Cross-domain Attention Network with Wasserstein Regularizers for E-commerce Search

Minghui Qiu[1,2], Bo Wang[1], Cen Chen[1,*], Xiaoyi Zeng[1], Jun Huang[1], Deng Cai[2], Jingren Zhou[1], and Forrest Sheng Bao[3*]
[1] Alibaba Group, China [2] Zhejiang University, China [3] Iowa State University
{minghui.qmh,boyi.wb,chencen.cc,yuanhan,huangjun.hj,jingren.zhou}@alibaba-inc.com
dengcai@cad.zju.edu.cn,forrest.bao@gmail.com

ABSTRACT

Product search and recommendation is a task that every e-commerce platform wants to outperform their peels on. However, training a good search or recommendation model often requires more data than what many platforms have. Fortunately, the search tasks on different platforms share the common underlying structure. Considering each platform as a domain, we propose a cross-domain learning approach to help the task on data-deficient platforms by leveraging the data from data-abundant platforms. In our solution, the importance of features in different domains is addressed by a domain-specific attention network. Meanwhile, a multi-task regularizer based on Wasserstein distance is introduced to help extract both domain-invariant and domain-specific features. Our model consistently outperforms the competing methods on both public and real-world industry datasets. Quantitative evaluation shows that our model can discover important features for different domains, which helps us better understand different user needs across platforms. Last but not least, we have deployed our model online in three big e-commerce platforms namely Taobao, Tmall, and Qintao, and observed better performance than the production models for all the platforms.

CCS CONCEPTS

• **Information systems → Electronic commerce**; • **Computing methodologies → Transfer learning**;

KEYWORDS

E-commerce Search, Transfer Learning, Wasserstein Regularizers

ACM Reference format:
Minghui Qiu, Bo Wang, Cen Chen, Xiaoyi Zeng, Jun Huang, Deng Cai, Jingren Zhou, and Forrest Sheng Bao. 2019. Cross-domain Attention Network with Wasserstein Regularizers for E-commerce Search. In *Proceedings of The 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 2019 (CIKM '19),* 7 pages.
https://doi.org/10.1145/3357384.3357809

*Corresponding author.

CIKM '19, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357809

1 INTRODUCTION

E-commerce platforms such as Amazon have become an important part of our life. To them, a fundamental task is to recommend items to users or predict user click-through-rate on items. Due to the high volume of users and items on these platforms, the task requires sufficient, sometimes massive amounts of, data to train an effective model, making it impractical for platforms of small amounts of data. For example in our dataset, we have collected 52G instances from platform Taobao[1], but only 350M instances (around 0.7% of Taobao's) from the other platform Qintao[2]. Small platforms like Qintao are in a disadvantageous position in building effective recommendation models. Fortunately, the underlying nature of the tasks is the same across platforms, and data from large platforms can be used to level the playground. Considering each platform as a domain, we propose a cross-domain multi-task learning approach to leverage data from data-abundant domains to assist the task on data-deficient platforms. Our approach can be particularly useful to companies managing a portfolio of websites, e.g., Expedia.com also runs Hotels.com, Hotwire.com, Orbitz.com, etc.

Our approach is inspired by the success of transfer learning (TL) in many NLP and CV tasks [15, 16, 23]. A typical transfer learning approach uses a fully-shared network to learn shared features for both source and target domains [15, 23]. An improved framework is the *shared-private model*, i.e., a shared network and domain-private networks, to derive shared and domain-private features [14]. Unlike typical transfer learning settings focusing on the target domain only, our approach builds a unified model that is helpful for both the source and target domains, hoping to save the cost of deploying multiple models online.

Building such a unified model is not trivial because user behaviors on different domains/platforms vary drastically. For example, our experiments use collected data from three platforms, Taobao, Tmall [3] and Qintao. They cover similar product categories but offer general-purpose, high-end, and budget products, respectively. User behaviors are thus quite different on them. Users of platform Tmall usually value quality over price, while price is important to users of platform Qintao. This motivates us to capture domain-private feature importance, such that our model can be adapted when ported from one platform to another. Specifically, we build a cross-domain attention layer that assigns each type of feature with a domain-private weight to capture its importance in each domain.

