Curriculum Learning for Graph Neural Networks: A Multiview Competence-based Approach

Nidhi Vakil
Department of Computer Science
University of Massachusetts Lowell
nvakil@cs.uml.edu

Hadi Amiri
Department of Computer Science
University of Massachusetts Lowell
hadi@cs.uml.edu

Abstract

A curriculum is a planned sequence of learning materials and an effective one can make learning efficient and effective for both humans and machines. Recent studies developed effective data-driven curriculum learning approaches for training graph neural networks in language applications. However, existing curriculum learning approaches often employ a single criterion of difficulty in their training paradigms. In this paper, we propose a new perspective on curriculum learning by introducing a novel approach that builds on graph complexity formalisms (as difficulty criteria) and model competence during training. The model consists of a scheduling scheme which derives effective curricula by accounting for different views of sample difficulty and model competence during training. The proposed solution advances existing research in curriculum learning for graph neural networks with the ability to incorporate a fine-grained spectrum of graph difficulty criteria in their training paradigms. Experimental results on real-world link prediction and node classification tasks illustrate the effectiveness of the proposed approach.[1]

1 Introduction

Graph Neural Networks (GNNs) are generally trained using stochastic gradient descent (SGD), where the standard approach is to iteratively use the *entire* training data to optimize model's objective until convergence. Curriculum learning techniques improve this training process by scheduling examples for training, e.g., by gradually learning from easier examples before training with harder ones. Such curricula can be predefined by humans (Bengio and LeCun, 2007; Bengio et al., 2009) or dynamically derived from data during training (Jiang et al., 2018; Castells et al., 2020).

Curriculum learning for graph neural networks is an emerging area of research. Recently, Chu et al. (2021) employed a traditional curriculum learning approach introduced in (Bengio et al., 2009) to improve negative sampling for graph classification. Wang et al. (2021) proposed to estimate the difficulty of graph entities–nodes, edges or subgraphs–based on the intra- and inter-class distributions of their embeddings in supervised settings, and developed a smooth-step function to gradually introduce harder examples to GNNs during training. Vakil and Amiri (2022) developed a loss-based curriculum learning approach that dynamically adjusts the difficulty boundaries of training samples based on their sample-level loss trajectories obtained from recent training dynamics of GNN models.

To the best of our knowledge, existing curriculum learning approaches often employ a *single* criterion of difficulty in their curriculum learning framework, e.g., prediction loss (Wu et al., 2021), consistency in prediction loss (Xu et al., 2020), moving average of loss (Zhou et al., 2020) or transformations of loss (Vakil and Amiri, 2022). We address this gap by developing a new curriculum learning approach for GNNs titled **M**ultiview **C**ompetence-based **C**urriculum **L**earning (MCCL) that builds on the complexity formalisms of graph data. By leveraging rich graph structures, graph complexity formalisms and model *competence* (learning progress), we will design robust curricula for training GNNs. Table 1 shows three subgraphs ranked differently according to different graph complexity indices. If complexity is measured by *node degree*, then G1 and G2 are less complex than G3 because target nodes in these subgraphs have an overall smaller node degrees. However, if complexity is measured by *closeness centrality*[2], then G2 is more complex than G3 because the target nodes are less central in G2 than those in G3. It is evident that complexity indices (views) can vary significantly in their difficulty estimates of graph data.

[1] Code, data splits and guidelines are available at https://clu.cs.uml.edu/tools.html.

[2] Closeness centrality (Sabidussi, 1966) is smaller for central nodes–those that are closer to other nodes in the graph.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7036–7051

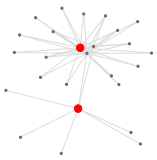
ID	subgraph	Sentence	Label	Degree	Centrality
G1		Prominent examples include mutations in the transporters for dopamine (DAT, **SLC6A3**), for creatine (CT1, SLC6A8), and for glycine (GlyT2, SLC6A5), which result in infantile dystonia, mental retardation, and **hyperekplexia**, respectively.	False	24	1.14
G2		Depletion of **PRODH** and GSALDH in humans leads to **hyperprolinemia**, which is associated with mental disorders such as schizophrenia.	True	69	1.46
G3		Lately, **ARMC5** was linked to the cyclic AMP signaling pathway, which could be implicated in all of mechanisms of cortisol-secreting by macronodules **adrenal hyperplasia** and the molecular defects in: G protein aberrant receptors; MC2R; GNAS; PRKAR1A; PDE11A; PDE8B.	True	122	1.40

Table 1: The difficulty of training examples–target node pairs in red–can be assessed based on their subgraphs, k-hop neighbors of target nodes. For brevity, we show two views: node degree and closeness centrality. Boldfaced tokens indicate nodes in the graph, Label indicates if the sentence reports a causal relation between the nodes, and Degree and Centrality report the sum of degree and closeness centrality scores of the target nodes in their subgraphs. Each subgraph provides a structural view of the target nodes in sentences. The relative difficulty of examples is different across views, e.g., G2 is less difficult than G3 according to Degree but more difficult according to Centrality.

The objective of this work is improve the training process of GNNs by strategically and dynamically (during training) prioritizing key complexity indices, aiming to guide the model toward better minima within its parameter space. Graph complexity is a well-established area of research and our review of relevant literature suggests that there exist various techniques that employ structural properties of nodes, edges and subgraphs to quantify complexity of graph data (Kim and Wilhelm, 2008; Vishwanathan et al., 2010; Newman, 2018; Kriege et al., 2020). We build on these indices to design our curriculum learning framework which treats each complexity index as a view of difficulty. Our approach consists of a novel data scheduling scheme which derives effective curricula based on given views of sample difficulty and model competence during training. Specifically, given a downstream GNN model, our data scheduler gradually selects training examples from a graph complexity view based on competency of the GNN model during training. The model updates its competency and the scheduler determines the next best view for training the model. As model competency gradually increases, the scheduler allows using more signals from different views.

The contributions of this paper are as follows:

- A new curriculum learning approach that effectively leverages complexity formalisms of graph data, taking into account multiview difficulty of training data samples and model's learning progress, and
- Key insights into important complexity indices for effective training of graph neural networks for NLP applications.

