DYNAMIC PARAMETERIZED NETWORK FOR CTR PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning to capture feature relations effectively and efficiently is essential in click-through rate (CTR) prediction of modern recommendation systems. Most existing CTR prediction methods model such relations either through tedious manually-designed low-order interactions or through inflexible and inefficient high-order interactions, which both require extra DNN modules for implicit interaction modeling. In this paper, we proposed a novel plug-in operation, Dynamic Parameterized Operation (DPO), to learn both explicit and implicit interaction instance-wisely. We showed that the introduction of DPO into DNN modules and Attention modules can respectively benefit two main tasks in CTR prediction, enhancing the adaptiveness of feature-based modeling and improving user behavior modeling with the instance-wise locality. Our Dynamic Parameterized Networks significantly outperforms state-of-the-art methods in the offline experiments on the public dataset and real-world production dataset, together with an online A/B test. Furthermore, the proposed Dynamic Parameterized Networks has been deployed in the ranking system of one of the world’s largest e-commerce companies, serving the main traffic of hundreds of millions of active users.

1 INTRODUCTION

Click-through rate (CTR) prediction, which aims to estimate the probability of a user clicking an item, is of great importance in recommendation systems and online advertising systems (Cheng et al., 2016; Guo et al., 2017; Rendle, 2010; Zhou et al., 2018b). Effective feature modeling and user behavior modeling are two critical parts of CTR prediction.

Deep neural networks (DNNs) have achieved tremendous success on a variety of CTR prediction methods for feature modeling (Cheng et al., 2016; Guo et al., 2017; Wang et al., 2017). Under the hood, its core component is a linear transformation followed by a nonlinear function, which models weighted interaction between the flattened inputs and contexts by fixed kernels, regardless of the intrinsic decoupling relations from specific contexts (Rendle et al., 2020). This property makes DNN learn interaction in an implicit manner, while limiting its ability to model explicit relation, which is often captured by feature crossing component (Rendle, 2010; Song et al., 2019). Most existing solutions exploit a combinatorial framework (feature crossing component + DNN component) to leverage both implicit and explicit feature interactions, which is suboptimal and inefficient (Cheng et al., 2016; Wang et al., 2017). For instance, wide & deep combines a linear module in the wide part for explicit low-order interaction and a DNN module to learn high-order feature interactions. Follow-up works such as Deep & Cross Network (DCN) follows a similar manner by replacing the wide part with more sophisticated networks, however, posits restriction to input size which is inflexible.

Above-mentioned methods pay little attention to user behavior modeling. Recently, attention-based methods like DIN and DIEN have attracted many interests that attempt to capture user preferences based on users’ historical behaviors (Zhou et al., 2018b; 2019; Feng et al., 2019). With regard to the interaction of characteristics, the use of attention mechanisms in these methods can be treated as an explicit modelling of the interaction of characteristics while neglecting the modelling of implicit interactions of characteristics.

The methods mentioned above either model implicit and explicit feature interactions isolated or adopt a suboptimal way to combine them, which can be inefficient. In this work, we aim to address these problems by introducing a small MLP layer that dynamically generates kernels conditioned...
by the current instance to capture both implicit and explicit feature interactions. The core idea is
to first generate context weights and biases from the context stream, and then aggregate them with
the input stream adaptively. We formulate a generic function and implement it with an efficient
dynamic parameterized operation (DPO). The first weight generator projects contextual features
into high-dimensional representation, which models implicit conditional bias. The second feature
aggregator aims to fuse input features and projected contextual representation in a multiplicative way,
e.g., matrix multiplication and convolution, maintaining both low- and high-order information.

For feature-based modeling, we introduce feature-based DPO where the weight-generate operation
dynamically produces instance-wise filters conditioned on the embedded context. The feature-aggregate function then applies instance-wise filters to the flattened input by matrix multiplication,
allowing to learn multiplicative features. In particular, we further propose a new class of DPO, called
field-based DPO, which is not only instance-specific but also field-specific. In that case, the filters
vary from field to field and from instance to instance, allowing more complex interactions along the
field dimension.

For user behavior modeling, we introduce sequence-based DPO that consists of two variants: behavior-behavior dynamic operation and query-behavior dynamic operation. A representative method of
dynamic convolution \cite{Chen2020, Yang2019} shares the convolution kernel, which
is generated by the global average of the inputs. Similarly, \cite{Wu2019} proposed DyConv, a
lightweight fine-grained convolution that depends only on time-step, reinforcing the encoder-based
language modeling framework. However, our methods incorporate both local and global information
as they jointly use locality-aware methods (e.g., convolution or separable convolution) followed
by a global average pooling layer to produce instance-wise weights. The query-behavior dynamic
operation is specialized designed for the decoder-based framework in CTR prediction, aiming to
capture target-behavior dependency.