Moreover, a vanilla transfer learning model may mix private features into shared features and Wasserstein distance regularizer is a

[1] www.taobao.com
[2] www.qintao.taobao.com
[3] www.tmall.com

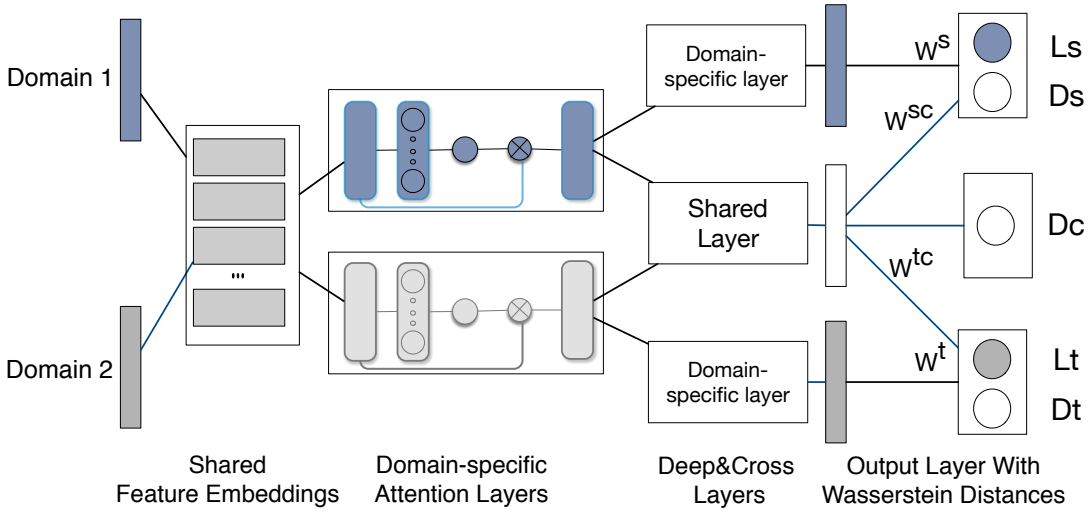

Figure 1: Our proposed Wasserstein regularized Cross-domain Attention Network (WE-CAN).

good way to help learn better shared features [18]. We argue that for improving model performance on multiple domains, it is also important to capture domain-private features. Inspired by this, we propose a new Wasserstein distance based multi-task regularizer to help extract both domain-shared and domain-private features. Our intuitions are: (i) shared features from both the source and target domains should be close to each other, and (ii) domain-private features should differ from the shared features. To achieve so, we propose a new Wasserstein regularizer with three terms: D_c for the difference between source-shared features and target-shared features, D_s for the difference between source-specific features and source-shared features, and D_t for the difference between target-specific features and target-shared features. They are proven to improve the performance of our models in multiple domains in experiments. Our full model is denoted as the Wasserstein regularized Cross-domain Attention Network (WE-CAN).

In summary, we have made these three contributions in this study:

- Unlike traditional transfer learning tasks that only focus on target domains, our cross-platform recommender system is helpful for multiple domains/platforms. It addresses domain-private feature importance and leverages Wasserstein regularizers to help learn domain-shared and domain-private features for multiple domains.
- We evaluate our method on both public and real-world industry datasets. Experiments show that our model outperforms the competing methods, and improves multiple domains. Quantitative evaluation shows our model can discover important features for different domains which help to understand different user needs in these domains.
- We have also deployed our method in the real-world e-commerce platforms and observed better performance than the baseline models.

2 MODEL

Our model is designed to address the following general problem. Given a user and an item, we would like to infer, as a binary classification problem, whether the user is interested in the item. Specifically, given an input instance $X_i^d = [x_i^0, x_i^1, \cdots, x_i^n]$ from domain $d \in \{s, t\}$, where s and t are the source and target domains, respectively, our model seeks to predict the label $y_i^d \in \{0, 1\}$. Each feature x_i^j is a feature related to either a user and/or an item, such as user gender, education, item categories, etc. Note that, unlike a typical transfer learning model that focuses on the performance on the target domain only, we prefer the model to function well on both the source and target domains.

As illustrated in Figure 1, our model first obtains hidden representations for each feature by looking up its feature embeddings. They are then fed into attention layers to obtain domain-specific feature representations. These features are passed to hidden layers (Deep & Cross layers in our case) to obtain high-level representations. The output layers are applied on the high-level representations to generate output labels. To encourage learning domain-shared/specific features, we add new Wasserstein regularizers to the model. Below we present our model in details.

