DCAF-BERT: A Distilled Cachable Adaptable Factorized Model For Improved Ads CTR Prediction

Aashiq Muhamed
muhaaash@amazon.com
Amazon
Palo Alto, California, USA

Jaspreet Singh
jazsingh@amazon.com
Amazon
Palo Alto, California, USA

Shuai Zheng
shzheng@amazon.com
Amazon Web Services
Santa Clara, California, USA

Iman Keivanloo
imankan@amazon.com
Amazon
Seattle, Washington, USA

Sujan Perera
peresuja@amazon.com
Amazon
Palo Alto, California, USA

James Mracek
jmracek@amazon.com
Amazon
Palo Alto, California, USA

Shuai Zheng
shzheng@amazon.com
Amazon Web Services
Santa Clara, California, USA

Yi Xu
yxaamzn@amazon.com
Amazon
Seattle, Washington, USA

Qingjun Cui
qjgjunc@amazon.com
Amazon
Palo Alto, California, USA

Santosh Rajagopalan
syrgn@amazon.com
Amazon
Palo Alto, California, USA

Belinda Zeng
zengb@amazon.com
Amazon
Seattle, Washington, USA

Trishul Chilimbi
trishulc@amazon.com
Amazon
Seattle, Washington, USA

ABSTRACT

In this paper we present a Click-through-rate (CTR) prediction model for product advertisement at Amazon. CTR prediction is challenging because the model needs to a) learn from text and numeric features, b) maintain low-latency at inference time, and c) adapt to a temporal advertisement distribution shift. Our proposed model is DCAF-BERT, a novel lightweight cache-friendly factorized model that consists of twin-structured BERT-like encoders for text with a mechanism for late fusion for tabular and numeric features. The factorization of the model allows for compartmentalised retraining which enables the model to easily adapt to distribution shifts. The twin encoders are carefully trained to leverage historical CTR data, using a large pre-trained language model and cross-architecture knowledge distillation (KD). We empirically find the right combination of pretraining, distillation and fine-tuning strategies for teacher and student which leads to a 1.7% ROC-AUC lift over the previous best model offline. In an online experiment we show that our compartmentalised refresh strategy boosts the CTR of DCAF-BERT by 3.6% on average over the baseline model consistently across a month.

1 INTRODUCTION

At Amazon, ads for sponsored products are served on the landing page of a given product (Fig. 1). The selection of sponsored products depends on the current product being viewed and the order is determined by an ML model operating on a host of features including textual features like product title and description, contextual features like advertiser name, and historical features like total sales value. CTR prediction is challenging due to (i) deployment constraints – the model must serve millions of requests per second at high throughput and low (<5 ms) average latency, and (ii) user preferences that can experience temporal shifts due to special events, new campaigns, seasonality and other factors (e.g., a pandemic). To adapt to this changing distribution, the model must be refreshed often by retraining at a daily or hourly cadence.

This implies CTR models must be lightweight (fast inference), be economical to train/re-train (reduce cost), and improve performance (better customer experience). In this paper we outline how we address these 3 core tenants by leveraging large language models (LLM), knowledge distillation, and compartmentalized training using DCAF-BERT – a novel lightweight factorized model that consists of twin-structured BERT-like encoders with a mechanism for late fusion for tabular and numeric features. Factorizing the text and tabular features allows the BERT-based text features to be pre-computed offline and cached in memory. The cost at inference-time is therefore from the late fusion layer only. This fusion layer...
A pair of Page and Ad products (refer Fig. 1).

Problem Definition

2 APPROACH

We propose a knowledge distillation approach to train DCAF-BERT. This is motivated by the fact that we have large quantities of past click data that can be leveraged by large language models. We then distill it to smaller dual encoders using cross-architecture distillation which has shown to be more effective than training from scratch [9].

Teacher Model Training: Our teacher model is a single large pre-trained BERT tower with an additional layer norm before the MLP layers. During finetuning, the features (textual, contextual and historical) are transformed into their string representation and concatenated along with their feature names. The tokens from Page and Advertised products are separated with the [SEP] token. An example input would look as follows: “[CLS] Page_product_title: title Page_product_num_feat1: num_feat, Page_product_cat_feat1: cat_feat [SEP] Ad_product_title: title Ad_product_num_feat1: num_feat, Ad_product_cat_feat1: cat_feat”. Training with MLM across fields, enables learning strong cross-feature representations [7] through the attention mechanism.

