EdgeDIPN: a Unified Deep Intent Prediction Network Deployed at the Edge

Long Guo, Lifeng Hua, Rongfei Jia
Fei Fang, Binqiang Zhao
Alibaba Inc
{leo.gl,issac.hlf,rongfei.jrf,mingyi.ff,binqiang.zhao}@alibaba-inc.com

Bin Cui
School of EECS & Key Laboratory of High Confidence Software Technologies (MOE), Peking University
bin.cui@pku.edu.cn

ABSTRACT

With the rapid growth of e-commerce in recent years, e-commerce platforms are becoming a primary place for people to find, compare and ultimately purchase products. To improve online shopping experience for consumers and increase sales for sellers, it is important to understand user intent accurately and be notified of its change timely. In this way, the right information could be offered to the right person at the right time. To achieve this goal, we propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge, i.e., mobile device, and able to monitor multiple user intent with different granularity simultaneously in real-time. We propose to train EdgeDIPN with multi-task learning, by which EdgeDIPN can share representations between different tasks for better performance and saving edge resources in the meantime. In particular, we propose a novel task-specific attention mechanism which enables different tasks to pick out the most relevant features from different data sources. To extract the shared representations more effectively, we utilize two kinds of attention mechanisms, where the multi-level attention mechanism tries to identify the important actions within each data source and the inter-view attention mechanism learns the interactions between different data sources. In the experiments conducted on a large-scale industrial dataset, EdgeDIPN significantly outperforms the baseline solutions. Moreover, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing results in several business scenarios reveal the potential of monitoring user intent in real-time. To the best of our knowledge, EdgeDIPN is the first full-fledged real-time user intent understanding center deployed at the edge and serving hundreds of millions of users in a large-scale e-commerce platform.

PVLDB Reference Format:
Long Guo, Lifeng Hua, Rongfei Jia, Fei Fang, Binqiang Zhao, and Bin Cui. EdgeDIPN: a Unified Deep Intent Prediction Network Deployed at the Edge. PVLDB, 14(3): 320-328, 2021.
doi:10.14778/3430915.3430922

1 INTRODUCTION

E-commerce platforms are becoming a primary place for people to find, compare and ultimately purchase products. One of the fundamental questions that arises in e-commerce is to understand user intent, which allows for providing better services for both sellers and customers. In reality, a user's intent is changing all the time when surfing the e-commerce app. For example, a user's intent may change from browsing to buying when she finds some goods of interest. It is important to be notified of the change of user intent timely, which can provide users with brand new proactive experiences by offering the the right information at the right time. However, previous work only deals with the offline user intent prediction [7, 18, 23, 24, 27] and cannot monitor user intent in real-time. The reasons are twofold. First, previous work cannot predict users' real-time intent with a high accuracy limited by the representation capability of traditional coarse-grained behavior data. Second, it is difficult to monitor user intent in real-time in traditional cloud-based computing architecture due to the high communication cost.

In order to overcome the aforementioned difficulty when monitoring user intent in real-time, we propose to deploy our user intent understanding system at the edge, i.e., mobile devices, following the idea of edge computing. There are several advantages to deploy the system at the edge. First, it offers us the opportunity to unlock the potential of the vast untapped data created by the connected devices. In our system, we collect a new type of user interactive behavior data, which we call touch-interactive behavior and contributes a lot when predicting user's real-time intent. Second, it can reduce the communication cost between the cloud and the devices significantly. During the Alibaba 2019 Double 11 Shopping Festival, EdgeDIPN serves more than 0.5 billion customers without suffering from the traditional peak traffic problem of the platform on that day. Third, it can greatly increase the response speed of EdgeDIPN by moving the model to the data instead of the data to the model. The response time of EdgeDIPN is between 20 $\sim$ 50 ms on different devices, which is immune to the influence of the network traffic and can achieve the real-time requirement. Last but not least, it enables us to create new business scenarios in the e-commerce platform based on the real-time user intent.

However, monitoring user intent in real-time at the edge is challenging. First, it is necessary to build a computationally lightweight and memory efficient system, because edge resources are typically resource-constrained. Second, the touch-interactive behavior is more fine-grained and contains less semantic information compared with traditional browse-interactive behavior. Therefore, it is challenging to extract useful features from these data to improve the prediction performance. To solve these challenges, we propose a unified deep intent prediction network, named EdgeDIPN, which is able to monitor multiple user intent with different granularity simultaneously in real-time. To save edge resources, we propose

Proceedings of the VLDB Endowment, Vol. 14, No. 3 ISSN 2150-8097.
doi:10.14778/3430915.3430922

to train EdgeDIPN with multi-task learning. In more details, we extract a universal shared representation based on users' behavior data from several views, based on which multiple user intent with different granularity could be derived. In this way, edge resources can be reduced dramatically, because we do not need to build a model for each intent separately. We improve the performance of multi-task learning from two aspects. First, we utilize two kinds of attention mechanisms to extract the shared representation more effectively, where the multi-level attention mechanism tries to identify the important actions within each data source and the inter-view attention mechanism learns the interactions between different data sources. Second, we propose a novel task-specific attention mechanism which enables different tasks to pick out the most relevant features from different data sources. The contribution of the paper can be summarized as follows:

- We propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge and able to monitor multiple user intent with different granularity simultaneously in real-time.
- We propose to train EdgeDIPN with multi-task learning by sharing representations among several intent, which can save edge resources dramatically. In addition, several attention mechanisms are proposed to improve the performance.
- We evaluate the performance of EdgeDIPN by conducting extensive experiments on a large-scale industrial dataset. The results show the superiority of EdgeDIPN in predicting user intent. In particular, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing results show the benefits of monitoring user intent in real-time.

2 PROBLEM STATEMENT

2.1 Dataset

We build two types of user interactive behavior dataset which depict users from different views, i.e., the novel touch-interactive behavior collected from the edge and the traditional browse-interactive behavior. In particular, the touch-interactive behavior is further divided into the swipe and tap interactive behavior. The touch-interactive behavior records a user's fine-grained behavior when surfing in the app and thus contains rich user behavior patterns.