2.1 Base Model

The base model mainly contains two major components: attention layer and Deep & Cross layers [21]. The former obtains a gated representations of input features and the latter obtains abstract features from the representations.

2.1.1 Attention Layer. Recall that different features play different importance in different domains. For example, features related to item quality are important in platform Tmall, while item price is more important in platform Qintao. To model this, we consider a domain-specific attention layer to weight feature importance. For an input $X_i^d = [x_i^1, x_i^2, \cdots, x_i^n]$, our model first looks up feature

embedding $e_i^j \in \mathbb{R}^{K\times 1}$ for each feature x_i^j where $j \in [1, n]$. For each feature x_i^j, its feature embedding e_i^j is fed into an attention layer to obtain a gated representation (weighted) $g_i^j \in \mathbb{R}^{K\times 1}$.

Formally, we define the attention layer as follows:

$$\begin{aligned} h_0 &= \sigma(W^0 e_i^j + b^0), \\ h_1 &= \sigma(W^1 h_0 + b^1), \\ a_i^j &= \sigma(W^2 h_1 + b^2), \\ g_i^j &= a_i^j e_i^j, \end{aligned} \tag{1}$$

where a_i^j is a scalar that weights the importance of the embedding of the j-th feature e_i^j. Note that, different domains have different sets of weights $W^{\{0..2\}}$, we omit domain label here for clarity. We then concatenate all weighted embeddings to form new feature embeddings $x_i^g = g_i^1 \oplus g_i^2 \oplus \cdots \oplus g_i^n$.

2.1.2 Deep & Cross Layer. The above gated representations need to be passed to another layer to extract abstract features. We employ Deep & Cross network [21] here because of its capability to efficiently learn feature interactions. Note that our framework is general for cross-domain tasks where the Deep & Cross layer can be replaced by any deep neural networks. Hence, finally, we obtain the output as $O_i = \text{Deep\&Cross}(x_i^g)$.

2.2 Joint Training

Our model considers a joint training setting with one source domain and one target domain. We leave the exploration of more domains to our future work. We aim to improve our model performance on both domains. Similar to [14], we use one shared network and two domain-specific networks to derive shared features and domain-specific features respectively. Specifically, the predictions for source and target domains are, respectively:

$$\begin{aligned} \hat{y}^s &= \sigma(W^{sc} O^{sc} + W^s O^s + b^s), \\ \hat{y}^t &= \sigma(W^{tc} O^{tc} + W^t O^t + b^t), \end{aligned} \tag{2}$$

where all the $O^{(\cdot)}$'s are obtained from the base model defined in §2.1 above: O^{sc} and O^{tc} are from the shared network, O^s and O^t are from domain-specific networks; $W^{(\cdot)}$ are the output weights and $b^{(\cdot)}$ is the biases.

During the training of the model, the prediction loss is:

$$L = -\sum_{d\in\{s,t\}} \frac{1}{n_d} \sum_{j=1}^{n_d} \frac{1}{2} \left[y_j^d \log \hat{y}_j^d + (1 - y_j^d) \log(1 - \hat{y}_j^d) \right], \tag{3}$$

where n_d is the number of instances in domain d.

2.3 Wasserstein Regularizer

Note that the above method does not force the shared features to be domain-invariant. Some domain-specific features may be mixed into the shared features. The study in [18] considers Wasserstein distance to learn better domain-shared features. However, we argue that to excel in each domain, domain-specific and domain-shared features are equally important to knowledge transfer. We then propose new Wasserstein regularizers to our specific-shared model to capture both domain-shared and domain-specific features. Our intuitions are: (i) shared features trained from both source domain and target domain should be close, and (ii) domain-specific features should differ from domain-shared features.

To capture the first intuition, we consider a Wasserstein distance based critic, D_c. Let $f_g^{(\cdot)}$ be the feature generators in our model which correspond to the gated embedding and Deep & Cross network in §2.1. Denote the shared, source-specific and target-specific feature generators as f_g^c, f_g^s, and f_g^t, respectively. For an input X^s from the source domain, our model learns a feature representations O^{sc} corresponding to source-shared features through the feature generator f_g^c. Similarly we have O^{tc} from a target input X^t. Here O^{sc} and O^{tc} are:

$$O^{sc} = f_g^c(X^s), \quad O^{tc} = f_g^c(X^t).$$

Let f_w be a domain regressor which maps the feature representation to a real number with parameters θ_w. As shown in [18], to maximize the Wasserstein distance between O^{sc} and O^{tc} is to maximize the following domain critic loss D_c.