To our best knowledge, this is the first attempt to extend the business of dynamic neural networks
to CTR prediction with extensive experiments on two fundamental scenarios. The comprehensive
study against existing solutions validates the superiority of our proposed method. Moreover, we
demonstrate that incorporating DPO into the real-world ranking system is beneficial.

Our contribution can be summarized as followed:

- We propose a generic formulation for capturing multiplicative interaction via weight-
generate and feature-aggregate function, termed DPO.
- For feature-based modeling, we propose two variants, named field-based and feature-based
  DPO, offering a unifying view of implicit and explicit feature interaction. Decomposing
  these operations, we find they implicitly inherit low- and second-order representation.
- For user behavior modeling, we propose two sequence-based variants: behavior-behavior
  and query-behavior DPO. The first one computes locally perceptual dynamic filters and the
  second one learns target-behavior dependency in a multiplicative manner. We demonstrate
  that such operations can benefit the self-attention layers by higher computational efficiency
  through modeling locality as inductive bias.
- The proposed dynamic parameterized networks outperform state-of-the-art methods by a
  significant margin on both public and real-world production datasets. We also give a compre-
  hensive study about the relationship of our proposed methods to previous Factorization
  Machine \cite{Rendle2010} and CrossLayer \cite{Wang2017}. We further demonstrate the
effectiveness and superiority of our method with an online A/B test in real-world applications
by incorporating it into the fine-rank stage of the real-world ads system.

2 Method

We first review the mainstream approaches of feature-based and user behavior (sequence-based)
modeling under the situation of CTR prediction\footnote{Related work is in Appendix A due to space limitation.}. After that, we introduce DPO and provide several
specific instantiations designed for traditional feature-based and sequence-based modeling.
2.1 Preliminary

Traditional CTR prediction methods mainly predict a probability of a user click an item, which serves as a fundamental evaluation criterion for computing advertising systems. Typically, in a given scenario (the contexts), users click on certain items (item profiles) based on their own needs (query) and past preferences (user profiles). Consequently, a model considers four fields of features, i.e., query, user profile, item profile, and contexts to predict:

\[ CTR = F(query, user\ profile, item\ profile, contexts) \]  

(1)

where item and user profiles contain up to tens of fine-grained static attributes depending on the specific circumstances.

Sequence-based CTR prediction involves user behaviors additionally:

\[ CTR = F(query, user\ behavior, user\ profile, item\ profile, contexts) \]  

(2)

where the models can learn from the behaviors that have occurred under certain contexts and query in the past to make judgments on the current items. As mentioned in KFAtt (Liu et al., 2020), the behavior module can be formulated as: \( \hat{v}_q = UserBehavior(q, k_{1:T}, v_{1:T}) \), where \( k_{1:T} \) and \( v_{1:T} \) are given \( T \) historical clicked items and corresponding query words. The most used strategy is to adopt the the self-attention mechanism (Vaswani et al., 2017), which naturally learns multiplicative interaction between query and the historical behavior.

2.2 Formulation

Namely, multiplicative interaction (Jayakumar et al., 2020) has been proposed to fuse two different sources of information with the goal of approximating function \( f_{target}(x, z) \in \mathbb{R}^c \), where \( x \) and \( z \) are the input and context respectively. Similarly, we give a generic formulation of DPO in CTR prediction task as:

\[ y_i = \frac{1}{C(z)} \sum_{\forall j} f(x_i; g_i(z_j; \theta)) \]  

(3)

Here \( i \) is the index of a position (in the field, or sequence), whose response is calculated with the generated output of \( z \) over all existing positions. \( x \) is the input embedding, while \( z \) denotes any specified context. The generate function \( g \) aims to compute dynamic weights and bias followed as one of the inputs of the pairwise aggregate function \( f \), which learns the interactive features reflecting the relationship between \( x_i \) and \( z_j \). The output is finally normalized by a factor \( C(z) \).

MLP and convolution typically process input and context features in an additive way with fixed weights. While in [Eqn. (3)] using instance-wise generated weights and bias from contexts \( z \), the additive nature is transformed to multiplicative. DPO is also different from bilinear layer (Lin et al., 2013; He & Chu, 2017), for [Eqn. (3)] computes representation based on the generated weights over all positions, whereas bilinear layer aggregates information over all positions between \( x \) and \( z \), leading to large memory consumption. Furthermore, our generated dynamic weights can maintain more local information, which complements the global counterpart, e.g., self-attention. DPO is a flexible block and can easily work together with MLP and self-attention layers.