The swipe-interactive behavior includes four types of basic actions, i.e., *Open Page*, *Leave Page*, *Swipe* and *Tap*. A user's swipe-interactive track is a time sequence of actions, consisting of these four basic types of actions. Each action has a timestamp and a page index to identify when and where the action occurs. In addition, the positional coordinates of the action on the touch screen are also recorded (i.e., we record the tap position for the *Tap* action and the start and end positions for the *Swipe* action.). The duration presents how long the action lasts. For each action at a data point, we extract 14 features. Among these features, the time duration of a swipe, time gap between two actions and positional coordinates of actions (i.e., tap position X/Y, swipe start position X/Y, swipe end position X/Y and swipe length on X/Y) are continuous variables. Page indices, action indices and swipe directions (i.e., left/right and up/down) are categorical variables. We conduct discretization on all the raw features to ensure unified inputs for EdgeDIPN. The discretization of the continuous variables is described as follows:

- **Position.** The positional coordinates of actions are discretized according to the resolution of the touch screen. We divide the width and length of the screen into 17 and 25 uniform segments corresponding to X and Y, respectively, for one-hot vectors encoding.
- **Swipe Length.** The length of a swipe is encoded into a one-hot vector, the length of which is as twice as the length of the one-hot vectors of position encoding. The reason of applying twice length is that, for a swipe track, we consider the direction of the swipe.
- **Time Gap and Duration.** We apply a step function to encode time gaps between actions and swipe duration as follow:

$$y = \begin{cases} x/f_s, & x < f_b \\ x/f_b + 9, & f_b \le x < 10 \times f_b \\ 19, & x \ge 10 \times f_b \end{cases}$$

  where $\{f_s = 100, f_b = 1000\}$ are used for time gap, and $\{f_s = 25, f_b = 250\}$ are used for time duration.

The tap-interactive behavior records the information associated with the tap actions. A user's tap-interactive track is a time sequence of tap actions. Each action has a timestamp and page index to identify when and where the action occurs. There is also an event id to identify whether a user taps on a page or a button. If a button is tapped, the button name is also recorded. We extract 3 raw features (i.e., event index, page index and button index) for each tap action, all of which are categorical variables.

The browse-interactive behavior represents the typical behavior users conduct on products. It includes five types of actions, i.e., *Browse*, *Search*, *collect*, *Add to cart* and *Purchase*. A user's browse-interactive track is a time sequence of these actions. We extract 6 raw features for each action (i.e., type index, top category index, leaf category index, page index, page stay time and timestamp), where the type index represents the type of an action.

2.2 Problem Formulation

EdgeDIPN is used to monitor multiple user intent with different granularity in real-time, which enables us to send the right information to the right user at the right time. The typical predicted user intent is listed as follows.

- **Real-time Purchasing Intent**: whether a user would buy some goods within one hour.
- **Long-time Purchasing Intent**: whether a user would buy some goods within the current day.
- **Real-time Cart Intent**: whether a user would add some goods to cart within one hour.
- **Retention Intent**: whether a user would visit the app the next day.

Besides the above global intent, EdgeDIPN also predicts some fine-grained intent such as category preference. Note that a user's category preference is measured by the probability of whether the user would buy some goods belonging to the category. We also set the category preference as real-time and long-time preference, which means the probability of the order being placed within one hour and one week, respectively. In addition, the category is further divided into the top category and leaf category, following the settings of the category label system in Taobao app.

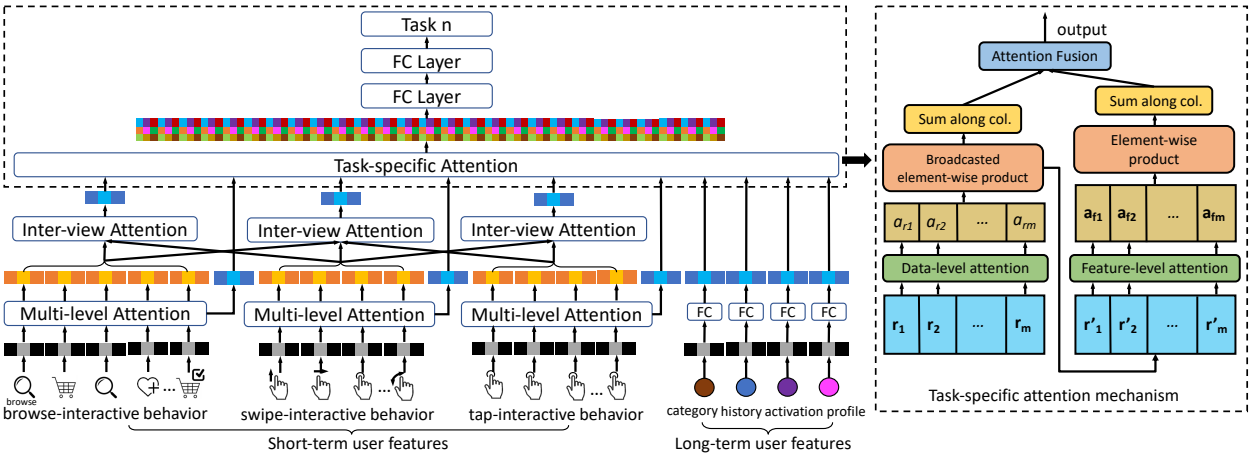

Figure 1: The model architecture of EdgeDIPN.

In our work, each intent is treated as a separate task and we adopt the binary classification for each task. Note that we do not use multi-label classification for the category preference, because the softmax operation would make the model too big to fit at the edge due to the large number of categories to be predicted.

3 MODEL ARCHITECTURE

We propose a unified deep intent prediction network deployed at the edge, named EdgeDIPN, to predict multiple user intent with different granularity simultaneously, as shown in Figure 1.

