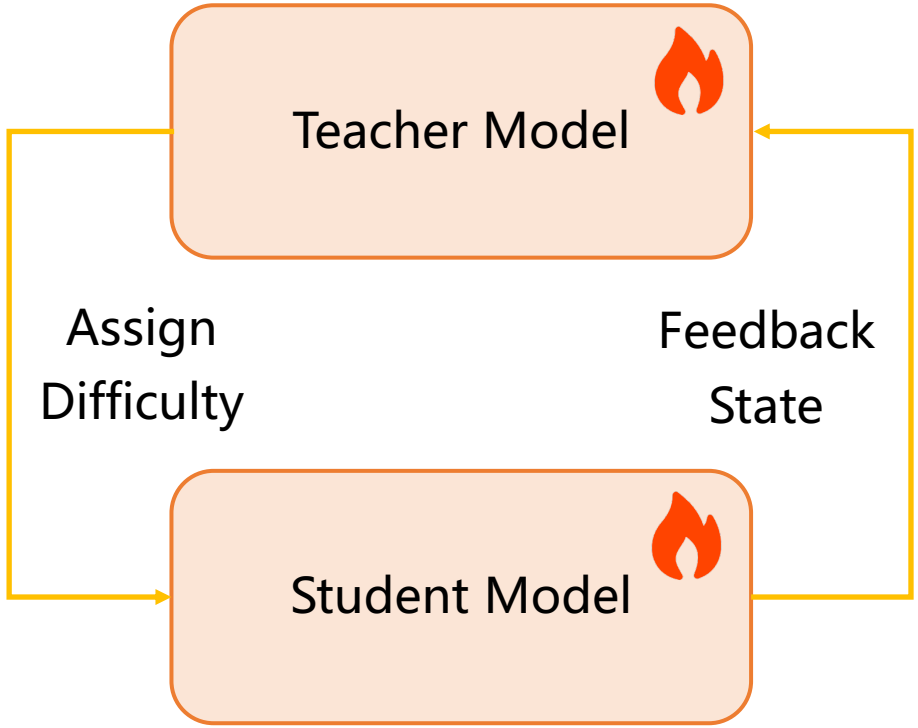


Automatic Curriculum: RL Teacher

- Transfer Teacher



- Reinforcement Learning Teacher



Automatic Curriculum: RL Teacher

Intrinsically Motivated Curriculum Learning

- Hand-chosen curriculum introduces hard-to-tune parameters and unknown or variable thresholds across tasks.
- This work proposes to treat the decision about which task to study next as a stochastic policy, adapted to optimize learning progress, e.g.,
  - Prediction gain:

$$\nu_{PG} := \mathcal{L}(x, \theta) - \mathcal{L}(x, \theta')$$

  - Self prediction gain:

$$\nu_{SPG} := \mathcal{L}(x', \theta) - \mathcal{L}(x', \theta')$$

  - Variational Complexity Gain

$$\nu_{VCG} := KL(P_{\theta'} || Q_{\psi'}) - KL(P_{\theta} || Q_{\psi})$$

Algorithm 1 Intrinsically Motivated Curriculum Learning

Initially: $w_i = 0$ for $i \in [N]$

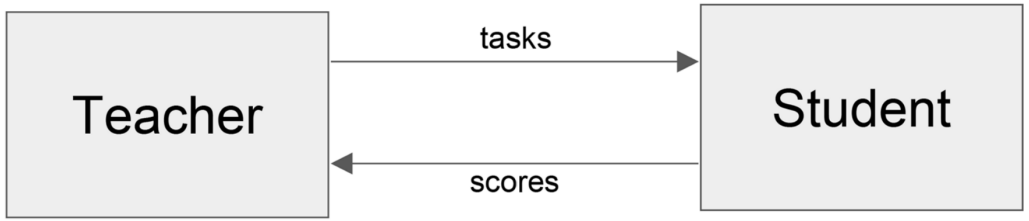
for $t = 1 \ldots T$ **do**
  $\pi(k) := (1-\epsilon)\frac{e^{w_k}}{\sum_i e^{w_i}} + \frac{\epsilon}{N}$
  Draw task index k from π
  Draw training sample $\mathbf{x}$ from D_k
  Train network p_θ on $\mathbf{x}$
  Compute learning progress ν (Sections 3.1 & 3.2)
  Map $\hat{r} = \nu/\tau(\mathbf{x})$ to $r \in [-1, 1]$ (Section 2.3)
  Update w_i with reward r using Exp3.S (1)
end for

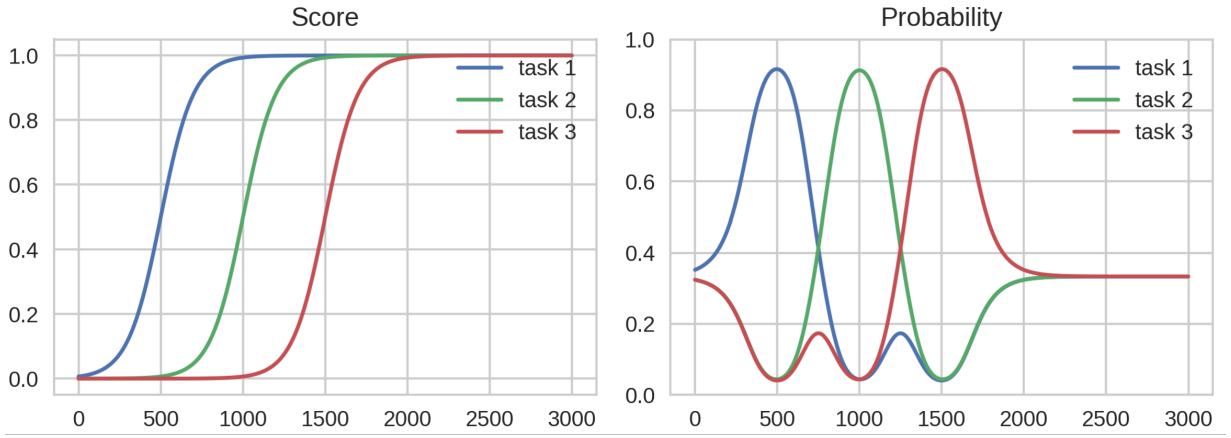
Automated curriculum learning for neural networks. ICML 2017.

Automatic Curriculum: RL Teacher

Teacher-Student Curriculum

- Tasks with sparse rewards remain challenging to solve, for the number of samples needed increases exponentially with the number of steps to get a reward.
- The main idea is that the student should practice more the tasks on which it is making fastest progress and the tasks where the performance is getting worse:
  - The learning curve slope is highest.
  - The learning curve slope is negative.





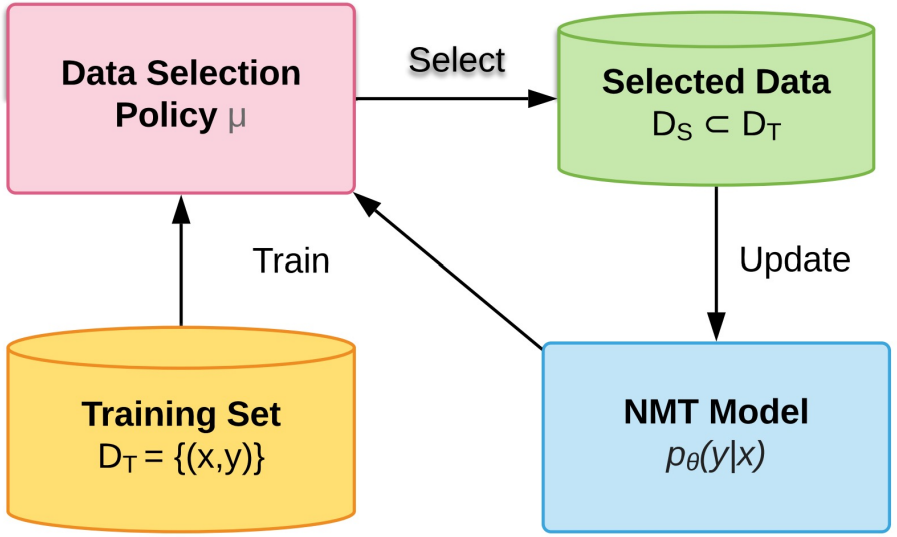
Teacher-Student Curriculum Learning. NeurIPS 2017.

Automatic Curriculum: RL Teacher

Reinforced Curriculum on Pre-trained Model

- Existing curriculum methods require training from scratch, while in practice most NMT models are pre-trained on big data already.
- This work designs a curriculum to improve pretrained model by re-selecting influential data samples from the original training set and formulate this task as a RL problem:

Example 1	
zh	xianzai ta zheng kaolv zhe huijia.
zh-gloss	Now he is thinking about going home.
en	He is thinking about going home now.
Example 2	
zh	wo yao chi niupai.
zh-gloss	I want eat steak
en	I want a steak. Get me a coke.

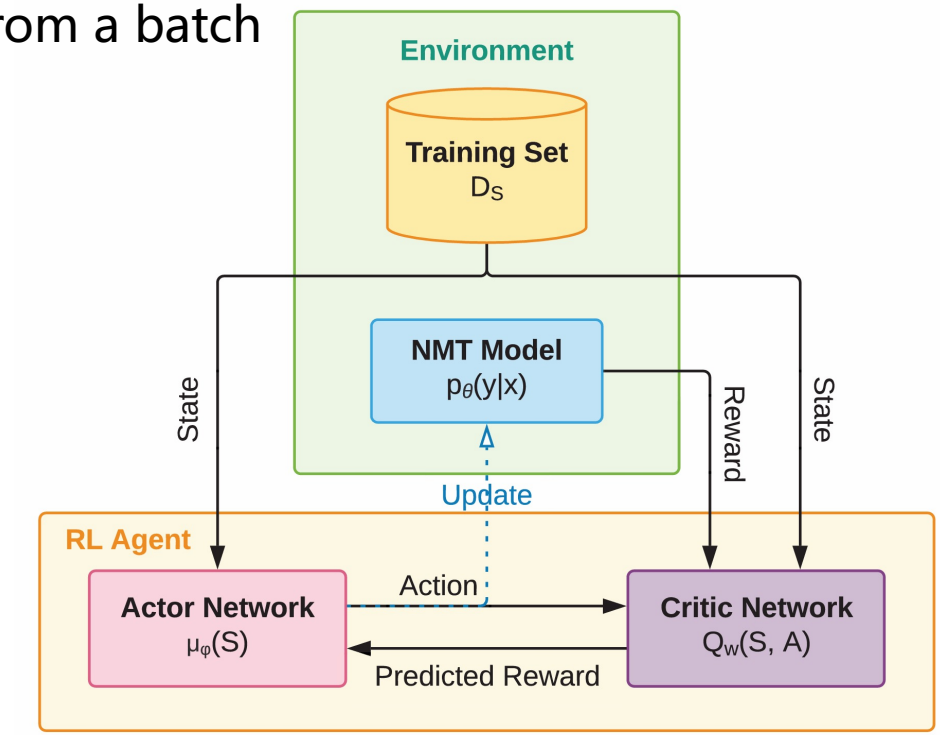


Reinforced Curriculum Learning on Pre-trained Neural Machine Translation Models. ACL 2020.

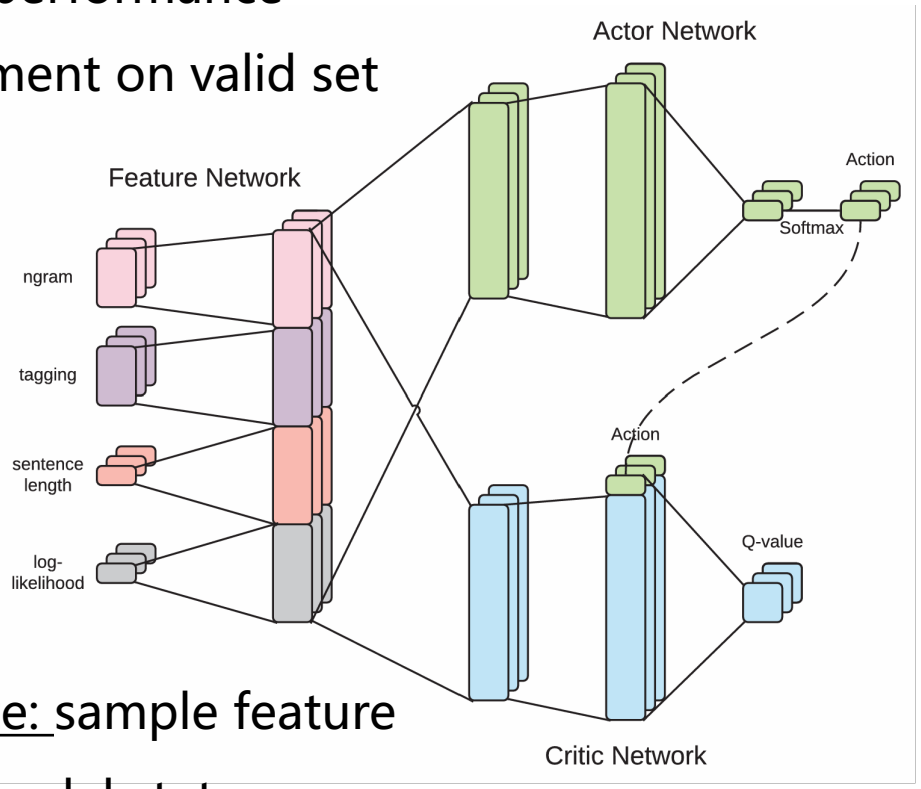
Automatic Curriculum: RL Teacher

Curriculum for Pre-trained Model

Action: select the best data samples from a batch



Reward: performance improvement on valid set



State: sample feature & model state

Reinforced Curriculum Learning on Pre-trained Neural Machine Translation Models. ACL 2020.

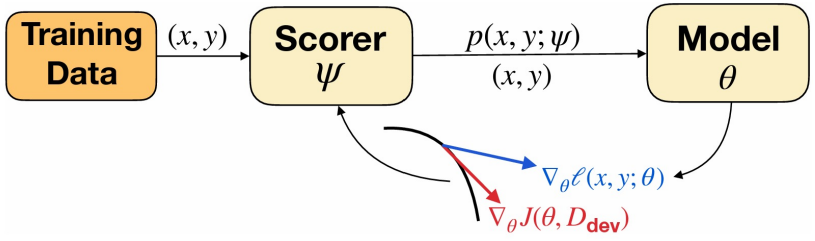
Automatic Curriculum: RL Teacher

Differentiable Data Selection (DDS)

- Single reward signal provided by dev set accuracy at the end of training can be noisy, which is not able to achieve results competitive with simpler heuristic training methods.
- This work proposes to use the gradient alignment between the training set and the dev set as a reward signal for a parametric scorer network:

$$\psi^* = \arg\min_{\psi} J(\theta^*(\psi), \mathcal{D}_{dev})$$

$$\theta^*(\psi) = \arg\min_{\theta} \mathbb{E}_{x,y\sim P(X,Y;\psi)}[\mathcal{L}(x, y; \theta)]$$

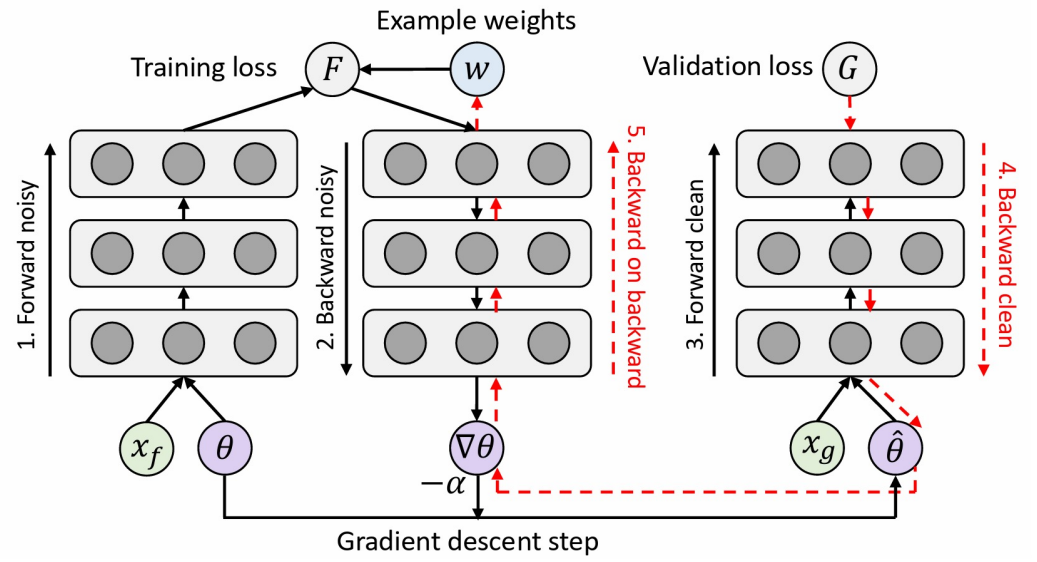


Optimizing Data Usage via Differentiable Rewards. ICML 2020.

Automatic Curriculum: RL Teacher

Learning to Reweight Examples (LRE)

- Data reweighting algorithms require careful tuning of additional hyperparameters.
- This work proposes a meta-learning algorithm that assigns weights to training data based on the gradient directions.



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$

Ensure: θ_T

for $t = 0 \ldots T-1$ **do**

$\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$

$\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$

$\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$

$\epsilon \leftarrow 0;\ l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$

$\hat{\theta}_t \leftarrow \theta_t - \alpha \nabla\theta_t$

$\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$

$l_g \leftarrow \frac{1}{m} \sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$

$\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$

$\tilde{w} \leftarrow \max(-\nabla\epsilon, 0);\ w \leftarrow \frac{\tilde{w}}{\sum_i \tilde{w} + \delta(\sum_i \tilde{w})}$

$\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$

$\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$

$\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$

end for

① Local update with perturbed example weights

② Calc. $\nabla\epsilon_i$ based on clean valid set, meta-update w_i

③ Use new w_i to meta-update model paras θ

Learning to reweight examples for robust deep learning. ICML 2018.

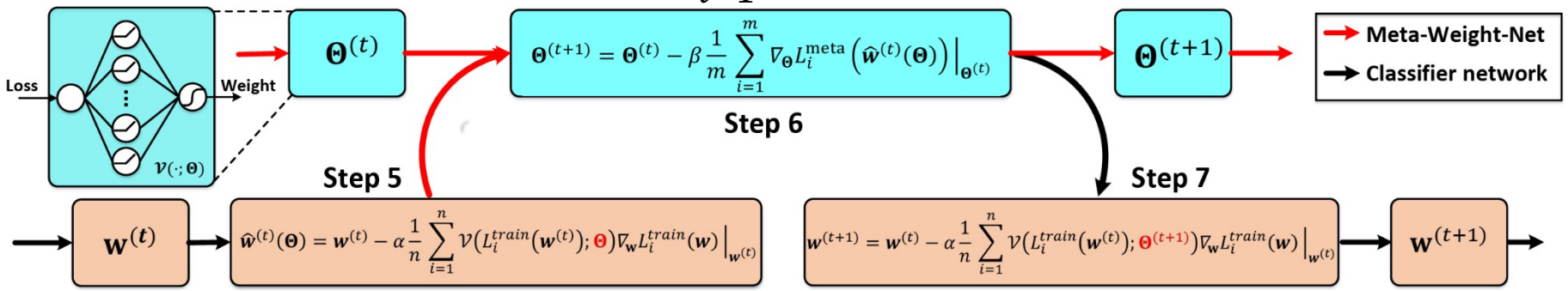
Automatic Curriculum: RL Teacher

Meta-Weight-Net (MW-Net)

- Current reweighing approaches need manually pre-specify the weighting function as well as its additional hyper-parameters.
- This work proposes to adaptively learn an explicit weighting function directly from data loss:

$$\min_{\boldsymbol{w},\boldsymbol{v}} \mathbb{E}(\boldsymbol{w}, \boldsymbol{\Theta}; \lambda) = \sum_{i=1}^{N} h(\mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i), \boldsymbol{\Theta}) \mathcal{L}(f_{\boldsymbol{w}}(x_i), y_i)$$

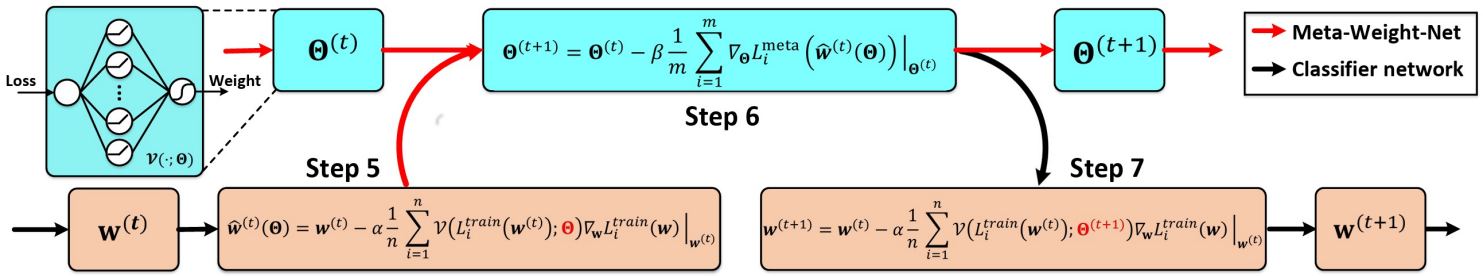
$$\Theta^* = \underset{\Theta}{argmin} \sum_{i=1}^{M} \mathcal{L}\left(f_{\boldsymbol{w}}\left(x_i^{(meta)}\right), y_i^{(meta)}\right)$$

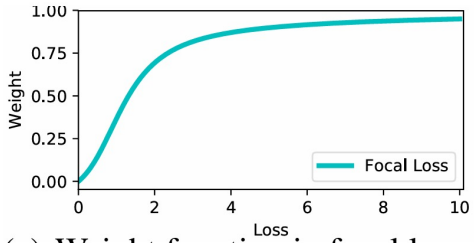


Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. NeurIPS 2019.

