Knowledge Transfer by Discriminative Pre-training for Academic Performance Prediction

Byungsoo Kim, Hangyeol Yu, Dongmin Shin, Youngduck Choi
Riiid! AI Research
{byungsoo.kim,hangyeol.yu,dm.shin,youngduck.choi}@riiid.co

ABSTRACT

The needs for precisely estimating a student's academic performance have been emphasized with an increasing amount of attention paid to Intelligent Tutoring System (ITS). However, since labels for academic performance, such as test scores, are collected from outside of ITS, obtaining the labels is costly, leading to label-scarcity problem which brings challenge in taking machine learning approaches for academic performance prediction. To this end, inspired by the recent advancement of pre-training method in natural language processing community, we propose DPA, a transfer learning framework with **D**iscriminative **P**re-training tasks for **A**cademic performance prediction. DPA pre-trains two models, a generator and a discriminator, and fine-tunes the discriminator on academic performance prediction. In DPA's pre-training phase, a sequence of interactions where some tokens are masked is provided to the generator which is trained to reconstruct the original sequence. Then, the discriminator takes an interaction sequence where the masked tokens are replaced by the generator's outputs, and is trained to predict the originalities of all tokens in the sequence. We conduct extensive experimental studies on a real-world dataset obtained from a multi-platform ITS application and show that DPA outperforms the previous state-of-the-art generative pre-training method with a reduction of 4.05% in mean absolute error and more robust to increased label-scarcity.[1]

Keywords

Academic Performance Prediction, Deep Learning, Transfer Learning, Discriminative Pre-training

1. INTRODUCTION

Predicting a student's future academic performance is a fundamental task for developing modern Intelligent Tutoring System (ITS) which aims to provide personalized learning

---

[1]For more detailed descriptions of experimental settings and results, please refer the arXiv version of this paper.
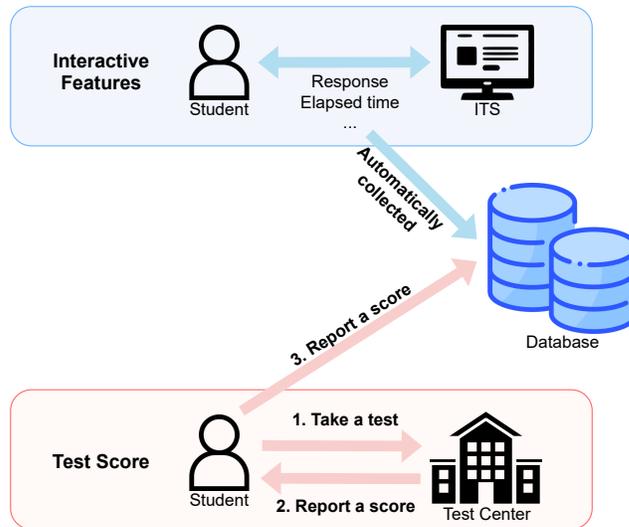
Figure 1: Interactive features, such as student response and elapsed time for the response, are automatically recorded to the database whenever a student interacts with ITS. On the other hand, more complicated steps are necessary to obtain a test score: a student should take the test in the designated test center, receive the test score, and report the score to ITS.

experience by supporting educational needs of each individual. However, labels for academic performance, such as test scores, are often scarce since they are external to ITS. For example, as shown in Figure 1, test scores are not automatically collected inside of ITS. Obtaining a test score requires a student to take the test in the designated test center, receive the score, and report the score to ITS. Transfer learning is a commonly taken approach to address such label-scarcity problems across different domains of machine learning. In this framework, a model is first pre-trained to optimize auxiliary objectives with abundant data, and then fine-tuned on the task of interest. In Artificial Intelligence in Education (AIEd) community, [3] introduced Assessment Modeling (AM), a set of pre-training tasks for label-scarce educational problems including academic performance prediction. AM proposed a pre-training method where first, a masked interaction sequence is generated by replacing a set of interactive features which can serve as criteria for pedagogical evaluation with artificial mask tokens. Then, given the

masked interaction sequence, a model is pre-trained to predict the masked interactive features. The idea was borrowed from the Masked Language Modeling (MLM) pre-training method proposed in [7]. In the MLM pre-training method, given a masked word sequence where some words in the sequence are replaced with an artificial mask token, a model is pre-trained to predict the masked words. However, recently, [6] pointed out that the MLM pre-training method has poor sample efficiency and suffers from pre-train/fine-tune discrepancy due to the artificial mask token, and proposed a new discriminative pre-training method. Considering the problems are also inherent in AM, potential gains are expected to be obtainable when the discriminative pre-training method is applied to academic performance prediction.