3.1 Embedding Layer

In the discretization process, the raw values of every feature are encoded into one-hot vectors. Then the one-hot vectors are used as the inputs of EdgeDIPN. As the inputs are high dimensional binary vectors with limited representation capacity, we use the embedding layer to transform them into low dimensional dense representations. The embedding operation follows the table lookup mechanism. In more details, each feature is corresponding to one embedding matrix. For example, the embedding matrix of the *Button Index* feature in the tap-interactive behavior is represented as $\mathrm{E}_{button} = [e_1; e_2; ...; e_{n_b}] \in \mathbb{R}^{n_e \times n_b}$, where $e_i \in \mathbb{R}^{n_e}$ represents an embedding vector with dimension n_e (i.e., 10), and n_b (i.e., 500) represents the number of buttons that a user can tap. The embedding vector of the *Button Index* feature can then be obtained as $\mathrm{e}_{\mathrm{button}} = \mathrm{E}_{button} \cdot \mathrm{b}_{\mathrm{button}} \in \mathbb{R}^{n_e}$, where $b_{button} \in \mathbb{R}^{n_b}$ is the one-hot vector of the *Button Index* feature. In addition, to capture the temporal order information in users' behavior sequence used by the following multi-level attention mechanism, we add the time feature to each action defined as the elapsed time (i.e., seconds) between the moment the action happens and the current moment. However, the model size would become too large after the embedding operation if we directly use the original elapsed time as the input feature. We notice that the impact of the time feature to the current intent decays as the elapsed time increases. Therefore, we propose a new discretization method which would not influence the model performance and can save model space significantly at the same time. Specifically, we split the time feature into multiple granularity, i.e., the continuous time feature in range of $[2^k, 2^{(k+1)})$ is mapped to the discrete feature $(k+1)$ while $[0, 1)$ is mapped to 0. At last, for each action, all the embedded features are concatenated into a vector and fed into a fully-connected layer for reshape. Specifically, each behavior sequence $\mathbf{S}$ can be encoded as a matrix $\mathbf{E}$, i.e., $\mathbf{E} = \{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_n}\} \in \mathbb{R}^{n \times d_e}$, where d_e is the dimension of $\mathbf{e_i}$ and n is the number of actions contained in $\mathbf{S}$. Moreover, as shown in Figure 1, except for the behavior sequences which can capture the short-term user features, we also collect some statistics for long-term user features.

3.2 Multi-level Attention Layer

The user interactive behaviors are all time sequence of actions. A traditional method is to use RNN to model the long-term dependencies between actions. The adoption of RNN can eliminate the need for extensive feature engineering. However, there are two limitations which hinder us from adopting RNN in EdgeDIPN. First, edge resources are typically resource-constrained and thus places a high demand on the memory cost and inference time of the model. Under this context, RNN is not suitable for use at the edge due to the complex data dependencies and limited parallelism. Second, RNN cannot achieve the best performance because of the unique characteristics of the touch-interactive behavior sequence with plenty of noisy actions operated by users inadvertently.

Therefore, we propose to use the attention mechanism to identify the important actions in the behavior sequence and reduce the interference from the noisy actions. Specifically, we use a softmax layer to determine the weights of different behavior actions. In this case, those actions contributing more to the intent understanding are given larger weights. The weights $\mathbf{a} = \{a_1, a_2, ..., a_n\}$ are computed as follows:

$$a_t = \mathrm{softmax}(\mathbf{h_t}), \mathbf{h_t} = \boldsymbol{w}_{s2} \tanh(\boldsymbol{W}_{s1} \mathbf{e_t}), \tag{1}$$

where $\mathbf{e_t}$ is embedding vector of the t-th action, $\boldsymbol{W}_{s1} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w}_{s2} \in \mathbb{R}^{d_e}$, and a_t is the weight calculated for $\mathbf{e_t}$. Then we can sum up the behavior sequence $\mathbf{E}$ according to the calculated weights to get a representation vector $\mathbf{r} \in \mathbb{R}^{d_e}$, i.e., $\mathbf{r} = \sum_{i=1}^{n} a_i \mathbf{e_i}$.

However, there is a limitation in the representation vector $\mathbf{r}$, which usually focuses on a specific component of the sequence, like a special set of actions. In reality, there can be multiple components in a sequence that together form the overall preferences of a user. More importantly, EdgeDIPN is proposed to predict multiple user intent and different intent may correspond to different component of the behavior sequence. Therefore, we perform multiple hops of attention and get multiple representation vectors that focus on different component of the behavior sequence, following the idea of the sentence embedding task in [15]. Suppose we want h different components to be extracted from the sequence, we just need to extend $\boldsymbol{w}_{s2}$ into a matrix $\boldsymbol{W}_{s2} \in \mathbb{R}^{h \times d_e}$. Formally, we have

$$\mathbf{A} = \text{softmax}(\boldsymbol{W}_{s2}\text{tanh}(\boldsymbol{W}_{s1}\mathbf{E}^T)),\ \mathbf{R} = \mathbf{AE}, \tag{2}$$

where the softmax() is performed along the second dimension of its input. As a result, the weight vector $\mathbf{a} \in \mathbb{R}^n$ becomes a weight matrix $\mathbf{A} \in \mathbb{R}^{h \times n}$, and the representation vector $\mathbf{r} \in \mathbb{R}^{d_e}$ becomes a embedding matrix $\mathbf{R} \in \mathbb{R}^{h \times d_e}$. After the multi-level attention layer, for each behavior sequence, we can get two outputs, i.e., the representation vector $\mathbf{r_f} \in \mathbb{R}^{d_e}$ by flattening and reshaping $\mathbf{R}$, and the reweighed sequence by applying the element-wise product on $\mathbf{E}$ and $\mathbf{R}$ as follows:

$$\mathbf{r_m} = \text{reduce\_mean}(\mathbf{R}),\ \mathbf{V} = \mathbf{E} \odot \mathbf{r_m}. \tag{3}$$

The reweighed sequences are then sent to the following layer.