$$\begin{aligned} D_c =& \frac{1}{n_s} \sum_{X^s} f_w(O^{sc}) - \frac{1}{n_t} \sum_{X^t} f_w(O^{tc}) \\ =& \frac{1}{n_s} \sum_{X^s} f_w\left(f_g^c(X^s)\right) - \frac{1}{n_t} \sum_{X^t} f_w\left(f_g^c(X^t)\right). \end{aligned} \tag{4}$$

To capture the second intuition regarding the difference between domain-specific features and shared features, we consider two more domain critics: D_s for source domain features and source-shared features, and D_t for target domain features and target-shared features. They are defined as follows:

$$\begin{aligned} D_s =& \frac{1}{n_s} \sum_{X^s} f_w(O^{sc}) - \frac{1}{n_s} \sum_{X^s} f_w(O^s), \\ D_t =& \frac{1}{n_t} \sum_{X^t} f_w(O^{tc}) - \frac{1}{n_t} \sum_{X^t} f_w(O^t). \end{aligned} \tag{5}$$

In all, let θ_g be parameters for the feature generator f_g, θ_w be the parameters for the domain regressor f_w, and θ_o be the weights and biases for the output layer defined in Eqn. (2), we seek to optimize these parameters by solving the following objective function.

$$J = \min_{\theta_g, \theta_o} \{L + \lambda \max_{\theta_w} (D_c - D_s - D_t) + L_{reg}\}, \tag{6}$$

where the term L is from Eqn. (3), L_{reg} is a regularization term, and D_c, D_s, and D_t are Wasserstein regularizers defined in Eqs. (4)-(5).

The domain regressor and feature generator play a minimax game to optimize D_c, D_s, and D_t. By fixing domain regressor parameters θ_w, minimizing $D_c - D_s - D_t$ can be achieved by minimizing D_c (difference between source-shared and target-shared features) while maximizing $D_s + D_t$ (difference between domain-shared and domain-specific features where task refers to the source or target). By fixing θ_g, the regularizer seeks to maximize $D_c - D_s - D_t$ which provides challenges for θ_g to optimize. Since Wasserstein distance is continuous and differentiable, the parameters can be optimized using stochastic gradient descent based methods. Specifically, we

derive the optimization formulas for θ_g, θ_o and θ_w as follows [4].

$$\begin{aligned}\theta_w^* &= \theta_w + \alpha_1 \nabla_{\theta_w}[D_c - D_s - D_t],\\ \theta_o^* &= \theta_o - \alpha_2 \nabla_{\theta_o}[L],\\ \theta_g^* &= \theta_g - \alpha_3 \nabla_{\theta_g}[L + D_c - D_s - D_t].\end{aligned} \tag{7}$$

This process is like Generative Adversarial Nets (GANs) [8] where the generator (θ_g in our case) attempts to generate high-quality features to fool the discriminator (θ_w in our case), and the discriminator improves its discriminativeness by picking out generated features. The interactions between generator and discriminator improve both of them simultaneously.

3 EXPERIMENTS

In this section, we conduct experiments to examine the following questions:
(Q1) Does our base model perform well without transfer learning?
(Q2) Does our method outperform other transfer learning methods?
(Q3) Does our method have similar improvements for items with different popularity?
(Q4) Are the domain-specific attention weights learned interpretable and insightful?
(Q5) Does our model work well in real-world systems?
(Q6) Is our model sensitive to parameters?

3.1 Experiment Settings

Datasets: The first dataset used is an open dataset, Criteo Display Ads[5]. We follow the settings from [21] to use the first 6 days for training and randomly split day 7 data into validation and test sets of equal size.

Due to the lack of public transfer learning datasets in e-commerce domain, we create the second dataset by crawling one-week user click-through data from three large e-commerce websites, namely Taobao (https://taobao.com), Tmall (https://tmall.com), and Qintao (https://qintao.taobao.com), referred to as A, B, and C respectively. Each data instance has 254 types of features including:

- Statistic features w.r.t. user and items, e.g., item historical sales, exposure information, etc.;
- User and item profiles, e.g., user age, user education information, item price, and item title;
- User behavioral features such as click, purchase.