Student Model Training: The DCAF-BERT student model architecture is designed for the online inference scenario. It is a cache-friendly model with two separate arms for the Page Product and Advertised Product respectively. The textual embeddings corresponding to the [CLS] token in each arm (highlighted in gray) can be computed and cached in advance for every product in the given e-commerce catalog. At inference time, these representations are retrieved from the cache and concatenated with the rest of the features via a late-fusion layer for the final CTR prediction. The cost at inference time is therefore from the late fusion layer only.

We train the student using cross-architecture distillation or distillation from the large cross-attention teacher model to the DCAF-BERT student model. More specifically, we seek a BERT-CTR student pθs that is close to the teacher pθt. Let the temperature softened outputs for the student/teacher network denoted by f be given by pt(τ) = softmax(zt/τ), where zt is the corresponding logits vector. To learn the optimal student θs, we minimize a combination of the cross entropy (CE) loss and Kullback-Leibler (KL) divergence loss with the teacher [5].

\[ θ^∗ = \arg\min_θ E_{(x,y)}−p(1−α)L_{CE}(p_θ^s(1), y) + αL_{KL}(p_θ^t(τ), p_θ(τ)) \]

\[ L_{CE}(p_θ^s(1), y) = \sum_j y_j \log p_θ^s(1) \]

\[ L_{KL}(p_θ^s(τ), p_θ^t(τ)) = T^2 \sum_j p_θ^t(τ) \log \frac{p_θ^s(τ)}{p_θ^t(τ)} \]

Here the summation j is over the number of classes.

Table 1 provides a high level summary of the various input features. For our example in Fig. 1, ‘iPhone Charger [Apple ... White’ is a textual feature (i.e., the product title) denoting the Page Product, its rating value (11.3 K) is a numeric feature and ‘Cable Type’ is a categorical feature. The features can also be categorized based on their distribution shift time scales — textual features like product title change slowly, often remaining unchanged for periods longer than a month while historical features like sale value change rapidly as people purchase products every hour.

2.1 DCAF-BERT

We train the student using cross-architecture distillation or distillation from the large cross-attention teacher model to the DCAF-BERT student model. More specifically, we seek a BERT-CTR student pθs that is close to the teacher pθt. Let the temperature softened outputs for the student/teacher network denoted by f be given by pt(τ) = softmax(zt/τ), where zt is the corresponding logits vector. To learn the optimal student θs, we minimize a combination of the cross entropy (CE) loss and Kullback-Leibler (KL) divergence loss with the teacher [5].

\[ θ^∗ = \arg\min_θ E_{(x,y)}−p(1−α)L_{CE}(p_θ^s(1), y) + αL_{KL}(p_θ^t(τ), p_θ(τ)) \]

\[ L_{CE}(p_θ^s(1), y) = \sum_j y_j \log p_θ^s(1) \]

\[ L_{KL}(p_θ^s(τ), p_θ^t(τ)) = T^2 \sum_j p_θ^t(τ) \log \frac{p_θ^s(τ)}{p_θ^t(τ)} \]

Here the summation j is over the number of classes.

Table 1: Summary of CTR features for product advertisement

<table>
<thead>
<tr>
<th>Feature group</th>
<th>Sample features</th>
<th>Feature count</th>
<th>Distribution shift time scales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual</td>
<td>Product title, description</td>
<td>4</td>
<td>Long (~1 month)</td>
</tr>
<tr>
<td>Contextual</td>
<td>Application name, advertiser name</td>
<td>3</td>
<td>Medium (~1 week)</td>
</tr>
<tr>
<td>Historical</td>
<td>Total sale value (per product)</td>
<td>13</td>
<td>Short (~1 hour)</td>
</tr>
</tbody>
</table>

is a lightweight multi-layer perceptron (MLP), that can meet the stringent inference latency and throughput requirements.

Refreshing or retraining a large BERT-based model at an hourly cadence is quite expensive and infeasible without expensive GPU hardware. We observed that textual features such as product title and description exhibit distribution shift over longer time scales than numeric features like total sale value in the past day. Taking advantage of the factorized nature of our model, we can refresh the late fusion layer (MLP) at an hourly cadence and refresh the twin-BERT backbone at a longer cadence (eg. monthly). To further reduce costs while keeping the benefits of LLMs, we leverage 2 key insights: (i) we have an abundance of stale historical CTR data that can be leveraged by large language models. We then distill it to smaller dual encoders using cross-architecture distillation which has shown to be more effective than training from scratch [9].