3.3 Inter-view Attention Layer

To better fuse the views extracted from different behavior sequences, we adopt the inter-view attention mechanism [9] that learns the inter-view relations between different views, as shown in Figure 1. The inter-view attention takes two views as inputs, and emphasizes the important interactions between the two views using attention scores. In particular, for each action in one view, we calculate its relevance with all the actions in the other view. In this way, the asynchronous interactions between actions in these two sequences can be captured effectively. The inter-view attention $\text{IA}(\mathbf{V_s}, \mathbf{V_b})$ is formulated as follows, taking the swipe-interactive view $\mathbf{V_s}$ and the browse-interactive view $\mathbf{V_b}$ for example:

$$\begin{aligned}
\text{IA}(\mathbf{V_s}, \mathbf{V_b}) &= \text{A}_\text{s}(\mathbf{V_b}, \mathbf{V_s}, \mathbf{V_s}) \odot \text{A}_\text{b}(\mathbf{V_s}, \mathbf{V_b}, \mathbf{V_b}), \\
\text{A}_\text{s}(\mathbf{V_b}, \mathbf{V_s}, \mathbf{V_s}) &= \text{softmax}(\frac{\mathbf{V_b}\mathbf{V_s}^T}{\sqrt{2d}})\mathbf{V_s}, \\
\text{A}_\text{b}(\mathbf{V_s}, \mathbf{V_b}, \mathbf{V_b}) &= \text{softmax}(\frac{\mathbf{V_s}\mathbf{V_b}^T}{\sqrt{2d}})\mathbf{V_b},
\end{aligned} \tag{4}$$

where $\text{A}_\text{s} \in \mathbb{R}^{n \times d_e}$ and $\text{A}_\text{b} \in \mathbb{R}^{n \times d_e}$ are attentive representations of $\mathbf{V_s}$ and $\mathbf{V_b}$, respectively, and the element-wise product $\odot$ is used to model the interactions between A_s and A_b. Note that $\text{IA}(\mathbf{V_s}, \mathbf{V_b})$ is a symmetric operation and returns the representation $\mathbf{R_{sb}}$ between $\mathbf{V_s}$ and $\mathbf{V_b}$. We can calculate $\mathbf{R_{st}}$ and $\mathbf{R_{tb}}$ following the same procedure. At last, we can get three representations $\mathbf{r_{sb}} \in \mathbb{R}^{d_e}$, $\mathbf{r_{st}} \in \mathbb{R}^{d_e}$ and $\mathbf{r_{tb}} \in \mathbb{R}^{d_e}$ by applying the reduce_mean operation.

3.4 Multi-task Layer

In this work, we need to monitor multiple user intent simultaneously. Considering the limited computational and memory resources at the edge, it is not reasonable to build a model for each intent separately. Therefore, we propose to train EdgeDIPN with multi-task learning, by which a universal shared representation could be learned and used to derive multiple intent simultaneously In this way, edge resources can be reduced dramatically.

Prior approaches to simultaneously learning multiple tasks commonly concatenate the learned representations from several data sources and send the concatenated representation to the following task-specific output layers [2, 8, 18]. However, we find that different data representations have different contributions in different tasks. Therefore, the performance of each task is highly dependent on an appropriate choice of weighting for each representation. Searching for an optimal weighting is prohibitively expensive and difficult to resolve with manual tuning. Therefore, we propose a novel task-specific attention mechanism to learn the weights for each task automatically and enable each task to pick out the most relevant features from different data representations.

The task-specific attention consists of two components, i.e., data-level attention and feature-level attention, as shown in Figure 1. These two components allow each task the selection of effective data representations and latent features, respectively.

Data-level Attention. The data-level attention is used to learn the weights for each data representation given a task. Suppose we have m representations extracted from different data sources shown as the blue rectangles in Figure 1, i.e., $\mathbf{D} = \{\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_m}\}$, the weight vector $\mathbf{a_r}$ for the given task on different data representations is:

$$\begin{aligned}
\mathbf{a_r} &= \{a_{r1}, a_{r2}, ..., a_{rm}\}, \\
a_{ri} &= \frac{\exp(e(\mathbf{r_i}))}{\sum_{j=1}^{m}\exp(e(\mathbf{r_j}))}, \\
e(\mathbf{r_i}) &= \boldsymbol{w}_{r2}\text{tanh}(\boldsymbol{W}_{r1}\mathbf{r_i} + b_{r1}),
\end{aligned} \tag{5}$$

where $\boldsymbol{W}_{r1} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w}_{r2} \in \mathbb{R}^{d_e}$, and a_{ri} is the weight calculated for the data representation $\mathbf{r_i}$. After getting the weight vector $\mathbf{a_r} \in \mathbb{R}^m$, we extract a weighed data representation matrix $\mathbf{M}$ by applying the broadcasted element-wise product on $\mathbf{D}$ and $\mathbf{a_r}$.

Feature-level Attention. After getting the weighed data representation $\mathbf{M}$, we go one step further and let each task to pick out the relevant features from $\mathbf{M}$. The intuition behind this idea is that different features in the same data representation may have different impact on different tasks. The feature-level attention is a natural extension of the data-level attention at the feature level. Specifically, instead of computing a single scalar weight for each weighed data representation $\mathbf{r'_i}$, the feature-level attention computes a feature-wise weight vector for $\mathbf{r'_i}$, formalized as follows:

$$\begin{aligned}
\mathbf{a_f} &= \{\mathbf{a_{f1}}, \mathbf{a_{f2}}, ..., \mathbf{a_{fm}}\}, \\
\mathbf{a_{fi}} &= \frac{\exp(e(\mathbf{r'_i}))}{\sum_{j=1}^{m}\exp(e(\mathbf{r'_j}))}, \\
e(\mathbf{r'_i}) &= \boldsymbol{W}_{f2}\text{tanh}(\boldsymbol{W}_{f1}\mathbf{r'_i} + b_{f1}),
\end{aligned} \tag{6}$$

where $\boldsymbol{W}_{f1} \in \mathbb{R}^{d_e \times d_e}$, the vector $\boldsymbol{w}_{r2} \in \mathbb{R}^{d_e}$ in Eq. 5 is replaced with the matrix $\boldsymbol{W}_{f2} \in \mathbb{R}^{d_e \times d_e}$ in Eq. 6, and the softmax() is performed along the second dimension of its input. As a result, we can get a feature-wise weight matrix $\mathbf{a_f} \in \mathbb{R}^{m \times d_e}$. Then the new feature-wise weighted data representation $\mathbf{M'}$ is obtained by applying the element-wise product on $\mathbf{M}$ and $\mathbf{a_f}$, i.e., $\mathbf{M'} = \mathbf{M} \odot \mathbf{a_f}$.

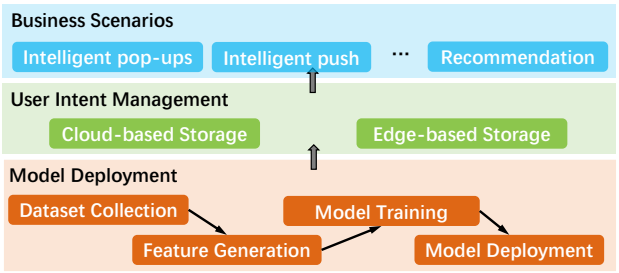

Figure 2: System overview under edge computing.