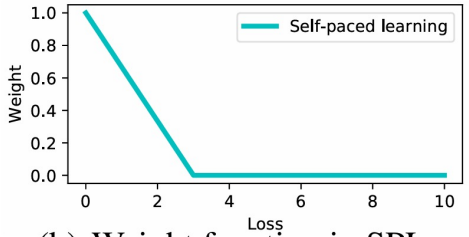
Automatic Curriculum: RL Teacher

Meta-Weight-Net (MW-Net)

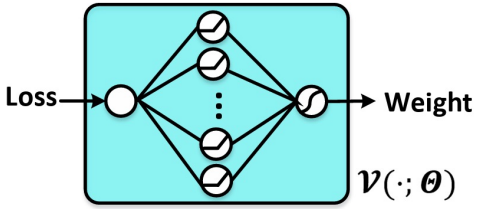




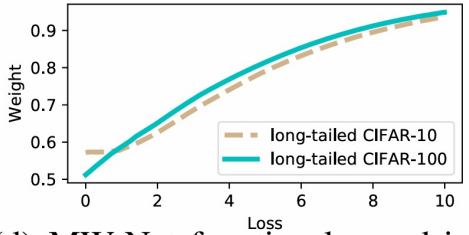
(a) Weight function in focal loss



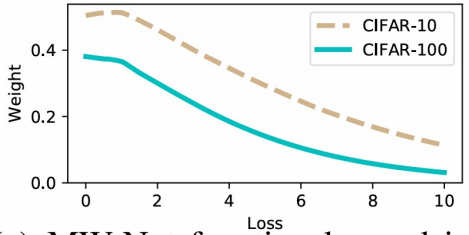
(b) Weight function in SPL



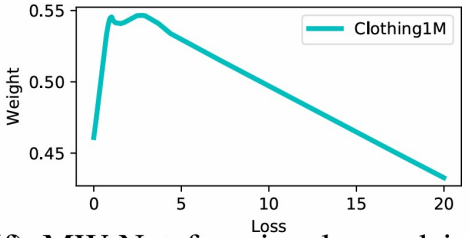
(c) Meta-Weight-Net architecture



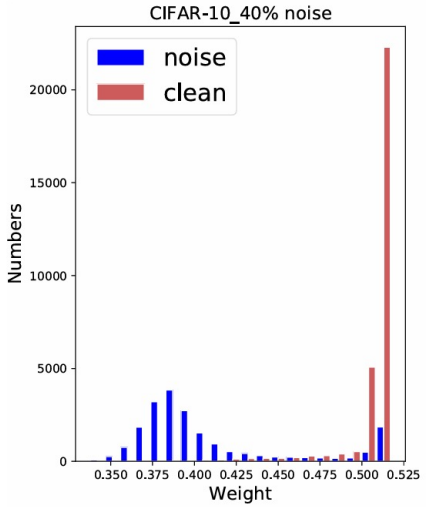
(d) MW-Net function learned in class imbalance case

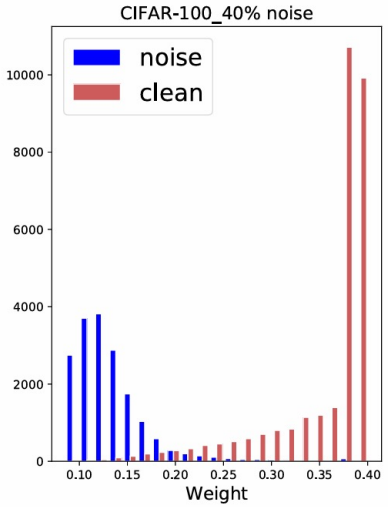


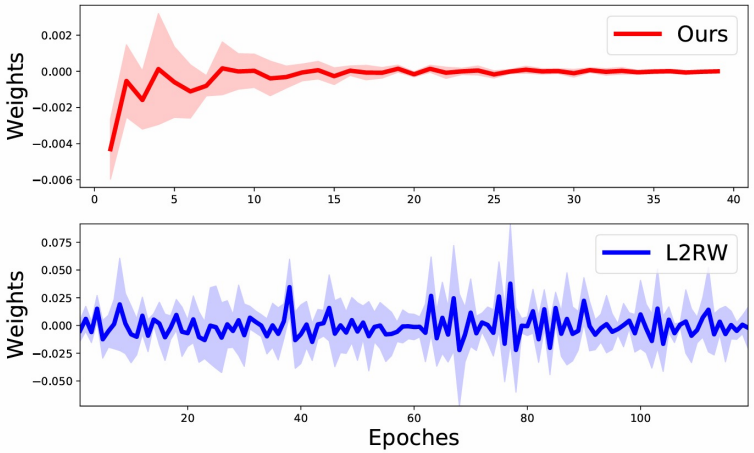
(e) MW-Net function learned in corrupter labels case



(f) MW-Net function learned in real Clothing1M dataset

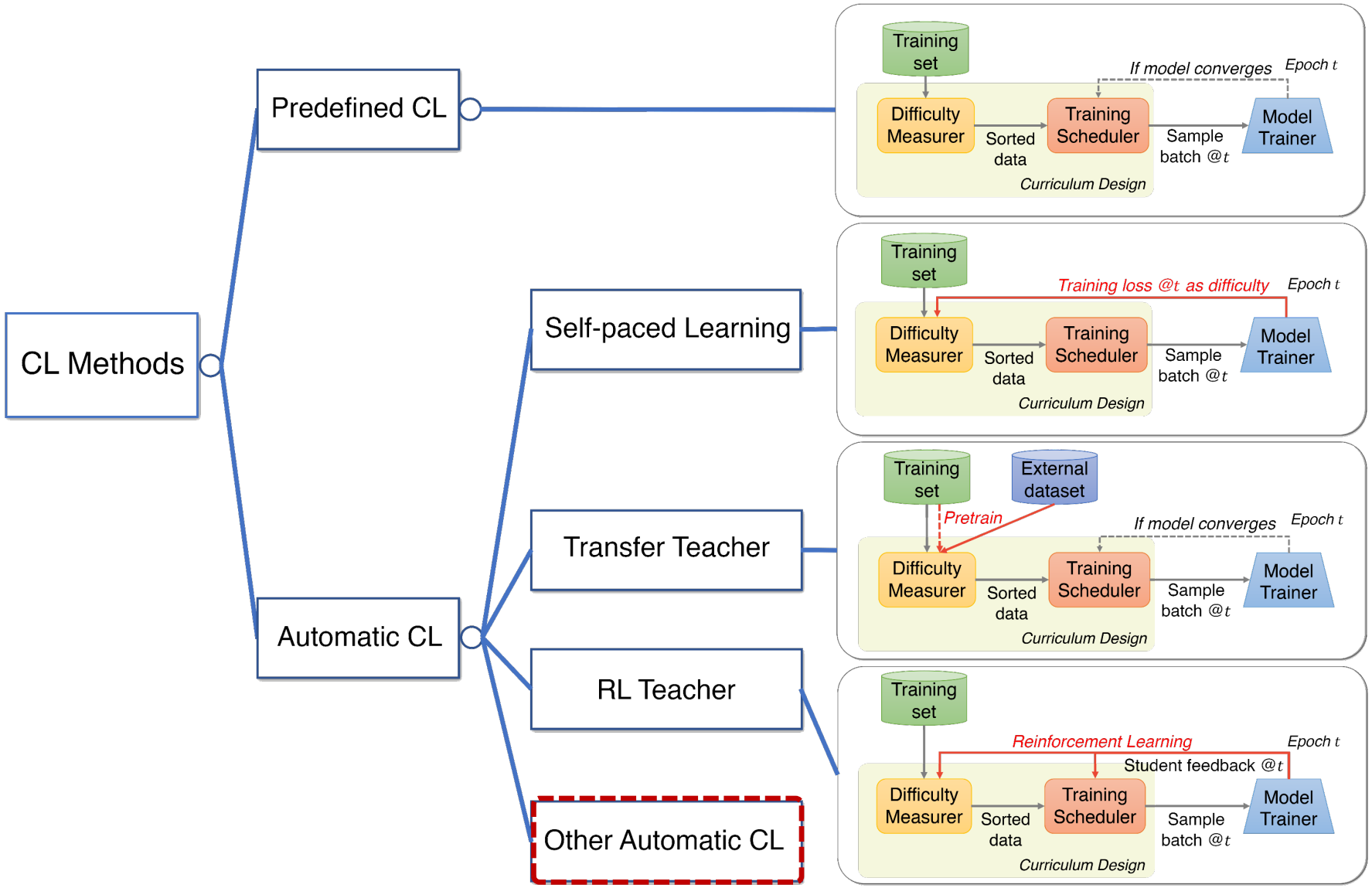






Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. NeurIPS 2019.

Automatic Curriculum: Others



Automatic Curriculum: Others

Recall: Generalized Curriculum Learning

- A curriculum is a sequence of training criteria over T training steps:

$$\mathcal{C} = \langle Q_1, \dots, Q_t, \dots, Q_T \rangle.$$

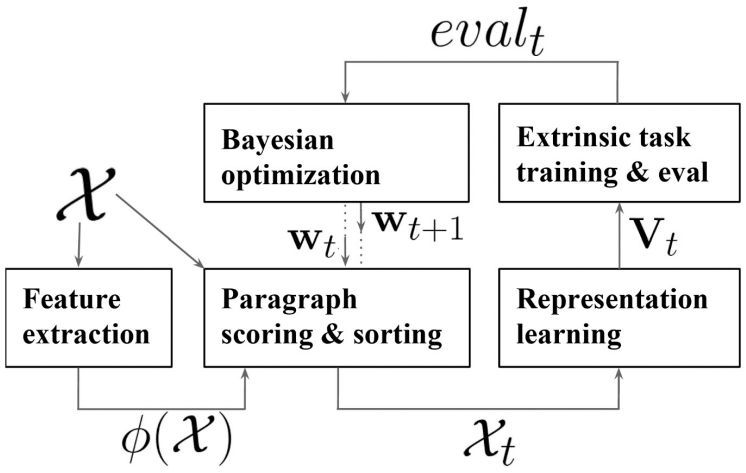
Model | **Data** | **Task**

Stage 1	Stage 2	Stage 3	… …	Stage T
Q_1	Q_2	Q_3	Q_t	Q_T

Others: Curriculum on Optimization

Bayesian Optimization (BO)

- Prior researches on curriculum in NLP relied too much on heuristics.
- This work uses Bayesian optimization by a linear ranking function to learn weight vectors that characterizes the different aspects of the complexity of each instance:
  - Acquisition function: Expected Improvement
  - Surrogate model: Tree-Structured Parzen Estimators



Algorithm 1 Bayesian optimization

$\mathcal{H} \leftarrow \emptyset$ ▷ Initialize observation history
$\mathcal{A} \leftarrow EI$ ▷ Initialize acquisition function
$\mathcal{S}_0 \leftarrow TPE$ ▷ Initialize surrogate model
for t $\leftarrow$ 1 to T **do**
  $\mathbf{w}_t \leftarrow \text{argmax}_{\mathbf{w}} \mathcal{A}(\mathbf{w}; \mathcal{S}_{t-1}, \mathcal{H})$ ▷ Predict $\mathbf{w}_t$ by optimizing acquisition function
  $eval(\mathbf{w}_t)$ ▷ Evaluate $\mathbf{w}_t$ on extrinsic task
  $\mathcal{H} \leftarrow \mathcal{H} \cup (\mathbf{w}_t, eval(\mathbf{w}_t))$ ▷ Update observation history
  Estimate $\mathcal{S}_t$ given $\mathcal{H}$
end for
return $\mathcal{H}$

Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning. ACL 2016.

Others: Curriculum on Loss Function

Dynamic Loss Functions (DLF)

- A good teacher not only provides his/her students with qualified teaching materials (e.g., data), but also sets up appropriate learning objectives (e.g., loss function).
- The main idea is that the loss function of a machine learning model (call it student) is defined by another machine learning model (call it teacher):
  - Original SGD process:

$$w_{t+1} = w_t - \eta_t \frac{\partial \mathcal{L}\left(f_{w_t}(x), y\right)}{\partial w_t}$$

  - Dynamic loss for SGD process:

$$w_{t+1} = w_t - \eta_t \frac{\partial \mathcal{L}_{h_\Theta(x,y)}\left(f_{w_t}(x), y\right)}{\partial w_t}$$

Learning to Teach with Dynamic Loss Functions. NeurIPS 2018.

Others: Curriculum on Loss Function

Dynamic Loss Functions (DLF)

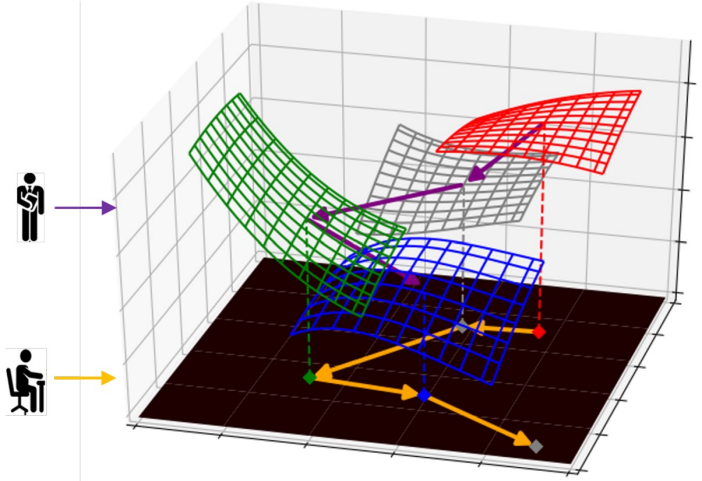
- Student model:

$$w_{t+1} = w_t - \eta_t \frac{\partial \mathcal{L}_{h_\Theta(x,y)}\left(f_{w_t}(x), y\right)}{\partial w_t},$$

- Teacher model:

$$\Theta = \arg\max_{\Theta} \text{Performance}(f_w, h_\Theta, \mathcal{D}_{dev}),$$

- which can be solved by Reverse-Mode Differentiation (RMD).



Algorithm 1 Training Teacher Model μ_θ

Input: Continuous relaxation $\tilde{m}$. Initial value of θ.
while Teacher model parameter θ not converged **do** ▷ One *teacher optimization step*
  Randomly initialize student model parameter ω_0.
  for each time step $t = 0, \cdots, T-1$ **do** ▷ Teach student model
    Conduct student model training step via Eqn. (6).
  end for
  $d\theta = 0$. Compute $d\omega_T$ via Eqn. (3).
  for each time step $t = T-1, \cdots, 0$ **do** ▷ Reversely calculating the gradient $d\theta$
    Update $d\theta$ as Eqn. (5).
    Compute $d\omega_t$ as Eqn. (7).
  end for
  Update θ using $d\theta$ via gradient based optimization algorithm.
end while
Output: the final teacher model μ_θ.

Learning to Teach with Dynamic Loss Functions. NeurIPS 2018.

Others: Curriculum on Model

Curriculum by Smoothing (CBS)

- When learning a network from scratch, the information propagated within the network during the earlier stages of training can contain distortion artifacts due to noise.
- This work proposes a model curriculum, controlling the amount of high frequency information propagated within the CNNs as training progresses by using low-pass filters.

```
# Use the Gaussian kernel after the convolution operation
h = gaussian_kernel(conv1(x))
# Add non-linearity and pooling
h = activation(pool(h))

# Same operation after each conv. layer
h = gaussian_kernel(conv2(h))
h = activation(pool(h))
```

$$h_i = ReLU\left(pool\left(\theta_{G_\sigma} \circledast (\theta_w \circledast x_i)\right)\right)$$

Gaussian kernel, known as a low-pass filter, smooths high frequency information

Curriculum by Smoothing. NeurIPS 2020.

Curriculum Approach: Summary

- Predefined Curriculum vs. Automatic Curriculum

	Predefined CL	Automatic CL
Applicability	Need human knowledge	General
Difficulty	Human predefined, fixed	Model decided, dynamic
Scheduler	Ignore model feedback, fixed	Consider model feedback, dynamic

- Main Methodologies for Automatic Curriculum

Self-Paced Learning	Transfer Teacher	RL Teacher	Others
Student model tell difficulty by itself	Teacher model judge difficulty	Teacher judge according to student feedback	Curriculum on optimization, loss, model, ……

Applications

Applications

- Application fields so far:
  - **Combinatorial Optimization**: traveling salesman problem, secretary problem, etc.
  - **Computer Vision**: image classification, object detection, semantic segmentation, face recognition, image generation and translation, video processing, etc.
  - **Natural Language Processing**: machine translation, question answering, etc.
  - **Multi-Modal Learning**: visual question answering, visual grounding, etc.
  - **Recommendation**: sequential recommendation, social recommendation, etc.
  - **Graph Machine Learning**: node classification, graph classification, link prediction, etc.
  - **Robotics / Reinforcement Learning**: navigation, control, games, etc.
  - **Other Cutting-edge Areas**: meta-learning, continual learning, neural architecture search, etc.

Applications

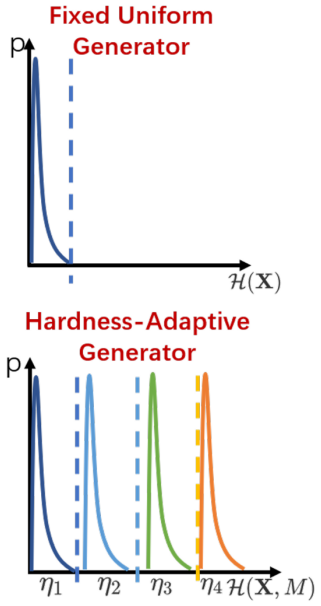
- Application fields so far: CO, CV, NLP, Graph, Robotics and Others…

Motivation	Effect	Scene	Example
To guide	make training possible/much better	when the target task is hard or has a different distribution	games, control, …
To denoise	make training faster, robust, generalizable	tasks with noisy, uneven quality, heterogeneous data (often large-scale, cheaply collected)	image classification, neural machine translation, question answering, …
To balance	make training robust on imbalanced data	tasks with class imbalance	classification, detection, …

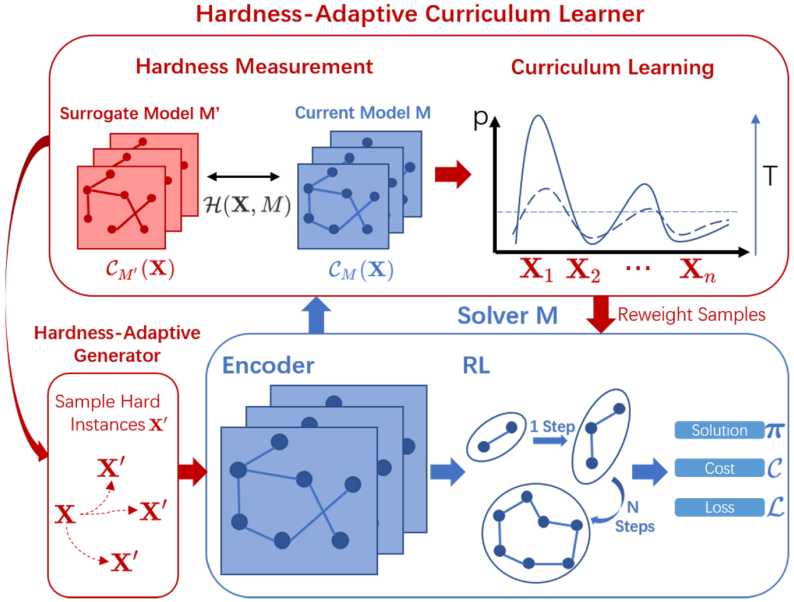
Applications: Combinatorial Optimization

Hardness-Adaptive Curriculum for Travelling Salesman Problem

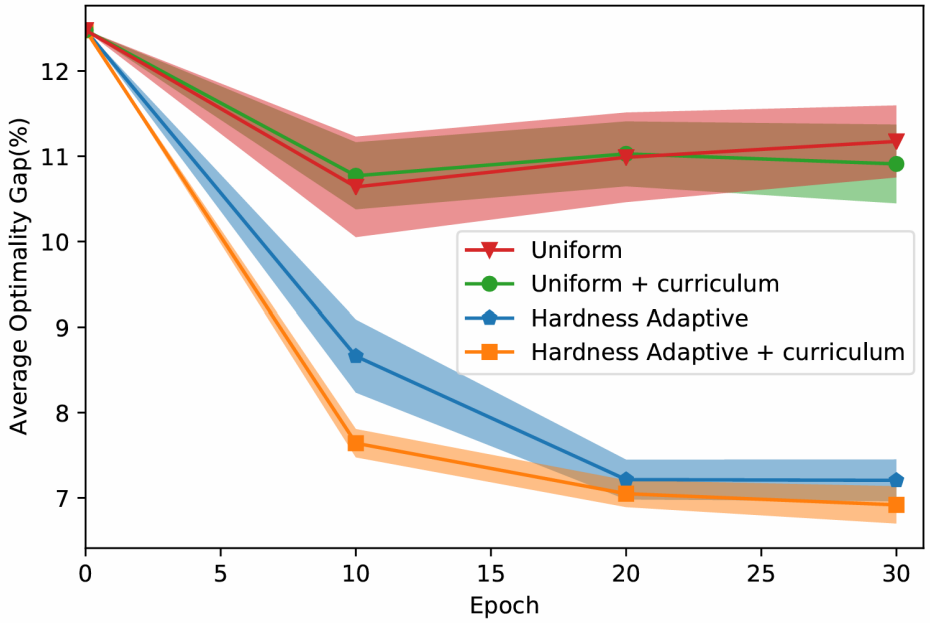
- Different training and testing distribution will result in more difficult TSP instances.
- This work proposes a principled hardness measurement to quantify the hardness of instances and a hardness-adaptive generator to generate instances with different hardness.