We conduct extensive experiments on real world datasets for link prediction and node classification tasks in text graph datasets. Our approach results in 3.3 and 1.8 absolute points improvements in F1-score over the state-of-the-art model on link prediction datasets and 6.7 and 4.9 absolute points improvement on the node classification dataset. The results show that the contribution of complexity indices in training depends on factors such as training stage and model behavior. When the scheduling criterion relies solely on complexity indices, the scheduler tends to initially focus on indices that operate locally around nodes, and later shifts to those that operate globally at graph level. Extending schedulers based on model dynamics (e.g., loss) results in both local and global indices being used throughout the training. These findings provide insights into the type of complexity information that GNNs learn at different stages of their training.

2 Competence-based Multiview Curricula

We present a competence-based multiview curriculum learning framework for training GNNs. At every training iteration, the framework selects a sub-set of training examples based on the best complexity index (view) and model's competence at that iteration. Algorithm 1 describes the overall approach. We first introduce our complexity indices and then present the model.

2.1 Graph Complexity Formalisms

Various graph complexity indices were introduced in graph theory (Kashima et al., 2003; Borgwardt and Kriegel, 2005; Vishwanathan et al., 2010; Kriege et al., 2020; Newman, 2018). We consider 26 of such indices which represent criteria of difficulty in our curriculum learning framework.[3] In what follows, we describe a few representative complexity indices and refer the reader to Appendix A for a full description of all indices.

Since GNNs train through neural message passing at subgraph level (Gilmer et al., 2017; Hamilton et al., 2017), we compute complexity indices with respect to the k-hop neighbors (subgraph) of target nodes. For tasks involving two nodes (e.g., relation extraction), we sum the scores computed for the node pairs. We use Networkx (Hagberg et al., 2008) to compute the indices:

- **Degree:** The number of immediate neighbors of a node in a graph.
- **Average neighbor degree:** Average degree of the neighbors of a node:

$$\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} k_j,$$

  where $\mathcal{N}_i$ is the set of neighbors of node i and k_j is the degree of node j.
- **Katz centrality:** The centrality of a node computed based on the centrality of its neighbors. Katz centrality computes the relative influence of a node within a network by measuring the number of immediate neighbors and number of walks between node pairs. It is computed as follows:

$$x_i = \alpha \sum_j \mathbf{A}_{ij} x_j + \beta,$$

  where x_i is the Katz centrality of node i, $\mathbf{A}$ is the adjacency matrix of Graph G with eigenvalues λ. The parameter β controls the initial centrality and $\alpha < 1 \,/\, \lambda_{max}$.
- **Resource allocation index:** For nodes i and j in a subgraph, the resource allocation index is defined as follows:

$$\sum_{k\in(\mathcal{N}_i \bigcap \mathcal{N}_j)} \frac{1}{|\mathcal{N}_k|},$$

  which quantifies the closeness of target nodes based on their shared neighbors.
- **Subgraph density:** The density of an undirected subgraph is computed as follows:

$$\frac{e}{v(v-1)},$$

  where e is the number of edges and v is the number of nodes in the subgraph.
- **Local bridge:** A local bridge is an edge that is not part of a triangle in the subgraph. We take the number of local bridges in a subgraph as a complexity index.
- **Subgraph connectivity:** Is measured by the *minimum* number of nodes that must be removed to disconnect the subgraph.
- **Eigenvector centrality:** Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is $Ax = \lambda x$. where A is the adjacency matrix of the graph G with eigenvalue λ.

We note that our approach does not depend on any specific index. However, we recommend considering indices that are computationally inexpensive for applicability to large graphs. The complexity scores of each index are normalized into [0, 1] range using L2 norm.

2.2 Model Competency

We define model competence at each training iteration t as the fraction of training data that can be used by the model at time t; we refer to this fraction by $c(t)$. Our curriculum learning framework employs difficulty indices to select $c(t)$ fraction of examples to train its downstream model (a GNN). We employ the following function (Platanios et al., 2019) to quantify competence:

$$c(t) = \min\left(1, \sqrt[p]{t\left(\frac{1-c_0^p}{T}\right) + c_0^p}\right), \qquad (1)$$

[3] Our list of complexity indices may not be exhaustive. However, our framework and implementation allows adding any number of additional indices.

where t is the training iteration, p controls the sharpness of the curriculum so that more time is spent on the examples added later in the training, T is the maximum curriculum length (number of iterations), and c_0 is the initial value of the competence. $c(t)$ gradually increases to achieve the maximum value of 1, which covers the entire training dataset. We set $p = 2$ and $c_0 = 0.01$ as suggested in (Platanios et al., 2019).

2.3 Prioritizing Important Difficulty Indices

Difficulty indices vary significantly in their difficulty estimates, owing to the complicated topology and indistinct patterns in graph data. Our framework strategically prioritizes key difficulty indices while training a GNN model. Specifically, the framework employs two mechanisms (see line 7 in Algorithm 1) to determine which index (i.e., top $c(t)$ portion of training data ranked by the difficulty scores obtained from the index) should be used for training the downstream GNN model at iteration t: (i) model-based and (ii) index-based approaches:

Model-based: This approach performs a forward pass on the selected portion of training data and calculates the average loss of the GNN on these examples. The index with the maximum (or minimum, depending on the curriculum) average loss will be selected at iteration t and it's top $c(t)$ examples will be used for training the downstream GNN. Minimum average loss prioritizes easier examples over harder ones for training. On the other hand, maximum average loss prioritizes harder examples (as in an anti-curriculum setting).

Index-based: This approach uses the actual difficulty scores obtained from indices. The index with minimum (or maximum) average difficulty score across its top $c(t)$ portion of training samples will be selected for training and calculating the error (see lines 9-14 in Algorithm 1). We note that the index-based approach is computationally inexpensive compared to the model-based approach, and results in comparable performance, see results in experiments (Tables 4)

2.4 Base Graph Neural Network Model

Our approach is model agnostic and can be applied to any GNN. We use the graph-text neural network (GTNN) model[4] from (Vakil and Amiri, 2022) as the base model because it is designed for text-graph data. The model integrates textual information with graph structure and directly uses text embeddings at prediction layer to avoid information loss in the iterative process of training GNNs. We use this model as a base model to compare our and baseline curriculum learning approaches on graph data.