Attention Fusion. This component is used to combine the two weighted data representation $\mathbf{M}$ and $\mathbf{M}'$, as shown in Figure 1. The combination is accomplished by a dimension-wise fusion gate:

$$\begin{aligned} \mathbf{F} &= \text{sigmoid}(\boldsymbol{W}\mathbf{M} + \boldsymbol{W}'\mathbf{M}' + b), \\ \mathbf{O} &= \mathbf{F} \odot \mathbf{M} + (1 - \mathbf{F}) \odot \mathbf{M}', \end{aligned} \tag{7}$$

where $\boldsymbol{W}, \boldsymbol{W}' \in \mathbb{R}^{d_e \times d_e}$ and $b \in \mathbb{R}^{d_e}$.

For each task, the extracted data representation $\mathbf{O}$ is then fed into two fully connected layers to further learn the combination of task-specific features automatically. Given T binary classification tasks, the loss function of task t_i is:

$$\mathcal{L}_i = -\frac{1}{N} \sum_{(x,y)\in\mathcal{D}}^{N} (y \log p_i(x) + (1-y)\log(1-p_i(x))) \tag{8}$$

where $\mathcal{D}$ is the training set with size D, x is the input of the network and y is the label, and $p_i(x)$ represents the predicted probability of sample x for task t_i. The global loss is:

$$\mathcal{L} = \sum_{i=1}^{T} \mathcal{L}_i + \lambda ||(\mathbf{A}\mathbf{A}^T - \mathbf{I})||_F^2 \tag{9}$$

where the term $||(\mathbf{A}\mathbf{A}^T - \mathbf{I})||_F^2$ is used to punish redundancy between different vectors in the weight matrix $\mathbf{A}$ mentioned in Section 3.2, and thus the multi-level attention mechanism can focus on different component of the behavior sequence.

4 SYSTEM OVERVIEW

The structure of our user intent prediction system is illustrated in Figure 2. For model deployment, the procedure is as follows. First, user behavior data is collected from the mobile app usage logs to construct the dataset. Second, features are generated to build the training samples. Third, EdgeDIPN is trained in the cloud based on the samples. Finally, the trained model is deployed to each user's mobile device. In addition, we build a platform where the model deployment procedure is automatically conducted. As a result, we have built a user intent understanding system in several apps owned by Alibaba for more than a year, such as Taobao, Tmall, Xianyu and Freshippo with little manual work. For user intent management, EdgeDIPN automatically predicts multiple user intent using the real-time features collected from the mobile device at a predefined time interval. The predicted user intent is stored in two forms, i.e., the cloud-based storage where only the predicted intent is sent to the cloud and the edge-based storage where the intent is directly stored at the edge, and used to support the above business scenarios.

5 EXPERIMENTS

5.1 Experimental Setup

Dataset Statistics. We conduct the experiments on a large-scale industrial dataset collected from Taobao. The dataset contains normal users' daily interaction information when using our app. We collect 800,000 users' behaviors for two weeks. For each user, we randomly truncate about 400 groups of samples on average. In total, we obtain 300 million groups of samples. Each group contains the user profile feature, the user statistic feature, and the three behavior sequences (the default length is 256 and padding 0 for short ones). The labels of the multiple tasks are then tagged for each group based on the timestamp of the last action. We use samples in the first 13 days for training and samples in the last day for evaluation.

Compared Methods. We compare EdgeDIPN with the following baseline methods. In particular, we conduct experiments to verify the effect of each component in EdgeDIPN. In the following, we introduce the compared methods briefly.

- **GBDT**, which is the competitive gradient boosting model widely used in the industrial environment.
- **EdgeDIPN-tradition**, which does not use the touch interactive behavior data. Our goal is to see the benefit of using the new touch-interactive behavior.
- **EdgeDIPN-single**, which does not use multi-task learning and train a separate model for each intent.
- **EdgeDIPN-v0**, which replaces the multi-level attention mechanism with the GRU architecture and removes the inter-view attention and task-specific attention mechanism from EdgeDIPN. This method is used to verify the effect of each component in EdgeDIPN.
- **EdgeDIPN-fastgrnn**, which replaces the GRU architecture in EdgeDIPN-v0 with a more advanced RNN architecture named Fastgrnn [13] with smaller model size and lower prediction costs suited to be used at the edge. In addition, we also test the low rank version of Fastgrnn which can further reduce the mode size, denoted as EdgeDIPN-fastgrnnlr.
- **EdgeDIPN-v1**, which replaces GRU in EdgeDIPN-v0 with the multi-level attention mechanism.
- **EdgeDIPN-v2**, which adds the inter-view attention mechanism to EdgeDIPN-v1.
- **EdgeDIPN-v3**, which adds the data-level attention mechanism to EdgeDIPN-v2.
- **EdgeDIPN-v4**, which adds the feature-level attention mechanism to EdgeDIPN-v3 but the feature-level attention mechanism picks out relevant features from the original data representations $\mathbf{D}$ instead of $\mathbf{M}$.
- **EdgeDIPN**, which is the complete model.

Evaluation Metrics. The metric used in our experiments is Area Under the Curve (AUC), which is insensitive to class imbalance and suitable to our experiments. Since EdgeDIPN is used to predict multiple intent simultaneously, we define a metric name for each intent, as shown in Table 1.

Experimental Details. The dimension of embedding features is 10 and the dimension of the reshaped embedding vector is set to $d_e = 64$. The parameters h and λ corresponding to the multi-level attention mechanism are set to 10 and 0.01, respectively. Layers of MLP is set by $128 \times 64 \times 2$. In our experiments, we test 8 binary

Table 1: The metric name of each user intent.