To this end, we propose DPA, a transfer learning framework with **D**iscriminative **P**re-training tasks for **A**cademic performance prediction. There are two models in DPA: a generator and a discriminator. In DPA's pre-training phase, the generator is trained to predict the masked interactive features in the same way as AM. Then, given a replaced interaction sequence which is generated by replacing the masked features with the generator's outputs, the discriminator is trained to predict whether each token in the sequence is the same as the one in the original interaction sequence. After the pre-training, the generator is thrown away and only the discriminator is fine-tuned on academic performance prediction. Also, we investigate diverse pre-training tasks for the generator and show that pre-training the generator to predict a student's response is more effective than to predict the correctness and timeliness of their response which were considered as the most pedagogical interactive features in AM. Extensive experimental studies conducted on a real-world dataset collected from a multi-platform ITS application show that DPA outperforms AM with a reduction of 4.05% in Mean Absolute Error (MAE) and more robust when the degree of label-scarcity increases.

2. SANTA: A SELF-STUDY SOLUTION EQUIPPED WITH AN AI TUTOR FOR ENGLISH EDUCATION

In this paper, we conduct experiments on a real-world dataset obtained from $Santa^2$, a multi-platform ITS with more than a million users in South Korea available through Android, iOS, and Web that exclusively focuses on the Test of English for International Communication (TOEIC) standardized examination. The publicly accessible version of the dataset was released under the name *EdNet* [4]. The TOEIC consists of two timed sections, Listening Comprehension (LC) and Reading Comprehension (RC). There are a total of 100 multiple choice exercises in each section, and the total score for each section is 495 in steps of 5 points. *Santa* provides learning experiences of solving exercises, studying explanations, and watching lectures. When a student consumes a specific learning content, *Santa* diagnoses their current academic status based on their learning activities records and recommends another learning content appropriate for their current position. *Santa* records diverse types of interactive features, such as student response, the duration of time the student took to respond, and the time interval between the current and previous learning activities. However, unlike

the interactive features automatically collected from *Santa*, obtaining the official TOEIC score requires more steps: a student should register and pay for the test, take the test in the designated test center, receive the test score from the Educational Testing Service, and report the score to *Santa* (Figure 1). *Santa* collected students' TOEIC score data by offering small gifts to students when they report their scores.

3. TRANSFER LEARNING FOR ACADEMIC TEST PERFORMANCE PREDICTION

To overcome the label-scarcity problem in academic test performance prediction, we consider burgeoning machine learning discipline of transfer learning. There is an open issue of what information to transfer or which pre-training task is the most effective for academic test performance prediction. Previous studies proposed two types of pre-training methods for AIEd Tasks: interaction-based method which models students' dynamic learning behaviors [13, 15, 8, 3], and content-based method which learns representations of learning contents [14, 23, 19, 24, 29]. [3] showed that interaction-based pre-training method outperforms content-based pre-training methods when the pre-trained model is fine-tuned on several label-scarce educational tasks including academic test performance prediction. Following this line of research, we propose a transfer learning framework where a model is pre-trained using only student interaction data, and fine-tune the pre-trained model on academic test performance prediction. In this paper, we consider the following interactive features:

- *eid*: A unique ID assigned to an exercise solved by a student. There are a total of 14419 exercises in the dataset.

- *part*: Each exercise belongs to a specific part that represents the type of the exercise. There are a total of 7 parts in the TOEIC.

- *response*: Since the TOEIC consists of multiple choice exercises and there are four options for each exercise, a student response for a given exercise is one of the options, 'a', 'b', 'c', or 'd'.

- *correctness*: Whether a student responded correctly to a given exercise. Note that *correctness* is a coarse version of *response* since *correctness* is processed by comparing *response* with a correct answer for a given exercise.

- *elapsed_time*: The amount of time a student spent on solving a given exercise.

- *timeliness*: Whether a student responded to a given exercise under the time limit. Note that *timeliness* is a coarse version of *elapsed_time* since *timeliness* is processed by comparing *elapsed_time* with the time limit recommended by domain experts for a given exercise.