Algorithm 1: Multiview Competence-based Curriculum Learning (MCCL).

input :
D: Training data of size n
L: Difficulty indices
M: GNN Model
O: easy-to-hard vs. hard-to-easy transition
output : Trained model M^*

1 Compute complexity scores for each index i in L and store the results in L_i
2 $L_i \leftarrow sort(L_i)$ *# in ascending or descending order.*
3 **for** $t \leftarrow 0$ **to** T **do**
4 $c(t) \leftarrow$ competence from Eq (1)
5 **foreach** *index in L* **do**
6 $l_i \leftarrow$ top $(c(t) \times n)$ examples from L_i
7 $e_i \leftarrow$ average loss or complexity of l_i
8 **end**
9 **if** *O = easy-to-hard* **then**
10 $j = \arg\min_i e_i$
11 **else**
12 $j = \arg\max_i e_i$
13 **end**
14 Train M with l_j samples
15 **end**

3 Experimental Results

3.1 Datasets

Gene Phenotype Relation (PGR) (Sousa et al., 2019): PGR is created from PubMed articles and contains sentences describing causal relations between genes and phenotypes (symptoms); see Table 1 for examples of this dataset.

Gene, Disease, Phenotype Relation (GDPR) (Vakil and Amiri, 2022): GDPR contains different types of relations among genes, diseases and phenotypes, and long texts describing them.

Cora (McCallum et al., 2000): Cora is a relatively small citation network, in which nodes are scientific papers and edges are citations among them. Each paper is categorized into one of the seven subject categories and is provided with a textual feature word vector obtained from the content of the paper.

Ogbn-arxiv (Hu et al., 2020): This Open Graph Benchmark dataset is a citation network between papers in the Computer Science domain. Each node in the graph is a paper and an edge represents

[4] https://github.com/CLU-UML/gtnn

	GDPR	PGR	Cora	Ogbn-Arxiv
Nodes	18.3K	20.4K	2.7K	169K
Edges	365K	605K	5.4K	1.1M
Train	30.1K	2.6K	2.1K	90K
Test	3.7K	155	271	49K
Val	3.7K	–	271	30K

Table 2: Dataset statistics.

	Link Prediction		Node Classification	
Model	**GDPR F1**	**PGR F1**	**Cora Acc**	**Ogbn-Arxiv Acc**
GTNN	82.4	93.4	91.5	71.6
CurGraph	81.0	80.3	88.6	68.7
SL	84.1	94.5	90.4	71.8
Trend-SL	84.6	94.5	90.4	71.5
MCCL	**85.7***	**95.2***	**98.2***	**76.5***

Table 3: Performance of curriculum models on **GDPR** and **PGR** datasets for link prediction, and **Cora** and **Ogbn-Arxiv** for node classification. The base model for all curriculum learning approaches is GTNN, which has a high score of F1 and Accuracy on the datasets using standard training. MCCL performs best compared to the other curricula methods. Asterisk marks (*) indicate significantly better performance compared to all other competing models.

a citation from one paper to another. Also, each paper contains 128 dimension embedding vector obtained by taking the average of the words present in the title and the abstract.

Table 2 shows the statistics of the above datasets. We use PGR and GDPR for link prediction and Cora and Ogbn-Arxiv for node classification.

3.2 Baselines

CurGraph (Wang et al., 2021) is a curriculum learning framework for graphs that computes difficulty scores based on the intra- and inter-class distributions of embeddings and develops a smoothstep function to gradually include harder samples in training. We report the results of our implementation of this approach.

SuperLoss (SL) (Castells et al., 2020) is a generic curriculum learning approach that dynamically learns a curriculum from model behavior. It uses a fixed difficulty threshold at batch level, determined by the exponential moving average of all sample losses, and assigns higher weights to easier samples than harder ones.

Trend-SL (Vakil and Amiri, 2022) is a curriculum learning approach which extends (Castells et al., 2020) by incorporating sample-level loss trends to better discriminate easier from harder samples and schedule them for training.

3.3 Settings

We consider 1-hop neighbors for PGR and GDPR and 2-hop neighbors for Cora and Ogbn-Arxiv to create subgraphs for computing complexity indices, see Section 2.1, and training the GTNN model, see Section 2.4. We train all models for a maximum number of 100 iterations for PGR and GDPR, and 500 iterations for Cora and Ogbn-Arxiv with model checkpoint determined by validation data for all models. We conduct all experiments using Ubuntu 18.04 on a single 40GB A100 Nvidia GPU.

We consider 26 complexity indices listed in Appendix A. Since some of the indices are highly co-related, we use k-means to group them based on the Pearson co-relations between their ranking of training samples. We categorize indices into 10 clusters through grid search, which effectively prevents any redundancy in the index space. We randomly select an index from each cluster to be used by our curriculum learning framework. Indices that are used by the framework are labeled by asterisks in Appendix A.

We report F1 score (on positive class) for PGR and GDPR datasets, and Accuracy score for Cora and Ogbn-Arxiv datasets. In addition, we use t-test for significance testing and asterisk mark (*) to indicate significant difference at $\rho = 0.01$.

3.4 Main Results

Table 3 shows the performance of the proposed MCCL method against other curriculum learning approaches. The results shows that applying curricula to the base model (GTNN) further improves its performance by 3.3 and 1.8 absolute points in F1 on GDPR and PGR datasets respectively, indicating the importance of curriculum learning for training GNNs. The corresponding improvement on Cora and Ogbn-Arxiv datasets are 6.7 and 4.9 absolute points in accuracy.

In addition, MCCL outperforms other curriculum learning approaches. Furthermore, as MCCL increasingly introduces more training instances at each iteration, it shows an overall faster training time compared to the other curriculum learning models, which iterate through all training examples at every iteration. See Section 4.4 for detail analysis on time complexity of different models.