User Intent	Metric Name
Real-time Purchasing Intent	RPAUC
Long-time Purchasing intent	LPAUC
Real-time Cart Intent	RCAUC
Retention Intent	RAUC
Real-time Top Category Preference	RTCAUC
Long-time Top Category Preference	LTCAUC
Real-time Leaf Category Preference	RLCAUC
Long-time Leaf Category Preference	LLCAUC

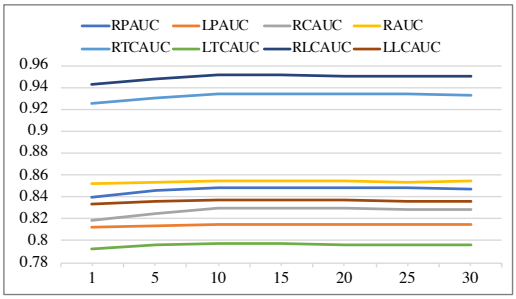

Figure 3: Impact of multiple components.

classification tasks, i.e., $T = 8$. EdgeDIPN is trained with SGD, using the Adam optimizer [12] with initial hyper-parameters of $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the mini-batch size to be 1024 and apply exponential decay, in which the learning rate starts at 0.001 and decay rate is set to 0.9. We train EdgeDIPN using a distributed TensorFlow with 1 parameter server and 200 workers.

5.2 Experimental Results

Results of different models. Table 2 shows the performance of the evaluated models. We have the following observations. (1) EdgeDIPN outperforms the baseline methods EdgeDIPN-tradition on all the tasks by a significant margin between 0.9% and 5.3% in terms of AUC. The improvement of EdgeDIPN over EdgeDIPN-tradition reveals the value of adopting the touch-interactive behavior to depict users from different views. (2) The multi-level attention mechanism is effective in identifying the important actions and reduce the interfence from the noisy actions. We can see that EdgeDIPN-v1 performs better than EdgeDIPN-v0, EdgeDIPN-fastgrnn and EdgeDIPN-fastgrnnlr for all the tasks. (3) The interview attention mechanism plays an important role in EdgeDIPN. As shown, EdgeDIPN-v2 outperforms EdgeDIPN-v1 for all the tasks. (4) With the help of multi-task learning, EdgeDIPN can increase AUC by about 0.85% on average compared with EdgeDIPN-single. This shows the superiority of adopting multi-task learning in EdgeDIPN, which can not only save edge resources dramatically but also improve the performance greatly. In particular, the comparison of EdgeDIPN, EdgeDIPN-v4, EdgeDIPN-v3 with EdgeDIPN-v2 demonstrate the effectiveness of our proposed task-specific attention mechanism. The reason is that different data representations as well as their features play different role in different tasks. By adopting the task-specific attention mechanism, we can let each task choose the most relevant data representations and features, and thus reduce the mutual influence between tasks. We can also see that EdgeDIPN outperforms EdgeDIPN-v4 for all the tasks. This is because the feature-level attention cannot consider the relationship between data representations in EdgeDIPN-v4, which can be solved by adopting the data-level attention first in EdgeDIPN.

Impact of sequence length (Scalability). In this section, we evaluate the impact of different sequence lengths on the performance of the models. The results are shown in Table 3. We evaluate four different models, where Att, RNN, FG and FG-lr denote the model with the the multi-level attention mechanism, GRU, Fastgrnn and Fastgrnnlr structure, repectively. As shown, the prediction performance [1] of each model increases when the sequence length increases. This is because more useful features can be extracted from longer sequences. However, longer sequences result in larger memory cost and longer inference time, which creates a heavy burden for the deployment at the edge. Compared to RNN, Att scales better when the sequence length increases. As shown, when the sequence length is 512, Att can reduce the model size and inference time by 43.9% and 56.2% , respectively. Although FG and FG-lr can reduce the model size and inference time significantly compared with RNN, they perform much worse in terms of AUC in our scenario. Compared with FG and FG-lr, Att not only performs much better in terms of AUC, but also has comparable and even better performance in model size and inference time, which further demonstrates its the effectiveness.

Impact of multiple components. In this section, we evaluate the impact of the parameter h in the multi-level attention mechanism. We vary h from 1 to 30 for each task and the results are shown in Figure 3. We can see the best performance appears after h reaches 10. This demonstrates that there can be multiple components in a sequence that together form the overall preferences of a user. However, it is not always true that a larger h results in better performance. As shown, there is no more benefit when h is larger than 10. Therefore, an appropriate h should be selected.

5.3 Online A/B Testing

5.3.1 Intelligent Pop-ups. Pop-ups are one of the most powerful tools on e-commerce platforms which can reach most of the traffic the platforms receive. If used intelligently, they can play an important role in turning the visitors into sales. In this section, we introduce a novel pop-up strategy based on the real-time category preference predicted by EdgeDIPN in online traffic of Taobao. One important task in Taobao is to promote the conversions of the inactive customers. Since these customers are hard to place an order, we design a special promotion strategy, introduced as follows. We first use the pop-ups to pop some product from the candidate products to the inactive customers. If one customer taps the pop-ups, she would be directed into a shopping page where the products are specially designed for the inactive customers and can promote the conversions effectively. Therefore, one of the key tasks in this strategy is to improve the click-through rate of the pop-ups.

[1] Due to the page space limitation, we only show the AUC for the real-time purchasing intent. The performance is simlar for other intent.

Table 2: Comparison of different models.

Model	RPAUC	LPAUC	RCAUC	RAUC	RTCAUC	LTCAUC	RLCAUC	LLCAUC
GBDT	0.7911	0.7747	0.8057	0.8421	0.8933	0.7826	0.9102	0.8203
EdgeDIPN-tradition	0.7960	0.7856	0.8159	0.8457	0.9069	0.7883	0.9267	0.8263
EdgeDIPN-single	0.8396	0.8068	0.8208	0.8512	0.9241	0.7889	0.9407	0.8286
EdgeDIPN-fastgrnn	0.8367	0.8049	0.8181	0.8496	0.9153	0.7834	0.9291	0.8214
EdgeDIPN-fastgrnnlr	0.8342	0.8037	0.8160	0.8490	0.9105	0.7807	0.9262	0.8185
EdgeDIPN-v0	0.8413	0.8093	0.8191	0.8519	0.9208	0.7895	0.9397	0.8278
EdgeDIPN-v1	0.8435	0.8097	0.8221	0.8522	0.9231	0.7901	0.9418	0.8283
EdgeDIPN-v2	0.8458	0.8106	0.8239	0.8527	0.9266	0.7909	0.9433	0.8292
EdgeDIPN-v3	0.8471	0.8126	0.8271	0.8534	0.9317	0.7947	0.9471	0.8341
EdgeDIPN-v4	0.8482	0.8138	0.8291	0.8538	0.9335	0.7955	0.9499	0.8357
EdgeDIPN	**0.8489**	**0.8152**	**0.8298**	**0.8543**	**0.9346**	**0.7969**	**0.9514**	**0.8368**

Table 3: Impact of sequence length.