(a) Generators



(b) Our proposed hardness-adaptive curriculum learner



Learning to Solve Travelling Salesman Problem with Hardness-Adaptive Curriculum. AAAI 2022.

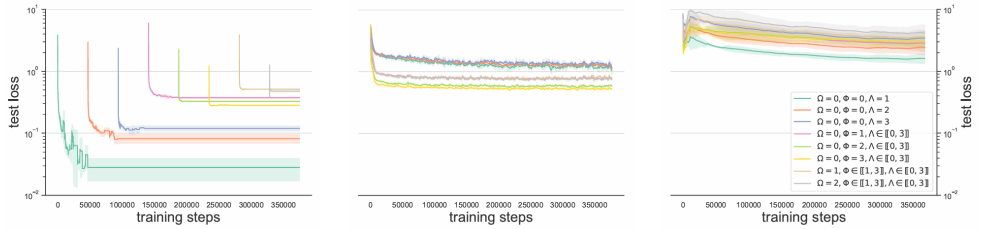
Applications: Combinatorial Optimization

Curriculum for Multilevel CO Problems

- Multilevel combinatorial optimization problems are the generalization of the already solved single-level optimization problems with only one player.
- The curriculum: if an agent knows how to estimate the value of instances with budgets up to B, then solving instances with budget $B+1$ can be done in polynomial time.

Algorithm 1: MultiL-Cur

Initialize the value-network $\hat{V}$ with weights $\hat{\theta}$;
Initialize the list of experts $\mathbb{L}_{\hat{V}}$ to be empty ;
for $j = B_{max}, ..., 2$ **do**
  Create $\mathcal{D}^j_{train}, \mathcal{D}^j_{val}$ by sampling $(s^r_j \sim \mathbb{D}^r_j, \texttt{GreedyRollout}^a(s^r_j, \mathbb{L}_{\hat{V}}))$;
  Initialize $\hat{V}_j$, the expert of level j with $\hat{\theta}_j = \hat{\theta}$;
  Initialize the loss on the validation set $\mathcal{L}^j_{val}$ to ∞ ;
  for *epoch* $e = 1, ..., E$ **do**
    for *batches* $(s_i, \hat{y}_i)_{i=1}^m \in \mathcal{D}^j_{train}$ **do**
      Update $\hat{\theta}$ over $\frac{1}{m}\sum_{i=1}^m \left(\hat{y}_i - \hat{V}(s_i)\right)^2$ with Adam [35] ;
      if *number of new updates* = T_{val} **then**
        if $\mathcal{L}^{new}_{val} = \frac{1}{N_{val}}\sum_{k=1}^{N_{val}} \left(\hat{y}_k - \hat{V}(s_k)\right)^2 < \mathcal{L}^j_{val}$ **then**
          $\hat{\theta}_j \leftarrow \hat{\theta}$; $\mathcal{L}^j_{val} \leftarrow \mathcal{L}^{new}_{val}$;
  Add $\hat{V}_j$ to $\mathbb{L}_{\hat{V}}$
return *the trained list of experts* $\mathbb{L}_{\hat{V}}$



opt gap (%)	approx ratio	opt gap (%)	approx ratio	opt gap (%)	approx ratio
0.51	**1.006**	3.34	1.036	14.53	1.158

				MCN						MCN$_{dir}$		MCN$_w$	
	exact	*random*		*DA-AD*			*cur*			*cur*		*cur*	
$\|V\|$	$t(s)$	$\eta(\%)$	ζ	$t(s)$	$\eta(\%)$	ζ	$t(s)$	$\eta(\%)$	ζ	$\eta(\%)$	ζ	$\eta(\%)$	ζ
20	29	68	3.32	6	**0.3**	**1.00**	**0.4**	0.5	1.00	5.7	1.07	6.9	1.07
40	241	52	2.64	13	7.6	1.09	**0.9**	**5.0**	**1.06**	11.9	1.13	6.5	1.07
60	405	68	3.24	38	7.3	1.09	**1.5**	**4.4**	**1.05**	4.4	1.05	3.7	1.04
80	636	55	2.28	60	3.8	1.04	**2.8**	**2.7**	**1.03**	1.6	1.02	2.8	1.03
100	848	45	1.86	207	**2.7**	**1.03**	**8.7**	49.6	1.50	1.8	1.02	4.1	1.05

Curriculum Learning for Multilevel Budgeted Combinatorial Problems. NeurIPS 2020.

Applications: Combinatorial Optimization

Curriculum in Policy Optimization Methods for Online CO Problems

- Theoretical study on the benefit of curriculum learning in RL for CO is still at its early stage, despite the emerging empirical evidence.
- This work explains a curriculum can find a strong sampling policy and reduce the distribution shift, a critical quantity that governs the convergence rate.

Algorithm 1 `NPG` (Full version: Algorithm 3)

Input: Environment E; learning rate η; episode number T; batch size N; initialization θ_0; sampler π_s; optimization domain $\mathcal{G}$.
for $t \leftarrow 0, 1, \ldots, T-1$ **do**
  For $0 \leqslant n \leqslant N-1$ and $0 \leqslant h \leqslant H-1$, sample $(a_h^{(n)}, s_h^{(n)})$ and estimate $\widehat{A}_{H-h}^{(n)}$ using Algorithm 4.
  Calculate:

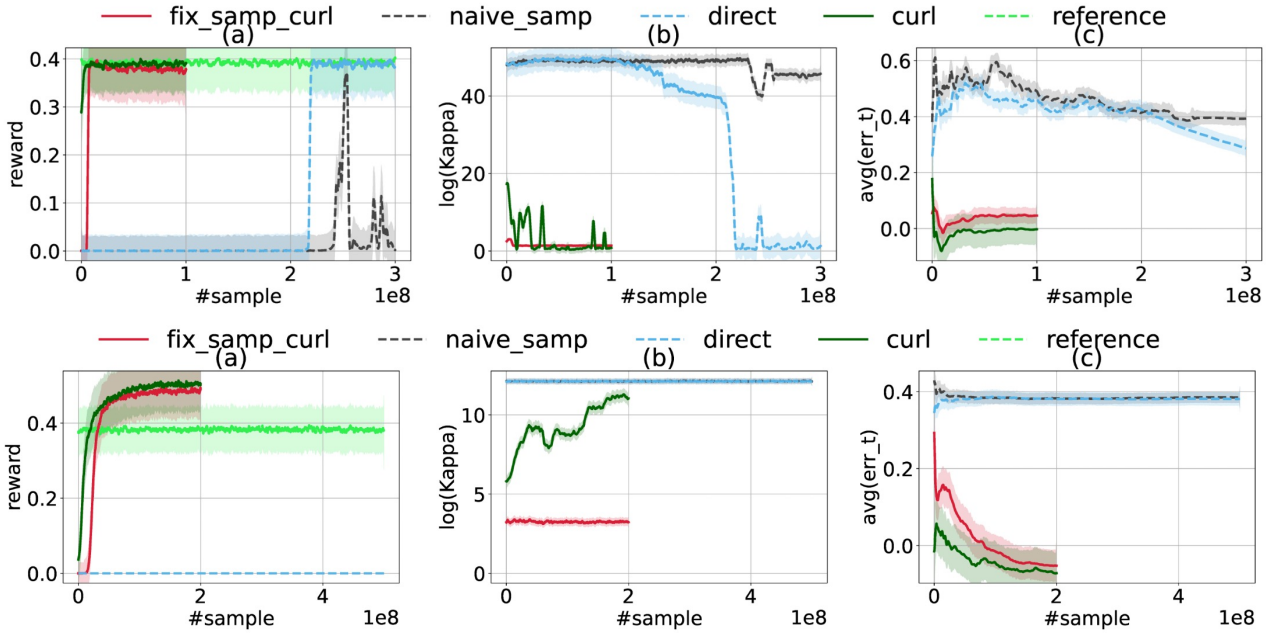
$$\widehat{F}_t \leftarrow \sum_{n=0}^{N-1} \sum_{h=0}^{H-1} (\nabla_\theta \ln \pi_{\theta_t}(a_h^{(n)} | s_h^{(n)}))^{\otimes},$$

$$\widehat{\nabla}_t \leftarrow \sum_{n=0}^{N-1} \sum_{h=0}^{H-1} \widehat{A}_{H-h}^{(n)} \nabla_\theta \ln \pi_{\theta_t}(a_h^{(n)} | s_h^{(n)}).$$

  Call any solver to get $\widehat{g}_t \leftarrow \arg\min_{g \in \mathcal{G}} g^\top \widehat{F}_t g - 2 g^\top \widehat{\nabla}_t$.
  Update $\theta_{t+1} \leftarrow \theta_t + \eta \widehat{g}_t$.
end for
Return: θ_T.

Algorithm 2 Curriculum learning framework.

Input: Environment E; learning rate η; episode number T; batch size N; sampler type $samp \in \{$ `pi_s, pi_t` $\}$; optimization domain $\mathcal{G}$.
Construct an environment E' with a task easier than E. This environment should have optimal policy similar to that of E.
$\theta_s \leftarrow$ `NPG` $(E', \eta, T, N, 0^d, \text{None}, \mathcal{G})$ (Algorithm 1).
if $samp =$`pi_s` **then**
  $\theta_T \leftarrow$ `NPG` $(E, \eta, T, N, 0^d, \pi_s, \mathcal{G})$.
else
  $\theta_T \leftarrow$ `NPG` $(E, \eta, T, N, \theta_s, \text{None}, \mathcal{G})$.
end if
Return: θ_T.



Understanding Curriculum Learning in Policy Optimization for Online Combinatorial Optimization. TMLR 2023.

Applications: Computer Vision

Curriculum for Image Classification

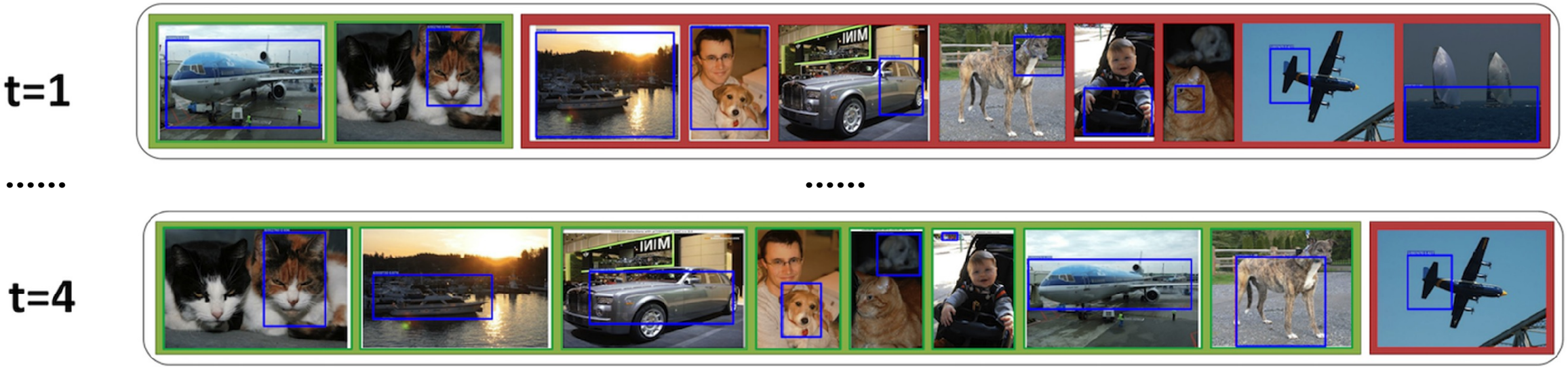
- As a popular machine learning task, image classification is widely used to validate the effectiveness of curriculum learning empirically.

Method	Conference	Involved Datasets	Involved Models
SPL	NeurIPS 2010	MUC6, UniProbe, MNIST, Mammals	SSVM
MCL	ICLR 2018	MNIST, CIFAR-10, STL-10, SVHN, …	LeNet5, CNN
DCL	NeurIPS 2019	CIFAR-10, CIFAR-100, ImageNet, …	VGG-16, SSDNet, ResNet-18, …
DIHCL	NeurIPS 2020	CIFAR-10, CIFAR-100, ImageNet, …	ResNet-50, WideResNet-16-8, …
Superloss	NeurIPS 2020	MNIST, CIFAR-10, CIFAR-100, …	ResNet-18, ResNet-50, …
CBS	NeurIPS 2020	CIFAR-10, CIFAR-100, ImageNet, …	VGG-16, ResNet-18, …
Adaptive CL	ICCV 2021	CIFAR-10, CIFAR-100, ImageNet, …	VGG-16, ResNet-18, …

Applications: Computer Vision

Curriculum for Object Detection

- In a weakly-supervised scenario, most object detectors are based on an iterative process, but the errors of an immature classifier can make the process drift.
- The main idea in this work is to iteratively select a subset of images and boxes that are the most reliable, based on the self-paced learning paradigm.



Self Paced Deep Learning for Weakly Supervised Object Detection. TPAMI 2019.

Applications: Natural Language Processing

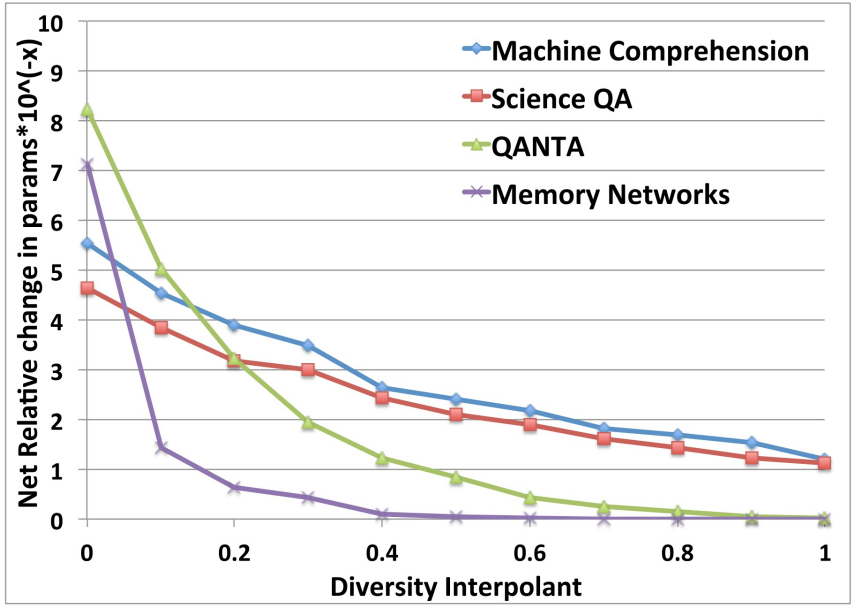
Question Answering

- This work evaluates the curriculum learning in the task of QA and NLP.
- This work proposes new heuristics for determining the easiness and highlights the diversity. The diversity is measured using the angle between the hyperplanes that the question examples induce in the feature space.

Text: … Natural greenhouse gases include carbon dioxide, methane, water vapor, and ozone ... CFCs and some other man-made compounds are also greenhouse gases …

Hypothesis: The important greenhouse gases are Carbon dioxide , Methane, Ozone and CFC

Q: What are the important greenhouse gases? → **A:** Carbon dioxide, Methane, Ozone and CFC

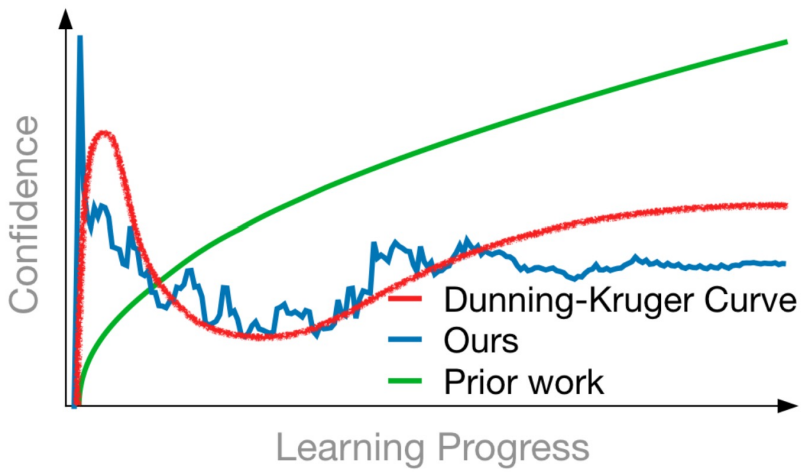


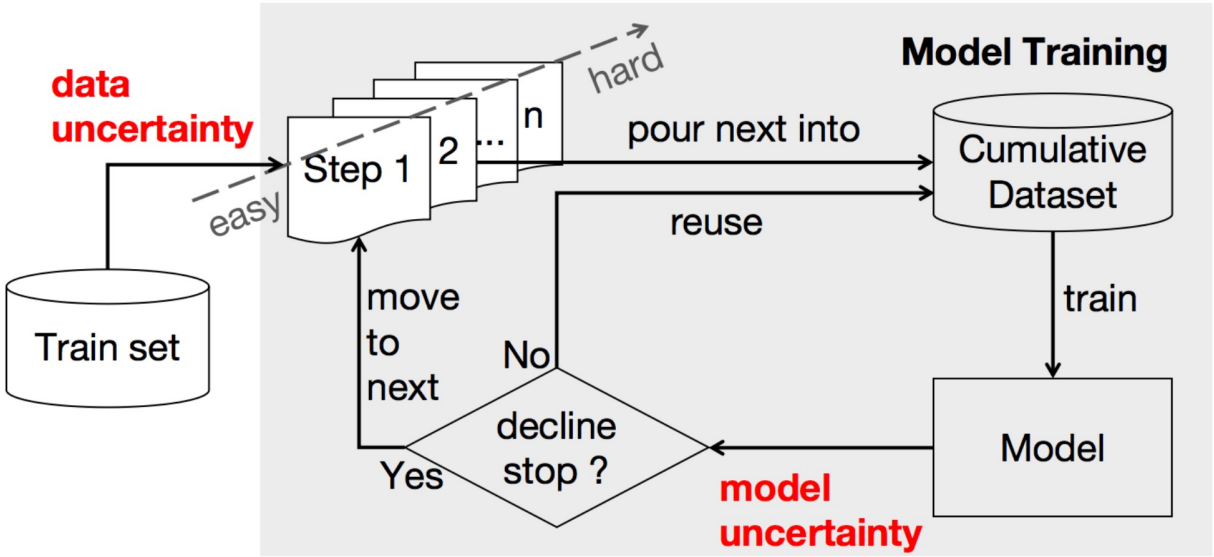
Easy Questions First? A Case Study on Curriculum Learning for Question Answering. ACL 2016.

Applications: Natural Language Processing

Neural Machine Translation

- The open problems in the integration of CL with NMT lie in the assessment of data difficulty and the program of learning schedule.
- This work proposes a uncertainty-aware curriculum where cross-entropy of data serves as its data difficulty and the variance of distributions over the weights of the network serves as the model uncertainty.