2.3 Feature-based DPO

Given \( x \in \mathbb{R}^m \) and \( z \in \mathbb{R}^n \) as inputs and context, due to the lack of position information, the generic formulation degrades as \( y = f(x; g(z; \theta)) \). For simplicity, we consider \( f \) in the form of a linear transformation: \( f(x; z) = W(z)x \), where \( W(z) \) is an instance-wise two-dimensional matrix generated by function \( g \). Now, We discuss the choice of function \( g \). Following the hypernetworks (Ha et al., 2016), a natural choice of \( g \) is a fully-connected layer to form dynamic weights and bias:

\[ y = (W_T z + b)_T x + (W_T z + b) = z^T \hat{W} x + \hat{W}_T z + \hat{b} \]  

(4)

where \((\hat{W}, \hat{b}, \hat{W}, \hat{b}) \in (\mathbb{R}^{n \times mc}, \mathbb{R}^{mc}, \mathbb{R}^{mc}, \mathbb{R}^c)\). However, the size of \( \hat{W} \) has quadratic space complexity, unsuitable for deployment in real-world application. Here, we consider a low-rank method in practice, e.g., a two-layer MLP:

\[ g(z) = \sigma(W_2^T z + b_1) + b_2 \]  

(5)
where \((W_1, b_1) \in (\mathbb{R}^{n \times l}, \mathbb{R}^l)\) and \((W_2, b_2) \in (\mathbb{R}^{l \times (mc+c)}, \mathbb{R}^{mc+c})\), \(\sigma\) is a non-linear function. Then, we can decompose the output into explicit dynamic weights and bias. The right inductive bias depends on how we select context \(z\) and \(g\). We denote the complexity of \(f\) is \(O(mc)\) less than \(O(mc + ln)\) of plain MLP layer, while \(g\) scales up to \(O(lmc + ln)\). To reduce the complexity, we set \(l\) as a small number and use a multi-head mechanism \((\text{Vaswani et al., 2017})\).

**Relation to Cross Network:** A cross layer \((\text{Wang et al., 2017})\) take the feature interaction formulation as \(x_{i+1} = x_0 \odot x_i + b_1 + x_i\), where \(x_i\) is scalar. We prove CrossLayer is the simplest formulation of DPO. Let’s take \(x_0\) as \(z\), \(x_i\) as \(x\) and only use 1-layer MLP as weight-generate function, whose hidden states are 1, i.e., \(z = x_0, W \in \mathbb{R}^n, \hat{W} = 0, \hat{B} = 1\). Thus, we get a scalar output of \(g\) as the same as the multiplicative term of CrossLayer. In this way, DPO aims to imitate multiplicative operation.

**2.4 Field-based Dynamic Parameterized Operation**

Given \(X \in \mathbb{R}^{t_1 \times m}\) and \(Z \in \mathbb{R}^{t_2 \times n}\) as inputs and context, where \(t_1\) and \(t_2\) represent the field numbers respectively, our goal is modeling the interaction between \(x_i\) and \(z_j\) over all field positions. A simple idea is to treat field-based operation as multiple feature-based operations followed by summation over all output. Thus, Eqn. (3) can be expressed as \(y_i = f(x_i; \sum_{j} g(z_j; \theta))\), which means all fields share the same instance-wise weights.

However, the field-based operation interacts between all fields, which sometimes introduces unnecessary feature coupling (i.e., multiplicative interaction of brand ID and time, etc.). The empirical evidence finds over-coupling brings more noise and then results in underfitting, albeit their capacity of learning high-order features. A considerable method is to use Self-Field dynamic operation without heavily hand-crafted feature engineering, formulated as: \(y_i = f(x_i; g_i(x_i; \theta))\) by removing cross-field interactions. Apart from Summation-based and Self-based methods, a more attentive solution can be used to aggregate the dynamic attributes: \(y_i = f(x_i; \sum_{j} h(z_j; w_j)g_j(z_j; \theta))\), where \(h\) is an attention layer. Beyond taking position into consideration, we can interact the whole context with inputs without explicit summation instead of concatenation, formulated as: \(y_i = f(x_i; g_i([x_1, z_2, ..., z_{f_2}]; \theta))\), where \([,]\) is a concatenation operation. These four methods learn pairwise field-based interaction from coarse to fine to model high-order representation, while the feature-based method combines both low- and high-order information over all fields. The complex weight-generate function can be designed for the right inductive bias, but we do not specifically consider such a method for online serving and leave it to future work.

**Relation to FM:** Here, we slightly modify the origin FM implementation \((\text{Rendle, 2010})\) as: \(y = \sum_{v_i \neq v_j} x_i^T x_j\) by removing the LR term, that takes interaction among all field positions into consideration. Given inputs and the context as \(x_i\) and \(\{x_j, \forall j \neq i\}\), the function \(f\) is simply matrix multiplication and \(g\) is the identity function, then Eqn. (3) can be decomposed to: \(y_i = \sum_{j \neq i} x_i^T x_j\). Thus, FM can be viewed as the self-excluded version of field-based dynamic operation, where the context is other field features different to the input features.