- *exp_time*: The amount of time a student spent on studying an explanation for an exercise they had solved.

- *inactive_time*: The time interval between the current and previous interactions.
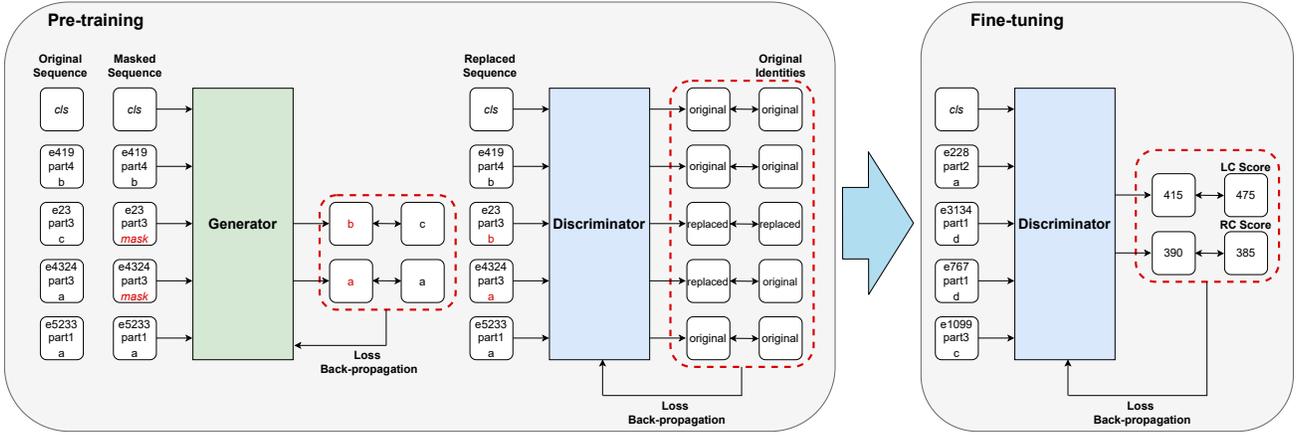
Figure 2: The overall pre-training/fine-tuning process of DPA when each token in an interaction sequence is a set of *eid*, *part*, and *response*, and *response* is a feature being masked. *mask* and *cls* are special tokens for mask and classification, respectively, which are the same as the ones used in [7].

In our experiments, we normalize the values of *elapsed_time*, *exp_time*, and *inactive_time* so they are between 0 and 1 to stabilize the training process.

4. PROPOSED METHOD

Figure 2 depicts our proposed method. There are two models in DPA: a generator and a discriminator. In pre-training phase, given a sequence of interactions $I = [I_1, \ldots, I_T]$, where each interaction $I_t = \{f_t^1, \ldots, f_t^k\}$ is a set of interactive features f_t^i, such as *eid*, *part*, and *response*, a masked interaction sequence $I^M = [I_1^M, \ldots, I_T^M]$ is generated by first randomly selecting a set of positions to mask $M = \{M_1, \ldots, M_m\}$ $(m < T)$, and for the masked position M_i, masking out a fixed set of features $\{f_{M_i}^1, \ldots, f_{M_i}^n\}$ $(n < k)$. For instance, in Figure 2, if the original interaction sequence is [(e419, part4, b), (e23, part3, c), (e4324, part3, a), (e5233, part1, a)] where each token in the sequence is a set of *eid*, *part*, and *response*, a masked interaction sequence where $M = \{2, 3\}$ and *response* as a masked feature is [(e419, part4, b), (e23, part3, *mask*), (e4324, part3, *mask*), (e5233, part1, a)]. Then, the generator takes the masked interaction sequence I^M as an input, and outputs predicted values O_{ij}^G for the masked features $f_{M_i}^j$. After that, a replaced interaction sequence $I^R = [I_1^R, \ldots, I_T^R]$ is generated by replacing the masked features $f_{M_i}^j$ with the generator's predictions O_{ij}^G. In Figure 2, since the generator's outputs for the masked features are 'b' and 'a', a replaced interaction sequence is [(e419, part4, b), (e23, part3, b), (e4324, part3, a), (e5233, part1, a)]. Then, the discriminator takes the replaced interaction sequence I^R as an input, and predicts whether each token in the sequence is the same as the one in the original interaction sequence (original) or not (replaced). After the pre-training, we throw away the generator and fine-tune the pre-trained discriminator on academic test performance prediction. We provide detailed explanations of each component in the generator and the discriminator, and training objective functions in the following subsections.

