AI Open 4 (2023) 56–63

Contents lists available at ScienceDirect

AI Open

journal homepage: www.keaipublishing.com/en/journals/ai-open

AdaDS: Adaptive data selection for accelerating pre-trained language model knowledge distillation

Qinhong Zhou [a], Peng Li [b,*], Yang Liu [b], Yuyang Guan [c], Qizhou Xing [c], Ming Chen [c], Maosong Sun [a], Yang Liu [a,b]

[a] Department of Computer Science and Technology, Tsinghua University, China
[b] Institute for AI Industry Research, Tsinghua University, China
[c] Beijing Sinovoice Technology Co., Ltd., China

ARTICLE INFO

Keywords:
Knowledge distillation
Pre-trained language model
Active learning

ABSTRACT

Knowledge distillation (KD) is a widely used method for transferring knowledge from large teacher models to computationally efficient student models. Unfortunately, the computational cost of KD becomes unaffordable as pre-trained language models (PLMs) grow larger. Computing KD loss on only part of the training set is a promising way to accelerate KD. However, existing works heuristically leverage only one static data selection strategy during the KD process, demonstrating inconsistent improvements across different distillation scenarios. In this work, we conduct a thorough study on various typical data selection strategies for KD, and show that this problem is due to the fact that the best data selection strategy is specific to various factors, including task, selected data size, and training stage. To automatically adapt to these factors, we propose a framework named AdaDS to learn to choose the data selection strategy adaptively during the KD process. Experimental results show that our proposed method is effective for various tasks and selected data sizes under both fine-tuning and pre-training stages, achieving comparable performance to DistilBERT with only 10% amount of queries to the teacher model.

1. Introduction

Due to the promising results on various natural language processing (NLP) tasks, pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020) have become a new paradigm for NLP. Recent works show that the performance of PLMs grows non-trivially with their parameter number (Radford et al., 2018, 2019; Brown et al., 2020; Raffel et al., 2020), making it a popular trend to explore even larger PLMs (Zhang et al., 2021; Chowdhery et al., 2022). However, it is difficult to deploy such huge PLMs, especially on mobile devices. Knowledge distillation (KD) (Hinton et al., 2015) is a popular method to alleviate the problem by compressing large teacher PLMs into small but effective student PLMs. Unfortunately, conventional KD methods need to query the large teacher PLMs with each training sample for at least one time, making KD itself computationally expensive for large PLMs. Therefore, how to accelerate KD for PLMs is an important problem with both significant research and practical values.

As a majority part of the computational cost of KD comes from and is roughly proportional to the number of times querying the teacher PLMs, computing KD loss on only part of the training set, mentioned as *data selection*, is a simple yet effective way for KD acceleration. Compared with KD that uses all training data, Xu et al. (2023) achieve better results with 79% computational cost on image classification tasks by combining data selection with mixup (Zhang et al., 2018). Li et al. (2021) achieve comparable KD performance with only 19% computational cost by selecting informative training data for PLMs. Therefore, accelerating KD with data selection is a promising research direction.

However, data selection for KD is still underexplored: (1) **Data selection strategies are underexplored**. Existing works mainly investigate uncertainty-based strategies (Settles, 2009; Settles and Craven, 2008; Xu et al., 2023; Li et al., 2021). However, data selection strategies based on other criteria have been widely studied in deep active learning works, demonstrating promising results on a wide range of datasets and tasks (Ren et al., 2022). Thus, more strategies need to be explored. (2) **Pre-training stage KD acceleration is underexplored**. As PLMs grow increasingly larger, building smaller PLMs by distillation at the pre-training stage from larger PLMs has significant practical value. However, KD acceleration for this scenario is uninvestigated in existing work. (3) **Dynamics of KD is underexplored**. Li et al. (2021) find

* Corresponding author.
E-mail address: lipeng@air.tsinghua.edu.cn (P. Li).

https://doi.org/10.1016/j.aiopen.2023.08.005
Received 22 May 2023; Received in revised form 16 July 2023; Accepted 9 August 2023
Available online 23 August 2023

that the most suitable teacher model and objective function evolves dynamically during the whole KD process. Our pilot study also shows even a manually designed simple dynamic data selection strategy can outperform existing single static strategies (Section 4.3). Therefore, we believe the dynamics of KD also plays an important role in KD acceleration and needs to be further explored.

To achieve a better understanding of data selection for KD, we conduct an empirical study of a broader range of data selection strategies for KD under both pre-training and fine-tuning stage KD settings. We empirically justify that the best-performed data selection strategy varies across (1) tasks, (2) selected data sizes, and (3) pre-training and fine-tuning stages. Therefore, the currently most widely used *single static data selection strategy paradigm* is sub-optimal. To the best of our knowledge, this is the first thorough investigation of data selection for PLM distillation acceleration.

Furthermore, to make the most use of all these data selection strategies and the dynamics of KD under different scenarios, we propose an adaptive data selection method named AdaDS to accelerate KD for PLMs, which chooses data selection strategies dynamically during the KD process. The appropriateness of a strategy is measured by its impact on student performance. As achieving the accurate appropriateness for each strategy is time-consuming, we calculate the student loss on only part of the training data to estimate the appropriateness, and propose a more efficient greedy reward design to make the student loss decrease the fastest. To further reduce the extra computational cost introduced by the data strategy selector, we leverage a parameter-efficient yet effective tabular Q-learning method (Matiisen et al., 2020) to optimize the selector. Experimental results show that our proposed AdaDS method achieves promising results across a wide range of tasks and selected data sizes under both pre-training and fine-tuning stage KD settings. Especially, we recover 99.9% performance of DistilBERT (Sanh et al., 2019) with only 10% queries to the teacher model.