The data statistics are presented in Table 1. The platforms A, B, and C have 52G, 4.9G, and 350M instances respectively [6]. Platform A is with much larger data instances than platform B and C, thus serving as the source domain.

Table 1: The E-commerce dataset statistics.

E-commerce Data	A (Taobao)	B (Tmall)	C (Qintao)
# instances	52,000M	4,900M	350M
# fields	254	254	254
# features	1,000M	1,000M	1,000M

[4]We omit the regularization term in the formula for clarity
[5] https://www.kaggle.com/c/criteo-display-ad-challenge
[6]Quantities are M (10^6) and G (10^9), not MB nor GB.

The sample features for our dataset are in Table 2. Five types of features: user profile, item profile, query statistics, item statistics, and user behaviors, are considered in this work.

Table 2: Sample features in our item recommendation dataset. PV: page view. IPV: item page view.

Category	Feature	Dimension	Type	#
User profile	user id	5×10^8	one-hot	1
	gender	2	one-hot	1
	age level	6	one-hot	1
	...	...	...	...
Item profile	item id	$\sim 2 \times 10^8$	one-hot	1
	price level	6	one-hot	1
	brand level	6	one-hot	1
	...	...	...	...
Query statistic	search PV 1 day	~ 100	one-hot	1
	search PV 7 days	~ 100	one-hot	1
	search PV 14 days	~ 100	one-hot	1
	...	...	...	...
Item statistic	IPV 1 day	~ 100	one-hot	1
	IPV 7 days	~ 100	one-hot	1
	IPV 14 days	~ 100	one-hot	1
	...	...	...	...
User behavior sequence	click items	$\sim 2 \times 10^8$	multi-hot	~ 50
	cart items	$\sim 2 \times 10^8$	multi-hot	~ 50
	pay items	$\sim 2 \times 10^8$	multi-hot	~ 50
	...	...	...	...

4 MODEL SETTINGS

For the item recommendation dataset, we treat user click or purchase behaviors as positive labels ($y = 1$), and others as negative labels ($y = 0$). Continuous features are discretized to categorical features.

For each type of discrete feature, the feature embedding is 64. For each domain, its attention layer is a dense layer with a structure of $64 \times 32 \times 1$. For the Deep & Cross network, the deep network is of size 512×256 while the cross network consists of two crossing layers. We use ReLU as the activation functions for the layers and Adam as the optimizer. The learning rate is set as 0.001. Note that for the Criteo dataset, all the Deep-only, and Deep & Cross models follow the settings in [21]. For evaluation, we consider Area Under Curve (AUC) and Logloss on the true label.

4.1 Comparison on Base Models (Q1)

We first compare our base model (§2.1) with the following classic baselines widely used in recommender system:

- **Deep-only**: a classic two-layer dense (or fully-connected) network with a structure of 512×256.
- **Wide & Deep**: an extended Deep-only model with a wide layer [4]. Here we feed all the original features to the wide layer.
- **DeepFM**: an extended Deep-only model with FM [9];
- **CAN-s**: a single domain version of our Cross-domain Attention Networks (CAN).

Table 3: Comparison on the base models of different neural architectures on two datasets. A larger AUC or a lower Logloss means better performance.

Method	E-commerce (AUC)			Criteo
	A	B	C	(Logloss)
Deep-only	0.7383	0.6833	0.6523	0.4428
Wide&Deep	0.7449	0.6935	0.6689	0.4424
DeepFM	0.7448	0.6998	0.6711	0.4421
CAN-s	**0.7456**	**0.7183**	**0.6734**	**0.4418**

The results from both Criteo dataset and e-commerce dataset are given in Table 3. Data from platform A alone is sufficient to train a good model as the AUC for A is always higher than that for B or C. On platforms A, B and C, Wide&Deep outperforms Deep-only, showing the necessity to consider wide features. DeepFM further pushes the performance to new highs on platforms B and C while keeping the performance on platform A, showing that DeepFM is more beneficial for domains with fewer data. Lastly, our base model outperforms all baselines on both Criteo and e-commerce datasets, demonstrating the effectiveness of our base model in learning feature interactions.