Teacher Model Training: Our teacher model is a single large pre-trained BERT tower with an additional layer norm before the MLP layers. During finetuning, the features (textual, contextual and historical) are transformed into their string representation and concatenated along with their feature names. The tokens from Page and Advertised products are separated with the [SEP] token. An example input would look as follows: “[CLS] Page_product_title: title Page_product_num_feat1: num_feat, Page_product_cat_feat1: cat_feat [SEP] Ad_product_title: title Ad_product_num_feat1: num_feat, Ad_product_cat_feat1: cat_feat”. Training with MLM across fields, enables learning strong cross-feature representations [7] through the attention mechanism.

Student Model Training: The DCAF-BERT student model architecture is designed for the online inference scenario. It is a cache-friendly model with two separate arms for the Page Product and Advertised Product respectively. The textual embeddings corresponding to the [CLS] token in each arm (highlighted in gray) can be computed and cached in advance for every product in the given e-commerce catalog. At inference time, these representations are retrieved from the cache and concatenated with the rest of the features via a late-fusion layer for the final CTR prediction. The cost at inference time is therefore from the late fusion layer only.

We train the student using cross-architecture distillation or distillation from the large cross-attention teacher model to the DCAF-BERT student model. More specifically, we seek a BERT-CTR student pθs that is close to the teacher pθt. Let the temperature softened outputs for the student/teacher network denoted by f be given by pt(τ) = softmax(zt/τ), where zt is the corresponding logits vector. To learn the optimal student θs, we minimize a combination of the cross entropy (CE) loss and Kullback-Leibler (KL) divergence loss with the teacher [5].

\[ θ^∗ = \arg\min_θ E_{(x,y)}−p(1−α)L_{CE}(p_θ^s(1), y) + αL_{KL}(p_θ^t(τ), p_θ(τ)) \]

\[ L_{CE}(p_θ^s(1), y) = \sum_j y_j \log p_θ^s(1) \]

\[ L_{KL}(p_θ^s(τ), p_θ^t(τ)) = T^2 \sum_j p_θ^t(τ) \log \frac{p_θ^s(τ)}{p_θ^t(τ)} \]

Here the summation j is over the number of classes.
Related Approaches There exists little work on using pre-trained LLMs akin to BERT for CTR prediction. While deep learning models have recently gained traction in CTR prediction [14], to meet latency requirements, the models are still relatively shallow by modern deep learning standards. The largest model in DCN-V2 [13], for example uses <5 million parameters. Other approaches like [8] utilize multi-head self attention layers to encode both text and other features but do not use pre-trained model representations. We point readers to [14] for an overview of existing approaches.

3 EXPERIMENTS
Motivated by the cost considerations in repeatedly finetuning large teacher models, we study the interaction between knowledge distillation and distribution shift in the setting where a large teacher model is trained once on out-of-distribution historical data and subsequently frozen. Given a frozen teacher trained on past data, we ask the question: What is the best training strategy to maximize performance on recent data for the student, where there is abundant past labeled data and limited recent labeled data?

3.1 Dataset
Past data We select train-test splits from 2020 online traffic. The data is temporally divided: after picking a reference point in 2020, the train set is uniformly sampled from data before this reference point in time and the test/val sets are sampled after the reference point. As the proportion of clicks is much lesser than non-clicks in the dataset, the two classes are balanced in the train set by down-sampling the dominant (non-clicks) class. We do not artificially balance the test set which remains skewed towards the non-clicks in a 95:5 ratio. The train set comprises of 1 billion data points and the test and validation sets comprise about 25 million points each.

Recent data The train-test splits are sampled from 2021 online traffic and balanced the same way as past data. The train set comprises 200 million data points and the test and validation sets comprise 25 million points each.

Preprocessing All text data is preprocessed using the Sentence-piece tokenizer [6]. A Byte Pair Encoding [12] subword vocabulary of 32000 tokens is constructed from the train corpus.

Metrics and evaluation All metrics are reported on the test set of Recent data. Since the task is binary classification, we report the ROC-AUC which is frequently used in imbalanced classification.

3.2 Models
Teacher: In all our experiments we use a 1.5 billion parameter teacher model pretrained using MLM on Amazon product data and finetuned on past data for 1 epoch. The teacher uses 48 layers, hidden size=1600, 25 attention heads and intermediate dim=6500. We use Adam optimizer with lr=2e-5, weight decay = 1e-2, and dropout = 0.1. [10] shows that scaling the teacher model size to 1.5 billion parameters, significantly improves performance (2.59% ROC AUC increase) over an MLP baseline.