2. Related work

Knowledge distillation (KD) (Hinton et al., 2015) aims to transfer knowledge from a cubersome teacher model to a lightweight student model, and has been shown effective for compressing pre-trained language models (PLMs) (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020). The efficiency issue of KD is first highlighted in computer vision (Wang et al., 2020a). The existing KD acceleration works either use uncertainty-based data selection to reduce the number of times querying the teacher (Wang et al., 2020a; Xu et al., 2023; Li et al., 2021), or reduce the cost of each query (Lin et al., 2022). In this paper, we choose the promising and easy-to-implement data selection method to accelerate KD. Wang et al. (2020a) first proposed to select samples with high uncertainty for KD to reduce computational cost, and achieved this by active mixup (Zhang et al., 2018) and search the mixup coefficient with the highest uncertainty. Based on this approach, Xu et al. (2023) achieved further acceleration by reducing the number of queries to student models. In natural language processing (NLP), Wang et al. (2021) analyzed the data selection problem for KD, but its word-level selection strategy is unable to reduce the computational cost of KD. To accelerate KD in NLP, Li et al. (2021) selected samples with the highest uncertainties during training and found this strategy effective.

Different from these existing KD acceleration works, our method is not limited to uncertainty-based data selection strategy. Instead, we explore several popular data selection strategies from active learning, which aims to reduce the amount of data required to train by allowing models to choose the data (Settles, 2009). In active learning, although uncertainty-based strategy (Lewis and Gale, 1994) is the simplest and most commonly used, there is no universal dominant strategy. Some works also focus on other strategies such as information density strategy (Settles and Craven, 2008), cluster-based strategy (Xu et al., 2003), query-by-committee strategy (Seung et al., 1992), and expected model change strategy (Settles and Craven, 2008). In this paper, we take the advantages of these strategies and dynamically select the most appropriate strategy during the distillation process.

Few-shot KD (Pan et al., 2021; Zhou et al., 2022; Sauer et al., 2022) is another related topic to this paper. These works emphasize data efficiency, aiming to achieve better performance with limited training data, without strict constraints on computational costs. In contrast, our focus lies in computational efficiency during KD, targeting at achieving better performance while keeping the computational costs limited.

3. Data selection for KD acceleration

3.1. Problem formulation

Knowledge distillation is generally performed by minimizing the Kullback–Leibler (KL) divergences of certain statistics between the student and teacher models on a training dataset $\mathcal{D} = \{(x_i, y_i)\}$, where x_i and y_i are the input and desired output respectively. Given a mini-batch $\mathcal{B} \subset \mathcal{D}$, data selection-based KD acceleration methods select a subset of samples $\hat{\mathcal{B}} \subseteq \mathcal{B}$ and conduct KD only on $\hat{\mathcal{B}}$ to reduce computational cost. Generally, the samples are ranked by a scoring function $\mathcal{S}(x_i, y_i, \mathcal{B}, f_{\theta_S})$, where f_{θ_S} is a student model, and those meet certain criteria are selected. For example, the widely used threshold-based selection methods can be formulated as

$$\hat{\mathcal{B}} = \left\{ (x_i, y_i) | \mathcal{S}(x_i, y_i, \mathcal{B}, f_{\theta_S}) < \delta \right\}, \tag{1}$$

where δ is the threshold. We will abbreviate $\mathcal{S}(x_i, y_i, \mathcal{B}, f_{\theta_S})$ to $\mathcal{S}(x_i)$ for brevity.

KD for PLMs can be roughly divided into two types: (1) *pre-training stage KD* (Sanh et al., 2019; Jiao et al., 2020; Liu et al., 2020; Wang et al., 2020b), which distills knowledge from a *raw* teacher PLM to a student PLM, and (2) *fine-tuning stage KD* (Tang et al., 2019; Sun et al., 2019), which distills knowledge from a *fine-tuned* teacher PLM to a downstream student model. The first type is usually conducted on unlabeled pre-training corpus and more computationally costly than the second, which is performed on labeled downstream task datasets. In the following sections, we will show that these two scenarios have distinct KD-related properties, which lead to different preferences on data selection.

3.2. Computational cost

For each sample, the major KD computational cost consists of three parts: (1) the student forward cost F_S, (2) the student backward cost B_S, and (3) the teacher forward cost F_T. Following Li et al. (2021), we formulate the total computational cost C for an epoch as:

$$C \approx |\mathcal{D}| \cdot (F_S + B_S + F_T). \tag{2}$$

Suppose there are $r \cdot |\mathcal{D}|$ samples are actually selected for KD, where $r \in (0, 1]$, the total computational cost $\hat{C}$ becomes

$$\hat{C} \approx |\mathcal{D}| \cdot F_S + r \cdot |\mathcal{D}| \cdot (B_S + F_T) + C_{sel}, \tag{3}$$

where C_{sel} is the extra computational cost introduced by the data selector. Obviously, we have to make r and C_{sel} small to achieve better speedup.

4. Dynamic data selection is essential

Most existing data selection works for KD leverage a single static data selection strategy. In this section, we will show that the best-performed data selection strategy varies dynamically across various aspects, including tasks, selected data sizes, pre-training and fine-tuning stages, and the stages in a single KD process.

4.1. Data selection strategies

Most existing KD acceleration methods only leverage uncertainty-based data selection strategies. However, a wider range of strategies have been investigated in other research fields (Ren et al., 2022), which are also worth to be explored in KD acceleration. The typical data selection strategies are summarized as follows[1]:

Random strategy. Samples are selected randomly. It is widely used as a baseline strategy.