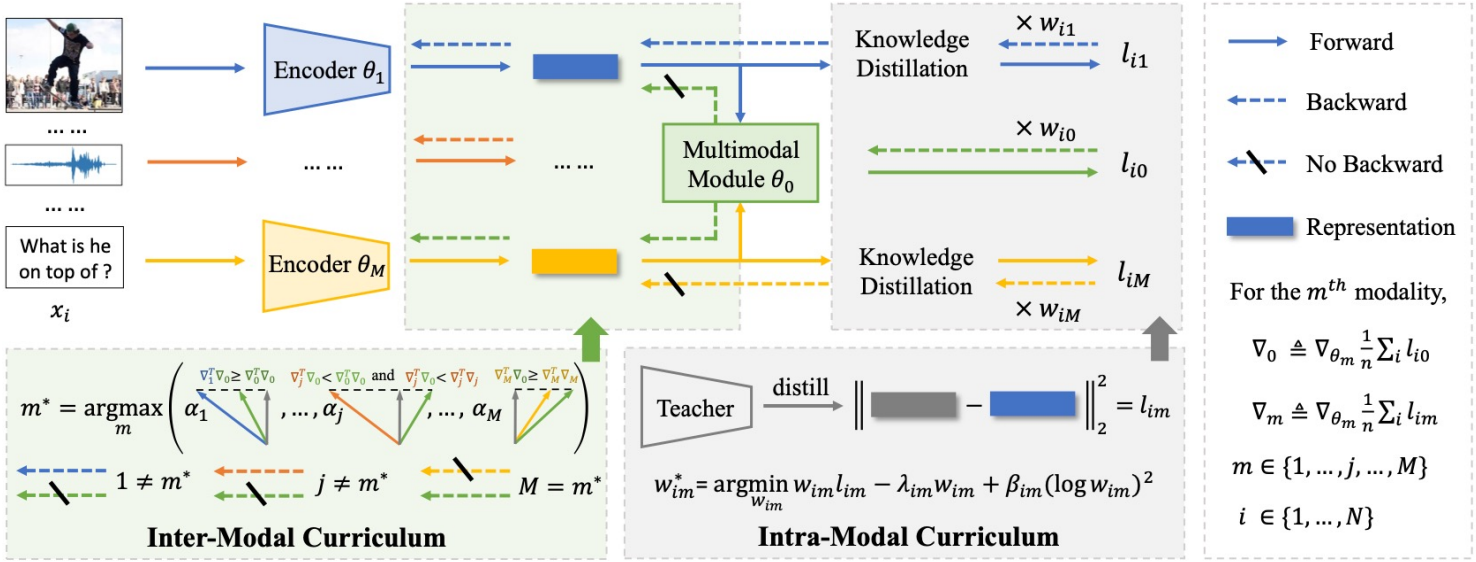


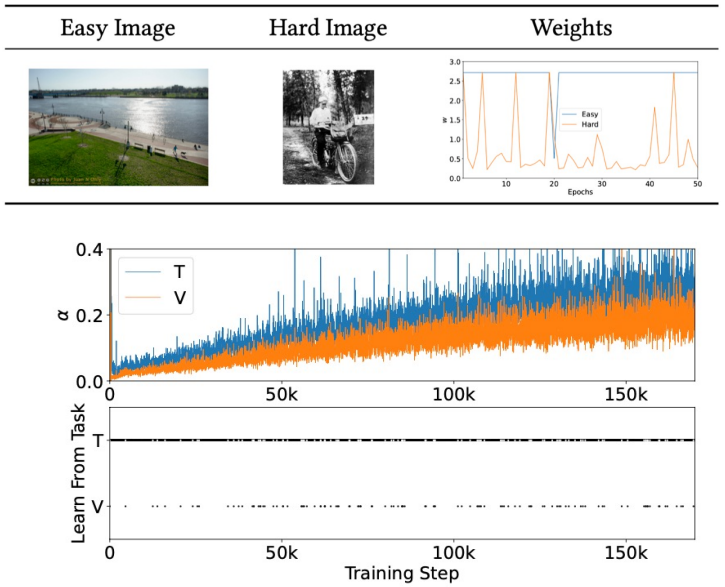
Uncertainty-Aware Curriculum Learning for Neural Machine Translation. ACL 2020.

Applications: Multimodal Learning

Multimodal Learning

- Modal encoders are under-optimized in multimodal learning in contrast to unimodal learning, especially when some modalities are dominant over others.
- This work proposes an intra- and inter-modal curriculum learning method which simultaneously considers both data difficulty and modality balance for multimodal



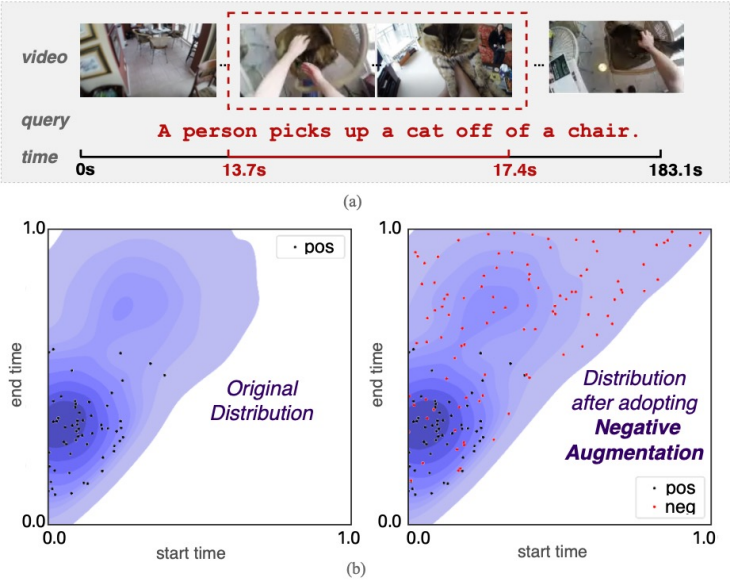


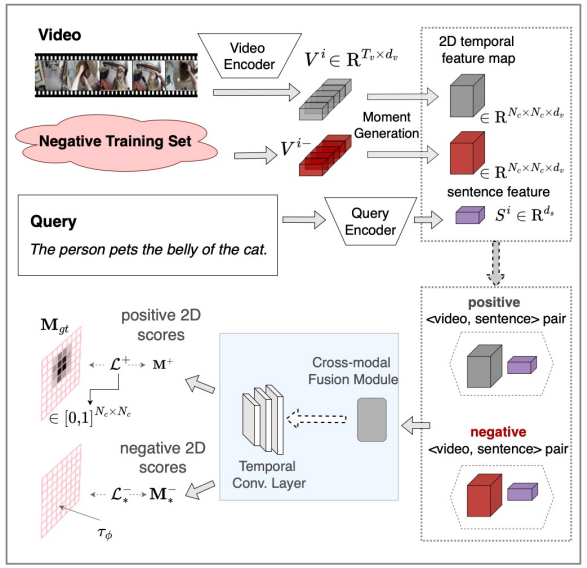
Intra- and Inter-Modal Curriculum for Multimodal Learning. ACMMM 2023.

Applications: Multimodal Learning

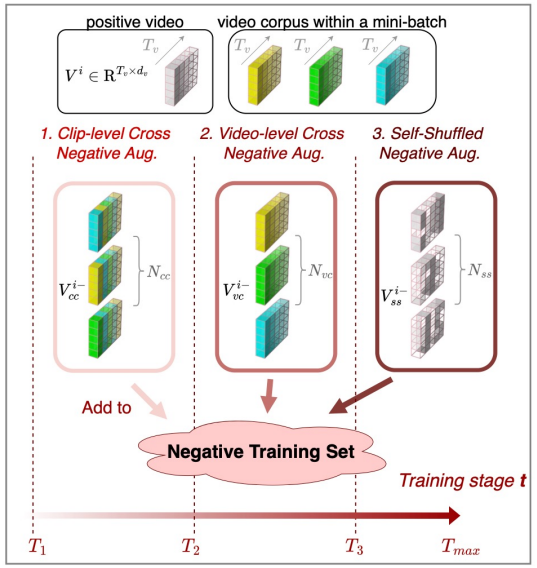
Video Grounding

- Current video grouding models are prone to over-rely the groundtruth moment annotation distribution biases in the training set.
- This work designs a multi-stage curriculum strategy to adaptively train the VG model from easy to hard negative augmentations.





The Overall Grounding Pipeline



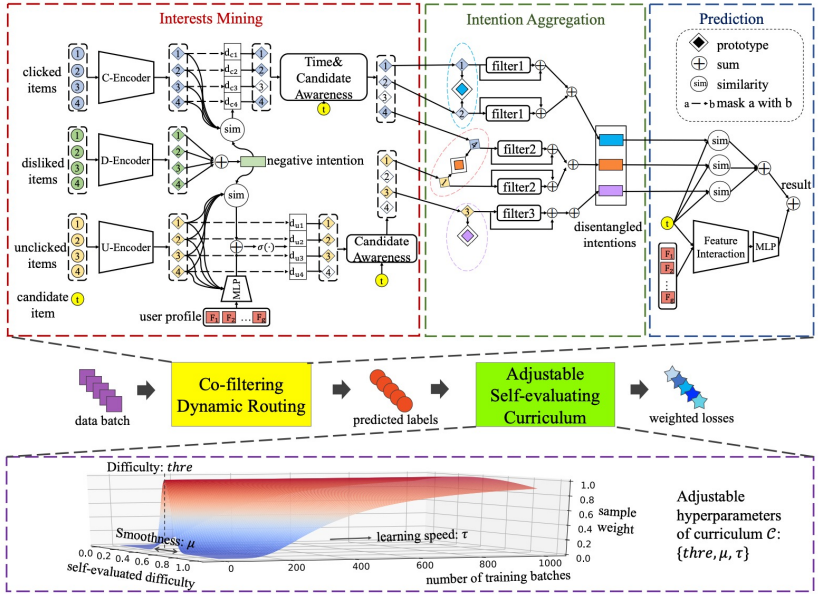
Multi-stage Curriculum Process with NA
(from *easy* to *hard*)

Curriculum Multi-Negative Augmentation for Debiased Video Grounding. AAAI 2023.

Applications: Recommendation

Recommendation with Noisy Multiple Feedback

- Existing recommendation works only focus on positive feedback, failing to handle the complex relations and noise hidden in multi-feedback data.
- This work presents an adjustable self-evaluating curriculum that can evaluate sample difficulties and conduct denoising in the label level via disregarding useless information.



Algorithm 1 Adjustable Self-evaluating Curriculum towards A Better Self

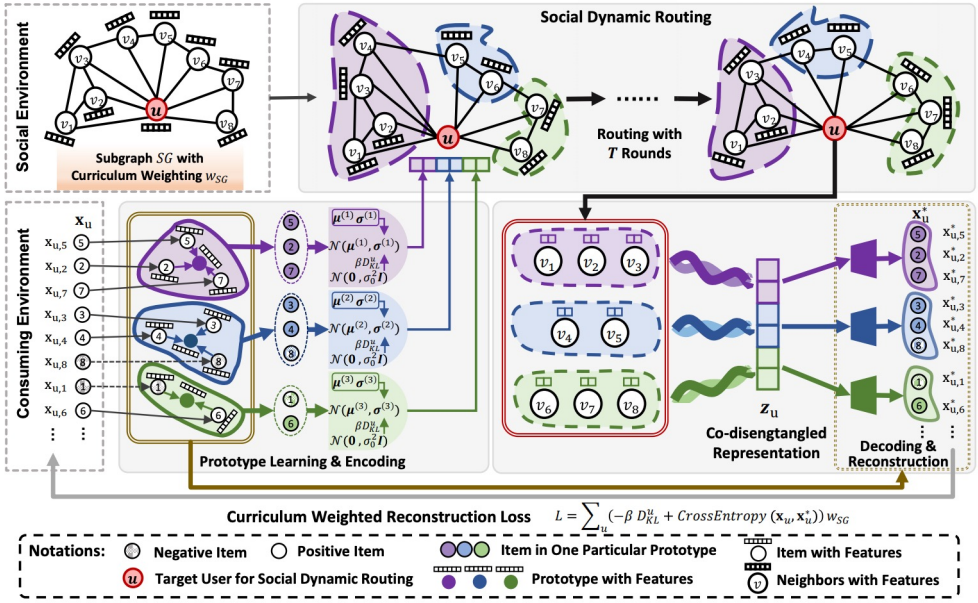
input: $\{(x_{vt}, y_{vt})\}_{(v,t)\in D_T}, \pi(\cdot), \ell(\cdot,\cdot), \tau < 1, counter, interval, thre, F$
initialize: θ, μ
for $B \subset D_T$ **do**
  for $(x_j, y_j) \in B$ **do**
    $s_j = F(x_j), d_j = abs(y_j - \sigma(s_j))$;
    $w_j = exp(-\mu \times (d_j - thre)^2)$;
    $w_j \leftarrow w_j / \sum_{j=1}^{|B|}(w_j)$;
  end for
  $\theta \leftarrow \theta - \pi(\nabla_\theta \sum_{j=1}^{|B|} w_j \ell(s_j, y_j) + \sum_{i=1}^{K} \sum_{j\neq i} \phi(\mathbf{m}_i, \mathbf{m}_j))$;
  $counter \leftarrow counter + 1$;
  if $counter\%interval == 0$ **then**
    $\mu \leftarrow \mu \times \tau$;
  end if
end for

Curriculum Disentangled Recommendation with Noisy Multi-feedback. NeurIPS 2021.

Applications: Recommendation

Recommendation across Multiple Environments

- In literature, there has been no work on social recommendation capable of disentangling user representations across consuming and social environments.
- The curriculum: partitioning the social graph into sub-graphs with minimum number of edges being cut, and measuring the complexity of subgraphs via Descartes' rule of signs.



Algorithm 1 Curriculum Subgraph Weighing

Input: $CurriculumEpoch$, λ;
Subgraphs $\mathcal{SG} = \{SG_1, SG_2, \cdots, SG_g\}$;
Subgraph difficulties $\Delta = \{\delta_{SG_1}, \delta_{SG_2}, \cdots, \delta_{SG_g}\}$;
Output: Subgraph weights $\mathcal{W}$;

function CURRICULUMSCHEDULER($epoch$)
  return $\min\left\{1, \lambda + \frac{(1-\lambda)*epoch}{CurriculumEpoch}\right\}$
$\mathcal{W} \leftarrow \{w_{SG_1}, w_{SG_2}, \cdots, w_{SG_g}\}$
  $\triangleright$ w_{SG_j} is the importance weight for subgraph SG_j.
$\Lambda \leftarrow$ CURRICULUMSCHEDULER($epoch$)
for $SG_j \in \mathcal{SG}$ **do**
  if $\delta_{SG_j} \leq \Lambda$ **then**
    $w_{SG_j} \leftarrow 1$
  else
    $w_{SG_j} \leftarrow 1 - \delta_{SG_j}$

Curriculum Co-disentangled Representation Learning across Multiple Environments for Social Recommendation. ICML 2023.

Applications: Graph Machine Learning

Survey on Curriculum Graph Machine Learning

- Most graph machine learning models are designed to conduct training with data samples in a random order, which may suffer from suboptimal performance due to **ignoring the importance of different graph data samples and their training orders** for the model optimization status.
- Problem Definition (Curriculum Graph Machine Learning):
  - Given the training set of data instances (i.e., nodes, links, or graphs) $D = \{(X_i, Y_i)\}_{i=1}^{|\mathcal{D}|}$, where X_i is the input instance and Y_i denotes the label, the goal is to optimize an optimal graph learning model guided by the curriculum $\mathcal{C}$, so as to achieve the best performance on testing graph data.

Curriculum Graph Machine Learning: A Survey. IJCAI 2024.

Applications: Graph Machine Learning

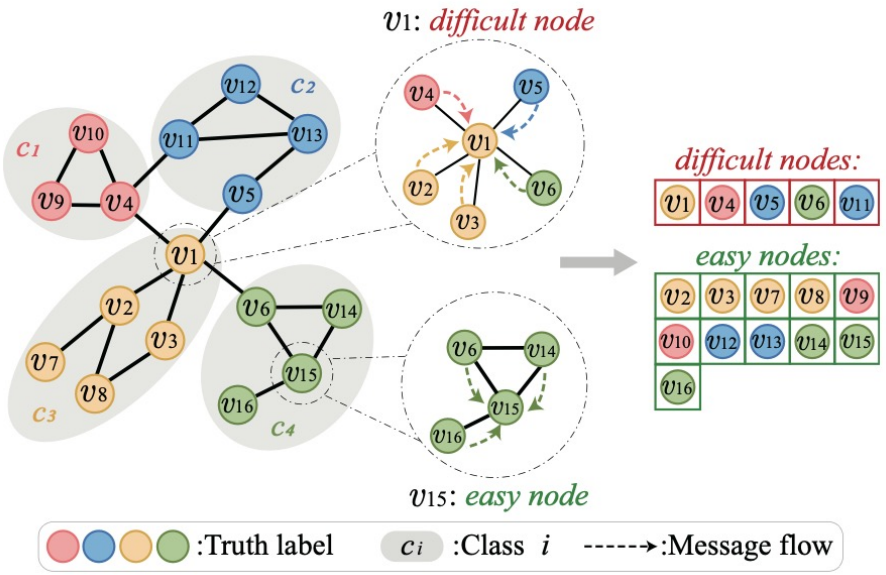
Method	Graph CL Type	Difficulty Measurer	Training Scheduler	Task	Need Label
Node-level Graph CL					
CLNode [2022]	Predefined	Label Distribution	Continuous	Node Classification	✓
GNN-CL [2022c]	Predefined	Sample Similarity	Continuous	Node Classification	✓
SMMCL [2019]	Predefined	Label Distribution	Discrete	Node Classification	✓
DiGCL [2021]	Predefined	Laplacian Perturbation	Continuous	Node Classification	✗
HSAN [2023]	Predefined	Sample Similarity	Discrete	Node Classification	✗
MTGNN [2020b]	Predefined	Step Length	Discrete	Time Series Forecasting	✓
MentorGNN [2022a]	Automatic	Attention Weight	Discrete	Node Classification	✗
RCL [2023]	Automatic	Self-supervised Loss	Continuous	Node Classification	✓
DRL [2018]	Automatic	Cumulative Reward	Discrete	Node Classification	✓
GAUSS [2022]	Automatic	Sample Loss	Discrete	Node Classification	✓
CGCT [2021]	Automatic	Sample Similarity	Discrete	Image Classification	✓
Link-level Graph CL					
GCN-WSRS [2018]	Predefined	Sample Similarity	Continuous	Link Prediction	✓
TUNEUP [2022]	Predefined	Node Degree	Discrete	Link Prediction	✓
CHEST [2023]	Predefined	Pretraining Task	Discrete	Link Prediction	✗
GTNN [2022]	Automatic	Sample Loss	Discrete	Relation Extraction	✓
Graph-level Graph CL					
CurGraph [2021b]	Predefined	Label Distribution	Discrete	Graph Classification	✓
CuCo [2021]	Predefined	Sample Similarity	Continuous	Graph Classification	✗
HACL [2022]	Predefined	Sample Size	Discrete	Graph Classification	✓
Dual-GCN [2021]	Predefined	BLEU Metric	Discrete	Image Captioning	✓
CurrMG [2022]	Automatic	Domain Knowledge	Continuous	Graph classification	✓
HAC-TSP [2022b]	Automatic	Solution Cost	Continuous	Travelling Salesman Problem	✗

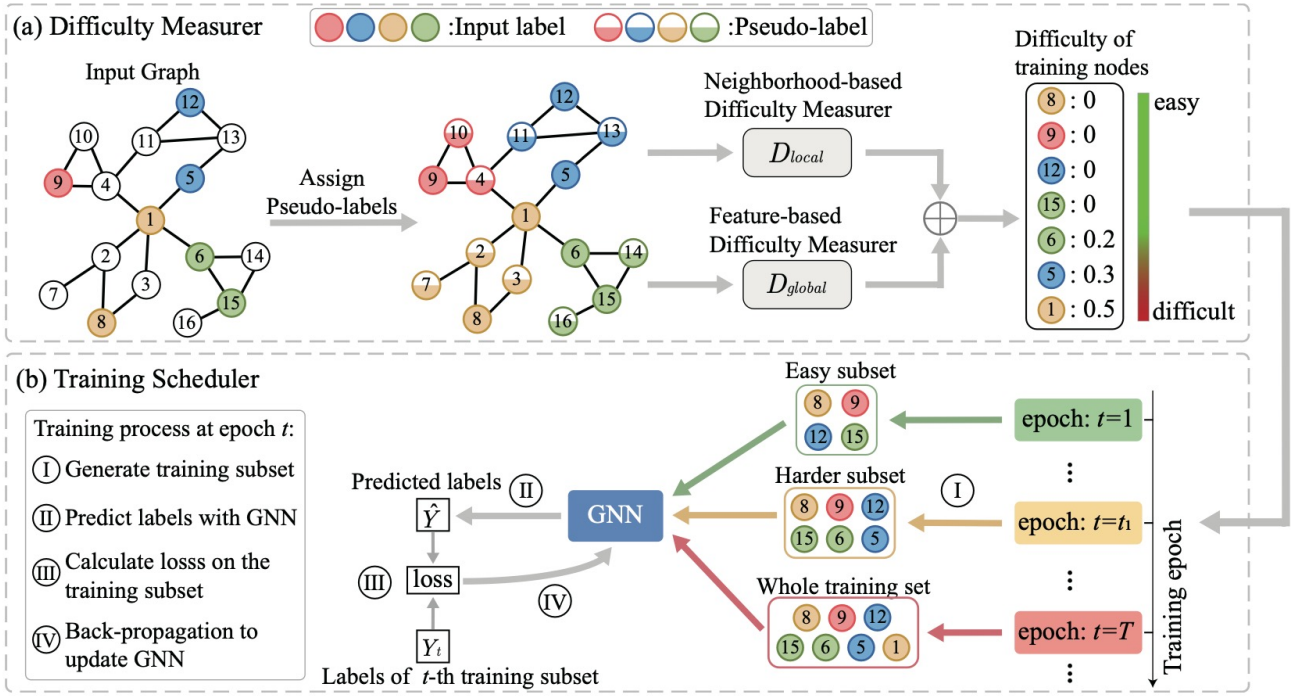
Curriculum Graph Machine Learning: A Survey. IJCAI 2024.

Applications: Graph Machine Learning

Node Classification

- The performance of GNNs could be harmed by two types of low-quality training nodes: (1) inter-class nodes situated near class boundaries. (2) mislabeled nodes.
- This work designs a multi-perspective measurer and a scheduler to select nodes.