Link Prediction				Node Classification			
Model	**Index Order**	**Transition Order**	**Avg F1**	**Model**	**Index Order**	**Transition Order**	**Avg Acc**
GTNN	–	–	87.9	**GTNN**	–	–	81.6
MCCL: Model-based	desc	max	89.4	**MCCL: Model-based**	desc	max	**87.3**
	desc	min	89.3		desc	min	86.7
	asc	max	**89.9**		asc	max	87.1
	asc	min	88.7		asc	min	86.5
MCCL: Index-based	desc	max	90.1	**MCCL: Index-based**	desc	max	**87.0**
	desc	min	89.2		desc	min	86.3
	asc	max	89.3		asc	max	86.9
	asc	min	**90.4**		asc	min	86.7

Table 4: Ablation analysis on the order by which training examples are sorted for complexity indices (ascending versus descending in the Index Order column, see line 2 in Algorithm 1), the training mechanism (model- versus index-based in the Model column, see line 7 in Algorithm 1) and the type of learning transition (easy-to-hard (Min error) versus hard-to-easy (Max error) in the Transition Order column, see lines 9–13 in Algorithm 1).

4 Multiview Curricula Introspection

We perform several ablation studies on the MCCL model, investigating genuine complexity indices compared to random ordering, multiview curricula versus anti-curricula, the impact of complexity indices in training, and the model's time complexity.

4.1 Model Prioritizes Genuine Complexity Indices over Random Ordering

In curriculum learning, effective training depends on the scheduler, which determines the set of examples and their order for training at each iteration. Hence, the performance of the model largely depends on the scheduling criteria used for training. To determine if our model can indeed prioritize better indices, we added a *fake* index named "Random" index to the list of our complexity indices. Training examples were randomly ordered in the Random index. We re-ran our model and checked whether it selects the Random index for training at any iteration. On Cora and Ogbn-Arxiv datasets, model selects the Random index at 17.6% and 12.8% of its training iterations. On PGR, the model never selects the Random index, and on GDPR, model selects the Random index at 8% of its training iterations. Specifically, toward the end of training at iterations [39, 46, 58, 71, 76, 83, 91, 93] with the best F1-score of 85.5% obtained at iteration 76. The fact that the model do not often select the Random index at many iterations is effectively inline with the core principle of curriculum learning–learning materials should be gradually learned in a properly-planned order. This sanity check indicates that the model prioritizes genuine complexity indices over random ordering.

4.2 Multiview Curricula vs. Anti-Curricula

We study the effect of different criteria in MCCL framework through ablation analysis on (a): the order by which training examples are sorted with respect to their complexity scores for each index (descending versus ascending, see line 2 in Algorithm 1), (b): the mechanism by which our framework prioritizes indices (model-based versus index-based, see line 7 in Algorithm 1 and Section 2.3), and (c): the type of learning transition in our framework (easy-to-hard versus hard-to-easy transition, see lines 9–13 in Algorithm 1).

Table 4 shows the result of this ablation analysis averaged over the PGR and GDPR datasets for link prediction, and Cora and Ogbn-Arxiv datasets for node classification respectively. The corresponding results for each dataset is reported in Appendix B. Overall, the ascending order results in the best average F1 score for link prediction while descending order performs better for node classification. In addition, in model-based training, hard-to-easy (max) transition order is more effective than easy-to-hard (min) transition order across both tasks. This is perhaps because harder examples are superior at helping the model find better local minima at the early stages of training. We also observe that easy-to-hard (min) transition for index-based training results in higher average F1-score than hard-to-easy (max) transition of the model-based training. This is because, in case of index-based ordering, the difficulty scores (which are obtained from indices) may provide a more accurate estimation of easiness to the model than hardness, i.e. easy examples are likely easy for the model in both ordering but this may not be true for hard examples.

4.3 Index Contributions to Training

To study the contributions of different complexity indices in the training process, we divide training iterations into three phases and create the histograms that show the number of times that each index is chosen at different stages of training: (i) Initial, (ii) Middle, and (iii) End phases of the training.

Figures 1 shows the results for different indices chosen by the best-performing MCCL model[5] for both index-based (where the criterion for selecting samples is merely based on their difficulty scores obtained from indices) and model-based (where the criterion for selecting samples is based on instantaneous loss) approaches across our four datasets. Our key observation was that MCCL mainly focused on indices that operate locally around nodes (such as density- or degree-based indices) at early stages of training and then focused on indices that operate globally at graph level (such as centrality-based indices) at later stages of training for both index-based and model-based training mechanisms. Additionally, we observed greater diversity in the sets of prioritized indices in case of the model-based training mechanism, which indicates MCCL encourages learning from diverse views of difficulty during training. This is mainly because the model-based training mechanism in MCCL allows the GNN to directly contribute in scheduling indices through its loss dynamics during training.

In addition, we note that on the Cora dataset the model merely focused on degree-based metrics throughout its training in case of index-based training and, used a smaller set of fundamentally-different indices in case of model-based training. On the Ogbn-Arxiv dataset, the model focuses only on the eigenvector centrality index throughout its training in case of index-based training and focuses on connectivity and centrality indices in the model-based training. Further analysis on this model behavior is the subject of our future work.

Overall, density, degree indices including degree and degree assortativity coefficient, and centrality indices including closeness centrality, group degree centrality and eigenvector centrality indices are often prioritized by the model across different mechanisms and datasets.