Uncertainty-based strategy. Samples with higher student model prediction uncertainty are selected. It is the only type of strategy that has been explored by existing KD acceleration works (Xu et al., 2023; Li et al., 2021). Three kinds of uncertainty score function $\mathcal{U}$ are used in previous works:

- **Entropy**, which is computed over the student prediction distribution:

$$\mathcal{U}(x) = -\sum_{y} P(y|x) \log P(y|x). \tag{4}$$

- **Margin**, which measures the probability margin between the most and second most probable labels y_1^* and y_2^*:

$$\mathcal{U}(x) = P(y_1^*|x) - P(y_2^*|x). \tag{5}$$

- **Least-Confidence (Conf)**, which measures the uncertainty of model about the most probable label y_1^*:

$$\mathcal{U}(x) = 1 - P(y_1^*|x). \tag{6}$$

Certainty-based strategy (Certainty). Samples with higher student model prediction *certainty* are selected (Attenberg et al., 2010). Generally, the negative prediction entropy is used as the measurement for certainty C

$$C(x) = \sum_{y} P(y|x) \log P(y|x). \tag{7}$$

Diversity-based clustering strategy (Cluster). Samples are clustered into groups and those closer to the clustering center are selected (Xu et al., 2003), hoping the selected samples have high representativeness. In this work, the samples are represented using the feature $\boldsymbol{x}$ from the last layer of the student model and clustered using K-means++ (Arthur and Vassilvitskii, 2007).

Density-based strategy (Density). Samples that have higher average similarities with all other ones are selected. Following the original information density method (Settles and Craven, 2008), we use cosine similarity as the similarity function

$$\mathtt{sim}_{cos}(x, x_i) = \frac{\boldsymbol{x}^\top \boldsymbol{x}_i}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{x}_i\|}. \tag{8}$$

4.2. Experimental settings

Fine-tuning stage distillation. We use a fine-tuned $\text{BERT}_{\text{BASE}}$ model as the teacher, and a 6-layer model as the student. Considering the size of the training corpus, we choose four tasks from the GLUE benchmark (Wang et al., 2018) which have sufficient training data, including MNLI (Williams et al., 2018), SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2016), and QQP (Chen et al., 2017). For each task, we train the student for 5 epochs with a 2×10^{-5} learning rate. Please refer to Wang et al. (2018) for the details of the evaluation metrics.

Pre-training stage distillation. We conduct pre-training experiments with the DistilBERT framework (Sanh et al., 2019). We use the 6-layer DistilBERT as the student architecture, and $\text{BERT}_{\text{BASE}}$ as the teacher. The training configurations are the same as DistilBERT, including parameters initialization, loss computation, and learning rate schedules. We use a concatenation of English Wikipedia and Toronto Book Corpus (Zhu et al., 2015) as our training corpus, which is the as the DistilBERT and BERT model. We evaluate pre-trained models on the GLUE benchmark, including two single sentence classification tasks: CoLA (Warstadt et al., 2019), SST-2, and three sentence similarity tasks: MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QQP, and four natural language inference tasks: MNLI, QNLI, RTE (Bentivogli et al., 2009), and WNLI (Levesque et al., 2012).

4.3. Experimental results

No strategy dominates in all tasks. As shown in Table 1, we find that when only using 10% data for KD, most examined data selection strategies have improvements over the *Random* selection strategy. Nevertheless, no approach emerges as the dominant strategy for all tasks. Most data selection strategies even have worse performance than *Random* strategy in one or some certain tasks.

Selected data size matters. Further results of fine-tuning stage distillation on the QQP task (Table 2) indicate that the appropriate selection strategy changes with selected data sizes. The *Certainty* strategy is the most efficient when selection ratio is 10% or 20%, while the uncertainty-based strategies, such as *Entropy* and *Margin*, dominate the larger selection ratios. The *Certainty* strategy is even the worst when selection ratio is 50%. This phenomenon indicates that the proper strategy changes with different selected data sizes.

Dynamics of KD matters. To explore the appropriate strategy in different training stages, we manually design simple dynamic data selection strategies, and the results are show in Table 3. Taking "*Entropy* → *Certainty*" as an example, it means choosing samples by the *Entropy* strategy in the first half of training, and selecting by the *Certainty* strategy in the second half. Surprisingly, we find the simple combination of strategies "*Certainty* → *Entropy*" outperform single strategies. These results shows that different selection strategies are needed in different training stages.

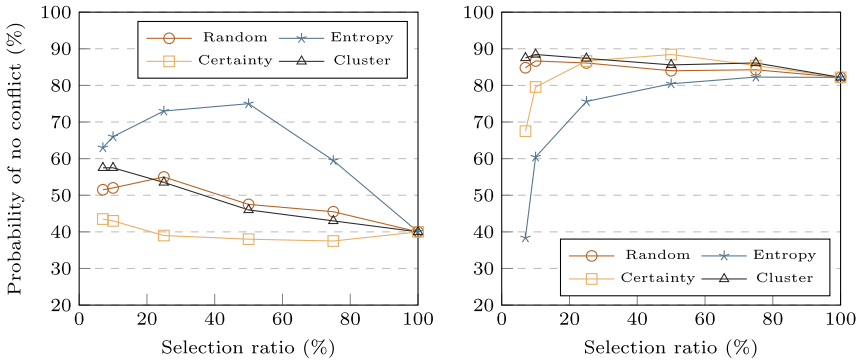
Pre-training and fine-tuning distillation differs. The results in Table 4 show that uncertainty-based strategies perform worse than the *random* strategy in pre-training stage distillation, while they are effective in the fine-tuning stage (Table 1). On the contrary, the *Cluster* strategy, which has poor performance in fine-tuning stage distillation (Table 1), performs relative better *Random* in pre-training stage distillation.

Apparently, the difference in training tasks and the complexity of optimization contribute to the phenomenon. However, the further reason behind this is unclear. Here we provide one possible explanation. Following Wang et al. (2021), we analyze the gradient of student model parameters ∇_{θ_S}. The gradient ∇_{θ_S} can be decomposed into two parts, including the gradient respect to golden cross-entropy loss $\nabla_{\theta_S} L_{ce}$, and the gradient from distillation loss $\nabla_{\theta_S} L_{kd}$. We randomly select 36K sentences from the training corpus, and calculate the probability that $\nabla_{\theta_S} L_{ce}$ shares the same direction with $\nabla_{\theta_S} L_{kd}$ on selected data.