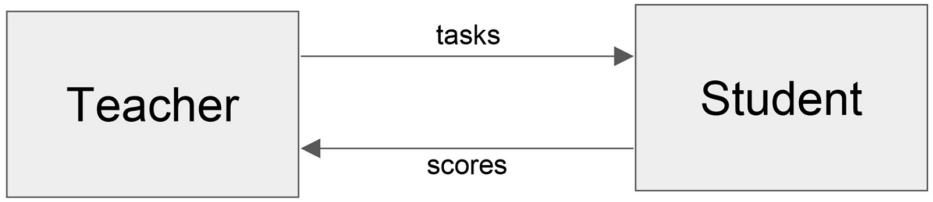


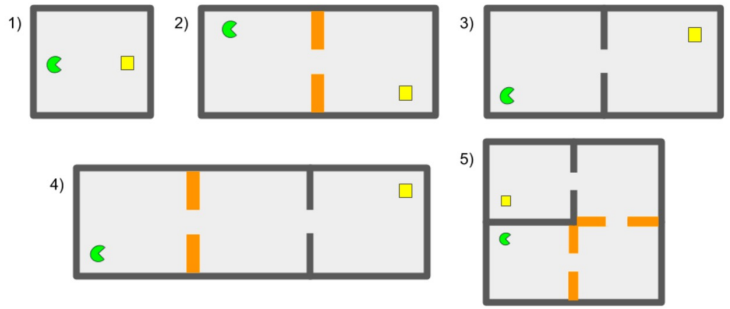
CLNode: Curriculum Learning for Node Classification. WSDM 2023.

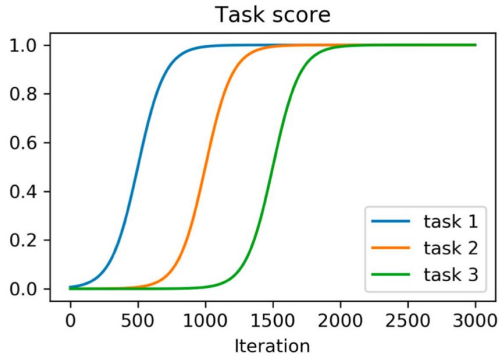
Applications: Reinforcement Learning

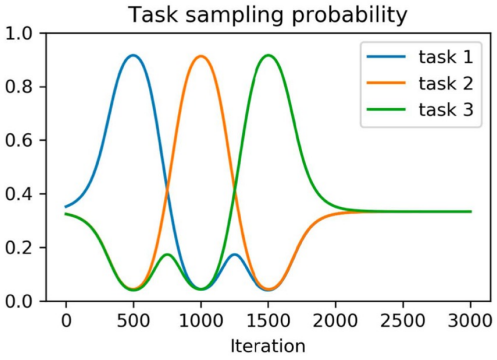
Navigation in Minecraft

- The method relies on the intuition that the Student should practice more those tasks on which it makes the fastest progress and the Teacher should address the problem of forgetting by also choosing tasks where the Student's performance is getting worse.







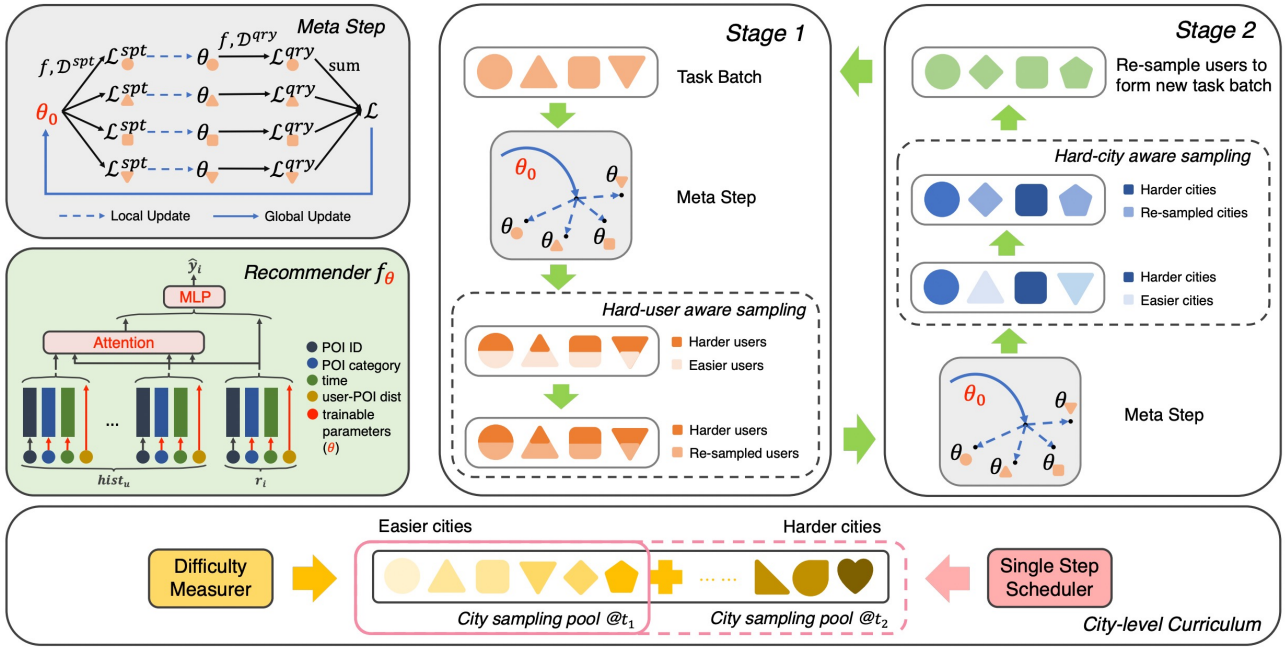


Teacher–Student Curriculum Learning. TNNLS 2019.

Applications: Other Aeras

Curriculum for Meta-Learning

- This work proposes a curriculum hardness aware meta-learning framework, which incorporates hard sample mining and curriculum learning into a meta-learning paradigm, considering binary-level hardness to enhance the meta-learning process.



Algorithm 1 Curriculum Hardness Aware Meta-Learning

Input: $C_{\mathcal{B}}, \mathbb{D}_{train}, N$: N base cities and all the samples; f: base recommender; α, β: learning rates; B_u, B_c: size of user/city batches; k_u, k_c: proportions of hard users/cities; M: max step of iterations; η: the number of iterations in each scheduler step; ξ: the fraction of cities in the initial scheduler step

Output: θ: optimal initial weights of f

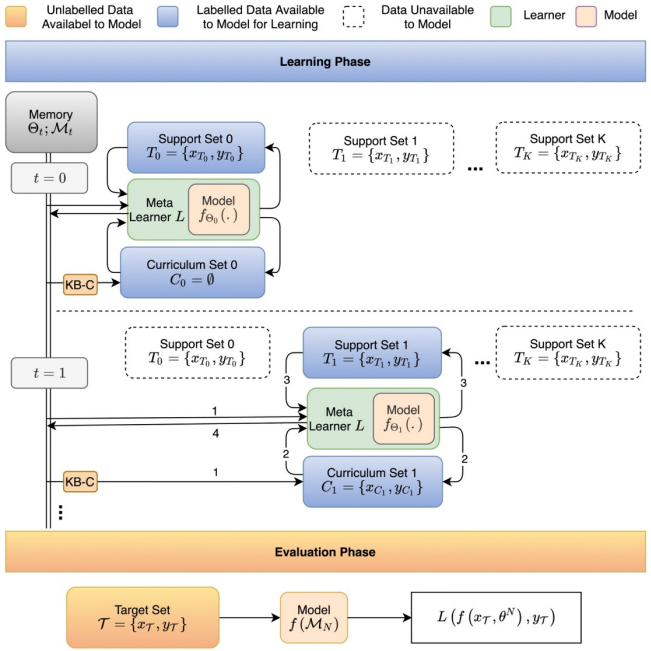
Randomly initialize θ.
Calculate the difficulty of each city by Eq.(10).
Randomly sample a batch of cities $\mathcal{B}_{init} \sim C_{\mathcal{B}}$.
Randomly sample a batch of users $\{u\} \subset \mathcal{U}_c$ from each city $c \in \mathcal{B}_{init}$ to form $\mathcal{T}_{init}$, $\mathcal{T}_{hard_city} = \mathcal{T}_{init}$.
repeat
  Update the city sampling pool as Eq.(11).
  [Stage 1]
  Do a meta step Eq.(5)(6) on $\mathcal{T}_{hard_city}$;
  Calculate user-level accuracies $\{acc_u\}$ on query samples of each user u;
  Select the $k_u B_u$ hardest users by $\{acc_u\}$ and re-sample new users in each city to form $\mathcal{T}_{hard_user}$ as Eq.(8).
  [Stage 2]
  Do a meta step Eq.(5)(6) on $\mathcal{T}_{hard_user}$;
  Calculate task-level accuracies $\{acc_t\}$ on $\mathcal{D}^{qry}$ of each task;
  Select the $k_c B_c$ hardest cities by $\{acc_t\}$ and re-sample new cities to form $\mathcal{B}_{hard_city}$ as Eq.(9);
  Randomly sample a batch of users $\{u\} \subset \mathcal{U}_c$ from each city $c \in \mathcal{B}_{hard_city}$ to form $\mathcal{T}_{hard_city}$.
until Max step of iterations M

Curriculum Meta-Learning for Next POI Recommendation. KDD 2021.

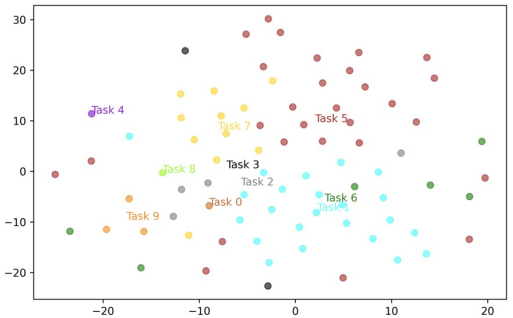
Applications: Other Aeras

Curriculum for Continual Learning

- Catastrophic forgetting and order-sensitivity are major problems in continual learning.
- This work proposes a curriculum-meta learning to adapt parameters to a new task and to reduce interference of previous tasks on the current task.



		Continual-FewRel				Continual-SimpQ				Continual-TACRED			
		Acc_w		Acc_a		Acc_w		Acc_a		Acc_w		Acc_a	
Setting	Model	Acc	EB	Acc	EB	Acc	EB	Acc	EB	Acc	EB	Acc	EB
Cluster	Vanilla‡	16.3	±4.10	19.7	±3.90	60.3	±2.52	58.3	±2.30	12.0	±3.21	8.7	±2.35
	EWC†	27.1	±2.32	30.2	±2.10	67.2	±3.16	59.0	±2.20	14.5	±2.51	14.5	±2.90
	AGEM†	36.1	±2.51	42.5	±2.63	77.6	±2.11	72.2	±2.72	12.5	±2.24	16.5	±2.20
	EA-EMR‡	59.8	±1.50	74.8	±1.30	82.7	±0.48	86.2	±0.33	17.8	±1.01	25.4	±1.17
	EMAR†	53.8	±1.30	68.6	±0.71	80.0	±0.83	76.9	±1.39	42.7	±2.92	**52.5**	±1.74
	MLLRE	56.8	±1.30	70.2	±0.93	84.5	±0.35	86.7	±0.46	34.4	±**0.49**	41.2	±1.37
	CML (ours)	**60.2**	±**0.71**	**76.0**	±**0.24**	**85.6**	±**0.34**	**87.5**	±**0.32**	**44.4**	±1.16	49.3	±**1.01**
Random	Vanilla‡	19.1	±1.20	19.3	±1.30	55.0	±1.30	55.2	±1.30	10.2	±2.02	10.4	±2.31
	EWC†	30.1	±1.07	30.2	±1.05	66.4	±0.81	66.7	±0.83	15.3	±1.70	15.4	±1.79
	AGEM†	36.9	±0.80	37.0	±0.83	76.4	±1.02	76.7	±1.01	13.4	±1.47	14.3	±1.62
	EA-EMR‡	61.4	±0.81	61.6	±0.76	83.1	±0.41	83.2	±0.47	27.3	±1.01	30.3	±0.70
	EMAR†	62.7	±0.63	62.8	±0.62	82.4	±0.86	84.0	±0.78	**45.1**	±1.48	**46.4**	±2.00
	MLLRE	59.8	±0.91	59.8	±0.94	85.2	±0.25	85.5	±0.31	36.4	±**0.66**	38.0	±**0.58**
	CML (ours)	**62.9**	±**0.62**	**63.0**	±**0.59**	**86.5**	±**0.22**	**86.9**	±**0.28**	43.7	±0.83	45.3	±0.72



Train		100		200	all
Memory		25	50	50	50
EA-EMR	Acc_a	70.7	75.5	74.8	73.9
	Acc_w	53.2	57.4	59.8	59.6
MLLRE	Acc_a	68.4	72.1	70.2	51.0
	Acc_w	51.9	57.8	56.8	47.3
EMAR	Acc_a	60.1	66.7	68.6	74.1
	Acc_w	43.7	51.2	53.8	57.7
CML	Acc_a	**73.6**	**76.4**	**76.0**	58.0
	Acc_w	**54.7**	**60.3**	**60.2**	49.1

Algorithm 1: Curriculum-Meta Learning

Input: Stream of incoming tasks $\mathcal{T}_1, \mathcal{T}_2, ..$; Classification model f_θ, Step size ϵ, Learning rate α, Relations embedding r, Memory buffer $\mathcal{M}$, Curriculum size k, Curriculum instance size n, Knowledge-based curriculum module g_ϕ

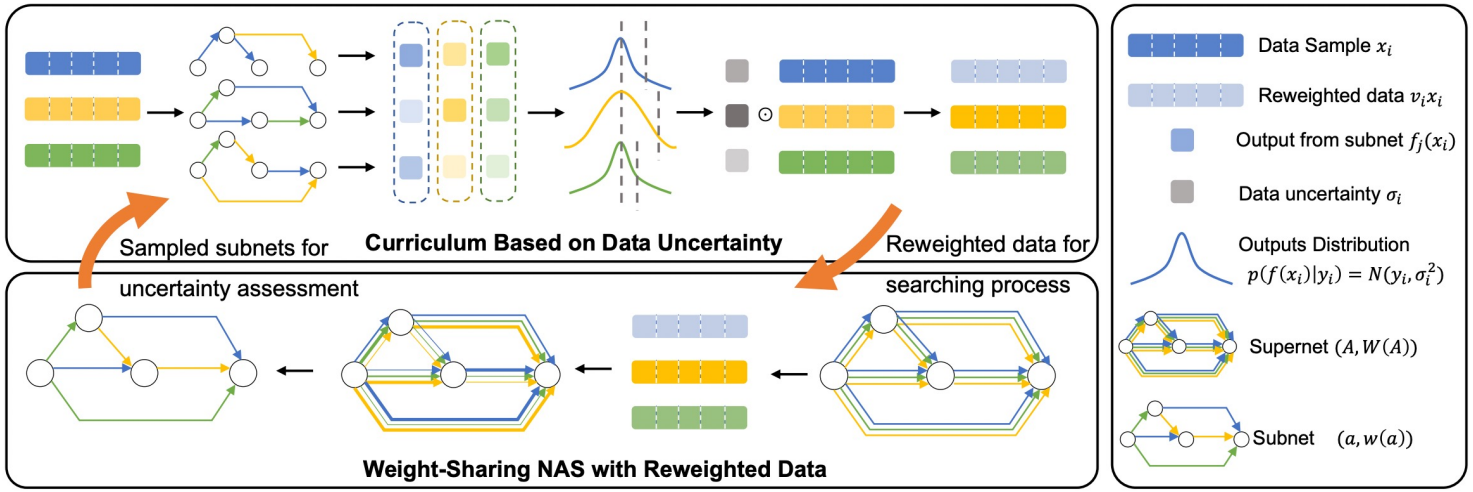
Initialize θ
while *there are still tasking* **do**
  Retrieve current task $\mathcal{T}_t$
  Initialize $\theta_t \leftarrow \theta$
  while *not convergence* **do**
    if $\mathcal{M}$ *is not empty* **then**
      Set $\mathcal{D}_t^{train} \leftarrow \mathcal{D}_t^{train} \cup \mathcal{M}$
    for *each relation* R_i *in* $\mathcal{T}_t$ **do**
      Initialize $\theta_{t_i} \leftarrow \theta_t$
      Sample $D_{t,i}^{train}$ from D_t^{train} for R_i
      if $\mathcal{T}_t$ *is not the first task* **then**
        Sample k curriculum relations from $\mathcal{M}$ with g_ϕ.
        Sample n instances for each curriculum relation
        Construct sorted mini-batches, each contains $D_{t,i}^{curri}$ of $k \times n$ instances.
        Evaluate $\nabla_{\theta_{t_i}} L_{R_i}(f_{\theta_{t_i}})$ using $D_{t,i}^{curri}$
        Update $\theta'_{t_i} = Adam(\theta_{t_i}, L_{R_i}(f_{\theta_{t_i}}), \alpha)$
      Evaluate $\nabla_{\theta'_{t_i}} L_{R_i}(f_{\theta'_{t_i}})$ using $D_{t,i}^{train}$
      Update $\theta^*_{t_i} = Adam(\theta'_{t_i}, L_{R_i}(f_{\theta'_{t_i}}), \alpha)$
  Update $\theta_{t+1} = \theta_t + \frac{\epsilon}{N} \sum_{i=1}^{N} (\theta^*_{t_i} - \theta_t)$
  Sample $subT_t$ from $\mathcal{T}_t$
  Update $\mathcal{M} \leftarrow \mathcal{M} \cup subT_t$
  Update $\theta \leftarrow \theta_t$
  Fine-tune θ on $\mathcal{M}$

Curriculum-Meta Learning for Order-Robust Continual Relation Extraction. AAAI 2021.

Applications: Other Aeras

Curriculum for Neural Architecture Search

- Existing weight-sharing NAS methods overlook the influence of data distribution and treat each data sample equally.
- This work proposes to utilize the multiple subnets included in NAS to jointly assess data uncertainty, which serves as the difficulty criterion in a curriculum manner.



Algorithm 1 Curriculum-NAS Algorithm

Require: The number of sampled subnets K, the original weight-sharing NAS algorithm like DARTS, ENAS, etc.

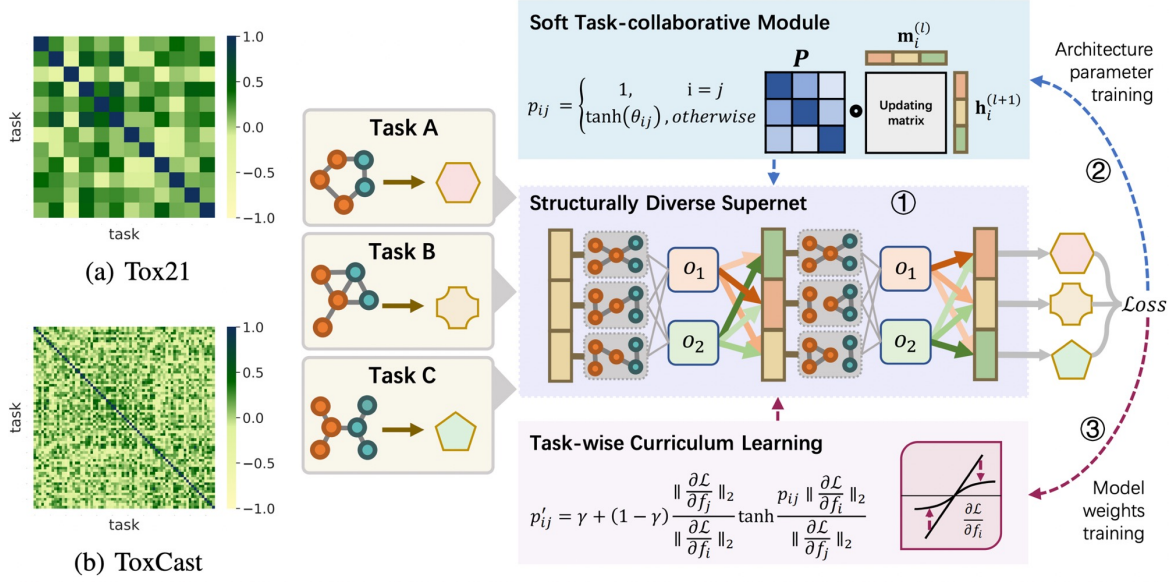
1: **Initialize** data weights $v = \{1\}^N$, supernet weights W, supernet architecture A, architecture searching policy π.
2: **while** not convergent **do**
3: Sample K subnets $f_k(w_k, a_k)$ via Eq. (13) and (14);
4: Assess data uncertainty σ with K subnets via Eq. (10);
5: Update data weights v via Eq. (11);
6: Update supernet weights $w \in W$ via Eq. (3b)
7: Update architecture searching policy π via the same way as the original weight-sharing NAS algorithm.
8: **end while**
9: Derive the final architecture $a^* \in A$ via Eq. (3a).

Curriculum-NAS: Curriculum Weight-Sharing Neural Architecture Search. ACMMM 2022.

Applications: Other Aeras

Curriculum for Multi-Task

- Multi-task Graph NAS faces the challenges of balancing the influence of different tasks on architecture optimization due to their diverse difficulties and optimization patterns.
- This work develops the task-wise curriculum training strategy to better optimize the supernet via reweighing the influence of different tasks on architecture optimization.