4.4 MCCL Has the Lowest Time Complexity

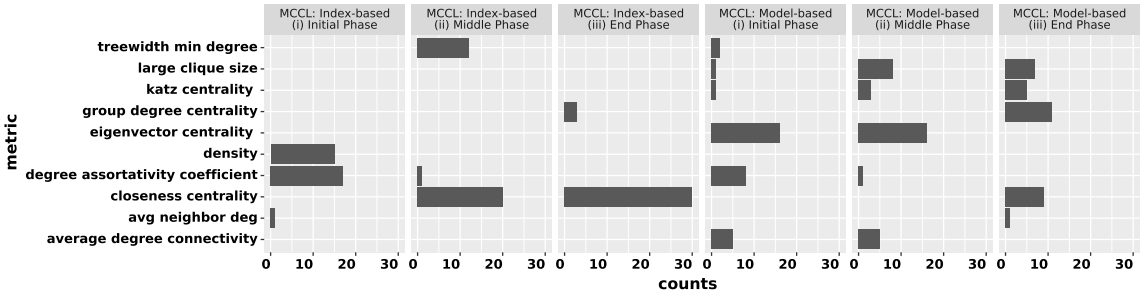
Let n be the number of training examples and e be the maximum number of iterations for training a neural network. The total of number of forward and backward passes required to train GTNN, SL, Trend-SL is $2 \times n \times e$. In contrast, MCCL trains from only a fraction of training examples at each iteration and do not need to "see" all the training examples at each iteration. If model-based mechanism is used to prioritize important indices for training, the total number of forward and backward passes of MCCL is $3 \times \sum_i (n \times \frac{i}{e})$, which amounts to $1.5 \times n \times (e-1)$; note that model-based approach requires $\sum_i (n \times \frac{i}{e}) = \frac{n \times (e-1)}{2}$ additional forward passes to select the best index. In case of the index-based training mechanism, no additional forward pass is required, resulting in a total of $2 \times \sum_i (n \times \frac{i}{e})$ passes, which amounts to $n \times (e-1)$ passes. In either case, MCCL has lower time complexity than other baselines. The turnaround time of our model ranges from 10 minutes to 2.5 hours, depending on the size of the input dataset.

5 Related Work

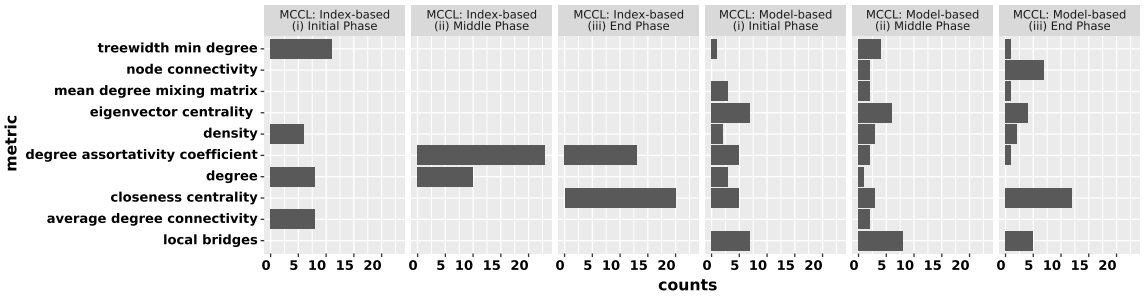
Curriculum learning (Bengio et al., 2009) aims to improve the generalizability of a model by gradually training it with easy examples followed by hard ones. Castells et al. (2020) introduced a generic loss function called SuperLoss (SL) which can be added on top of any target-task loss function to dynamically weight the training samples according to their difficulty for the model using a batch-wise threshold. Zhou et al. (2020) proposed dynamic instance hardness to determine the difficulty of an instance with running average of the hardness metric over training history.

Curriculum learning has been investigated in NLP (Elman, 1993; Sachan and Xing, 2016; Settles and Meeder, 2016; Amiri et al., 2017; Platanios et al., 2019; Amiri, 2019; Zhang et al., 2019; Lalor and Yu, 2020; Xu et al., 2020; Chu et al., 2021; Liu et al., 2021; Kreutzer et al., 2021; Agrawal and Carpuat, 2022; Maharana and Bansal, 2022). Specifically, Settles and Meeder (2016); Amiri et al. (2017) proposed spaced repetition-based curricula based on psycholinguistic theory where the training data is scheduled by increasing intervals of time between consecutive reviews of previously learned data samples. Zhang et al. (2019) investigated curriculum learning for domain adaptation in neural machine translation, where samples were grouped and ranked based on their similarity score such that more similar samples are seen earlier and more frequently during training. Platanios

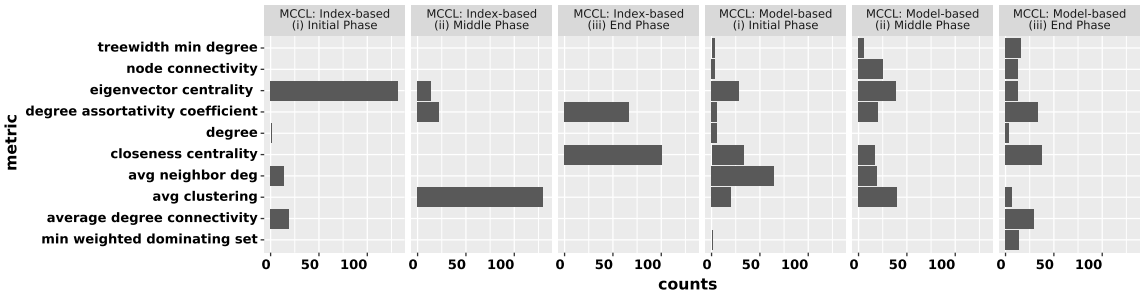
[5] According to the results in Tables 6 and 7 in Appendix B.