Fig. 1 represents the probability that the angle between $\nabla_{\theta_S} L_{ce}$ and $\nabla_{\theta_S} L_{kd}$ is less than 90°. We repeat the experiments with different data selection ratios. For uncertainty-based strategy, $\nabla_{\theta_S} L_{ce}$ and $\nabla_{\theta_S} L_{kd}$ are more likely to conflict with each other at the pre-training stage distillation, while the risk is lower than random in the fine-tuning stage. The conflict leads to a risk of introducing noise and disturbs the direction of parameter updating (Wang et al., 2021).

In conclusion, we find that the proper data selection strategy for KD changes for different tasks and selected data sizes. It also differs between pre-training and fine-tuning distillation. Therefore, dynamically adapting data selection strategy is essential for KD.

[1] Note that as C_{sel} in Eq. (3) should be small, we do not consider time-consuming strategies such as the expected model change approach (Freytag et al., 2014) in this section.



(a) Fine-tuning stage distillation (b) Pre-training stage distillation

Fig. 1. The probability for gradients of L_{ce} and L_{kd} have less than 90°.

Table 1
Experiments of fine-tuning stage distillation on MNLI, SST-2, QNLI, and QQP tasks when the selection ratio is set to 10%. † denotes statistically significant improvement over the random strategy. The best performances of static selection strategies are **bolded**, and performances worse than *Random* are underlined. Results of students are averaged over 5 models fine-tuned with different random seeds on the validation set, and **Speedup** is calculated by the average FLOPs per instance needed by KD.

Method	Speedup	MNLI-m/mm	SST-2	QNLI	QQP	Avg
Teacher	–	84.11/84.37	93.58	91.31	89.45	88.56
All Data	1.0×	82.35/82.52	90.99	88.20	89.06	86.62
Random	10.0×	80.16/80.66	89.96	86.34	87.47	84.92
Margin	5.0×	80.17/80.80	90.57	**86.86**	87.61	85.20
Conf		80.02/80.73	**90.64**	**86.86**	87.61	85.17
Entropy		79.99/80.70	90.55	86.80	87.63	85.13
Certainty		**80.97/81.05**	89.45	86.69	**88.64**†	**85.36**†
Cluster		80.28/80.42	90.02	86.23	87.19	84.83
Density		80.45/80.63	90.37	86.46	87.69	85.12†

Table 2
The best and the worst data selection strategies across different selected data sizes, which are presented as selection ratios. The strategies are evaluated on QQP dev set.

Selection ratio	10%	20%	30%	40%	50%
Best Strategy	Certainty	Certainty	Entropy	Margin	Entropy
Worst Strategy	Diversity	Diversity	Density	Density	Certainty

Table 3
Comparison between static and dynamic strategies on QQP dev set when selection ratio is set to 30%.

Strategy		QQP
Static	Random	88.25
	Entropy	88.54
	Conf	88.54
	Margin	88.53
	Certainty	88.64
	Cluster	88.23
	Density	88.28
Dynamic	Entropy → Certainty	88.61
	Certainty → Entropy	**88.70**

5. AdaDS: Adaptive data selection for KD

To obtain consistent improvement over various tasks, data sizes, and training steps, we propose a reinforced data selection framework AdaDS which learns to adapt data selection strategy dynamically during KD progress.

5.1. Overview

We view the data selection problem as a batch mode Partially Observable Markov Decision Process (POMDP) (Matiisen et al., 2020), and solve it with our data selection framework AdaDS (illustrated in Fig. 2). First, several basic selection strategies, which are typical active learning algorithms, provide different proposals of data selection from the current batch $\mathcal{B}$ with the help of feature maps and predictions given by the student model f_{θ_S}. Second, a reinforced data strategy selector (RL selector) chooses one of the proposals and only the selected data in the chosen proposal will be used in KD. To achieve consistent improvement across different distillation tasks, the AdaDS framework dynamically adjusts the RL selector by its reward, which comes from the observation of the student model performance change. We will describe the design of adaptive data selection framework in Section 5.2 and the definitions of observation and reward in Section 5.3.

5.2. Reinforce data selection strategy selector

One straightforward design for the adaptive data selection framework is directly rating each original sample through one data selector model, and selecting original samples with the highest rates. However, this design is impractical due to the complexity issues and the sparsity of rewards. Under this design, for each sample, the data selector needs to decide whether to select or not, so the data selector takes actions frequently. In contrast, the rewards available are sparse. Considering the computational cost required by each observation, to achieve KD acceleration, the number of observation steps in a complete KD process is limited. In addition, the dynamics of KD discussed in Section 4.3 also limits the reuse of historical rewards. Therefore, under this straightforward design, the data selector needs to make a tremendous number of actions (usually tens of thousands) to get one reward, which makes data selector model hardly to be trained.

In this paper, to reduce the complexity of data selection and the sparsity of rewards during training, we decompose the adaptive data selection framework into two components, including the *basic selection strategies* and *reinforce data selection strategy selector* (RL selector). Specifically, we introduce basic selection strategies from typical query

Table 4
Experiments of pre-training stage distillation with DistilBERT framework when selection ratio is set to 10%. **Speedup** is calculated by the average FLOPs per instance needed by KD. The best performances of static selection strategies are **bolded**, and performances worse than *Random* are underlined. The uncertainty-based strategy performs even worse than the random strategy.