**2.5 Sequence-based Dynamic Parameterized Operation**

User behavior modeling focus on learning from their historical actions to predict whether the users click the current items. As a comparison, transformer-based solutions \((\text{Liu et al., 2020}; \text{Zhou et al., 2018a})\) explored the encoder-decoder framework to learn long-range dependencies both source-to-
source and source-to-target, where the encoder exploits multi-head self-attention to extract session interest and the decoder aggregates the query-specific interest. Following the encoder-decoder framework, we consider two variants, i.e., homogeneous Behavior-Behavior and heterogeneous Query-Behavior dynamic operation (homo- and hetero- DPO). We show their multiplicative attributes on Appendix C.

**Homogeneous Behavior-Behavior DPO** aims to capture feature interaction at different time-steps of behavior. Given $X = Z \in \mathbb{R}^{t \times n}$ as inputs and context, where $t$ represents the behavior length. For user behavior modeling, our goal is to model the long- and short-term feature interaction. As mentioned above, a long-term function aims to learn non-local interaction between all positions while short-term ones only care about the local information. Thus, a natural way is to adopt global-aware weight-generate function $g$ and local-aware feature-aggregation $f$. Different to Section 2.3 and Section 2.4 we adopt convolution as $f$ which is widely used for modeling local sequence information with learned weights. For simplicity, we consider function $f$ in the form of a 1D-convolution with kernel size $k$, while feature-based and field-based methods only use MLP.

$$y_i = f(x_i; \frac{1}{C(z)} \sum_{j} g(z_j; \theta)) = \frac{1}{C(x)} \sum_{t=-\frac{k}{2}}^{\frac{k}{2}} \sum_{j=0}^{t} g_i(x_j; \theta)x_{i+t}$$

$$\sum_{j=0}^{t} g_i(x_j; \theta) = W_{2,l}^T \sigma(W_{1,l}^T x + b_{1,l}) + b_{2,l}$$

*Eqn. (6)* shows the function $f$ can act as a convolution operation without bias term which models local neighborhood by dynamic weight, where $x_{i-t}$ is the extracted behavior in position $i$. *Eqn. (7)* gives a instantiation of weight-generate function $g$. Firstly, we aggregate all sequence information and project them into a select operator $s \in \mathbb{R}^d$ by $W_{1,l}$ and $b_{1,l}$, where $W_{1,l} \in \mathbb{R}^{s \times d}$, $b_{1,l} \in \mathbb{R}^d$ and $\sigma$ is activation function. Secondly, we use $s$ to explicit aggregate expert weight, where $W_{2,l} \in \mathbb{R}^{d \times (nc)}$ and $b_{2,l} \in \mathbb{R}^{nc}$. To use dynamic depthwise-convolution, we can set $c = 1$. *Eqn. (7)* captures the multiplicative interaction correspond to global aggregation features. To further strengthen locality, we can adopt local-aware function to capture short-term information of context $x$ (e.g. convolution, separable convolution etc.).

**Heterogeneous Query-Behavior DPO** aggregates all sequential behaviors as context targeting to interaction with query. Given $x \in \mathbb{R}^m$ and $Z \in \mathbb{R}^{t \times n}$, we take function $f$ as linear transformation as mentioned in Section 2.3. *Eqn. (3)* learns interaction between query and behavior over all length followed by summation, and the simplest formulation can be derived as:

$$y = f(x; \frac{1}{C(z)} \sum_{j \in t} g(z_j; \theta)) = g(\frac{1}{C(z)} \sum_{j \in t} z_j; \theta)^T x$$

Similarly to feature-based and field-based methods, query-behavior dynamic operation can easily learn rich multiplicative interaction and conditional inductive bias. The weight-generate function $g$ aims to learn the weight representation $W_{g} \in \mathbb{R}^{m \times c}$. Typically, we can exploit a specific aggregation function, such as *Eqn. (7)*. Compared to self-attention in decoder [Liu et al, 2020], DPO focuses attention on instance-weights based on context, while self-attention takes bipartite attention matrix to aggregate value units. Thus, we conjecture they are two orthogonal and complementary solutions.

**Figure 2:** Illustration of homogeneous behavior and heterogeneous query-behavior dynamic parameterized operation.
We perform comprehensive experiments on feature modeling and user behavior modeling of public and real-world production CTR prediction datasets.