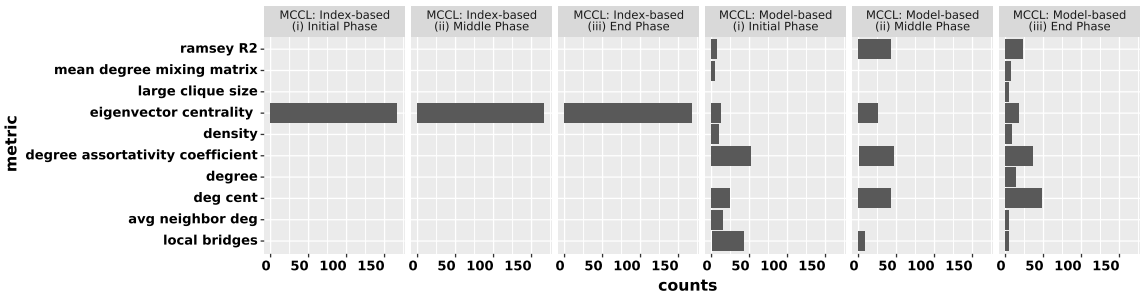
(a) GDPR dataset



(b) PGR dataset



(c) Cora dataset



(d) Ogbn-Arxiv dataset

Figure 1: Histogram of the different indices chosen by the best-performing MCCL model for both index-based (where samples are selected for training merely based on their difficulty scores obtained from indices) and model-based (where samples are selected based on instantaneous loss of the model) approaches across three datasets at the beginning, middle and end of training; see best models in Table 6, Appendix B. Dominance of a metric depends on the internal structure of the graph, underlying task, stage of training, and model behavior during training. (a): on GDPR, model mainly focuses on density- and degree-based indices (which operate locally around nodes) at the beginning and centrality-based indices (which operate globally at graph level) at later stages of index-based training (left three plots). In case of model-based training (right three plots), both degree- and centrality-based indices were selected throughout the training. (b): On PGR, we observed the same pattern except that locality indices such as number of local bridges were prioritized in case of model-based training. (c): On Cora, the model merely focuses on the degree-based metrics throughout its index-based training, and focuses on a mix of different indices at early stages of its model-based training and closeness centrality at the end of training. (d): On Ogbn-Arxiv, the model only focuses on eigenvector centrality metric throughout its index-based training and focuses on the local bridge and degree assortativity indices in the initial phase of the model-based training. In the middle and final phases, model-based training mainly focuses on degree based indices.

et al. (2019) proposed an approach to use competency function using rarity of words or length of a sentence for neural machine translation and inspired Liu et al. (2021) to define a curriculum based on multi-modal (text and image) data to choose which modality should be used for training. The model uses sample perplexity at batch level to select the modality for training. Linguistic features such as word rarity or length of sentence in (Platanios et al., 2019) and sample perplexity in (Liu et al., 2021) were used as measures of difficulty. Xu et al. (2020) designed a curriculum learning approach for NLP tasks using cross-view of training data to identify easy and hard examples and rearrange the examples during training. Other works of curriculum learning in NLP focused on machine translation and language understanding. Agrawal and Carpuat (2022) developed a framework to train non-autoregressive sequence-to-sequence model to edit text where a curriculum is designed to first perform easy-to-learn edits followed by increasing difficulty of training samples. Maharana and Bansal (2022) designed several curriculum learning approaches using teacher-student model where the teacher model calculates the difficulty of each training example using question-answering probability, variability, and out-of-distribution measures.

Curriculum learning for graph data is an emerging area of research. Chu et al. (2021) explored curriculum learning approach in the self-supervision settings where the difficulty measure evaluates the difficulty of negative samples which is calculated based in the embeddings's similarity between positive and negative examples. Wang et al. (2021) proposed a curriculum based subgraph classification approach, CurGraph, which first obtains graph-level embeddings via unsupervised GNN method and then uses neural density estimator to model embedding distributions. The difficulty scores of graphs are calculated by a predefined difficulty measure based on the inter- and intra-class distribution of sub-graph embeddings. In Vakil and Amiri (2022), we extended the SuperLoss approach developed in (Castells et al., 2020) by introducing a curriculum learning framework that dynamically adjusts the difficulty of samples during training with respect to the loss trajectory. We demonstrated the effectiveness of incorporating this strategy in graph curriculum learning settings. Previous work in graph curriculum learning has employed a single criterion of difficulty in their curriculum learning framework. Our work uses multiple criteria for curriculum learning on graphs. We encourage readers to see (Li et al., 2023) for a survey on graphs curriculum learning approaches.

Finally, in terms of datasets, Sousa et al. (2019) developed the PGR dataset, which we used in our experiments. They developed a transformer model to identify the relation between biomedical entities, genes and phenotypes, from scientific PubMed articles. For relation classification authors considered a pair of entities and the context from the corresponding sentence in which both entities occur.

6 Conclusion and Future work

We present a novel curriculum learning approach for training graph neural networks. Our approach combines well-established graph complexity indices (views) obtained from graph theory and demonstrates the effectiveness of learning from diverse difficulty views for the tasks of link prediction and node classification. Our approach improves over the state-of-the-art techniques for curriculum learning on graphs across several datasets. Ablation studies show that the model prioritizes genuine complexity indices over to random ordering, and effectively uses and learn multiview complexity indices in both curricula and anti-curricula settings, and has lower time complexity that competing models. In future, we will extend our approach to other graph processing tasks, focusing on NLP applications such as clustering and community detection, and investigate the effect of graph complexity indices in such tasks.

Limitation

Calculating complexity indices for large-scale graphs can be computationally expensive and time consuming. Some of the complexity indices show longer turnaround time when computed for denser areas in the graphs. In addition, as we mentioned in the paper, although we made sure our framework and implementation allows adding any number of additional indices in a modular way, there might be other effective complexity indices that are not included in this investigation. Furthermore, it should be noted that the model has been exclusively tested on graphs where nodes contain textual content, which may limit its application to more general graph types. Finally, the model has not been applied to other graph-based tasks such as clustering and graph-level classification.

References

Sweta Agrawal and Marine Carpuat. 2022. An imitation learning curriculum for text editing with non-autoregressive models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7550–7563.

Hadi Amiri. 2019. Neural self-training through spaced repetition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 21–31.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2017. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Yoshua Bengio and Yann LeCun. 2007. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*.

Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE.

Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. Superloss: A generic loss for robust curriculum learning. *Advances in Neural Information Processing Systems*.

Guanyi Chu, Xiao Wang, Chuan Shi, and Xunqiang Jiang. 2021. Cuco: Graph representation with curriculum contrastive learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, pages 2300–2306.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 321–328.