Method	MNLI-m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
Teacher	84.28/84.70	89.44	91.51	92.48	55.88	88.67	85.34	64.55	81.87
All Data	82.09/82.34	88.42	88.51	90.73	50.34	86.42	86.10	59.13	79.34
Random	81.90/82.36	88.30	88.64	91.19	48.13	85.22	85.79	57.76	78.81
Entropy	81.82/82.30	88.41	88.82	90.99	42.45	84.30	**86.34**	59.06	78.28
Margin	80.91/81.42	88.00	86.72	90.96	46.72	82.42	77.13	56.90	76.80
Conf	81.61/82.17	88.27	**88.91**	91.05	41.23	84.06	85.56	59.86	78.08
Certainty	81.13/81.77	88.12	87.88	90.87	48.72	83.87	82.39	**59.93**	78.30
Cluster	81.81/82.30	**88.43**	88.64	91.24	**49.13**	**85.48**	85.41	57.47	**78.88**
Density	**81.93/82.45**	88.35	88.67	**91.33**	46.74	84.36	84.61	58.34	78.53

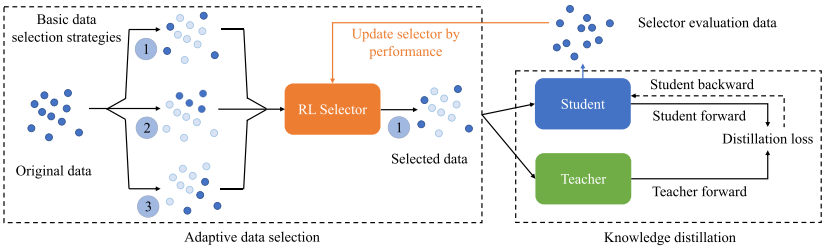


Fig. 2. Illustration of our reinforcement learning based method. In each training step, we choose the most useful data from the mini-batch. The reinforced data selection strategy selector (RL selector) chooses one of the basic data selection strategies to select samples. Only the selected data are used for the knowledge distillation process, including student forward, teacher forward, and student backward passes. During distillation, the RL selector is constantly updated according to the performance of the student model, which is evaluated on the instances randomly sampled from the training dataset (selector evaluation data).

strategies described in Section 4.1, including the uncertainty-based strategy, diversity-based clustering approach, and density-based strategy. Each strategy captures different attributes of samples, such as informativeness, density, and diversity, providing multiple selection criteria for the RL selector. Then, the RL selector only needs to choose one of these strategies at each observation step t, which is a multi-arm bandit problem.

We choose the simple and straightforward Online algorithm (Matiisen et al., 2020) to solve this multi-arm bandit problem. The Online algorithm is a simplified version of Q-learning for batch mode POMDP, where "Q" refers to the expected rewards for an action taken in a given state. The algorithm estimates the Q value by the Q from the last time step and the current reward r_t:

$$Q_{t+1}(a_t) = \alpha r_t + (1-\alpha) Q_t(a_t) \tag{9}$$

where a_t is the current action of step t, and α is the learning rate of the Online algorithm. After step t, a_{t+1} is selected by the ϵ-greedy algorithm, which chooses $\arg\max_a Q_{t+1}(a)$ with a probability of $1-\epsilon$ and a random action of ϵ.

5.3. Observation and reward

The objective of RL selector is for the student to achieve the lowest loss on the training dataset. Therefore, the ideal observation o_t for each observation step t should be the loss calculated on the entire training dataset

$$\frac{1}{|D|}\sum_{i=1}^{|D|} L(x_i, y_i, \theta_S), \tag{10}$$

and the ideal reward should be the performance of student model after the whole KD progress.

However, there are two problems with this design: (1) At each observation step t, it is difficult for us to directly get the observation o_t by computing the loss on the entire training set considering the computational complexity factor. (2) At the same time, due to the time complexity and the dynamic characteristics of knowledge distillation, it is also difficult for us to obtain the ideal reward for each action.

To address the former problem, we compute the loss of the student model only on a fraction of the dataset, named as *selector evaluation data*, thereby estimating the loss on the entire training dataset. In order to prevent the data leakage of the development set, we construct selector evaluation data with M randomly sampled training data points:

$$\frac{1}{M}\sum_{i=1}^{M} L(x_i, y_i, \theta_S). \tag{11}$$

To address the latter issue, we define the reward r in a greedy manner to make the loss of the student model on the training dataset drops the fastest. Specifically, we define the reward as the acceleration of the loss decay, i.e. the increase in the loss decaying speed v. To compute this acceleration, we use a windowing mechanism, which keeps the last K observations. Then we use the linear regression to calculate the loss decaying speed v, which equals to the negative slope of $\{o_{t-K+1}, o_{t-K+2}, \ldots, o_t\}$, where t is the current observation step. Finally, the current reward r_t can be described as $r_t = v_t - v_{t-1}$.

5.4. Computational cost of AdaDS

During the KD process, AdaDS requires a forward pass on all data within each training batch for data selection. Moreover, a forward pass is needed for each selector evaluation sample during the selector evaluation period. The complexity of selector update is negligible due to the utilization of a tabular-based Q-learning method. Consequently, the time complexity of the data selection process is

$$|D| \cdot F_S + r \cdot |D| \cdot (B_S + F_T) + n_{eval} M F_S, \tag{12}$$

where n_{eval} denotes the number of selector evaluation periods during one epoch.

6. Experiments of AdaDS

6.1. Baselines

We compare our method with static baseline strategies described in Section 4.1, including *Random*, *Entropy*, *Margin*, *Conf*, *Certainty*, *Cluster*,

Table 5
Experiments of fine-tuning stage distillation when the selection ratio is set to 10%. We include the *Best* and the *Worst* results of static selection strategies in Table 1. Notice that the **Avg** scores of *Best* and *Worst* are the best or worst among **Avg**, not the averages of the best or worst scores.

Method		Speedup	MNLI-m/mm	SST-2	QNLI	QQP	Avg
Teacher ($BERT_{BASE}$)		–	84.11/84.37	93.58	91.31	89.45	88.56
All Data		1.0×	82.35/82.52	90.99	88.20	89.06	86.62
Random		10.0×	80.16/80.66	89.96	86.34	87.47	84.92
Static	Best	3.6×	80.97/81.05	**90.64**	86.86	88.64†	85.36†
	Worst		79.99/80.42	89.45	86.23	87.19	84.83
Dynamic	Ensemble		80.11/80.55	89.95	86.26	87.25	84.82
	RandMix		80.58†/80.94	90.30	86.97†	87.85†	85.33†
Ours		3.4×	**81.19/81.39**	90.27	**87.17**†	**88.73**†	**85.75**†

and *Density*. To explore the effectiveness of our dynamic method, we also compare our method with the following simple dynamic strategies:

- **Ensemble**: a naive ensemble of basic strategies, using the majority voting ensemble to combine the choices of basic strategies.
- **RandMix**: a random schedule of basic strategies, using one of the basic strategies with the same probabilities at any training step.