4.1 Interaction Embeddings

The embedding layer produces a sequence of interaction embedding vectors by mapping each interactive feature to an appropriate embedding vector. We take two different approaches to embed the interactive features depending on whether they are categorical (*eid*, *part*, *response*, *correctness*, and *timeliness*) or continuous (*elapsed_time*, *exp_time*, and *inactive_time*) variables. If an interactive feature is a categorical variable, we assign unique latent vectors to possible values of the feature including special values for mask (*mask*) and classification (*cls*). Take *response* as an example, there is an embedding matrix $E_{response} \in \mathbb{R}^{6 \times d_{emb}}$ where each row vector is assigned to one of 'a', 'b', 'c', 'd', *mask*, and *cls*. If an interactive feature is a continuous variable, we assign a single latent vector to the feature. Then, an embedding vector for the feature is computed by multiplying the latent vector and a value of the feature. For instance, we compute an embedding vector for *elapsed_time* as $et * E_{elapsed\_time}$, where et is a specific value and $E_{elapsed\_time} \in \mathbb{R}^{d_{emb}}$ is a latent vector assigned to *elapsed_time*. Also, mask and classification for the continuous interactive features are indicated by setting their values to -1 and 0, respectively. Not only embeddings for interactive features, positional embeddings are also incorporated into Transformer-based models [27] to consider chronological order of each token. Rather than using conventional positional embeddings which stores an embedding vector for every possible position, we adopt axial positional embeddings [17] to further reduce memory usage. The final interaction embedding vector of dimension d_{emb} for each time-step is the sum of all embedding vectors in the time-step. The interaction embedding layer is shared by both the generator and the discriminator.

4.2 Performer Encoder

Since its successful debut in Natural Language Processing (NLP) community, Transformer's attention mechanism has become a common recipe adopted across different domains of machine learning including speech processing [18], computer vision [1, 9], and AIEd [21, 2, 11, 22]. Compared to Recurrent Neural Network (RNN) family models, Transformer's attention mechanism has benefits of capturing longer-range
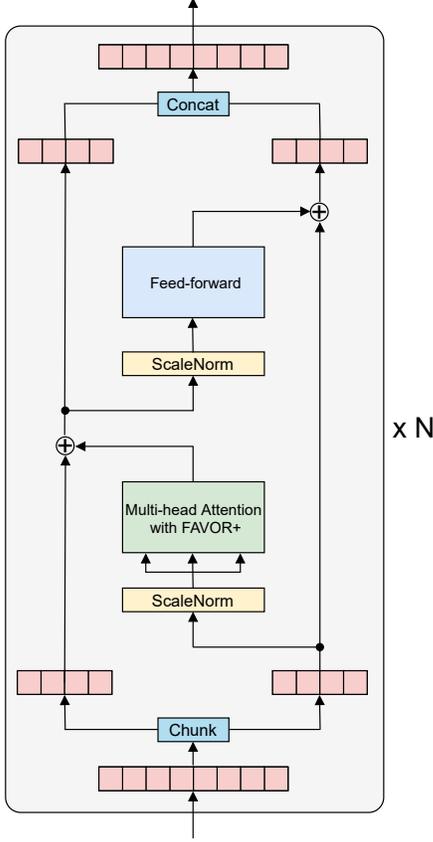
Figure 3: The reversible layer in the Performer encoder is composed of the FAVOR+-based multi-head attention layer and the point-wise feed-forward layer.

dependencies and allowing parallel training, which enables the model to achieve better performance with less training time. However, despite these advantages, the time and memory complexities of computing the attention grow quadratically with respect to input sequence length, requiring demanding computing resources for training the model on long sequences. For instance, if L is input sequence length and d is dimension of query, key, and value vectors, Transformer's attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)V,$$

where $Q, K, V \in \mathbb{R}^{L \times d}$. The time and memory complexities for computing $QK^\top$ in the above equation are $O(L^2 d)$ and $O(L^2)$, respectively. Therefore, the cost for training Transformer becomes prohibitive with large L, preventing training the model even on a single GPU.