Algorithm 1: MTGC^3

Input: Training dataset $\{G_i\}_{i=1}^M$ and $\{Y_i\}_{i=1}^M$, learning rates η_α, $\eta_{\mathbf{S}}$, η_θ and η_w

Initialize learnable parameters w, α, $\mathbf{S}$ and θ. Set $\gamma = 0$;
while *not converge* **do**
  Forward propagate by Eq.(5) and Eq.(9) to calculate loss $\mathcal{L}$;
  $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}(w, \alpha, \mathbf{S}, \theta)$, $\mathbf{S} \leftarrow \mathbf{S} - \eta_{\mathbf{S}} \nabla_{\mathbf{S}} \mathcal{L}(w, \alpha, \mathbf{S}, \theta)$, $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}(w, \alpha, \mathbf{S}, \theta)$;
  Using $\frac{\partial \mathcal{L}}{\partial f_j}$ to calculate p' by Eq.(11);
  Forward propagate by Eq.(5) and Eq.(11) to calculate loss $\mathcal{L}'$;
  $w \leftarrow w - \eta_w \nabla_w \mathcal{L}'(w, \alpha, \mathbf{S}, \theta)$;
  Increase γ;
end

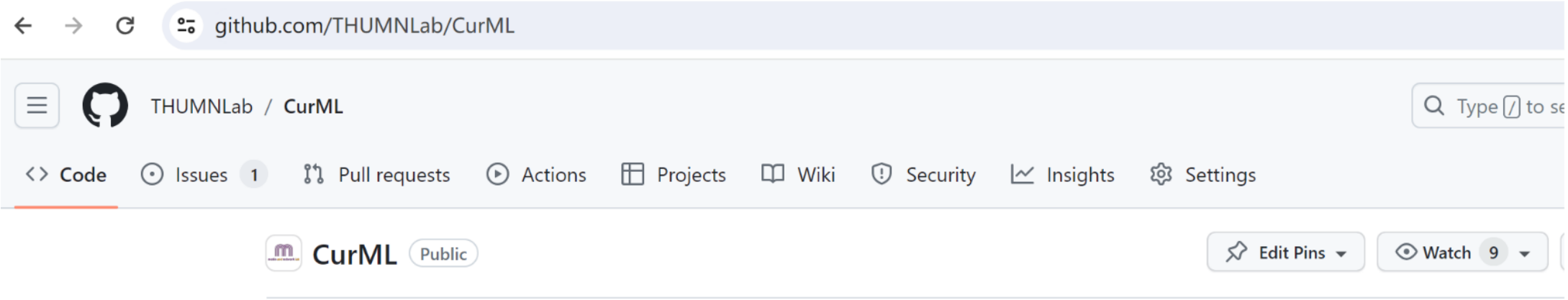
Multi-task Graph Neural Architecture Search with Task-aware Collaboration and Curriculum. NeurIPS 2023.

Tool and Benchmark

Tool: CurML

Introduction

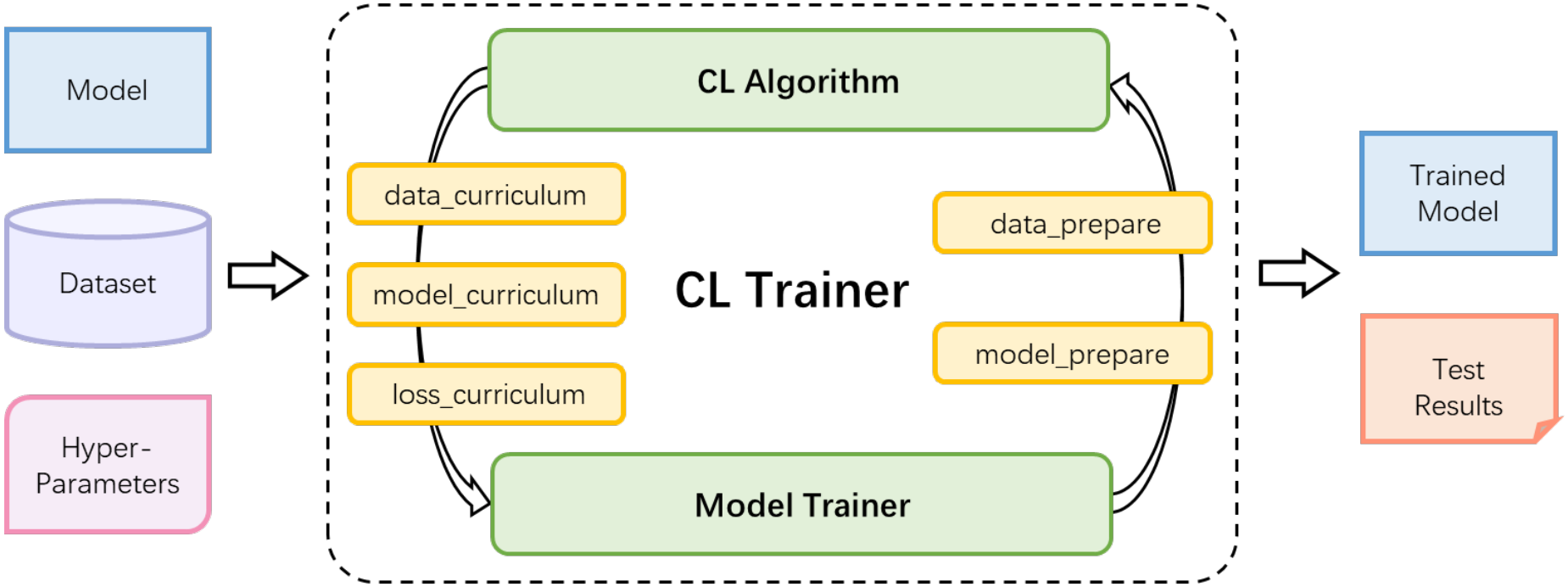
- As the approaches and applications of curriculum learning become increasingly diverse, it is more difficult to **reproduce, evaluate, compare and choose** the related methods.
- Therefore, we develop **Curriculum Machine Learning (CurML),** the first public open-source library for CL, which implements a considerable number of the existing CL algorithms through a extensible framework.



Tool: CurML

Framework

- The main body of the framework is a **CL Trainer** class, composed of a **CL Algorithm** class and a **Model Trainer** class, which interact with each other through **five key APIs** of the CL Algorithm class.

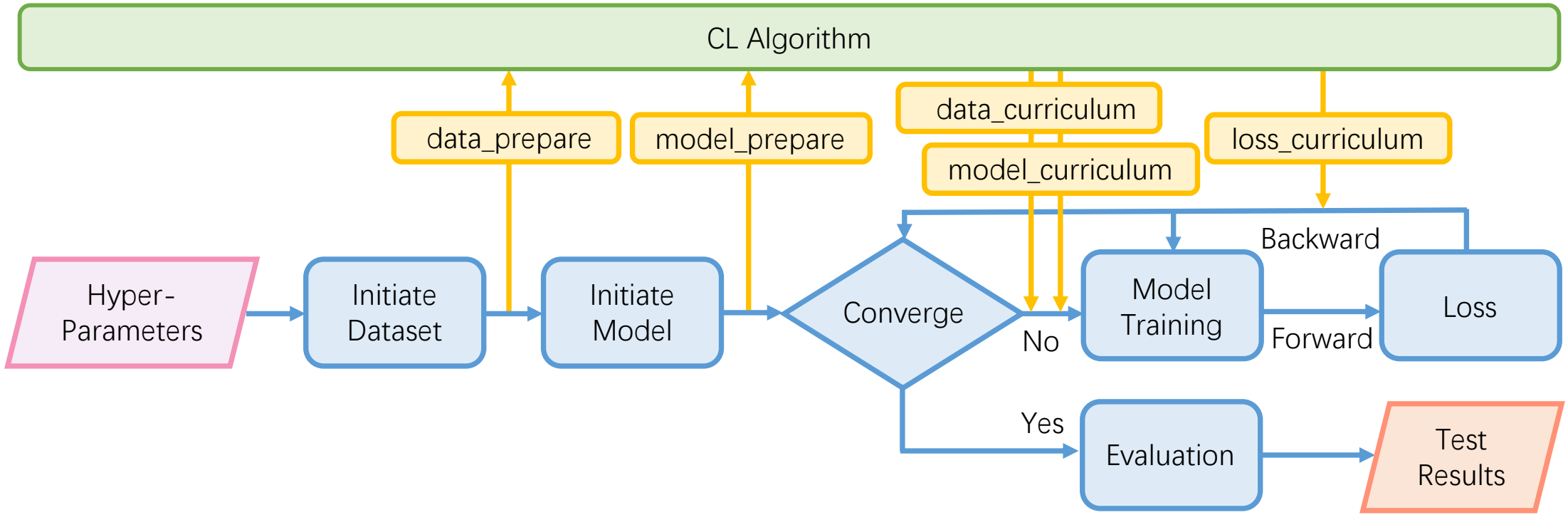


CurML: A Curriculum Machine Learning Library. ACMMM 2022.

Tool: CurML

Workflow

- In this workflow, **two prepare APIs** are called at the initiation and preparation stage and **three curriculum APIs** are called in every epoch of the training process.



CurML: A Curriculum Machine Learning Library. ACMMM 2022.

Tool: CurML

Usage

Example 1: Baby Step

```
trainer = BabyStepTrainer(
        data_name='cifar10',
        net_name='resnet',
        device_name='cuda:0',
        num_epochs=200,
        random_seed=42,
        start_rate=0.0,
        grow_rate=0.1,
        grow_interval=20,
)
trainer.fit()
trainer.evaluate()
```

Example 2: SuperLoss

```
trainer = SuperlossTrainer(
        data_name='cifar10-noise-0.4',
        net_name='efficientnet',
        device_name='cuda:1',
        num_epochs=200,
        random_seed=43,
        tau=math.log(10),
        lam=1.0,
        fac=0.0,
)
trainer.fit()
trainer.evaluate()
```

Benchmark: CurBench

Background

- Different methods vary in experimental setups, such as datasets and models.

Method	Conference	Involved Datasets	Involved Models
SPL	NeurIPS 2010	MUC6, UniProbe, MNIST, Mammals	SSVM
MCL	ICLR 2018	MNIST, CIFAR-10, STL-10, SVHN, …	LeNet5, CNN
DCL	NeurIPS 2019	CIFAR-10, CIFAR-100,	VGG-16, SSDNet, Re
DIHCL	NeurIPS 2020	CIFAR-10, CIFAR-100, ImageNet, …	ResNet-50, WideResNet-16-8, …
Superloss	NeurIPS 2020	MNIST, CIFAR-10, CIFAR-100, …	ResNet-18, ResNet-50, …
CBS	NeurIPS 2020	CIFAR-10, CIFAR-100, ImageNet, …	VGG-16, ResNet-18, …
DDS	ICML 2020	CIFAR0-10, ImageNet, TED Dataset	WideResNet-28, ResNet- 50, LSTM
Adaptive CL	ICCV 2021	CIFAR-10, CIFAR-100, ImageNet, …	VGG-16, ResNet-18, …

Same

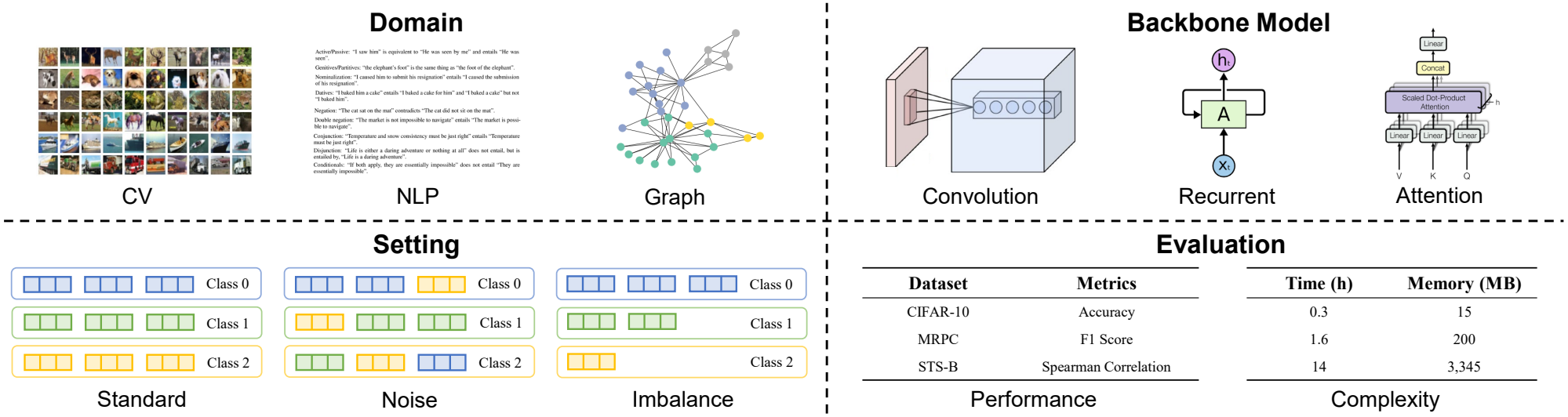
Different

CurBench: Curriculum Learning Benchmark. ICML 2024.

Benchmark: CurBench

Introduction

- To benchmark various curriculum learning algorithms in a systematic way, we develop CurBench on basis of CurML.
  - 15 datasets spanning 3 research domains: CV, NLP, and graph;
  - 3 settings: standard, noise, and imbalance;
  - 9 backbone models: LeNet, ResNet, ViT, BERT, GPT, GCN, GAT, and GIN;

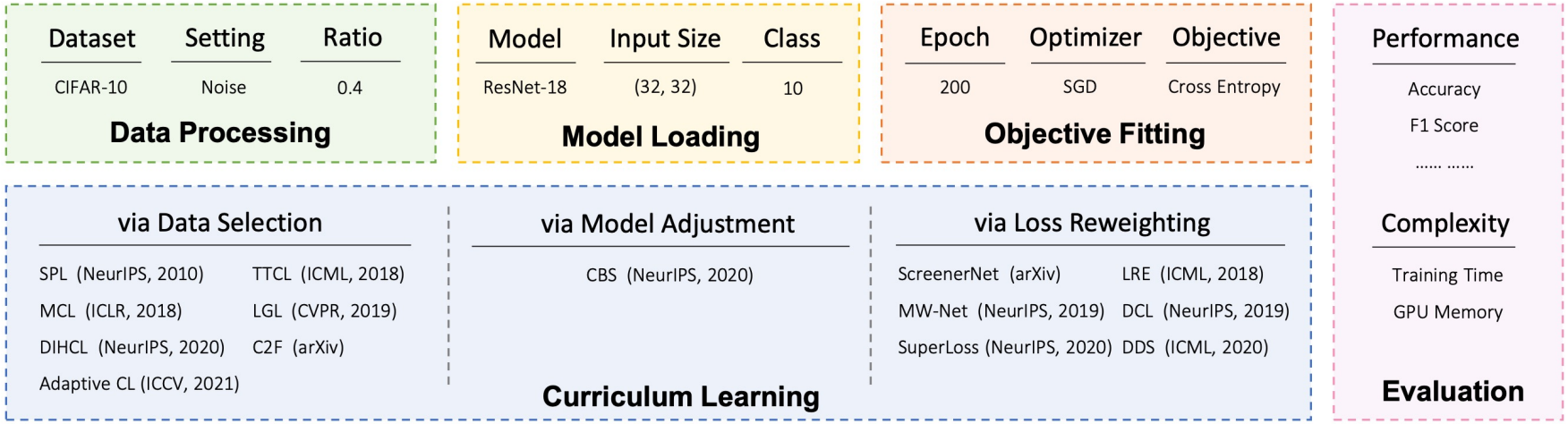


CurBench: Curriculum Learning Benchmark. ICML 2024.

Benchmark: CurBench

Toolkit

- For the easy use of CurBench, a companion toolkit is developed based on CurML for the entire pipeline of applying curriculum learning to various machine learning tasks.




```
from curbench.algorithms import SPLTrainer

# Instantiate curriculum learning class
trainer = SPLTrainer(
    # CIFAR-10 with 40% wrong labels
    data_name='cifar10-noise-0.4',
    # ResNet-18 with 32×32 input size
    net_name='resnet18',
    # Self-Paced Learning in a linear way
    start_rate=0.0,
    grow_epochs=100,
    grow_fn='linear',
    weight_fn= 'hard',
)
# Automatic, no need to specify:
# trainer._init_dataloader()
# trainer._init_model()

# Fitting and evaluating
trainer.fit()
trainer.evaluation()
```

CurBench: Curriculum Learning Benchmark. ICML 2024.

Benchmark: CurBench

- We obtain empirical performances of backbone models over different datasets in different settings with and without curriculum learning.

CV	CIFAR-10			CIFAR-100			Tiny-ImageNet		
	Standard	Noise-0.4	Imbalance-50	Standard	Noise-0.4	Imbalance-50	Standard	Noise-0.4	Imbalance-50
LeNet	$69.95_{1.00}$	$65.02_{1.12}$	$44.93_{0.56}$	$35.46_{0.70}$	$29.59_{0.40}$	$19.57_{0.64}$	$22.08_{0.61}$	$18.63_{0.43}$	$11.65_{0.30}$
LeNet + CL	$\mathbf{70.43}_{0.41}$	$\mathbf{65.93}_{0.57}$	$\mathbf{45.28}_{0.56}$	$\mathbf{35.63}_{0.78}$	$\mathbf{30.87}_{0.48}$	$\mathbf{19.74}_{0.17}$	$\mathbf{22.83}_{0.44}$	$\mathbf{19.91}_{0.26}$	$\mathbf{11.71}_{0.19}$
ResNet-18	$92.33_{0.16}$	$82.75_{2.06}$	$75.49_{0.87}$	$69.97_{0.27}$	$52.14_{0.39}$	$42.57_{0.68}$	$51.41_{1.74}$	$39.42_{0.21}$	$28.83_{0.38}$
ResNet-18 + CL	$\mathbf{92.74}_{0.20}$	$\mathbf{86.92}_{0.20}$	$\mathbf{76.43}_{0.96}$	$\mathbf{71.31}_{0.14}$	$\mathbf{58.56}_{0.60}$	$\mathbf{43.47}_{0.43}$	$\mathbf{53.61}_{0.48}$	$\mathbf{43.64}_{0.72}$	$\mathbf{30.82}_{0.36}$
ViT	$79.90_{0.38}$	$64.19_{0.51}$	$52.12_{0.81}$	$51.05_{0.62}$	$35.25_{0.24}$	$26.05_{0.52}$	$38.16_{0.53}$	$24.90_{0.26}$	$17.15_{0.31}$
ViT + CL	$\mathbf{80.66}_{0.27}$	$\mathbf{69.83}_{0.53}$	$\mathbf{52.85}_{0.81}$	$\mathbf{51.93}_{0.64}$	$\mathbf{39.15}_{0.30}$	$\mathbf{26.40}_{0.34}$	$\mathbf{38.92}_{0.53}$	$\mathbf{29.76}_{0.34}$	$\mathbf{17.47}_{0.14}$

Graph	MUTAG		PROTEINS		NCI1		ogbg-molhiv	
	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4
GCN	$73.68_{2.11}$	$66.31_{7.14}$	$70.71_{4.20}$	$63.57_{6.45}$	$69.59_{1.23}$	$55.23_{3.21}$	$75.84_{1.02}$	$64.29_{4.55}$
GCN + CL	$\mathbf{74.74}_{3.94}$	$\mathbf{71.58}_{5.37}$	$\mathbf{73.21}_{4.41}$	$\mathbf{71.61}_{6.62}$	$\mathbf{71.39}_{1.29}$	$\mathbf{67.98}_{2.01}$	$\mathbf{77.41}_{1.15}$	$\mathbf{72.81}_{1.14}$
GAT	$69.47_{6.14}$	$65.26_{5.37}$	$64.46_{2.96}$	$65.71_{9.13}$	$56.74_{2.86}$	$53.77_{2.12}$	$68.07_{2.34}$	$65.37_{2.66}$
GAT + CL	$\mathbf{72.63}_{8.42}$	$\mathbf{69.47}_{10.21}$	$\mathbf{69.82}_{7.13}$	$\mathbf{69.11}_{3.77}$	$\mathbf{59.37}_{1.59}$	$\mathbf{55.67}_{4.70}$	$\mathbf{72.64}_{1.16}$	$\mathbf{66.73}_{1.84}$
GIN	$86.84_{7.90}$	$78.95_{3.72}$	$74.11_{4.24}$	$69.82_{1.73}$	$79.32_{1.40}$	$60.24_{3.92}$	$74.72_{1.36}$	$63.07_{3.73}$
GIN + CL	$\mathbf{88.42}_{2.10}$	$\mathbf{81.58}_{4.56}$	$\mathbf{77.14}_{4.88}$	$\mathbf{73.93}_{1.82}$	$\mathbf{82.04}_{1.90}$	$\mathbf{62.14}_{6.47}$	$\mathbf{76.53}_{1.97}$	$\mathbf{65.53}_{1.61}$

CurBench: Curriculum Learning Benchmark. ICML 2024.

Benchmark: CurBench

- We obtain empirical performances of backbone models over different datasets in different settings with and without curriculum learning.