6.2. Results

AdaDS achieves consistent improvement over static strategies. Table 5 shows the performance of baseline strategies and our method when 10% data are selected. Our method achieves improvement over random strategy across all tasks and outperforms all static selection strategies in most tasks. The results in Fig. 3 justify that AdaDS is also effective over different selected data sizes. For the challenging pre-training distillation (in Fig. 4), in which static strategies fail to have significant improvement, AdaDS also surpasses Random in the averaged GLUE score. In summary, different from the static selection strategies, AdaDS achieves consistent improvement over different tasks, data sizes, and both pre-training and fine-tuning distillation.

AdaDS is effective among dynamic strategies. According to the results in Table 5, although basic strategies are effective, our method still surpasses the highest of those performances in most cases. This phenomenon indicates that the dynamic schedules has a higher upper bound of performance than static strategies. However, not all dynamical schedules are better than static. The results of *Ensemble* and *RandMix* strategies show that these simple combinations are unable to outperform static strategies. We argue the performance gap between AdaDS and dynamic baseline strategies comes from the effective schedule learned by the Online algorithm (described in Section 5.2). For example, Fig. 5(a) shows the strategy of our RL agent during fine-tuning stage distillation on the QQP task when only 10% data are selected. In fine-tuning stage distillation (see in Fig. 3), the *Certainty* strategy has more significant improvement over other strategies with limited data size. Our method has similar performances over these ratios because the Online algorithm simply learns to select *Certainty* strategy during training. The focus on the density attribute of samples near the end of training contributes to the slight improvement of AdaDS over *Certainty* in Table 5.

The schedules of AdaDS are dynamic and adaptive. To further explore the schedules learned by the Online algorithm, we observe the actions of the RL selector of AdaDS during training. Fig. 5 shows the learned schedules across different selection ratios and distillation tasks. We find all these schedules are dynamic, and the schedules for pre-training stage distillation (in Figs. 5(b) and 5(c)) differs from those for fine-tuning stage distillation (in Fig. 5(a)). The difference between Figs. 5(b) and 5(c) indicates that different strategies are needed for different selection ratios also.

However, it is difficult to give a thorough explanation of these schedules since the actions of the RL selector depends on the reward (defined in Section 5.3) history. Here we provide several influential factors. (a) The difference between choices in Figs. 5(a) and 5(b) is due to the different training tasks and data characteristics. As mentioned in Section 4.3, with a 10% selection ratio, the top 10% samples sorted by prediction entropy leads to a higher probability of conflict in gradient. During training, this problem can be mitigated under a larger batch size (pre-training stage), but aggravated under a smaller batch size (fine-tuning stage). (b) The difference between selection ratios (Figs. 5(b) and 5(c)) also has a complex influence. For example, considering the gradient conflict we mentioned before, instances with top 20% entropy have a much lower risk of conflict than those with 10% entropy (Fig. 1a). This lower risk of conflict is a positive factor in choosing the *Entropy* strategy.

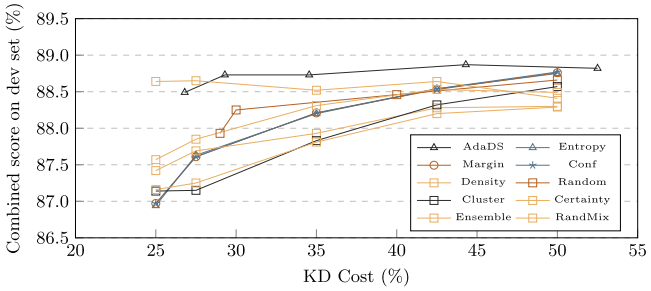


Fig. 3. The mean combined score on fine-tuning stage distillation (QQP task) over different selection ratios. Our method achieves higher performance than all basic strategies.

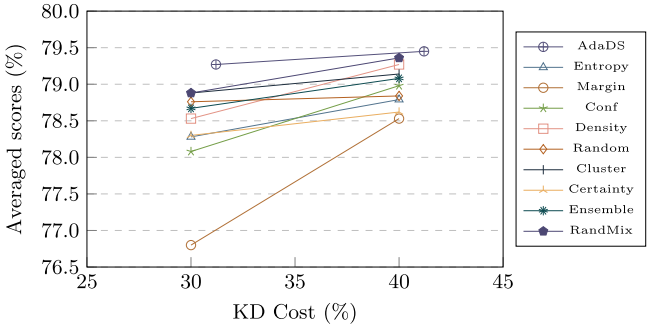
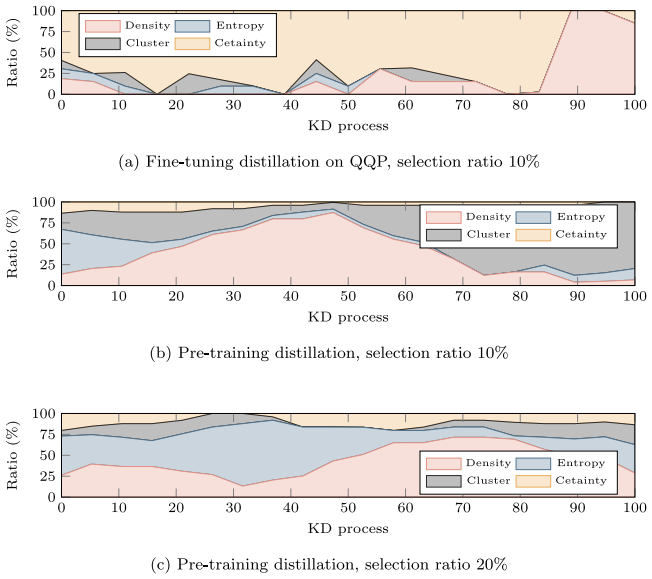


Fig. 4. Pre-training stage distillation experiments, the results are the averaged score on evaluation sets of 8 GLUE tasks. We record the results of 30% and 40% selection ratios for *Random* strategy, while 10% and 20% for other strategies.