Sequence Length	64	128	256	512
RPAUC (RNN)	0.8395	0.8418	0.8463	0.8499
RPAUC (FG)	0.8371	0.8384	0.8419	0.8432
RPAUC (FG-lr)	0.8353	0.8359	0.8396	0.8389
RPAUC (Att)	0.8413	0.8457	0.8489	0.8508
Model Size (RNN)	1.38MB	1.73MB	2.42MB	3.83MB
Model Size (FG)	1.06MB	1.38MB	1.69MB	2.45MB
Model Size (FG-lr)	0.88MB	1.19MB	1.53MB	2.21MB
Model Size (Att)	1.03MB	1.30MB	1.57MB	2.15MB
Infer Time (RNN)	19.5ms	32.9ms	59.8ms	111.6ms
Infer Time (FG)	15.6ms	24.5ms	41.3ms	81.7ms
Infer Time (FG-lr)	14.2ms	23.6ms	35.5ms	59.8ms
Infer Time (Att)	13.7ms	22.6ms	31.2ms	48.9ms

To improve the CTR of the pop-ups, one common method is to pop the hot products to the customers, which is usually a good strategy. A more effective method is to pop the products based on customers' preference. However, the inactive customers have very few history behaviors and thus traditional recommendation methods are not suited to be used in this scenario. The advantage of our EdgeDIPN is that we can utilize users' rich real-time behaviors to predict users' intent. Therefore, we propose to pop the products based on customers' real-time leaf category preference.

We set four pop-ups strategies as follows.

- **Hot Strategy** where the popped product is selected based on the purchase frequency of the products.
- **Category-1 Strategy** where the leaf category ranking first in the category preference is selected and then the hot product belonging to this category is selected.
- **Category-2** and **Category-3 Strategy** are used to compare with **Category-1**, where the leaf category ranking second and third are selected, respectively.

The online A/B testing results are shown in Figure 4, where the x-axis and y-axis represents the date time and the CTR, respectively. It is notable that the category-1 strategy contributes up to **50.1%** CTR promotion in average compared with the hot strategy in the online

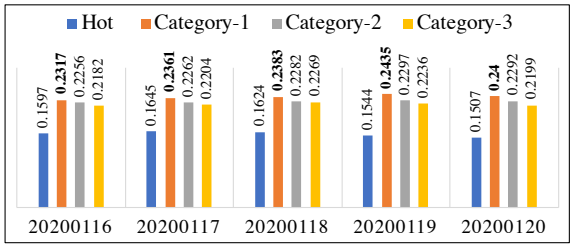

Figure 4: The results of different pop-up strategies.

traffic. The reason is that EdgeDIPN can understand a user's real-time preference by utilizing the rich real-time behaviors conducted by the user, which helps the pop-ups to select more appropriate product. In addition, the category-1 strategy increases the CTR by **4.4%** and **7.3%** compared with the category-2 and category-3 strategy, respectively. This indicates the effectiveness of EdgeDIPN in ranking user category preference.

5.3.2 Intelligent Push. Push notifications are another powerful tool to boost the user engagement on e-commerce platforms. Compared with the pop-ups, push notifications cause less disturbance to users and thus can be used at any moment when a user is visiting the app. In this section, we introduce a novel push notification strategy for the soon-to-expire coupons based on the real-time purchasing intent predicted by EdgeDIPN in online traffic of Taobao.

We choose two different types of coupons (denoted as C_1 and C_2) and set two push strategies to compare the performance.

- **Random Strategy** where the system chooses the moment randomly to push the notification of the soon-to-expire coupons to a user when she is visiting.
- **Model Strategy** where the system uses the score given by EdgeDIPN and a fixed threshold to decide the push moment.

The users selected to get this notification will be pushed a message in Taobao's mobile application. We use the CTR improvement I_C as the evaluation metric, defined as follows:

$$I_C = \frac{CTR_{model} - CTR_{rand}}{CTR_{rand}}, \tag{10}$$

where CTR_{model} and CTR_{rand} are the CTR of the push notifications with model and random strategy, respectively.

Table 4: The results of different push notification strategies.

	Num. of Users	I_{C_1}	I_{C_2}
Random Strategy	18.3M	/	/
Model Strategy	18.3M	**+9.02%**	**+5.32%**

We hypothesize that the users with a low purchasing intent may not be interest in these coupons and would be disturbed by these push notifications. On the contrary, the users with a high purchasing intent may be attracted by these coupons and click the related push notifications. Therefore, we set the threshold $t_l = 0.3$ which has the best online performance. The users whose real-time purchasing intent score given by EdgeDIPN is large than t_l are selected to get this push notification in the model strategy bucket.

As shown in Table 4, the model strategy contributes up to **9.02%** and **5.32%** *CTR* promotion compared with random strategy in the two different types of coupons in the large scale online traffic. The reason is that EdgeDIPN can help the system to understand users' real-time purchasing intent and push the coupon notification to the right person at the right time. Compared with random strategy, a reasonable model strategy relied on EdgeDIPN would result in a significant CTR improvement.