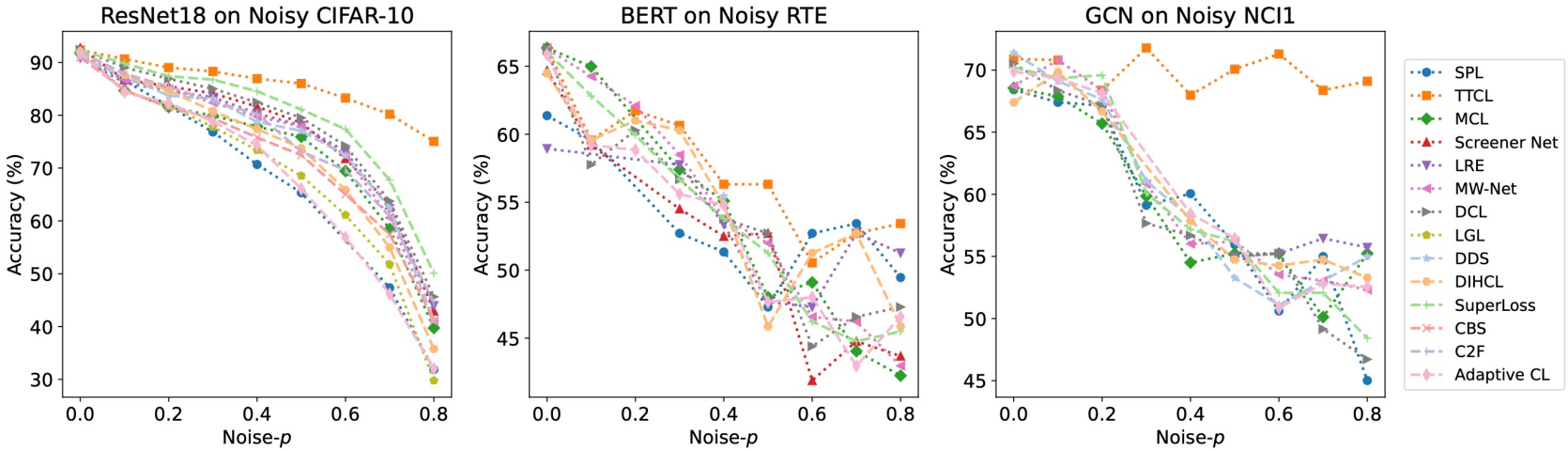
NLP	RTE		MRPC		STS-B		CoLA	
	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4
LSTM	$52.95_{1.34}$	$53.43_{1.77}$	$81.43_{0.14}$	$81.22_{0.00}$	$12.73_{0.72}$	$10.90_{1.19}$	$11.29_{1.27}$	$3.27_{1.68}$
LSTM + CL	$\mathbf{53.07}_{1.29}$	$\mathbf{54.22}_{1.77}$	$\mathbf{81.54}_{0.18}$	$\mathbf{81.24}_{0.05}$	$\mathbf{14.11}_{2.21}$	$\mathbf{11.75}_{1.61}$	$\mathbf{12.65}_{1.21}$	$\mathbf{8.55}_{2.10}$
BERT	$64.62_{3.33}$	$54.22_{3.14}$	$88.54_{0.45}$	$81.89_{0.83}$	$85.26_{0.22}$	$80.71_{1.01}$	$57.39_{1.30}$	$32.35_{0.79}$
BERT + CL	$\mathbf{66.35}_{1.76}$	$\mathbf{56.32}_{5.04}$	$\mathbf{88.69}_{1.24}$	$\mathbf{81.94}_{0.55}$	$\mathbf{85.42}_{0.22}$	$\mathbf{81.31}_{0.25}$	$\mathbf{57.80}_{1.96}$	$\mathbf{45.79}_{1.64}$
GPT2	$65.34_{1.95}$	$52.92_{4.49}$	$85.49_{0.86}$	$78.23_{1.72}$	$76.44_{1.20}$	$69.65_{1.85}$	$37.00_{3.72}$	$5.86_{1.69}$
GPT2 + CL	$\mathbf{66.35}_{2.10}$	$\mathbf{57.40}_{3.39}$	$\mathbf{86.29}_{0.36}$	$\mathbf{82.55}_{0.88}$	$\mathbf{80.82}_{1.39}$	$\mathbf{71.57}_{1.74}$	$\mathbf{39.95}_{3.16}$	$\mathbf{12.54}_{2.75}$

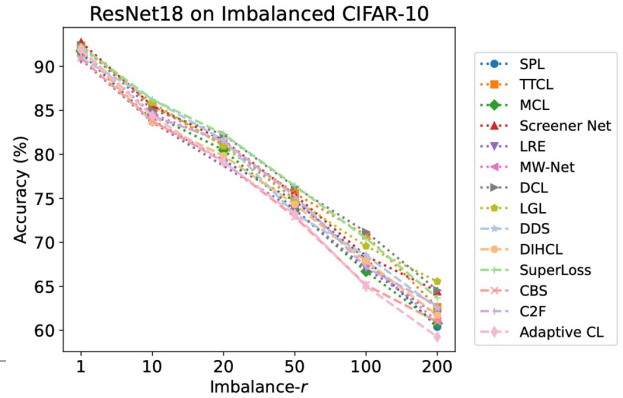
NLP	SST-2		QNLI		QQP		MNLI-(m/mm)	
	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4	Standard	Noise-0.4
LSTM	$81.67_{0.85}$	$64.36_{1.12}$	$50.54_{0.00}$	$50.62_{0.16}$	$75.69_{0.27}$	$60.72_{0.79}$	$61.38_{0.30}$ / $61.21_{0.45}$	$44.41_{0.51}$ / $44.83_{0.90}$
LSTM + CL	$\mathbf{82.87}_{0.88}$	$\mathbf{78.58}_{1.64}$	$\mathbf{51.02}_{0.46}$	$\mathbf{50.83}_{0.45}$	$\mathbf{75.73}_{0.21}$	$\mathbf{66.47}_{0.72}$	$\mathbf{62.47}_{0.36}$ / $\mathbf{62.33}_{0.42}$	$\mathbf{58.59}_{0.54}$ / $\mathbf{58.50}_{0.64}$
BERT	$92.66_{0.28}$	$87.22_{0.82}$	$91.21_{0.24}$	$81.21_{0.76}$	$88.05_{0.12}$	$76.23_{0.48}$	$83.89_{0.31}$ / $84.38_{0.29}$	$78.65_{0.70}$ / $79.21_{0.62}$
BERT + CL	$\mathbf{92.82}_{0.16}$	$\mathbf{91.25}_{0.59}$	$\mathbf{91.49}_{0.13}$	$\mathbf{89.45}_{0.44}$	$\mathbf{88.16}_{0.13}$	$\mathbf{84.50}_{0.25}$	$\mathbf{84.27}_{0.07}$ / $\mathbf{84.40}_{0.42}$	$\mathbf{81.73}_{0.31}$ / $\mathbf{82.25}_{0.40}$
GPT2	$91.95_{0.49}$	$85.83_{0.57}$	$87.92_{0.31}$	$78.72_{0.37}$	$86.00_{0.23}$	$75.40_{0.84}$	$81.53_{0.21}$ / $82.40_{0.21}$	$76.56_{0.15}$ / $77.69_{0.15}$
GPT2 + CL	$\mathbf{92.25}_{0.42}$	$\mathbf{90.34}_{0.53}$	$\mathbf{88.17}_{0.67}$	$\mathbf{84.00}_{0.70}$	$\mathbf{86.68}_{0.16}$	$\mathbf{82.16}_{0.35}$	$\mathbf{81.90}_{0.23}$ / $\mathbf{82.59}_{0.35}$	$\mathbf{78.36}_{0.19}$ / $\mathbf{79.62}_{0.44}$

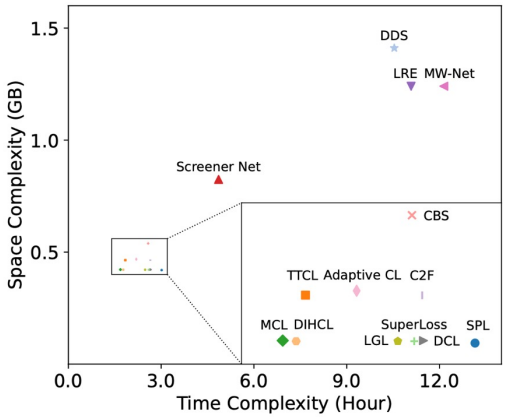
CurBench: Curriculum Learning Benchmark. ICML 2024.

Benchmark: CurBench

- Besides, experiments on noisy and imbalanced setting are conducted, as well as time and space complexity analysis.





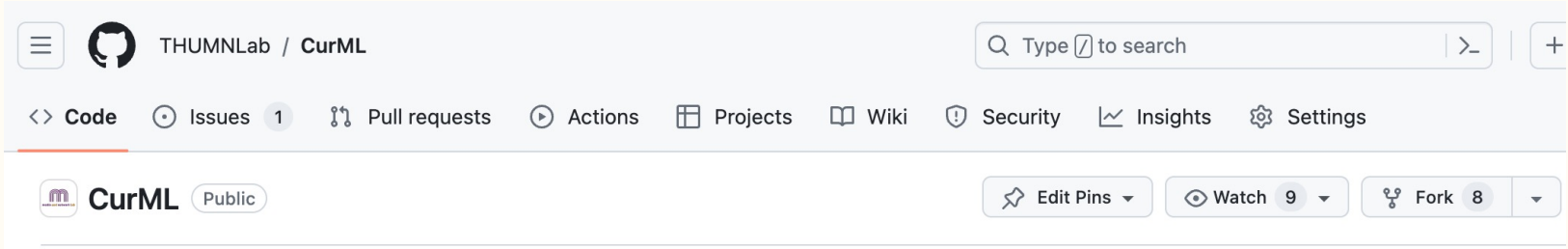


Benchmark: CurBench

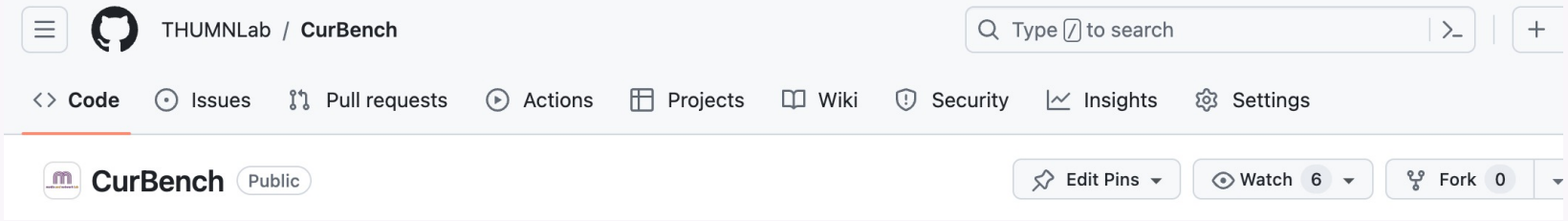
- We obtain some intriguing findings based on the empirical results:
  - There has been **no such method that outperforms others all the time**, and the effectiveness depends on specific scenarios.
  - Curriculum learning brings **more significant improvements in noise setting** than in standard and imbalance ones.
  - Methods by **teacher transferring have edges in the noise setting**, while methods by **reweighting perform relatively well in the imbalance setting**.
  - Methods involving gradient calculation and extra learnable networks generally have higher time and space complexity.

Tool and Benchmark

- Welcome to try our library and benchmark for curriculum learning~



https://github.com/THUMNLab/CurML



https://github.com/THUMNLab/CurBench

Discussion

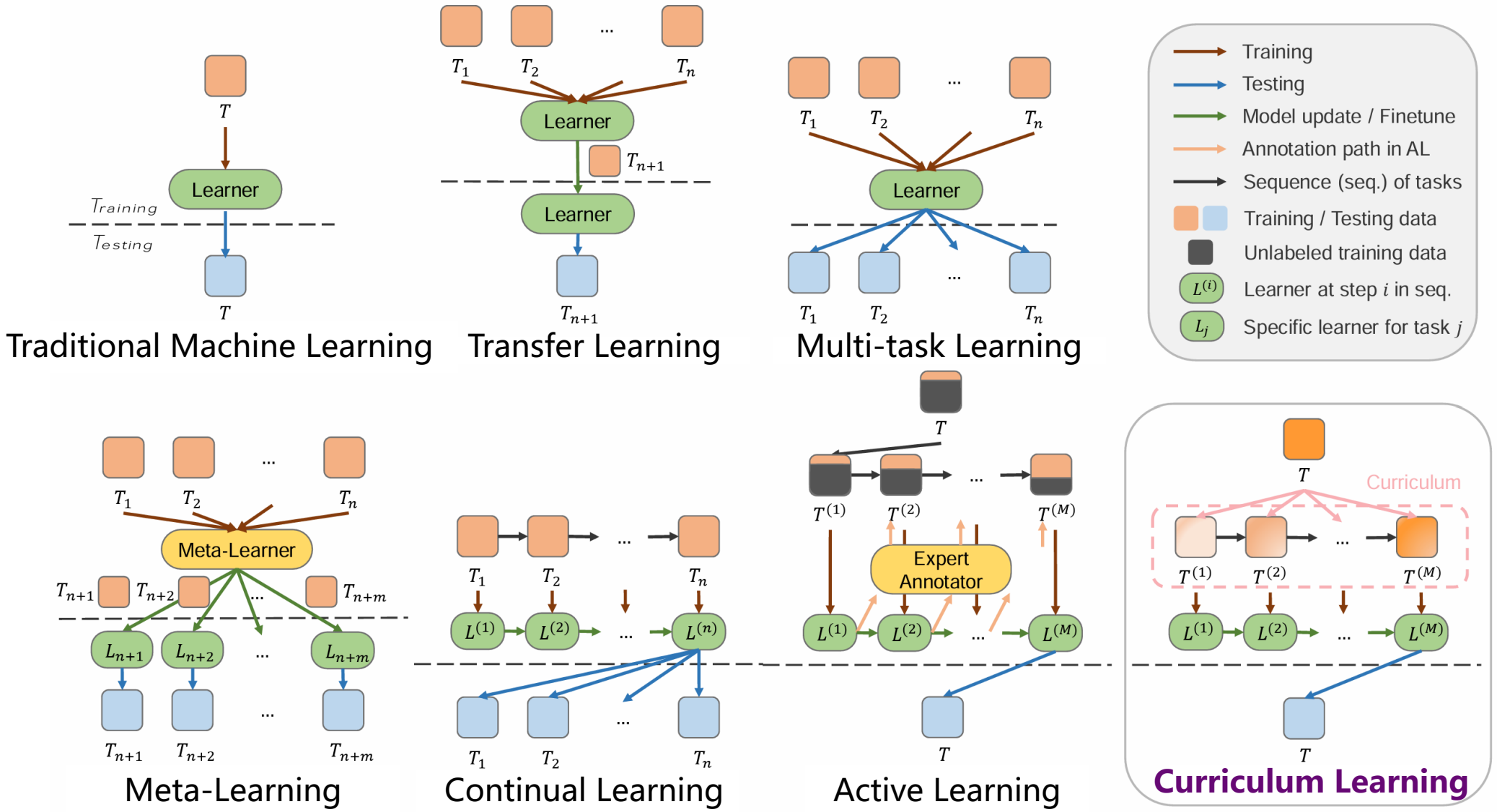
Does “easy to hard” always help?

- **Hard sample mining** (hard to easy) succeeds in many fields (e.g. object detection, boosting algorithm, etc.)
  - Difficulty defined by current loss / gradient magnitude, etc.
  - Literature also shows effectiveness of anti-curriculum.

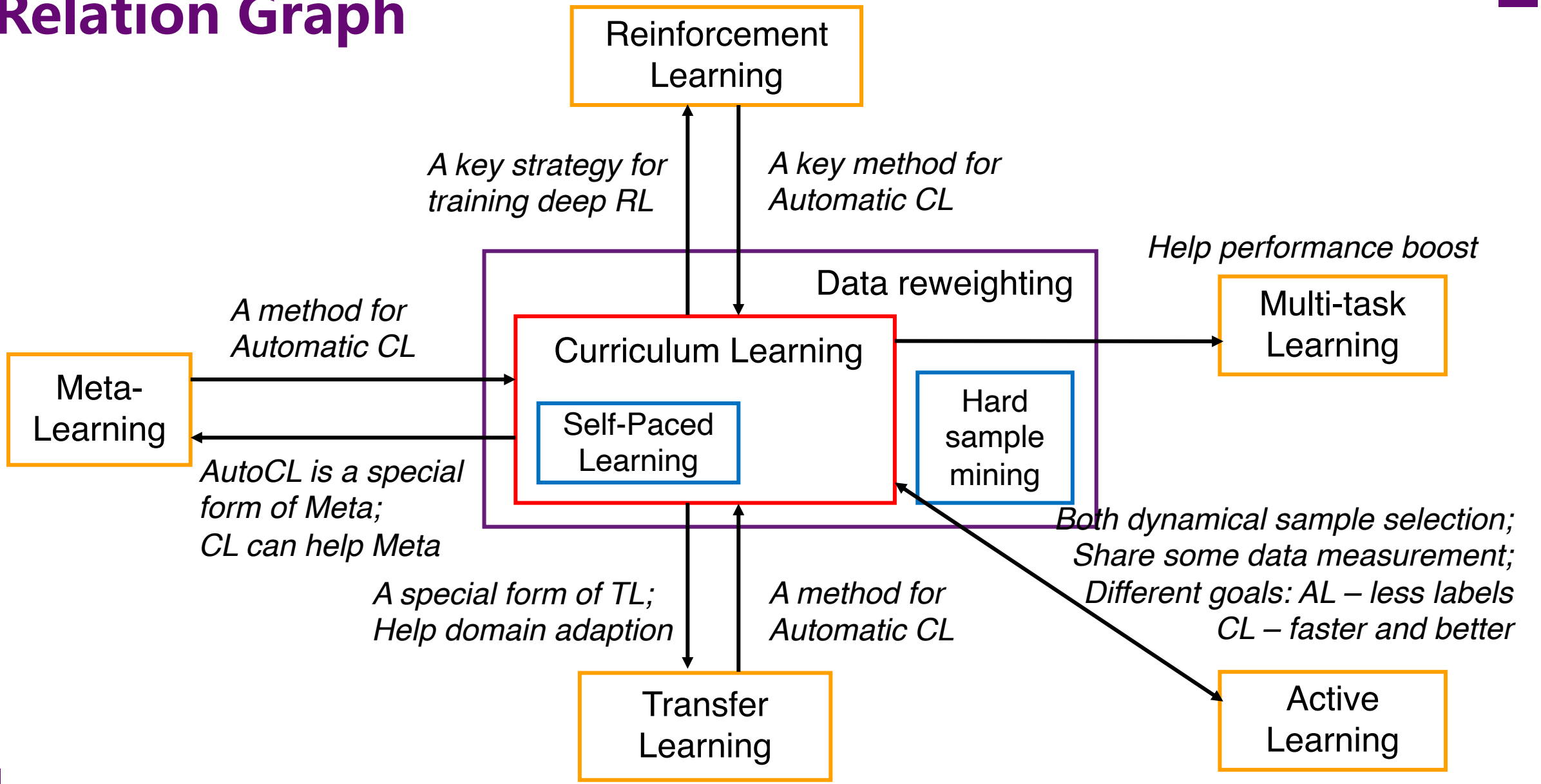
	Easier first (CL)	**Harder First (HSM)**
Suitable noisy level	Many noisy labels and outliers	Cleaner data
Suitable task difficulty	More difficult	More easy
Accelerate	√ (denoise, guide)	√ (informative, guide)
Better performance	√ (denoise, guide)	√ (informative, guide)

- However, there is currently no clear rule indicating when to use Curriculum Learning (CL) or when to use Hard Sample Mining (HSM).