3.1 EXPERIMENTS ON FEATURE MODELING

**Setting.** We evaluate with MovieLens-tag, Criteo, Avazu with following questions:

- How does DPN perform (effectiveness and efficiency) compared with other base models?
- How do different contexts and weight-generate functions influence the performance?
We use AUC and Logloss as metrics for public datasets. For all experiments, we evaluate the effectiveness of baseline models with the same training setting in AFN (Cheng et al., 2020) implemented by TensorFlow (Abadi et al., 2016). We adopt Adam (Kingma & Ba, 2014) as optimizer with best searched learning rate of a batch size 4096 for all models. We fix the embedding ranks as 10 across all datasets and use same deep neural network (e.g., 3 layers MLP of 400-400-400) with Batch Normalization and ReLU (Ioffe & Szegedy, 2015; Nair & Hinton, 2010) if without specifically noted. The details of our proposed methods are described in Appendix B.

Comparison with state-of-the-art results. Table 1 shows the results from AFN (Cheng et al., 2020) and our reimplemented results in the same setting. We note all these methods are single model without DNN components. First, our feature-based DPN consistently achieves better performance than other explicit interaction methods and also the implicit DNN baseline, which confirms the dynamic aspects contributes to implicit feature interaction. Additionally, when the train dataset and features get larger, the overwhelming margin get larger (e.g. 0.13% on MovieLens-tag, 0.23% on Avazu, 0.69% on Criteo), showing promising potential ability for applied in real-world production. Secondly, our field-based DPN perform better than the other explicit interaction module. We note field-based methods models the relation over different attributes (i.e., UserID, MovieID etc.) where low- and high-order information are captured in a totally different way. Specially, field-based DPN obtain additive module in parallel with multiplicative one while other high-order interaction methods follow an opposite stacked framework to learn the multiplicative features (Qu et al., 2016; Cheng et al., 2020; He & Chua, 2017).

Effectiveness of different instantiations of weight-generate function. Table 2a compares different types of a feature-based dynamic operation added to the DNN baseline (right after the embedding layer for replacing the fully-connected layer). After we search the best DNN baseline model, we replace a dynamic operation with the first fully-connected layer. We list the results of different weight-generate functions, where not all methods perform better than the baseline. We implement the hypernetworks-based idea (Ha et al., 2016) as HyperDense which slightly improve the baseline while add a big chunk of computation resulting for optimization difficulty. When we adopt our proposed simple and effective method as shown in Eqn. (5), gate mechanism can be exploited for better performance, which means mixture of kernels have better generality. Furthermore, we explore some approaches to reduce the complexity of $g$, such as Multi-Head mechanism, Matrix decomposition. However, they instead downgrade the performance even cannot compete on par with the baseline. We provide more ablation study on the Appendix D.

Multi-Layer Feature-based DPN with different contexts. Table 2b shows the results of deeper dynamic parameterized network with different context. We separately replace DPO with first, second, and all fully connected layers in 2-layer MLP. Table 2b shows that more feature-based DPO in general lead to better results regardless of context. We argue multiple feature-based operations can learn rich and high-order dynamic interaction by imitating MLP. High-order message can be processed with non-linear function layer by layer, which is hard to be found useful via multiplicative models.

In Table 2b, we also study the effectiveness of different context. We set the flattened inputs $x_0$ as the inputs of DPN and evaluate the performance of different contexts such as $x_0$ and $z_0$ where $x_0$ and $z_0$ is the flattened outputs of inputs and context embeddings respectively. We found they share similar results for most experiments while getting the best performance when we set the context as $z_t$ (i.e. use the tag information of context embeddings as context inputs). Under careful selection of hyperparameters, this best result mainly originates from the expert knowledge of MovieLens dataset and recommendation system. Meanwhile, it reveals a nice property of our methods: the intrinsic decoupling attribute can be more separably modeled. Interestingly, we find our methods improve results of the infrequent user on MovieLens datasets, as shown in Table 3. We believe the dynamic interaction can warm up the infrequent user embeddings, demonstrating the potential of our methods for the cold-start problem.

Effectiveness and Efficiency of Field-based DPNs. We design a family of field-based dynamic networks aiming to capture atomic relationships among fields’ features. Table 2c presents results of different aggregate function. We find that all dynamic operations with different context aggregate function-generate functions, where not all methods perform better than the baseline.
Table 4: Adapt sequence-based DPO (sDPO) on Transformer. We evaluate the effectiveness of combination of multi-head self-attention mechanism and sDPO.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Decoder</th>
<th>Homo</th>
<th>Heter</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSA</td>
<td>Homo</td>
<td></td>
<td></td>
<td>0.8718</td>
</tr>
<tr>
<td>MSA</td>
<td>Heter</td>
<td></td>
<td></td>
<td>0.8755</td>
</tr>
<tr>
<td></td>
<td>Homo</td>
<td></td>
<td></td>
<td>0.8728</td>
</tr>
<tr>
<td></td>
<td>Heter</td>
<td></td>
<td></td>
<td>0.8809</td>
</tr>
<tr>
<td></td>
<td>Homo</td>
<td></td>
<td></td>
<td>0.8731</td>
</tr>
<tr>
<td></td>
<td>Heter</td>
<td></td>
<td></td>
<td>0.8775</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8849</td>
</tr>
</tbody>
</table>