Jongkwang Kim and Thomas Wilhelm. 2008. What is a complex graph? *Physica A: Statistical Mechanics and its Applications*, 387(11):2637–2652.

Julia Kreutzer, David Vilar, and Artem Sokolov. 2021. Bandits don't follow rules: Balancing multi-facet machine translation with multi-armed bandits. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3190–3204, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. 2020. A survey on graph kernels. *Applied Network Science*, 5(1):1–42.

John P. Lalor and Hong Yu. 2020. Dynamic data selection for curriculum learning via ability estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 545–555, Online. Association for Computational Linguistics.

Haoyang Li, Xin Wang, and Wenwu Zhu. 2023. Curriculum Graph Machine Learning: A Survey. *arXiv e-prints*, page arXiv:2302.02926.

Fenglin Liu, Shen Ge, and Xian Wu. 2021. Competence-based multimodal curriculum learning for medical report generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3001–3012, Online. Association for Computational Linguistics.

Adyasha Maharana and Mohit Bansal. 2022. On curriculum learning for commonsense reasoning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 983–992, Seattle, United States. Association for Computational Linguistics.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Mark Newman. 2018. *Networks*. Oxford university press.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabás Póczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172.

Gert Sabidussi. 1966. The centrality index of a graph. *Psychometrika*, 31(4):581–603.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463, Berlin, Germany. Association for Computational Linguistics.

Burr Settles and Brendan Meeder. 2016. A trainable spaced repetition model for language learning. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1848–1858.

Diana Sousa, André Lamúrias, and Francisco M Couto. 2019. A silver standard corpus of human phenotype-gene relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Nidhi Vakil and Hadi Amiri. 2022. Generic and trend-aware curriculum learning for relation extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2021. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. 2021. When do curricula work? In *International Conference on Learning Representations*.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1903–1915, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. 2020. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*.

A Graph Difficulty Indices

Below are the list of 26 indices which we consider for Multiview Competence-based Curriculum Learning (MCCL) approach. All these indices are computed on the subgraph of the node or an edge. These definition and code to calculate the indices, we used Networkx package (Hagberg et al., 2008).

- * **Degree:** The number of immediate neighbors of a node in a graph.
- * **Treewidth min degree:** The treewidth of an graph is an integer number which quantifies, how far the given graph is from being a tree.
- * **Average neighbor degree:** Average degree of the neighbors of a node is computed as:

$$\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} k_j$$

  where $\mathcal{N}_i$ is the set of neighbors of node i and k_j is the degree of node j.
- * **Degree mixing matrix:** Given the graph, it calculates joint probability, of occurrence of node degree pairs. Taking the mean, gives the degree mixing value representing the given graph.
- * **Average degree connectivity:** Given the graph, it calculates the average of the nearest neighbor degree of nodes with degree k. We choose the highest value of k obtained from the calculation and used its connectivity value as the complexity index score.
- * **Degree assortativity coefficient:** Given the graph, assortativity measures the similarity of connections in the graph with respect to the node degree.
- * **Katz centrality:** The centrality of a node, i, computed based on the centrality of its neighbors j. Katz centrality computes the relative influence of a node within a network by measuring taking into account the number of immediate neighbors and number of walks between node pairs. It is computed as follows:

$$x_i = \alpha\sum_j A_{ij}x_j + \beta$$

  where x_i is the Katz centrality of node i, A is the adjacency matrix of Graph G with eigenvalues λ. The parameter β controls the initial centrality and $\alpha < 1$ / λ_{max}.

Degree based	**Computing based**
degree	ramsey R2
treewidth min degree	average clustering
degree mixing matrix	resource allocation index
average neighbor degree	**Basic properties**
average degree connectivity	density
degree assortativity coefficient	local bridges
Centrality	number of nodes
katz centrality	number of edges
degree centrality	large clique size
closeness centrality	common neighbors
eigenvector centrality	**Connectivity**
group degree centrality	subgraph connectivity
Flow property	local node connectivity
min weighted dominating set	
min weighted vertex cover	
min edge dominating set	
min maximal matching	

Table 5: Complexity indices used to capture diverse characteristics of a graph structure. For different graph structure, some index may represent the same order with another metric when sorted. To avoid this redundancy, we rank training data for each index and find Pearson corelation between each such ranking. We then used k-means to find the clusters.

- **Degree centrality:** Given the graph, the degree centrality for a node is the fraction of nodes connected to it.
- * **Closeness centrality:** The closeness of a node is the distance to all other nodes in the graph or in the case that the graph is not connected to all other nodes in the connected component containing that node. Given the subgraph and the nodes, added the values of the nodes to find the complexity index value.
- * **Eigenvector centrality:** Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is $Ax = \lambda x$. where A is the adjacency matrix of the graph G with eigenvalue λ.
- * **Group Degree centrality:** Group degree centrality of a group of nodes S is the fraction of non-group members connected to group members.
- **Ramsey R2:** This computes the largest clique and largest independent set in the graph G. We calculate the index value by multiplying number of largest cliques to number of largest independent set.

- *** Average clustering:** The local clustering of each node in the graph G is the fraction of triangles that exist over all possible triangles in its neighborhood. The average clustering coefficient of a graph G is the mean of local clusterings.

- **Resource allocation index:** For nodes i and j in a subgraph, the resource allocation index is defined as follows:

$$\sum_{k \in (\mathcal{N}_i \bigcap \mathcal{N}_j)} \frac{1}{|\mathcal{N}_k|},$$

  which quantifies the closeness of target nodes based on their shared neighbors.

- *** Subgraph density:** The density of an undirected subgraph is computed as follows:

$$\frac{e}{v(v-1)},$$

  where e is the number of edges and v is the number of nodes in the subgraph.

- *** Local bridge:** A local bridge is an edge that is not part of a triangle in the subgraph. We take the number of local bridges in a subgraph as a complexity score.

- **Number of nodes:** Given the graph G, number of nodes in the graph is chosen as the complexity score.

- **Number of Edges:** Given the graph G, number of edges in the graph is chosen as the complexity score.

- *** Large clique size:** Given the graph G, the size of a large clique in the graph is chosen as the complexity score.