Baseline: We use a 3-layer MLP baseline (65 million parameters) with ReLU activations carefully designed to meet the latency for the CTR task. The MLP resembles the DCAF-BERT fusion layer and uses a learnt word-embedding matrix to replace BERT embeddings. For offline experiments, we finetune the model for 2 epochs.

DCAF-BERT student: DCAF-BERT is a 70 million parameter student model, where the parameters between the two BERT towers are shared. DCAF-BERT uses 6 layers, hidden size=768, 16 attention heads and intermediate dim=3072. We use the Adam optimizer with weight decay = 1e-2, and dropout = 0.1. During distillation we use an lr=1e-4 and train for 3 epochs, while during finetuning we use an lr=1e-5 and train for 2 epochs. DCAF-BERT can be trained on 8 A100 GPUs in less than a day (<1000 USD).

3.3 Training Strategies
Using a past dataset from 2020 and a recent dataset from 2021, we examine the influence of various combinations of initialization, distillation and fine-tuning strategies on student performance. The student is first initialized with a selected initialization strategy, then pre-finetuned using a specified distillation strategy and finally finetuned using a fine-tuning strategy. In particular, we examine the following strategies applied in sequence:

(i) Initialization strategy: This can be one of (a) Random initialization, where the student is initialized with normally distributed weights similar to BERT initialization [2] before pretraining. (b) Masked Language Modeling (MLM) where we pretrain the student using MLM on Amazon product data, resembling the approach in [11]. (c) Supervised learning on labeled past data.

(ii) Distillation as a pre-finetuning strategy: (a) No distillation, (b) Past distillation where we train the student on soft labels from the past dataset teacher on past data, (c) Recent distillation where we distill using soft labels from the past dataset teacher on recent data, using the recent data soft-label validation loss for early stopping. We posit that recent data distillation can help the model see the recent data covariates during pre-finetuning, to further help downstream performance on recent data.

(iii) Finetuning strategy: (a) Vanilla fine-tuning on recent data, (b) Self-training where we use the finetuned recent data student to further label both the past and recent data, then use these labels for a second round of fine-tuning. Recent work in [3, 15] suggest that self-training provides gains orthogonal to supervised and self-supervised pretraining in the low transfer-data learning regime.

3.4 Results
We compare the performance of our model for the different combinations of initialization, distillation and fine-tuning strategies in Table 2. The Teacher model trained on past data without any finetuning on recent data achieves only 63.53% ROC-AUC. Both the MLP baseline and DCAF-BERT without MLM or distillation (Approach 1) achieve comparable ROC-AUC of 75.42%. Our best DCAF-BERT approach (MLM, Past KD, Self-training) achieves a 77.7% ROC-AUC.

a) Which initialization strategy gives best results? We compare 3 initialization strategies that correspond to a particular distillation strategy. When we examine approaches 1,2,3 for No KD, approaches 4,5,6 for Past KD and approaches 7,8,9 for Recent KD, we see that downstream performance of Random initialization < Supervised learning < MLM pretraining. This trend holds across all distillation and fine-tuning strategies. Both Supervised learning and MLM improve performance over random initialization. This performance gain is unsurprising as self-supervised pretraining and
Table 2: DCAF-BERT student performance for the various choices of initialization, distillation and fine-tuning strategies. We report the ROC-AUC on the recent data test set after finetuning.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Initialization</th>
<th>Distillation</th>
<th>Fine-tuning</th>
<th>Recent AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
<td>Random</td>
<td>No KD</td>
<td>No fine-tuning</td>
<td>63.53%</td>
</tr>
<tr>
<td>MLP baseline</td>
<td>Random</td>
<td>No KD</td>
<td>Vanilla</td>
<td>75.43%</td>
</tr>
<tr>
<td>1</td>
<td>Random</td>
<td>No KD</td>
<td>Vanilla</td>
<td>75.42%</td>
</tr>
<tr>
<td>2</td>
<td>MLM</td>
<td>No KD</td>
<td>Vanilla</td>
<td>75.93%</td>
</tr>
<tr>
<td>3</td>
<td>Supervised</td>
<td>No KD</td>
<td>Vanilla</td>
<td>75.48%</td>
</tr>
<tr>
<td>4</td>
<td>Random</td>
<td>Past KD</td>
<td>Vanilla</td>
<td>76.73%</td>
</tr>
<tr>
<td>5</td>
<td>MLM</td>
<td>Past KD</td>
<td>Vanilla</td>
<td>76.92%</td>
</tr>
<tr>
<td>6</td>
<td>Supervised</td>
<td>Past KD</td>
<td>Vanilla</td>
<td>76.84%</td>
</tr>
<tr>
<td>7</td>
<td>Random</td>
<td>Recent KD</td>
<td>Vanilla</td>
<td>75.60%</td>
</tr>
<tr>
<td>8</td>
<td>MLM</td>
<td>Recent KD</td>
<td>Vanilla</td>
<td>76.22%</td>
</tr>
<tr>
<td>9</td>
<td>Supervised</td>
<td>Recent KD</td>
<td>Vanilla</td>
<td>75.62%</td>
</tr>
<tr>
<td>10</td>
<td>Supervised</td>
<td>Past KD</td>
<td>Self-training</td>
<td>77.02%</td>
</tr>
<tr>
<td>11</td>
<td>MLM</td>
<td>Past KD</td>
<td>Self-training</td>
<td>77.07%</td>
</tr>
</tbody>
</table>