The problem of improving the efficiency of Transformer's attention mechanism is a common concern of machine learning community. Recent studies have proposed several methods to reduce the computing complexities lower than the quadratic degree with respect to input sequence length [17, 28, 16, 25, 5]. In this paper, we adopt Performer [5] since it uses reasonable memory and makes a better trade-off be-

tween speed and performance [26]. Performer approximates attention kernels through Fast Attention Via positive Orthogonal Random features (FAVOR+) approach. For those who want to know more about FAVOR+, please refer [5].

With the efficient attention mechanism by FAVOR+, we propose the Performer encoder which is stacks of several identical reversible layers described in Figure 3. The reversible layer is based on Reversible Transformer [12, 17] architecture to further improve memory efficiency in backpropagation. An input of the reversible layer $x \in \mathbb{R}^{L \times d_{hidden}}$ is first chunked to $x_1, x_2 \in \mathbb{R}^{L \times d_{hidden}/2}$. Then, scaled l_2 normalization (ScaleNorm) [20] and FAVOR+-based multi-head attention layer (MultiHeadAttn) are applied to x_2, and the result is added to x_1 to compute $y_1 \in \mathbb{R}^{L \times d_{hidden}/2}$.

$$y_1 = x_1 + \text{MultiHeadAttn}(\text{ScaleNorm}(x_2)).$$

After that, the scaled l_2 normalization and point-wise feed-forward layer (FeedForward) are applied to y_1, and the result is added to x_2, computing $y_2 \in \mathbb{R}^{L \times d_{hidden}/2}$.

$$y_2 = x_2 + \text{FeedForward}(\text{ScaleNorm}(y_1)).$$

An output of the reversible layer $y \in \mathbb{R}^{L \times d_{hidden}}$ is a concatenation of y_1 and y_2. We stack the reversible layer multiple times to allow the final model to sufficiently represent underlying data distribution.

4.3 Generator

The generator computes hidden representations $[h_1^G, \ldots, h_T^G]$ by feeding the masked interaction sequence I^M to a series of the interaction embedding layer (InterEmbedding), a point-wise feed-forward layer (GenFeedForward1), the Performer encoder (GenPerformerEncoder), and another point-wise feed-forward layer (GenFeedForward2):

$$[I_1^{ME}, \ldots, I_T^{ME}] = \text{InterEmbedding}([I_1^M, \ldots, I_T^M])$$
$$[h_1^{GF}, \ldots, h_T^{GF}] = \text{GenFeedForward1}([I_1^{ME}, \ldots, I_T^{ME}])$$
$$[h_1^{GP}, \ldots, h_T^{GP}] = \text{GenPerformerEncoder}([h_1^{GF}, \ldots, h_T^{GF}])$$
$$[h_1^G, \ldots, h_T^G] = \text{GenFeedForward2}([h_1^{GP}, \ldots, h_T^{GP}]),$$

where $I_t^{ME}, h_t^G \in \mathbb{R}^{d_{emb}}$ and $h_t^{GF}, h_t^{GP} \in \mathbb{R}^{d_{gen\_hidden}}$. Then, depending on whether the masked features are categorical or continuous variables, generator outputs are computed differently. If the masked features are categorical variables, the outputs are sampled from a probability distribution defined by the following softmax layer:

$$O_{ij}^G \sim P_G(f_{M_i}^j | I^M) = \text{softmax}(E_j h_{M_i}^G).$$

If the masked features are continuous variables, the outputs are computed by the following sigmoid layer:

$$O_{ij}^G = \text{sigmoid}(E_j^\top h_{M_i}^G).$$

Similar to the case of categorical masked features, one can sample the outputs from a probability distribution defined by I^M and parameters of the generator when the masked features are continuous variables. For instance, the outputs can be sampled from the Gaussian distribution where the mean and the variance are determined by I^M and the generator's parameters. However, we make the outputs deterministic because sampling the outputs underperforms in our preliminary experiments when the masked features are continuous variables.