AdaDS is efficient. As Figs. 3 and 4 show, AdaDS achieves the same performance with much less overall computational cost, or better performance with the same overall computational cost. Although extra computational cost is introduced by the observation and reward calculation (describe in Section 5.3) of AdaDS, it is tolerable considering the performance improvements.

Applications on other KD methods and architectures. Moreover, our method can be used simultaneously with other knowledge distillation approaches such as BERT-PKD (Sun et al., 2019), TinyBERT (Jiao



(a) Fine-tuning distillation on QQP, selection ratio 10%

(b) Pre-training distillation, selection ratio 10%

(c) Pre-training distillation, selection ratio 20%

Fig. 5. Schedules of selection strategy change across different selection ratios and distillation tasks. For each schedule, the dominant strategy also changes during the training process.

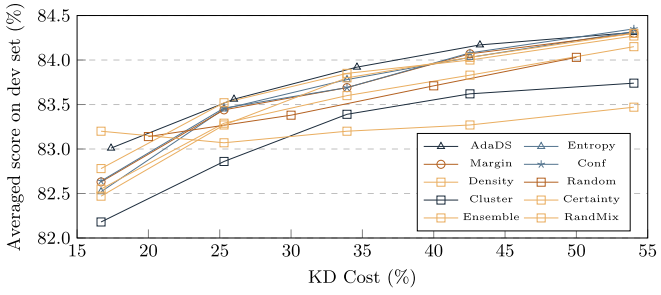


Fig. 6. The averaged score on TinyBERT architecture over four different NLU tasks. Our method achieves higher performance than all basic strategies on most selection ratios.

et al., 2020), and MetaKD (Pan et al., 2021). To validate the applicability of our method across various KD frameworks, we conduct experiments on the TinyBERT architecture using QQP, MNLI, QNLI, and SST-2 datasets. We summarize the averaged scores on the four datasets in Fig. 6. The results illustrate that our method surpasses all baseline strategies in the most of selection ratios, demonstrating it as the most efficient data selection strategy on TinyBERT architecture.

In summary, with the help of dynamic and adaptive schedules, the AdaDS method is effective and can outperform both static and dynamic baseline strategies in most cases.

7. Conclusion

In this paper, we propose an adaptive data selection framework AdaDS to accelerate knowledge distillation. AdaDS is effective across different tasks, selected data sizes, pre-training, and fine-tuning stages, while the performance of the existing acceleration methods is inconsistent. AdaDS framework also outperforms static methods by utilizing the dynamics of the distillation process.

In future, we will investigate and further improve the effectiveness of our method on extremely large PLMs such as GPT-3 (Brown et al., 2020). And we also plan to explore a larger action space for the adaptive selector.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tsinghua University, Beijing Sinovoice Technology co., Ltd, Peking University, Tencent, Baidu, WeBank, Shanghai Artificial Intelligence Laboratory, University of Massachusetts Amherst, IBM.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61925601, 62276152). We thank all anonymous reviewers for their valuable comments and suggestions on this work.

References

Arthur, David, Vassilvitskii, Sergei, 2007. K-Means++: The advantages of careful seeding. In: SODA 2007.

Attenberg, Josh, Melville, Prem, Provost, Foster, 2010. A unified approach to active dual supervision for labeling features and examples. In: ECML PKDD 2010.

Bentivogli, Luisa, Clark, Peter, Dagan, Ido, Giampiccolo, Danilo, 2009. The fifth PASCAL recognizing textual entailment challenge. In: TAC 2009.

Brown, Tom, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared D, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel, Wu, Jeffrey, Winter, Clemens, Hesse, Chris, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, Amodei, Dario, 2020. Language models are few-shot learners. In: NeurIPS 2020.

Cer, Daniel, Diab, Mona, Agirre, Eneko, Lopez-Gazpio, Iñigo, Specia, Lucia, 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: SemEVal 2017.

Chen, Zihang, Zhang, Hongbo, Zhang, Xiaoji, Zhao, Leqi, 2017. Quora question pairs.

Chowdhery, Aakanksha, Narang, Sharan, Devlin, Jacob, Bosma, Maarten, Mishra, Gaurav, Roberts, Adam, Barham, Paul, Chung, Hyung Won, Sutton, Charles, Gehrmann, Sebastian, et al., 2022. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, Toutanova, Kristina, 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT 2019.

Dolan, Bill, Brockett, Chris, 2005. Automatically constructing a corpus of sentential paraphrases. In: IWP 2005.

Freytag, Alexander, Rodner, Erik, Denzler, Joachim, 2014. Selecting influential examples: Active learning with expected model output changes. In: ECCV 2014.

Hinton, Geoffrey, Vinyals, Oriol, Dean, Jeff, 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Jiao, Xiaoqi, Yin, Yichun, Shang, Lifeng, Jiang, Xin, Chen, Xiao, Li, Linlin, Wang, Fang, Liu, Qun, 2020. TinyBERT: Distilling BERT for natural language understanding. In: Findings of EMNLP 2020.

Levesque, Hector J., Davis, Ernest, Morgenstern, Leora, 2012. The winograd schema challenge. In: KR 2012.

Lewis, David D., Gale, William A., 1994. A sequential algorithm for training text classifiers. In: SIGIR 1994.