Differences from Other Learning



Relation Graph

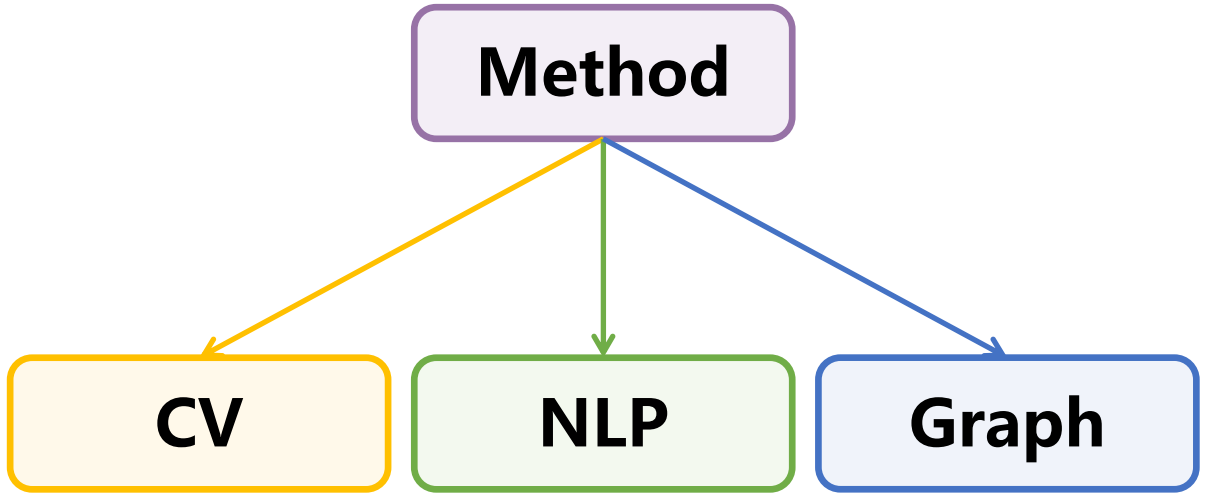


Summary

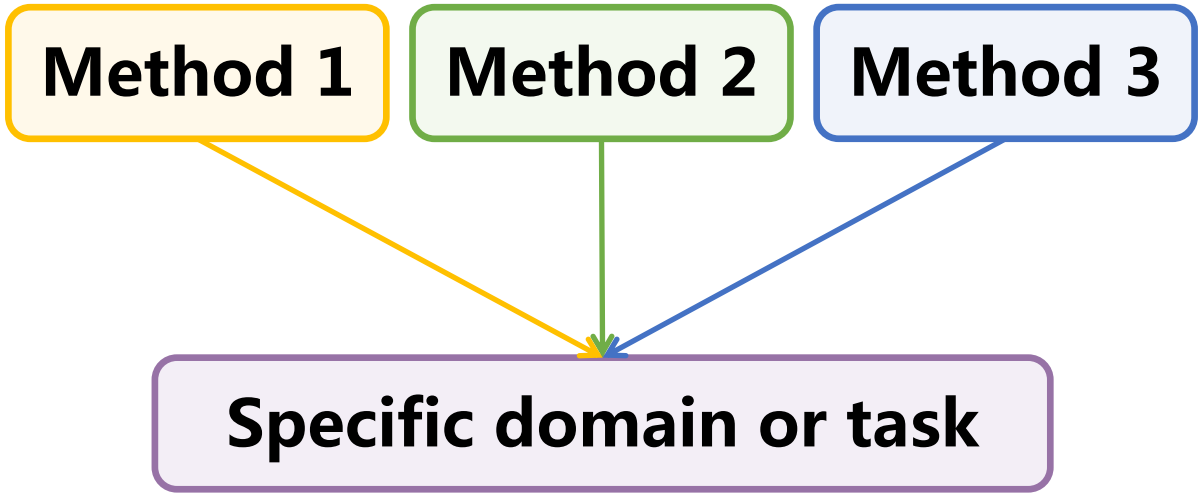
- Curriculum learning is about how to teach a model.
- Predefined or automatic curriculum differs mainly in the way of teaching.
  - Automatic methods are typically more flexible and general.
  - Predefined methods may also yield effective results.
- Try to find hyperparameters of teaching:
  - Data selection — Teaching material
  - Weights of data — Importance of materials
  - Training period — Semester
  - Loss function — Exam design
  - ……
- The collaboration between teaching & learning makes perfect.

Future Directions

Future Directions: Application Benchmarks



CurBench



Application Benchmark

Future Directions: More Advanced Theories

- Existing theoretical analyses provide different angles for understanding the curriculum.

Convergence of Risk

Theorem 4.6. *Suppose $\{\mathbf{z}_i\}_{i=1}^m$ are i.i.d. samples drawn from the pace distribution Q_λ. Let m_+/m_- be the number of positive/nagetive samples, $m^* = \min\{m_-, m_+\}$, and $\mathcal{H}$ the function family projecting to $\{-1,1\}$. Then for any $\delta > 0$ and $f \in \mathcal{H}$, with confidence at least $1-2\delta$ we have:*

$$\mathcal{R}(f) \le \frac{1}{2}\mathcal{R}^+_{emp}(f) + \frac{1}{2}\mathcal{R}^-_{emp}(f) + \frac{1}{2}\mathfrak{R}_{m_+}(\mathcal{H}) + \frac{1}{2}\mathfrak{R}_{m_-}(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{m^*}} + (1-\alpha_\lambda)\sqrt{1-\exp\{-D_{KL}(P^+_{\text{target}} \,\|\, E^+)\}} + (1-\alpha_\lambda)\sqrt{1-\exp\{-D_{KL}(P^-_{\text{target}} \,\|\, E^-)\}}, \quad (14)$$

and

$$\mathcal{R}(f) \le \frac{1}{2}\mathcal{R}^+_{emp}(f) + \frac{1}{2}\mathcal{R}^-_{emp}(f) + \frac{1}{2}\hat{\mathfrak{R}}_{m_+}(\mathcal{H}) + \frac{1}{2}\hat{\mathfrak{R}}_{m_-}(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{m^*}} + (1-\alpha_\lambda)\sqrt{1-\exp\{-D_{KL}(P^+_{\text{target}} \,\|\, E^+)\}} + (1-\alpha_\lambda)\sqrt{1-\exp\{-D_{KL}(P^-_{\text{target}} \,\|\, E^-)\}}, \quad (15)$$

where E^+, E^- denote the error distribution corresponding to P^+_{target}, P^-_{target}, and $\mathcal{R}^+_{emp}(f)$, $\mathcal{R}^-_{emp}(f)$ denote the empirical risk on positive samples and negative samples, respectively.

Optimization Landscape

Proposition 1 *The difference between the expected utility when computed with and without prior $\mathbf{p}$ is the covariance between the two random variables $U_\vartheta(X)$ and $p(X)$.*

Proposition 2 *For any curriculum satisfying (5):*

1. $\tilde{\vartheta} = \arg\max_{\vartheta} \mathcal{U}_p(\vartheta)$
2. $\mathcal{U}_p(\tilde{\vartheta}) - \mathcal{U}_p(\vartheta) \ge \mathcal{U}(\tilde{\vartheta}) - \mathcal{U}(\vartheta) \quad \forall \vartheta$

Corollary 1 [illegible] *proposition 2 holds if the variance of the utility function is roughly constant in the relevant range of plausible parameter values.*

From (6) and (7) we can also conclude the following

Proposition 3 *When using the ideal curriculum*

$$\mathcal{U}_p(\tilde{\vartheta}) - \mathcal{U}_p(\vartheta) \ge \mathcal{U}(\tilde{\vartheta}) - \mathcal{U}(\vartheta) \quad \forall \vartheta : \mathrm{Cov}[U_\vartheta, U_{\tilde{\vartheta}}] \le \mathrm{Var}[U_{\tilde{\vartheta}}]$$

Convergence Rate

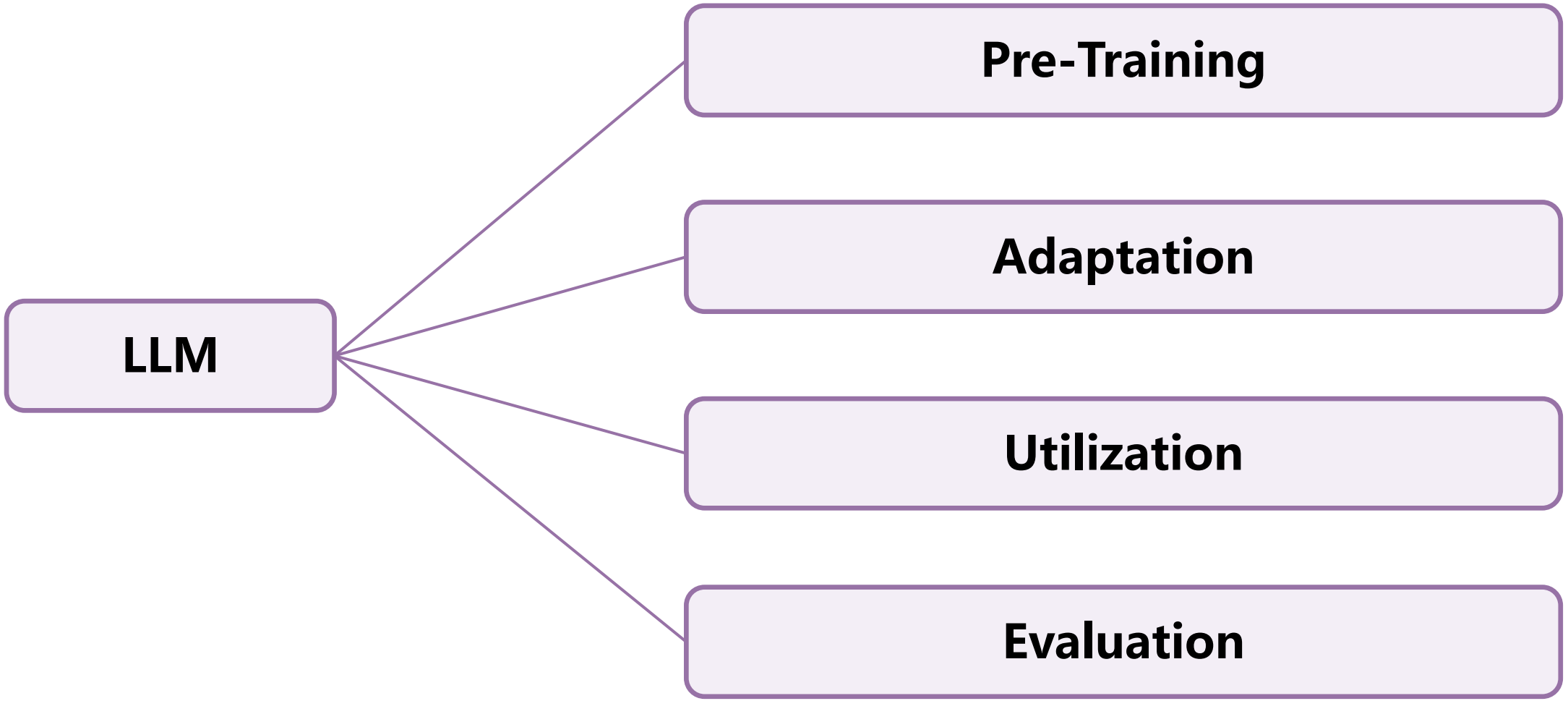
Theorem 1. *At time t the expected convergence rate for training point $\mathbf{x}$ is monotonically decreasing with the Difficulty Score Ψ of $\mathbf{x}$. If the step size coefficient is sufficiently small so that $\eta \le \frac{\mathbb{E}[r^2\cos^2\vartheta]}{\mathbb{E}[r^4\cos^2\vartheta]}$, it is likewise monotonically increasing with the distance λ between the current estimate of the hypothesis $\mathbf{w}_t$ and the optimal hypothesis $\bar{\mathbf{w}}$.*

Theorem 2. *Assume that the gradient step size is small enough so that we can neglect second order terms $O(\eta^2)$, and that $\frac{\partial \nabla}{\partial \Upsilon} \ge \frac{\psi}{\Upsilon} - \Upsilon \; \forall \Upsilon$. Fix the difficulty score at Ψ. At time t the expected [illegible] rate is monotonically increasing with the loss Υ of the training point $\mathbf{x}$.*

Theorem 3. *Assume that $\mathcal{D}(\mathbb{X})$ is continuous and that $\bar{\mathbf{w}}$ is realizable. Then there are always hypotheses $\mathbf{w} \in \mathcal{H}$ for which the expected convergence rate under $\mathcal{D}(\mathbb{X})$ is monotonically decreasing with the loss Υ of the sampled points.*

- However, more theories are still required to help us reveal why typical CL is effective:
  - If the dataset has no noise, are there any bounds for the effectiveness of CL?
  - What is the actual effect of increasing dataset size/variance and increasing difficulty?
  - What is the relation between the data distribution, task objective, and the best training strategy among "easier first" (CL), "harder first" (HEM)?

Future Directions: Application on LLM



A Survey of Large Language Models. ArXiv.

Future Directions: Application on LLM

LLM

Pre-Training

Motivation: There is **a wide variation in difficulty** among the data, containing a significant amount of noise, irrelevant, and harmful information.

Challenge: Due to the enormous volume of data, it is impractical to **rank all data** by difficulty or assign **weights to each data point**.

Adaptation

Utilization

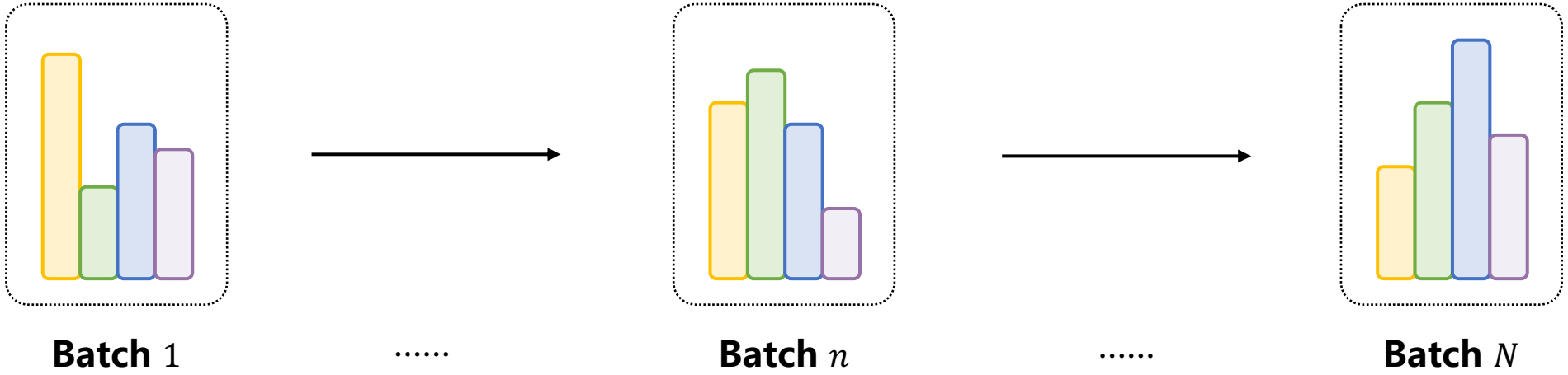
Evaluation

A Survey of Large Language Models. ArXiv.

Future Directions: Application on LLM

Curriculum for Pre-training

- **Example: Batch Difficulty Measurement**
  - Consider deciding data weights in each batch instead of the whole training process, according to data feature, data source, model state, model loss, etc.



Future Directions: Application on LLM

LLM

Pre-Training

Adaptation (e.g., Fine-Tuning)

Motivation: **Similarities or differences between the distribution of fine-tuning data and pre-training data** may lead to variations in the model's handling of data difficulty.

Challenge: **Deciding the difficulty of data within a limited number of iterations** and designing a learning sequence accordingly is a challenging task.

Utilization

Evaluation

A Survey of Large Language Models. ArXiv.

Future Directions: Application on LLM

Curriculum for Adaptation (e.g., Fine-tuning)

- **Example:** Data distribution from being close to pre-trained stage to the specific task.

- A clear and well-documented LATEX document is presented as an article formatted for publication by ACM in a conference proceedings or journal.
- Based on the “acmart” document class, this article presents and explains many of the common variations, as well as many of the formatting elements.
- ACM’s consolidated article template, provides a consistent LATEX style for use across ACM publications, and incorporates Digital Library endeavors.

Pre-training Stage

- Numerous ACM and SIG-specific LATEX templates have been examined, and their unique features incorporated into this single new template.
- If you are new to publishing with ACM, this document is a valuable guide to the process of preparing your work for publication.
- If you have published with ACM before, this document provides insight and instruction into more recent changes to the article template.

Fine-tuning Stage

- Each author must be defined separately for accurate metadata identification.
- Authors’ names should not be abbreviated; use full first names wherever possible.
- Grouping authors’ names or e-mail addresses is not acceptable.

- Include authors’ e-mail addresses whenever possible.
- Multiple authors may share one affiliation.

Future Directions: Application on LLM

LLM

Pre-Training

Adaptation

Utilization (e.g., Prompt)

Motivation: The concept of “Chain-of-Thought” suggests that breaking down tasks into smaller steps for large language models is an effective approach.

Challenge: Prompt design requires manual effort and practical experience, consuming considerable time and energy.

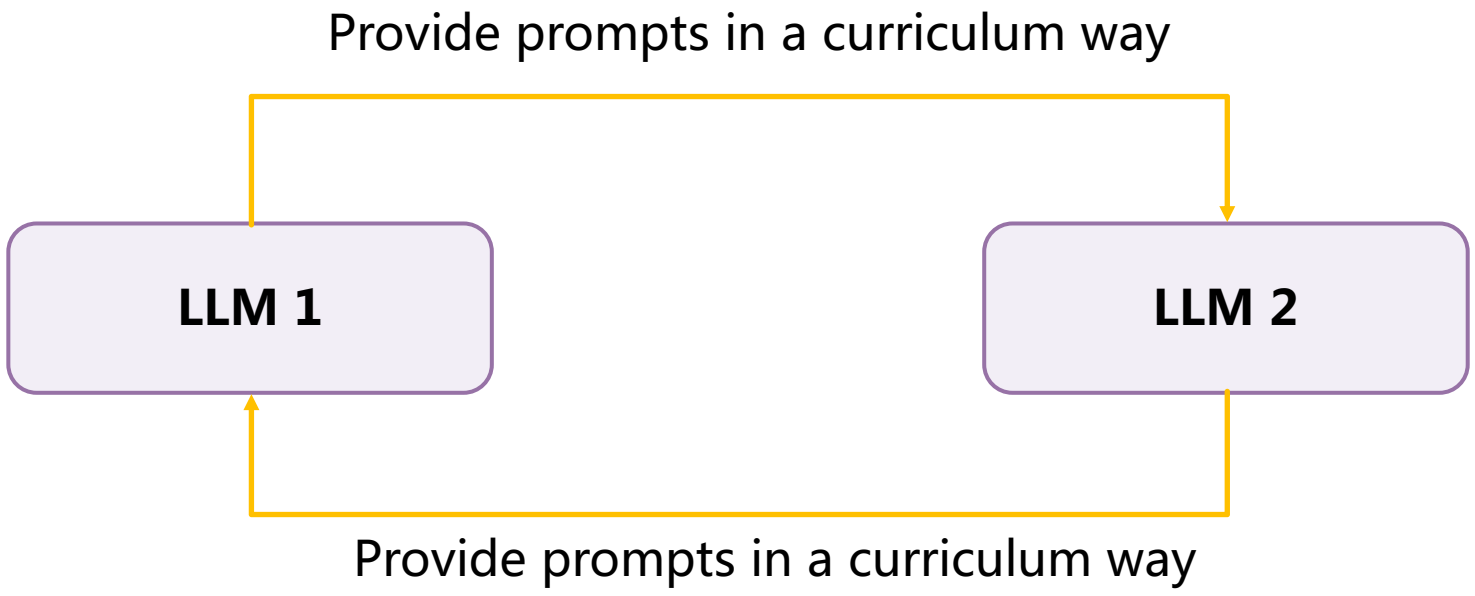
Evaluation

A Survey of Large Language Models. ArXiv.

Future Directions: Application on LLM

Curriculum for Utilization (e.g., Prompt)

- **Example: Dual Distillation**
  - Consider using one LLM to providing suitable task prompts for the other LLM, and the latter can also design prompts for the former, implementing a dual distillation.



Future Directions: Application on LLM

LLM

- **Pre-Training**
- **Adaptation**
- **Utilization**
- **Evaluation**

Motivation: Different capabilities correspond to different levels of difficulty:

Basic: Language Generation, Knowledge Utilization, Complex Reasoning, etc.

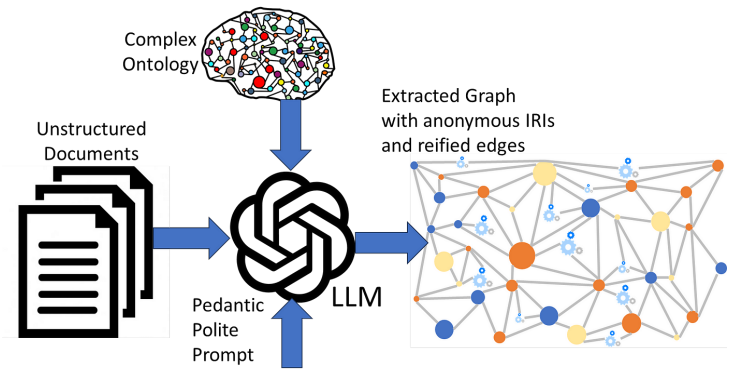
Advanced: Human Alignment, Interaction with External Environment, Tool Manipulation, etc.

Challenge: How to design evaluation and benchmark in a progressive way.

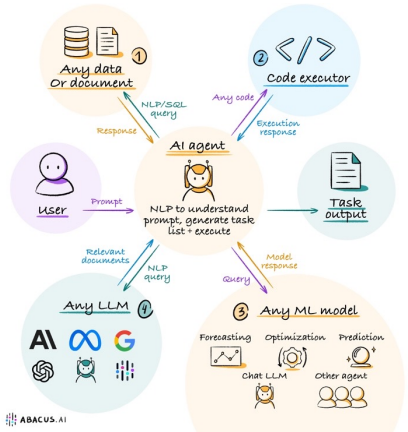
Future Directions: Application on LLM

Curriculum for Evaluation

- **Example 1: Phased Evaluation**
  - Observing the performance of the model as it processes increasingly complex data.
- **Example 2: Capability Benchmarking**
  - Designing a series of benchmarks across various levels of difficulty.
- **Example 3: Adaptive Capacity Testing**
  - Gradually introducing new or unseen types of tasks during the evaluation, and observing how quickly the model can adapt to novel tasks.



Knowledge Utilization



Tool Manipulation

Thanks for listening!

Any questions are welcome

Xin Wang
xin_wang@tsinghua.edu.cn
https://mn.cs.tsinghua.edu.cn/xinwang

Yuwei Zhou
zhou-yw21@mails.tsinghua.edu.cn
https://github.com/zhouyw16

Hong Chen
h-chen20@mails.tsinghua.edu.cn
https://github.com/forchchch

Wenwu Zhu
wwzhu@tsinghua.edu.cn
https://scholar.google.com/citations
?user=7t2jzpgAAAAJ