4.2 Cross-domain Performance Comparison (Q2)

We further evaluate our model performance on multiple domains using our collected e-commerce dataset. We compare our full model Wasserstein regularized cross-domain attention network, referred to as WE-CAN, with the following baselines:

- **Src-only**: a CAN-s model that uses only data from platform A to examine whether platform A is helpful for other domains.
- **Self-train**: a CAN-s model that uses its own data for training. This is identical to a supervised method that does not consider transfer learning.
- **FineTune**: a typical transfer learning method that trains on a source domain and finetunes the networks in the target domain [24].
- **DANN**: the domain-adversarial neural network (DANN) in [7], including a fully-shared model with both source and target data with a gradient reversal layer as adversarial training. For fair comparison, we also adopt Deep&Cross network in DANN.
- **TL-WD**: the model proposed in [18] that uses Wasserstein distance to learn domain-shared features. For fair comparison, we adopt Deep&Cross network in TL-WD.

The cross-domain results are given in Table 4. First, Src-only performs worse than Self-train which means simply using data from platform A for B or C is not effective for cross-domain tasks. This validates the domain shift between A and B/C, the challenge for using TL to solve our problem. Second, between Self-train and FineTune, FineTune degrades the performance on the source domain A (e.g., 0.7435 vs. 0.7456 for A & B), but improves the performance on the target domains B and C. This is because the model is fine-tuned by the target data at a later stage which may not be ideal for source domain. Third, despite the domain shift, DANN is able to leverage knowledge from the source domain to boost performance. Meanwhile, TL-WD outperforms DANN, showing that the Wasserstein regularizer can help better capture domain-shared features than DANN. Furthermore, it is also worth noting that, our WE-CAN has better performance than TL-WD because of our proposed Wasserstein regularizers that consider both domain-shared and domain-specific features. Last, our method consistently outperforms all the other baselines demonstrates the advantage of our proposed method.

Table 4: Comparison with different cross-domain models. Domain A is jointly trained with domain B and with domain C, respectively, as domain A has the largest data instances.

AUC|	Joint A & B		Joint A & C	
	A	B	A	C
Src-only (A)	0.7456	0.6892	0.7456	0.6621
Self-train (CAN-s)	0.7456	0.7183	0.7456	0.6734
FineTune	0.7435	0.7382	0.7398	0.6869
DANN	0.7489	0.7356	0.7459	0.6873
TL-WD	0.7490	0.7371	0.7460	0.6891
WE-CAN	**0.7501**	**0.7414**	**0.7489**	**0.7011**

The benefits of joint training can be seen from two aspects. First, while most of the performance improvement is on the target domain, the joint training strategy still boosts the performance on the source domain where the room for improvement is small because it is large enough. Second, a smaller target domain can benefit more from joint training than a bigger target domain. Our model shows more performance gain for 'A & C' than for 'A & B'. A potential reason is that domain C has less data than B, thus benefits more from the source domain. To further study this, we examine our model performance on items of different popularity in the following section.

4.3 Cold-start vs. Warm-start Items (Q3)

We are further interested in how the popularity of an item impacts the gain of our approach in joint training of platform A and B by comparing to the Self-train method (training on platform B only). Our hypothesis is that the benefit of leveraging data and knowledge from a large domain is more prominent for unpopular (cold-start) items from a small domain than for popular (warm-start) items, because the data in the small domain does not have sufficient data to train a model that can perform well on the unpopular items. To investigate, we categorize items into 6 bins (bin 0 as coldest and bin 6 as warmest) based on popularity and examine the performance of our model on different bins.

The result in Table 5 validates our hypothesis above that the less popular items are, the more beneficial our approach becomes. The performance improvement drops to its lowest on the warmest bin. For bin 0, the improvement is not as high as those in bin 1 to bin 3. One explanation is that these cold-start items are quite different from those in the large domain where the knowledge is

Table 5: AUC on items with different popularity

AUC	Bins (0: least popular, 5: most popular)					
	0	1	2	3	4	5
Self-train	0.694	0.692	0.682	0.682	0.709	0.738
WE-CAN	0.719	0.720	0.712	0.715	0.731	0.748
Improvment	3.6%	4.0%	4.4%	4.8%	3.1%	1.2%

transferred from. Hence the transfer is not very effective. Still, the improvement on bin 0 is higher than those on popular items from bin 4 or 5.

Similarly we have examined how item popularity impacts our approach in joint training of platform A and C. We also observed that our method helps more for items with less popularity (cold-start items). The findings are quite intuitive as less popular items are with less user behaviors, thus can potentially benefit more from the source domain.