- **Common neighbors:** Given the graph and the nodes, it finds the number of common neighbors between the pair of nodes. We chose number of common neighbors as the complexity score.

- *** Subgraph connectivity:** is measured by the *minimum* number of nodes that must be removed to disconnect the subgraph.

- **Local node connectivity:** Local node connectivity for two non adjacent nodes s and t is the minimum number of nodes that must be removed (along with their incident edges) to disconnect them. Given the subgraph and the nodes, gives the single value which we used as complexity score.

- **Minimum Weighted Dominating Set**: For a graph $G = (V, E)$, the weighted dominating set problem is to find a vertex set $\mathcal{S} \subseteq V$ such that when each vertex is associated with a positive number, the goal is to find a dominating set with the minimum weight.

- **Weighted vertex cover index:** The weighted vertex cover problem is to find a vertex cover $\mathcal{S}$–a set of vertices that include at least one endpoint of every edge of the subgraph–that has the minimum weight. This index and the weight of the cover $\mathcal{S}$ is defined by $\sum_{s \in \mathcal{S}} w(s)$, where $w(s)$ indicates the weight of s. Since $w(s) = 1, \forall s$ in our unweighted subgraphs, the problem will reduce to finding a vertex cover with minimum cardinality.

- **Minimum edge dominating set:** Minimum edge dominating set approximate solution to the edge dominating set.

- **Minimum maximal matching:** Given a graph G = (V,E), a matching M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. That is, out of all maximal matchings of the graph G, the smallest is returned. We took the length of the set as the complexity index.

B Multiview Curricula Ablation Analysis

The results of our study the effect of different curriculum criteria in our competence-based multiview curriculum learning framework. We conduct ablation analysis on (a): the order by which training examples are sorted with respect to their complexity scores for each index (descending versus ascending, see line 2 in Algorithm 1), (b): the mechanism by which our framework prioritizes indices (model-based versus index-based, see line 7 in Algorithm 1 and Section 2.3), and (c): the type of learning transition in our framework (easy-to-hard versus hard-to-easy transition, see lines 9–13 in Algorithm 1).

Dataset	Model	Index Order	Transition Order	P	R	F1
PGR	GTNN	–	–	93.6	93.2	93.4
PGR	MCCL: Model-based	descending	max	95.8	93.2	94.5
PGR	MCCL: Model-based	descending	min	93.3	94.6	94.0
PGR	MCCL: Model-based	ascending	min	95.9	94.6	**95.2**
PGR	MCCL: Model-based	ascending	max	97.2	93.2	**95.2**
PGR	MCCL: Index-based	descending	min	94.5	93.2	93.9
PGR	MCCL: Index-based	ascending	max	97.2	93.2	**95.2**
PGR	MCCL: Index-based	ascending	min	95.9	94.6	**95.2**
PGR	MCCL: Index-based	descending	max	97.1	91.9	94.4
GDPR	GTNN	–	–	77.1	88.5	82.4
GDPR	MCCL: Model-based	descending	max	80.6	88.2	84.3
GDPR	MCCL: Model-based	descending	min	83.0	86.3	84.6
GDPR	MCCL: Model-based	ascending	min	78.9	85.7	82.1
GDPR	MCCL: Model-based	ascending	max	80.6	89.0	84.6
GDPR	MCCL: Index-based	descending	min	82.4	86.5	84.4
GDPR	MCCL: Index-based	descending	max	86.2	85.2	**85.7**
GDPR	MCCL: Index-based	ascending	min	84.6	86.5	85.5
GDPR	MCCL: Index-based	ascending	max	84.2	82.6	83.4

Table 6: Ablation analysis on PGR and GDPR datasets with respect to the order by which training examples are sorted for complexity indices (ascending versus descending, see Index Order column and line 2 in Algorithm 1), the mechanism by which indices are prioritized (model-based versus index-based, see Model column and line 7 in Algorithm 1) and the type of learning transition (easy-to-hard (Min error) versus hard-to-easy (Max error) transition, see Transition Order column and lines 9–13 in Algorithm 1).

Dataset	Model	Index Order	Transition Order	Acc
Ogbn-Arxiv	GTNN	–	–	71.6
Ogbn-Arxiv	MCCL: Model-based	descending	max	**76.5**
Ogbn-Arxiv	MCCL: Model-based	descending	min	76.0
Ogbn-Arxiv	MCCL: Model-based	ascending	max	76.4
Ogbn-Arxiv	MCCL: Model-based	ascending	min	76.2
Ogbn-Arxiv	MCCL: Index-based	descending	max	76.2
Ogbn-Arxiv	MCCL: Index-based	descending	min	76.3
Ogbn-Arxiv	MCCL: Index-based	ascending	max	76.4
Ogbn-Arxiv	MCCL: Index-based	ascending	min	76.0
Cora	GTNN	–	–	91.5
Cora	MCCL: Model-based	descending	max	**98.15**
Cora	MCCL: Model-based	descending	min	97.4
Cora	MCCL: Model-based	ascending	max	97.8
Cora	MCCL: Model-based	ascending	min	96.7
Cora	MCCL: Index-based	descending	max	97.8
Cora	MCCL: Index-based	descending	min	96.3
Cora	MCCL: Index-based	ascending	max	97.4
Cora	MCCL: Index-based	ascending	min	97.4

Table 7: Ablation analysis on Ogbn-Arxiv and Cora datasets with respect to the order by which training examples are sorted for complexity indices (ascending versus descending, see Index Order column and line 2 in Algorithm 1), the mechanism by which indices are prioritized (model-based versus index-based, see Model column and line 7 in Algorithm 1) and the type of learning transition (easy-to-hard (Min error) versus hard-to-easy (Max error) transition, see Transition Order column and lines 9–13 in Algorithm 1).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section Limitation

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3.1 Datasets

☑ B1. Did you cite the creators of artifacts you used?
Section 3.1 Datasets

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Table 2 Section 3.1

C ☑ Did you run computational experiments?

Section 3.4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3.3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3.4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.