Table 5: Comparison with state-of-the-art on Amazon dataset for user behavior modeling. We record the mean AUC over 5 runs. We mainly compare our methods with a well-known attention mechanism.

<table>
<thead>
<tr>
<th>Model</th>
<th>All</th>
<th>New</th>
<th>inFreq</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN (Zhou et al., 2018)</td>
<td>0.8292</td>
<td>0.8029</td>
<td>0.7937</td>
</tr>
<tr>
<td>DIEN (Zhou et al., 2019)</td>
<td>0.8675</td>
<td>0.8457</td>
<td>0.8375</td>
</tr>
<tr>
<td>Trans (Vaswani et al., 2017)</td>
<td>0.8718</td>
<td>0.8522</td>
<td>0.8438</td>
</tr>
<tr>
<td>KFAtt (Liu et al., 2020)</td>
<td>0.8789</td>
<td>0.8578</td>
<td>0.8496</td>
</tr>
<tr>
<td>DIN + Heter</td>
<td><strong>0.8836</strong></td>
<td>0.8554</td>
<td><strong>0.8583</strong></td>
</tr>
<tr>
<td>DIEN + Heter</td>
<td>0.8693</td>
<td>0.8476</td>
<td>0.8414</td>
</tr>
<tr>
<td>Trans + Heter</td>
<td>0.8809</td>
<td>0.8608</td>
<td>0.8526</td>
</tr>
<tr>
<td>Trans + Homo</td>
<td>0.8728</td>
<td>0.8530</td>
<td>0.8440</td>
</tr>
<tr>
<td>sDPN</td>
<td><strong>0.8849</strong></td>
<td>0.8615</td>
<td>0.8590</td>
</tr>
</tbody>
</table>

Adaptation to Self-Attention. Table 4 presents us how sDPO performs when incorporated with self-attention mechanism. For homogeneous DPO, we find it performs slightly better than MHSA counterparts. When we use session-wise representation for user behavior modeling, self-attention can capture local information over handcraft scope of time designed by experts while narrow neighborhood interaction of convolution may not contribute to learning users’ attention due to the short session. For heterogeneous DPO, we find it can effectively facilitate the decoder counterpart no matter which encoder we adopt. Overall, the sequence-based DPN(sDPN) achieve best results than other combination, which shows the effectiveness of our propose homogeneous and heterogeneous DPO.

3.2 EXPERIMENTS ON USER BEHAVIOR MODELING

Experiment setting. We evaluate sequence-based DPN (sDPN) on Amazon Electronics Datasets. We only adopt AUC as metrics with the same training setting in KFAtt (Liu et al., 2020) implemented by TensorFlow. The task is to predict whether a user will write a review for the target item after reviewing historical items. We refer the readers to KFAtt (Liu et al., 2020) for more details.

Comparison with state-of-the-art. From Table 5 we can see that sequence-based DPN achieves the best performance than all state-of-the-arts on total situations, including the strong baseline KFAtt. When incorporating heterogeneous DPO into the attention mechanism, we find both DIN (Zhou et al., 2018) and DIEN (Zhou et al., 2019) perform better than the origin baseline where the armed DIN outperform all the base models on Amazon datasets by a large margin, which shows that heterogeneous DPO can effectively learn complementary representation which can benefit the attention mechanism.

Adaptation to Self-Attention. Table 4 presents us how sDPO performs when incorporated with self-attention mechanism. We may hypothesize that additional contexts can benefit the field features after having been processed for imitating linear transformation, containing multiplicative interaction between inputs and contexts. However, we find the time-consuming is worrying in the CPU machine when the dimensions of outputs are relatively large, making it venturesome to be applied on real-world production.