4.4 Discriminator

In pre-training, outputs of the discriminator $O^D = [O_1^D, \ldots,$ $O_T^D]$ is computed by applying a series of the interaction embedding layer (InterEmbedding), a point-wise feed-forward layer (DisFeedForward1), the Performer encoder (DisPerformerEncoder), and another point-wise feed-forward layer (DisFeedForward2) to the replaced interaction sequence I^R:

$$[I_1^{RE}, \ldots, I_T^{RE}] = \text{InterEmbedding}([I_1^R, \ldots, I_T^R])$$
$$[h_1^{DF}, \ldots, h_T^{DF}] = \text{DisFeedForward1}([I_1^{RE}, \ldots, I_T^{RE}])$$
$$[h_1^{DP}, \ldots, h_T^{DP}] = \text{DisPerformerEncoder}([h_1^{DF}, \ldots, h_T^{DF}])$$
$$[O_1^D, \ldots, O_T^D] = \text{DisFeedForward2}([h_1^{DP}, \ldots, h_T^{DP}]),$$

where $I_t^{RE} \in \mathbb{R}^{d_{emb}}$, $h_t^{DF}, h_t^{DP} \in \mathbb{R}^{d_{dis\_hidden}}$, $O_t^D \in \mathbb{R}$, and the sigmoid is applied to the last layer of the discriminator. After the pre-training, we slightly modify the discriminator by replacing the last layer with a layer having appropriate dimension for academic test performance prediction.

4.5 Training Objectives

The objective for pre-training is to minimize the following loss function:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \text{GenLoss}(O_{ij}^G, f_{M_i}^j) + \lambda \sum_{t=1}^{T} \text{DisLoss}(O_t^D, \mathbb{1}(I_t^R = I_t)),$$

where GenLoss is the cross entropy (or mean squared error) loss function if the masked features are categorical (or continuous) variables, DisLoss is the binary cross entropy loss function, and $\mathbb{1}$ is the identity function. For ease of notation, we omit an index for each input sample in the above equation. If there are more than one masked features in each time-step ($n > 1$), the generator is trained under the multi-task leaning scheme. The objective for fine-tuning is to minimize the mean squared error loss between the model's predictions and score labels.

5. EXPERIMENTS

5.1 Effects of Generator's Pre-training Tasks

There are multiple interactive features to be masked in each token of the interaction sequence, which raises a question of how to construct a set of masked interactive features, and accordingly, which pre-training task for the generator is the most effective for academic test performance prediction. By default, all interactive features listed in Section 3 are taken as inputs for both the generator and discriminator. However, if response (or elapsed_time) is masked, correctness (or timeliness) is excluded from the inputs and vice versa since there is an overlap of information that the features represent. For example, when both response and correctness are taken as inputs, and correctness is masked, the generator can predict the masked correctness by only looking at eid and response without considering other interactions, which leads to poor pre-training. The results are described in Table 1.

The best result was obtained under the pre-training task of predicting response alone, which is slightly better than that of predicting correctness, and both response and correctness. Predicting correctness of student response is an important task in AIEd as can be seen from the large volume of studies about Knowledge Tracing. Also, [3] empirically showed

Table 1: Comparison between different pre-training tasks.

Pre-training task	MAE
response	**50.65** $\pm$ 1.26
response + elapsed_time	54.86 $\pm$ 1.64
response + timeliness	52.91 $\pm$ 1.38
response + exp_time	57.54 $\pm$ 1.47
response + inactive_time	60.69 $\pm$ 1.74
correctness	51.36 $\pm$ 0.97
correctness + elapsed_time	53.36 $\pm$ 1.43
correctness + timeliness	52.60 $\pm$ 1.20
correctness + exp_time	54.36 $\pm$ 1.62
correctness + inactive_time	55.04 $\pm$ 1.58
response + correctness	51.13 $\pm$ 1.60
response + correctness + elapsed_time	52.15 $\pm$ 1.43
response + correctness + timeliness	53.05 $\pm$ 1.81
response + correctness+ exp_time	53.09 $\pm$ 1.25
response + correctness + inactive_time	56.41 $\pm$ 1.72

that student response correctness is the most pedagogical interactive feature for academic test performance prediction. However, rather than pre-training a model to predict whether a student correctly responded to a given exercise, the pre-training task of predicting student response itself injects more fine-grained information into the model, which leads to the more effective pre-training for academic test performance prediction. Interestingly, the underperformed results were obtained when predicting elapsed_time or timeliness in pre-training despite the benefits their information bring to several AIEd tasks [10, 30, 22]. We hypothesize that elapsed_time and timeliness may introduce irrelevant noises and thus guide the model towards a direction inappropriate for academic test performance prediction. In the case of exp_time and inactive_time, we observed that the generator failed to learn to predict their values when only given the interactive features listed in Section 3, which leads to unstable pre-training. From these observations, in the following subsections, we conduct experimental studies based on the pre-training task of predicting response alone.