supervised training helps the model learn features that generalize to out-of-distribution data\[1\]. Our findings suggest in particular that the MLM objective is better than supervised learning objectives even when abundant labeled past data is available.

b) Which distillation pre-finetuning strategy gives best results? When we compare approaches 1,4,7; approaches 2,5,8 and approaches 3,6,9 we see across the different initialization strategies that performance of models pre-finetuned with No KD < Recent KD < Past KD. Both Past KD and Recent KD help learn from the Past data teacher, and boost model performance over No KD. This is consistent with findings in literature \[4\]. Why does Recent KD perform worse than Past KD? We speculate that in the low in-distribution Recent data regime, to effectively learn from the Past data teacher, the student needs a larger data distribution support.

c) Does self-training provide an additional boost? We find that self-training finetuning after vanilla finetuning gives an additional performance boost over approaches 5 and 6 on recent data. This shows that self-training is another effective method to learn from labeled past data and the performance gain is complementary to the gains from initialization and distillation.

d) Does knowledge distillation boost performance over supervised learning on the Past dataset? If we had to pick between initialization and distillation pre-finetuning, comparing approaches 2 (MLM), 3 (Supervised learning), 4 (Past KD) we find that the best strategy to learn from the Past dataset is to first train a large teacher (MLM, vanilla fine-tuning) and then distill this teacher to a small student, which performs much better than MLM initialization or supervised learning alone. Training one large teacher on past data can significantly boost performance on recent data.

3.5 Online evaluation

We performed an online experiment where DCAF-BERT was tested within the Amazon e-commerce detail page service. The embeddings for the Page product (previous 7 days) and Ad Product (previous 1 day) were prepared offline using a 70 million parameter DCAF-BERT backbone and indexed in a distributed database for serving. The embeddings in the index were recomputed at a regular cadence (every 3.5 days) to serve fresh ads using the most recently refreshed BERT backbone. The DCAF-BERT MLP layer and the MLP baseline were re-trained daily while the DCAF-BERT BERT backbone was re-trained monthly. The index regeneration period was determined based on a hit rate criterion and the model refresh period was determined based on a CTR-lift criterion.

Figure 2 shows the average CTR of DCAF-BERT relative to the MLP over a week along with the Page and Ad product embedding index hit rates also shown. The vertical dashed line denotes a point where both the BERT backbone was re-trained and the embeddings index was regenerated. We observe that the BERT refresh helps increase the CTR-lift of DCAF-BERT relative to the baseline and this lift persists for extended periods of time without additional backbone retraining. In between index regenerations, the page and ad hit rates decrease which causes the CTR-lift of DCAF-BERT relative to the baseline to decrease. When compared against the baseline, DCAF-BERT improves CTR by 3.6% on average. Furthermore, we observed an 8+% CTR lift on tail traffic which indicates that DCAF-BERT generalizes better than the baseline.

4 CONCLUSION

In this paper, we address some significant challenges \[14\] in the application of large language models to CTR prediction. We propose a novel, lightweight factorized DCAF-BERT model that meets online latency requirements while being inexpensive to adapt to CTR distribution shift. Through an extensive empirical study using labels from a stale pretrained LLM teacher, we show that student MLM pretraining, distillation pre-finetuning and self-training can help learn representations that maximize performance when subject to distribution shift.

5 ACKNOWLEDGEMENT

We would like to thank RJ George, Sandy Cheng, and Anthony Ko for their help with training infrastructure and data pipelines.
REFERENCES