3.3 EXPERIMENTS ON REAL-WORLD PRODUCTION DATASET AND ONLINE A/B TESTING

We conduct all feature-based, field-based, and sequence-based experiments on the Real-world Production dataset. In the offline experiments, we observe the progressive improvement from modern rank models consisting of advanced user behavior and multi-modal features model in our advertising system. Table 6 shows our DyMLP and DyJoint get significant advancement compared to origin implementation. For feature interaction modeling, DyMLP shows better performance than the commonly used DNN component while only add a little extra cost. Despite we don’t explore the effectiveness of ensemble DPN models in a public dataset, Table 6 presents even one lightweight field-based DPO that can benefit the generality. To reduce the complexity, we use a multi-head mechanism in feature-based DPN which may influence the effectiveness.
Table 6: Results on Real-world Production dataset. We show the details of how to incorporate DPNs into online ads system in Appendix F.1. For feature modeling, we name DPNs as DyMLP. For both feature and user behavior modeling together with online model, we name DPNs as DyJoint.

<table>
<thead>
<tr>
<th>Module</th>
<th>MLP</th>
<th>Feature</th>
<th>Field</th>
<th>KFAAtt</th>
<th>Homo</th>
<th>Heter</th>
<th>Auc(+gain)</th>
<th>Throughput (batch/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>0.7523</td>
<td>101</td>
</tr>
<tr>
<td>DyMLP</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>0.7530(↑0.07)</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>0.7550(↑0.27)</td>
<td>88</td>
</tr>
<tr>
<td>Online</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>0.7598</td>
<td>55</td>
</tr>
<tr>
<td>DyJoint</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>0.7609(↑0.11)</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>0.7624(↑0.26)</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>0.7633(↑0.35)</td>
<td>46</td>
</tr>
</tbody>
</table>

For sequence-based DPO, we conduct more ablation studies in Appendix F.3. Our homogeneous DPO can act as a specific form like dynamic convolution. Incorporating it with a session-based self-attention encoder, we can inject inductive bias learned from local neighborhood information into global long-term dependencies based on Transformer-like models (Vaswani et al., 2017; Feng et al., 2019). Beyond that, heterogeneous DPO can learn more conditional multiplicative interaction which models the user interest on given items, showing greater power than the homogeneous component. Combined with all techniques, we get the best results by a large margin to the online model.

In the online A/B test, Table 7 shows DyJoint contributes 1.0% CTR gain against the online component, which demonstrates the superiority over the highly optimized base model on our ad system. However, DyJoint leads to larger online latency compared to the base model due to the increment of model parameters and memory access.

4 CONCLUSION

In this paper, we describe a new class of neural networks that captures both explicit and implicit interaction via dynamic parameterized operation. Our proposed block can be easily inserted into existing CTR predicting architectures for fusing features from different modalities. Our experiments show that it overwhelms the existing feature-crossing-based and attention-based models on two fundamental tasks. Furthermore, we confirm its representation effectiveness in the real-world production dataset. For the theoretical understanding, we decouple dynamic operation for comprehensive study with high-order feature-crossing methods and self-attention. Overall, we open a new era where current mainstream solutions are dominated by self-attention mechanisms and MLP in CTR prediction.
REFERENCES


Table 8: Statistics of datasets for feature modeling.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>instance</th>
<th>fields</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteo</td>
<td>4530240</td>
<td>39</td>
<td>2086936</td>
</tr>
<tr>
<td>Avazu</td>
<td>4042896</td>
<td>22</td>
<td>1544250</td>
</tr>
<tr>
<td>Movielens-tag</td>
<td>2006859</td>
<td>3</td>
<td>90445</td>
</tr>
<tr>
<td>Movielens-1M</td>
<td>739012</td>
<td>7</td>
<td>3529</td>
</tr>
<tr>
<td>Real-world Production</td>
<td>12 Billion</td>
<td>96</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 3: Illustration of stacked, parallel and ours paradigm for traditional CTR prediction.

A RELATED WORK

Feature Crossing as high-order interaction has been widely explored in CTR prediction and recommendation system by capture a cross feature which can be defined as $\prod_{i=0}^{n} x_i^{a_i}$. Factorization Machine (Rendle, 2010) was firstly proposed to calculate similarity by using inner product between two different features (e.g. item feature and user feature) borrowed from the collaborative-filtering based model (Sarwar et al., 2001). With the rise of deep neural network, recent works focus on combining feature crossing with Embedding+MLP paradigm for better performance (i.e. stacked and parallel paradigm) as shown in Fig. (3a) and Fig. (3b). Late fusion is mostly used in parallel paradigm. Wide and deep (Cheng et al., 2016) propose wide component to form order-2 features by cross-product transformation between sparse inputs and a deep layer for order-1 features. DCN (Wang et al., 2017) propose to use parallel structure composed by bit-wise cross-layer and deep neural network. DeepFM (Guo et al., 2017) adopts a parallel structure to fuse the FM and DNN outputs. To facilitating the high-order representation, xDeepFM (Lian et al., 2018) are proposed to generate vector-wise high-order feature, achieving further advancement by stacking multiple interaction layers. Contrast to polynomial networks, AutoInt (Song et al., 2019) exploit a multi-head self-attention layer to automatically learn the high-order feature interaction of all atom features (e.g. item id, user id, brand id etc.). Ensembled with DNN, AutoInt achieve better result. Constrast to parallel ensemble methods, stacked paradigm uses an explicit layer to extract cross features followed by an implicit layer to process them, i.e. DNN. NeuFM (He & Chua, 2017) uses Bi-Interaction layer with stacked deep neural network for bridging generalized linear models and deep learning. AFN (Cheng et al., 2020) models cross features of adaptive orders followed by 3-layer MLP. Despite the inherit strong generality incorporated with DNN, stack paradigm often resorts to further ensemble with DNN for better results.