5.2 DPA vs. Baseline Methods

We compare DPA with the following pre-training methods:

- No pre-training: We train the fine-tuning models only on the fine-tuning dataset.

- Autoencoding: Autoencoding (AE) is a generative pre-training method widely used across different domains of machine learning including AIEd [13, 8]. Given an unmasked interaction sequence, AE pre-trains a model to reconstruct the input interaction sequence.

- Assessment Modeling: Assessment Modeling (AM) [3] is the previous state-of-the-art generative pre-training method for academic test performance prediction. In AM, a model takes a masked interaction sequence as an input and is pre-trained to predict masked features. AM is exactly the same as fine-tuning the pre-trained generator in DPA.

Also, we investigate whether DPA is effective with the following different fine-tuning models:

Table 2: Comparison of DPA with baseline methods.

Pre-training method	Fine-tuning model	MAE
No pre-training	MLP	82.89 ± 3.23
	BiLSTM	84.05 ± 2.06
	Transformer encoder	107.06 ± 2.52
	Performer encoder	81.76 ± 1.24
AE	MLP	79.46 ± 1.15
	BiLSTM	85.64 ± 1.89
	Transformer encoder	75.13 ± 3.10
	Performer encoder	64.80 ± 1.43
AM	MLP	77.17 ± 2.14
	BiLSTM	58.16 ± 1.28
	Transformer encoder	57.16 ± 2.08
	Performer encoder	52.79 ± 1.39
DPA	MLP	77.24 ± 1.59
	BiLSTM	57.59 ± 1.76
	Transformer encoder	55.99 ± 1.62
	Performer encoder	$\mathbf{50.65} \pm 1.26$

- MLP: Multi-Layer Perceptron (MLP) is stacks of simple fully-connected layers. Given an interaction sequence, interaction embedding vectors of all time-steps are summed together to compute a fixed-dimensional vector which is fed to a series of the fully-connected layers.

- BiLSTM: Bi-directional Long Short-Term Memory (BiLSTM) is a model widely used for time series data prediction tasks. The global max pooling layer is applied on top of the BiLSTM layer to obtain a fixed-dimensional intermediate representation from an input sequence of varying length.

- Transformer Encoder: Transformer Encoder is a series of several identical layers composed of a multi-head self-attention layer with the softmax attention kernel and a point-wise feed-forward layer. We set the Transformer encoder's attention window size to 512 due to the out of GPU memory occuring when training the Transformer encoder of 1024 attention window size on our single GPU machine.

As described in Table 2, transferring the pre-trained knowledge brings better results in most cases, and the best result is obtained from DPA. Especially, when the Performer encoder, the best performing fine-tuning model, is used as the fine-tuning model, DPA reduces MAE by 4.05%, 21.84%, and 38.05% compared to AM, AE, and No pre-training, respectively. Among the baseline pre-training methods excluding No pre-training, the worst result is obtained from AE beacuse the pre-training task of AE is much easier than that of AM and DPA. We observed that the loss curve of AE converged to near zero within the first pre-training evaluation.

5.3 Robustness to Increased Label-scarcity
Since the motivation behind our proposal of DPA is the label-scarcity problem, we investigate how MAE changes at varying degrees of label-scarcity. Figure 4 and Table 3
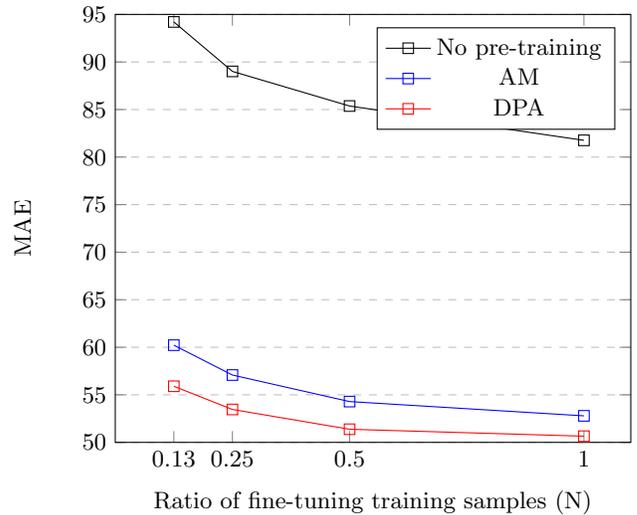

Figure 4: The black, blue, and red lines represent MAEs for No pre-training, AM, and DPA, respectively, when the number of fine-tuning training samples becomes 1/2, 1/4, and 1/8 of the entire dataset.