6 RELATED WORK

6.1 Multi-task Learning

Multi-task learning has been used successfully across various applications of machine learning. Ruder [20] presents an overview of multi-task learning in deep learning, where multi-task learning is typically done with either hard or soft parameter sharing of hidden layers. The hard parameter sharing method is the most commonly used multi-task learning approach, which shares the hidden layers between all tasks and keeps several task-specific output layers [4, 5, 8, 18]. The shared-bottom model structure suffers from optimization conflicts caused by task differences. To overcome this issue, Ma et al. [16] propose the MMoE method by sharing the expert submodels across all tasks with a gating network trained to optimize each task. However, MMoE is originally designed for DNNs where the shared-bottom model can be easily partitioned into small expert submodels to reduce the parameter size. This is not suited for EdgeDIPN with a complicated bottom model, which would result in a model with a large size and cannot be deployed at the edge. In soft parameter sharing, each task has its own model with its own parameters where the distance between the parameters is regularized [6, 17, 25]. However, compared to the hard parameter sharing methods, the soft parameter sharing methods have much more task-specific parameters and thus are not suited to be adopted at the edge. Recently, some work [3, 21] try to optimize multi-task learning from the perspective of loss functions. Previous methods usually use a weighed sum of losses, where the loss weights are uniform or manually tuned, and is hard to achieve the best performance. To overcome this issue, Kendall et al. [3] propose to automatically learn the weights by using the homoscedastic task uncertainty, while Sener et al. [21] cast multi-task learning as multi-objective optimization and aim to find a Pareto optimal solution. These methods are orthogonal to our work and can be adopted in EdgeDIPN seamlessly. Compared with previous work, EdgeDIPN proposes a task-specific attention mechanism to let each task pick out the most relevant features from the extracted data representations and thus reduce the conflict between different tasks. Moreover, the task-specific attention mechanism is light-weight and very friendly to be used at the edge.

6.2 Mobile User State Prediction

Another related work is user state prediction based on the mobile phone usage history. In these studies, the usage history is used as features for machine learning models and the targeted user state varies for each study, such as emotion state [26], user trait [22], and mental health [1]. However, none of these studies implement an on-device inference for a real-time prediction [14]. Current on-device applications are mainly image recognition [10] and natural language processing such as smart reply [11]. Recently, Ochiai et al. [19] implement a MLP model on Android smartphone for the smartphone operation troubleshooting recommendation. We are the first to monitor multiple user intent on a large-scale e-commerce platform at the edge. In [9], we make a preliminary attempt at predicting user's real-time purchasing intent with a model named DIPN. EdgeDIPN distinguishes from DIPN in several aspects. First, EdgeDIPN is a full-fledged real-time user intent understanding center and demonstrates its effectiveness in several business scenarios while DIPN just focuses on the purchasing intent prediction. Second, EdgeDIPN replaces the RNN component with the multi-level attention mechanism, which can not only improve prediction performance but also save edge resources. Last but not least, EdgeDIPN proposes a novel task-specific attention mechanism for the multi-task learning.

7 CONCLUSION

In this paper, we propose a unified deep intent prediction network, named EdgeDIPN, which is deployed at the edge and able to monitor multiple user intent with different granularity simultaneously in real-time. To save edge resources, we propose to train EdgeDIPN with multi-task learning by sharing representations among several intent. In addition, several attention mechanisms are proposed to improve the performance of EdgeDIPN. Experimental results on a large-scale industrial dataset shows the superiority of EdgeDIPN. In particular, EdgeDIPN has been deployed in the operational system of Alibaba. Online A/B testing on several business scenarios shows the benefits of monitoring users' intent in real-time.

Note that EdgeDIPN is a universal and scalable architecture which extracts an effective user embedding from various user behaviors and thus new user intent can be easily integrated. In addition, EdgeDIPN is the first full-fledged on-device attempt at the domain of e-commerce, which can offer the opportunity to create more novel business scenarios in the e-commerce platforms.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61702016 and 61832001, the National Key Research and Development Program of China (No.2018YFB1004403),Beijing Academy of Artificial Intelligence (BAAI), and Okawa Research Grant.

REFERENCES

[1] B. Cao, L. Zheng, C. Zhang, P. S. Yu, A. Piscitello, J. Zulueta, O. Ajilore, K. Ryan, and A. D. Leow. Deepmood: Modeling mobile phone typing dynamics for mood detection. In *KDD*, page 747–755, 2017.

[2] Y. Chen, Y. Ma, X. Mao, and Q. Li. Multi-task learning for abstractive and extractive summarization. *Data Science and Engineering*, 4(1):14–23, 2019.

[3] R. Cipolla, Y. Gal, and A. Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491, June 2018.

[4] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[5] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *ICASSP*, 2013.

[6] L. Duong, T. Cohn, S. Bird, and P. Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL-IJCNLP*, pages 845–850, 2015.

[7] Z. Gharibshah, X. Zhu, A. Hainline, and M. Conway. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering*, 5(1):12–26, 2020.

[8] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[9] L. Guo, L. Hua, R. Jia, B. Zhao, X. Wang, and B. Cui. Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *KDD*, page 1984–1992, 2019.

[10] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool. Ai benchmark: Running deep neural networks on android smartphones. In *ECCV Workshops*, 2018.

[11] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, M. Ganea, P. Young, and et al. Smart reply: Automated response suggestion for email. In *KDD*, page 955–964, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *NIPS*, pages 9017–9028. 2018.

[14] S. Li, D. Zhai, P. Du, and T. Han. Energy-efficient task offloading, load balancing, and resource allocation in mobile edge computing enabled iot networks. *Science China Information Sciences*, 62(2):29307, 2018.

[15] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[16] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *KDD*, page 1930–1939, 2018.

[17] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. *CoRR*, abs/1604.03539, 2016.

[18] Y. Ni, D. Ou, S. Liu, X. Li, W. Ou, A. Zeng, and L. Si. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *KDD*, pages 596–605, 2018.

[19] K. Ochiai, K. Senkawa, N. Yamamoto, Y. Tanaka, and Y. Fukazawa. Real-time on-device troubleshooting recommendation for smartphones. In *KDD*, page 2783–2791, 2019.

[20] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[21] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *CoRR*, abs/1810.04650, 2018.

[22] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti. Predicting user traits from a snapshot of apps installed on a smartphone. *SIGMOBILE Mob. Comput. Commun. Rev.*, 18(2):1–8, 2014.

[23] A. Toth, L. Tan, G. D. Fabbrizio, and A. Datta. Predicting shopping behavior with mixture of rnns. In *ACM SIGIR Forum*, 2017.

[24] R. Xu, J. Du, Z. Zhao, Y. He, Q. Gao, and L. Gui. Inferring user profiles in social media by joint modeling of text and networks. *Science China Information Sciences*, 62(11):219104, 2019.

[25] Y. Yang and T. M. Hospedales. Trace norm regularised deep multi-task learning. *CoRR*, abs/1606.04038, 2016.

[26] X. Zhang, W. Li, X. Chen, and S. Lu. Moodexplorer: Towards compound emotion detection via smartphone sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4), 2018.

[27] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068, 2018.