Li, Lei, Lin, Yankai, Ren, Shuhuai, Li, Peng, Zhou, Jie, Sun, Xu, 2021. Dynamic knowledge distillation for pre-trained language models. In: EMNLP 2021.

Lin, Wenye, Li, Yangming, Liu, Lemao, Shi, Shuming, Zheng, Hai-tao, 2022. Efficient sub-structured knowledge distillation. arXiv preprint arXiv:2203.04825.

Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, Stoyanov, Veselin, 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.

Liu, Weijie, Zhou, Peng, Wang, Zhiruo, Zhao, Zhe, Deng, Haotang, Ju, Qi, 2020. FastBERT: a self-distilling BERT with adaptive inference time. In: ACL 2020.

Matiisen, Tambet, Oliver, Avital, Cohen, Taco, Schulman, John, 2020. Teacher–student curriculum learning. IEEE Trans. Neural Netw. Learn. Syst..

Pan, Haojie, Wang, Chengyu, Qiu, Minghui, Zhang, Yichang, Li, Yaliang, Huang, Jun, 2021. Meta-KD: A meta knowledge distillation framework for language model compression across domains. In: ACL 2021.

Radford, Alec, Narasimhan, Karthik, Salimans, Tim, Sutskever, Ilya, et al., 2018. Improving language understanding by generative pre-training.

Radford, Alec, Wu, Jeffrey, Child, Rewon, Luan, David, Amodei, Dario, Sutskever, Ilya, et al., 2019. Language models are unsupervised multitask learners.

Raffel, Colin, Shazeer, Noam, Roberts, Adam, Lee, Katherine, Narang, Sharan, Matena, Michael, Zhou, Yanqi, Li, Wei, Liu, Peter J, et al., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.

Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, Liang, Percy, 2016. SQuAD: 100, 000+ questions for machine comprehension of text. In: EMNLP 2016.

Ren, Pengzhen, Xiao, Yun, Chang, Xiaojun, Huang, Po-Yao, Li, Zhihui, Gupta, Brij B., Chen, Xiaojiang, Wang, Xin, 2022. A survey of deep active learning. ACM Comput. Surv..

Sanh, Victor, Debut, Lysandre, Chaumond, Julien, Wolf, Thomas, 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: NeurIPS 2019 Workshop on Energy Efficient Machine Learning and Cognitive Computing.

Sauer, Anna, Asaadi, Shima, Küch, Fabian, 2022. Knowledge distillation meets few-shot learning: An approach for few-shot intent classification within and across domains. In: Proceedings of the 4th Workshop on NLP for Conversational AI.

Settles, Burr, 2009. Active Learning Literature Survey. Computer Sciences Technical Report, University of Wisconsin–Madison.

Settles, Burr, Craven, Mark, 2008. An analysis of active learning strategies for sequence labeling tasks. In: EMNLP 2008.

Seung, H. S., Opper, M., Sompolinsky, H., 1992. Query by committee. In: COLT 1992.

Socher, Richard, Perelygin, Alex, Wu, Jean, Chuang, Jason, Manning, Christopher D, Ng, Andrew Y, Potts, Christopher, 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP 2013.

Sun, Siqi, Cheng, Yu, Gan, Zhe, Liu, Jingjing, 2019. Patient knowledge distillation for BERT model compression. In: EMNLP-IJCNLP 2019.

Sun, Zhiqing, Yu, Hongkun, Song, Xiaodan, Liu, Renjie, Yang, Yiming, Zhou, Denny, 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In: ACL 2020.

Tang, Raphael, Lu, Yao, Liu, Linqing, Mou, Lili, Vechtomova, Olga, Lin, Jimmy, 2019. Distilling task-specific knowledge from bert into simple neural networks. arXiv preprint arXiv:1903.12136.

Wang, Dongdong, Li, Yandong, Wang, Liqiang, Gong, Boqing, 2020a. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In: CVPR 2020.

Wang, Alex, Singh, Amanpreet, Michael, Julian, Hill, Felix, Levy, Omer, Bowman, Samuel, 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: EMNLP 2018 Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.

Wang, Wenhui, Wei, Furu, Dong, Li, Bao, Hangbo, Yang, Nan, Zhou, Ming, 2020b. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: NeurIPS 2020.

Wang, Fusheng, Yan, Jianhao, Meng, Fandong, Zhou, Jie, 2021. Selective knowledge distillation for neural machine translation. In: ACL 2021.

Warstadt, Alex, Singh, Amanpreet, Bowman, Samuel, 2019. Neural network acceptability judgments. In: TACL 2019.

Williams, Adina, Nangia, Nikita, Bowman, Samuel, 2018. A broad-coverage challenge corpus for sentence understanding through inference. In: NAACL 2018.

Xu, Guodong, Liu, Ziwei, Loy, Chen Change, 2023. Computation-efficient knowledge distillation via uncertainty-aware mixup. Pattern Recognit..

Xu, Zhao, Yu, Kai, Tresp, Volker, Xu, Xiaowei, Wang, Jizhi, 2003. Representative sampling for text classification using support vector machines. In: ECIR 2003.

Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N., Lopez-Paz, David, 2018. Mixup: Beyond empirical risk minimization. In: ICLR 2018.

Zhang, Zhengyan, Gu, Yuxian, Han, Xu, Chen, Shengqi, Xiao, Chaojun, Sun, Zhenbo, Yao, Yuan, Qi, Fanchao, Guan, Jian, Ke, Pei, Cai, Yanzheng, Zeng, Guoyang, Tan, Zhixing, Liu, Zhiyuan, Huang, Minlie, Han, Wentao, Liu, Yang, Zhu, Xiaoyan, Sun, Maosong, 2021. CPM-2: Large-scale cost-effective pre-trained language models. AI Open.

Zhou, Wangchunshu, Xu, Canwen, McAuley, Julian, 2022. BERT learns to teach: Knowledge distillation with meta learning. In: ACL 2022.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S., 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: ICCV 2005.