Table 3: Comparison of DPA with AM and No pre-training at varying degrees of label-scarcity.

N	No pre-training	AM	DPA
1/8	94.21 ± 8.40	60.22 ± 1.86	55.90 ± 1.97
1/4	89.01 ± 2.14	57.08 ± 1.75	53.46 ± 1.45
1/2	85.37 ± 1.15	54.29 ± 1.50	51.38 ± 1.16
Full	81.76 ± 1.24	52.79 ± 1.39	50.65 ± 1.26

describe the results when using 1/2, 1/4, and 1/8 of the total number of fine-tuning training samples. In all degrees of label-scarcity, DPA consistently outperforms AM. Also, DPA fine-tuned on 1/2, 1/4, and 1/8 of the dataset outperforms AM fine-tuned on the entire dataset, 1/2, and 1/4 of the dataset, respectively, which shows that DPA is more robust to label-scarcity than AM. Compared with No pre-training, the gap between No pre-training and the other two pre-training methods increases as the number of labels becomes scarce. Furthermore, the other two pre-training methods fine-tuned on 1/8 of the dataset outperform No pre-training fine-tuned on the entire dataset.

6. CONCLUSION
In this paper, we proposed DPA, a transfer learning framework with discriminative pre-training tasks for academic performance prediction. Our experimental results showed the effectiveness of DPA for the label-scarce academic performance prediction task over the previous state-of-the-art generative pre-training method. Avenues of future research include investigating more effective pre-training tasks for academic performance prediction and pre-train/fine-tune relations in AIEd.

7. REFERENCES
[1] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining

from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[2] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 341–344, 2020.

[3] Y. Choi, Y. Lee, J. Cho, J. Baek, D. Shin, S. Lee, Y. Cha, B. Kim, and J. Heo. Assessment modeling: Fundamental pre-training tasks for interactive educational systems. *arXiv preprint arXiv:2002.05505*, 2020.

[4] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo. Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*, pages 69–73. Springer, 2020.

[5] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[6] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] M. Ding, Y. Wang, E. Hemberg, and U.-M. O'Reilly. Transfer learning using representation learning in massive open online courses. In *Proceedings of the 9th international conference on learning analytics & knowledge*, pages 145–154, 2019.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009.

[11] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[12] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. *arXiv preprint arXiv:1707.04585*, 2017.

[13] B. Guo, R. Zhang, G. Xu, C. Shi, and L. Yang. Predicting students performance in educational data mining. In *2015 International Symposium on Educational Technology (ISET)*, pages 125–128. IEEE, 2015.

[14] Z. Huang, Q. Liu, E. Chen, H. Zhao, M. Gao, S. Wei, Y. Su, and G. Hu. Question difficulty prediction for reading problems in standard tests. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[15] X. J. Hunt, I. K. Kabul, and J. Silva. Transfer learning for education data. In *Proceedings of the ACM SIGKDD Conference, El Halifax, NS, Canada*, volume 17, 2017.

[16] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[17] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[18] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713, 2019.

[19] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019.

[20] T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[21] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[22] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi. Saint+: Integrating temporal features for ednet correctness prediction. *arXiv preprint arXiv:2010.12042*, 2020.

[23] Y. Su, Q. Liu, Q. Liu, Z. Huang, Y. Yin, E. Chen, C. Ding, S. Wei, and G. Hu. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[24] C. Sung, T. I. Dhamecha, and N. Mukhi. Improving short answer grading using transformer-based pre-training. In *International Conference on Artificial Intelligence in Education*, pages 469–481. Springer, 2019.

[25] Y. Tay, D. Bahri, L. Yang, D. Metzler, and D.-C. Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[26] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[28] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[29] Y. Yin, Q. Liu, Z. Huang, E. Chen, W. Tong, S. Wang, and Y. Su. Quesnet: A unified representation for heterogeneous test questions. In *Proceedings of the 25th ACM SIGKDD International*

Conference on Knowledge Discovery & Data Mining, pages 1328–1336, 2019.

[30] L. Zhang, X. Xiong, S. Zhao, A. Botelho, and N. T. Heffernan. Incorporating rich features into deep knowledge tracing. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*, pages 169–172, 2017.