4.4 Qualitative Evaluation (Q4)

To qualitatively evaluate domain-specific attention weights, we choose five types of features and illustrate the attention weights learned in Figure 2 [7]. Because of domain shift, features do have different weights in different domains. For example, for the high-end platform B, item brand and user background features play more important role than item price, whereas for the low-end platform C, the item price is more important than item brand. Since the platform A targets a wide range of users, overlapping with those on B and C, its attention weights on different features are also in the middle of those of B and C. In all, the attention weights learned by our model are interpretable and provides insights on user needs at different platforms.

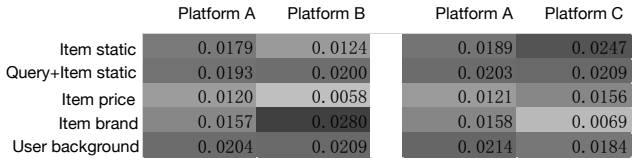

Figure 2: Attention weights of features on different platforms. The gray-scale background is proportional to the weight. The darker a cell is, the heavier the weight is.

4.5 Online Deployment (Q5)

Last but not least, we have deployed our model online on three aforementioned e-commerce platforms: Taobao, Tmall and Qintao and examined the model performance. Our full model is compared with a baseline model which is a variant of our model without joint training through a 7-day A/B testing. In terms of click-through-rate, our method outperforms the online model by 7%+ on platform Qintao on average, 4%+ on platform Tmall, and around 2% on the platform Taobao. Such improvements are considerably big for these platforms. In line with earlier observations, the biggest improvement happens on platform Qintao which has the least data and benefits the most from our approach. According to the statistics, this brings to an increase of 2.4%+ in terms of gross merchandise volume in total, which is considered to be a big improvement for the platforms.

[7] For each type of features, we average all the related feature attention weights.

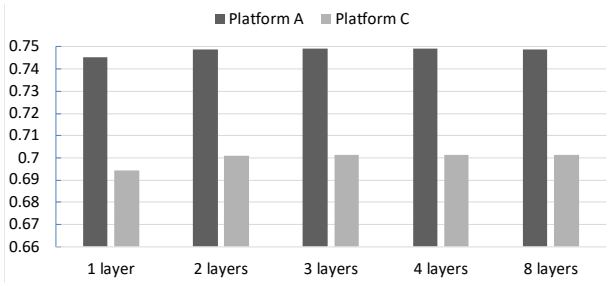

Figure 3: Parameter sensitivity w.r.t. different numbers of layers in a joint training for platforms A and C.

4.6 Parameter Sensitivity (Q6)

A good model also needs to be stable or insensitive to the parameters. According to our experience, the performance of a model is usually sensitive to the number of layers in neural networks. Hence we test our model's performance in relation to different numbers of layers.

As shown in Figure 3, our model is actually very stable. The model achieves a reasonably good performance with two layers (Deep&Cross networks). Increasing the number of layers beyond two does not significantly improve the performance. A potential reason is that the raw feature space is already very large and the interactions between them have already provided relatively sufficient data to train a model.

5 RELATED WORK

E-commerce Search. The core task for E-commerce search is to rank items according to the user and item features. This task is close to click-through-rate (CTR) prediction task, as the core problem is to evaluate the click or purchase probability of an item for a given user query. The task is a large-scale problem in the industry as usually millions of items and features are used. Due to the large feature space and data size, a number of methods are used to avoid extensive feature engineering, typical methods include: FM [3, 13, 17], Wide & Deep [4], DeepFM [9, 10, 22], and Deep & Cross [21]. The FM based methods [13, 17] embed sparse features with low-dimensional dense vectors and learns feature interactions based on the vectors. Still these models are considered as shallow and may have large computational cost by extending to higher orders [3]. With the popularity of deep neural networks (DNN), there are growing number of studies working on using DNN to learn high-degree feature interactions [4, 9, 21]. A recent study in [21] shows a cross network can help to learn high-degree feature interactions. Combining the cross network with the deep network, the resulting Deep&Cross method shows to have better performance than other methods. Hence we adopt this method as our hidden representation

layer. Note that this can be replaced with other models. Different from original Deep & Cross network, our method jointly learns domain-specific and shared embeddings together with attention layers to learn domain-specific feature representations.

Joint Training. We study cross-domain learning in our task, which is close to transfer learning [16] or multi-task learning [26]). There are generally two lines of studies for transfer learning. The first line of work is supervised domain adaptation, which assumes to have enough labeled data from a source domain and also a little labeled data from a target domain [6]. And the latter assumes to only have labeled data from source domain but may also have some unlabeled data from a target domain [2, 25]. Our study is close to supervised domain adaptation, which assumes to have enough labeled data from a source domain and also a little labeled data from a target domain [2, 5, 6, 25]. For studies in this line, a majority of recent studies are to find a shared feature space which can reduce the divergence between the distribution of the source and the target domains [1, 20]. In our study, we use deep neural network (DNN) models to learn shared feature representations, as DNN models are shown to be effective for learning transferable features in many tasks [12, 24].

With the popularity of deep learning, a great amount of neural network based methods are proposed for TL [11, 12, 24]. A simple but widely used framework is referred to as *fine-tuning* approaches, which first use the parameters of the well-trained models on the source domain to initialize the model parameters of the target domain, and then fine-tune the parameters based on labeled data in the target domain [15, 24]. A more effective method is to use a shared network to learn shared features for both source and target domains [15, 23]. Another approach is to use a specific-shared model that contains both a shared network and domain-specific networks to derive shared and domain-specific features [14]. Some recent studies [7, 14, 18, 19, 27] consider using adversarial network to learn more robust shared features across domains. Inspired the recent success of considering Wasserstein distance to learn domain-shared features [18], we extend the method to our specific-shared model to help learn both domain-shared features and domain-specific features. Different from [18], the cross-domain model in our setting is a specific-shared model. And we consider regularizer to capture two important intuitions: (i) domain specific features should be different than the shared features, and (ii) source domain and target domain shared features should be close to each other.

To the best of our knowledge, our work is the first to study cross-domain attention network with Wasserstein regularizer for e-commerce search. We consider both public and real-world industry datasets to evaluate the efficiency and effectiveness of our model. Experiments show that our model achieves significantly better results than the competing baseline methods.

6 CONCLUSION

In this work, we study cross-domain e-commerce search for building a model that is helpful for multiple domains. We proposed a cross-domain attention network for modeling the importance of features across different domains, and new Wasserstein regularizers to learn domain-invariant and domain-specific features. Quantitative and qualitative evaluations show that our model is both effective and insightful. We have also found our method helps more for cold-start items than warm-start items. We have deployed our method in real-world e-commerce sites and observed significant improvement over the production systems. In the future, we will examine the generality of our model by evaluating its performance on other tasks beyond e-commerce such as text retrieval.

REFERENCES

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*.

[2] John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[3] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In *Advances in Neural Information Processing Systems 29 (NIPS)*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 3351–3359. http://papers.nips.cc/paper/6144-higher-order-factorization-machines.pdf

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (2016).

[5] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. *KDD '07* (2007).

[6] Hal Daume III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR* 17, 59 (2016), 1–35.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017).

[10] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR'17*. 355–364.

[11] Xin Huang, Yuxin Peng, and Mingkuan Yuan. 2017. Cross-modal Common Representation Learning by Hybrid Transfer Network. *CoRR* abs/1706.00153 (2017).

[12] Xin Huang, Yuxin Peng, and Mingkuan Yuan. 2017. MHTN: Modal-adversarial Hybrid Transfer Network for Cross-modal Retrieval. *CoRR* abs/1708.04308 (2017).

[13] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *RecSys '16*. 43–50.

[14] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *ACL*.

[15] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications?. In *EMNLP'16*.

[16] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE* 22, 10 (2010), 1345–1359.

[17] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. 995–1000.

[18] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *AAAI*. AAAI Press.

[19] Yaniv Taigman, Adam Polyak, and Lior Wolf. 2017. Unsupervised cross-domain image generation. *ICLR* (2017).

[20] Chang Wang and Sridhar Mahadevan. 2008. Manifold alignment using procrustes analysis. In *ICML*.

[21] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD'17*. Article 12, 7 pages.

[22] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. *CoRR* abs/1708.04617 (2017).

[23] Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *ICLR* (2017).

[24] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *NIPS*.

[25] Jianfei Yu and Jing Jiang. 2016. Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification. In *EMNLP*.

[26] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv:1707.08114* (2017).

[27] Yu Zhang and Dit-Yan Yeung. 2010. A convex formulation for learning task relationships in multi-task learning. In *UAI*.