However, due to either the low capacity or high latency of so-called interaction layer, those methods typically failed in real-world rank system. Recent application of polynomial networks on CTR prediction can be seen as a special case of multiplicative interaction to formulate scalar-wise, element-
wise or feature-wise multiplier fusion between projected embeddings. We prove that our methods can
degrade as the bit-wise cross-layer \cite{Wang2017} and pairwise interaction layer \cite{Rendle2010}.

**User behavior modeling** aims to extract users’ dynamic and evolving interest over items. Earlier
works \cite{Covington2016,Song2016,Yu2016} use a target-independent manner to aggregate total behaviors, failed to learn the interaction between target and past behaviors. Recent
works \cite{Zhou2019,Zhou2018,Zhou2018b,Liu2020} achieve massive improvement by adopting
attention mechanism for modeling long-range representation between user historical behaviors and
other target features. Orthogonally, our proposed methods can be formulated as multiplicative
interaction between two different streams.

**Dynamic Neural Networks** enjoy preferable properties than static ones, whose computation graph
and forward parameters are fixed with limited representation power \cite{Han2021}. Instance-wise
dynamic neural networks typically have dynamic structure \cite{Tanno2019}, dynamic weights
\cite{Yang2019,Ha2016,Chen2020}, or dynamic layers \cite{Huang2018} in
inference time. Jayakumar \cite{Jayakumar2020} provide theoretical comprehension about why
dynamic weights are beneficial: HyperNetworks \cite{Ha2016} can be thought of generalised
multiplicative interaction in the non-factorised sense, whose filters are generated by conditional
inputs.

**B IMPLEMENTATION DETAILS OF DPN**

**B.1 DATASETS AND EXPERIMENTAL PROTOCOLS**

**Public dataset for feature-interaction modeling.** We conduct experiments on three public real-
world datasets in Section 3.1 as shown in Table 8. MovieLens-tag\(^2\) has been prepared into (user,
movie, tag) format for personalized recommendation. Totally, there are 2006859 tuples, constituting
of 3 categorical fields: UserID, MovieID and tag. Avazu\(^3\) is click-through rate dataset including 22
feature fields of user features and advertisement attributes. There are about 40 million instances in
total. Criteo\(^4\) is a benchmark dataset for displayed ads CTR prediction, containing 26 categorical
fields and 13 numeric fields. It has about 45 million user-click records on displayed ads. We split the
dataset in 7:2:1 for training, validation and testing respectively.

**Public dataset for user behavior modeling** We conduct sequence modeling on Amazon book
dataset \cite{McAuley2015} to learn users’ interest. We use the 5-core Electronics subset, including
1689188 instances with 192403 users and 63001 goods from 801 categories. The task is to predict
the intent of writing users’ review for a target item regard to the historical reviews. In practice, we
use the last review of each user as test split while others as train split. The negative instances are
randomly sampled from not-reviewed items.

**Real-world Production dataset** is a daily traffic log generated from the search advertising system.
Typically, we sample negatively a first 32-days click-through logs for training data containing 2.4
billion samples and non-negatively the next-day click-through logs for test data containing about 1
million samples. For the user-behavior data setting, we follow the KFAtt \cite{Liu2020} to choose
user clicks/queries in previous 70 days as behavior \(v/k\). The other features consist of the query, user
and ad profile, ad image and contexts, built for modeling feature interaction.

**Metrics:** AUC is almost the default offline evaluation metrics in advertising system due to the direct
relation to online performance. Thus, we use AUC as a evaluation metric for evaluation on both
public dataset and real-world production dataset. For fair comparison, we additionally add Logloss as
another metric when evaluated on public dataset. We denote that 0.001-level improvement of AUC
on CTR prediction task is significant, which has been a common sense in previous works \cite{Cheng2016,Guo2017,Lian2018,Wang2020,Wang2017}.

\(^2\)https://grouplens.org/datasets/movielens/
\(^3\)https://www.kaggle.com/c/avazu-ctr-prediction
\(^4\)https://www.kaggle.com/c/criteo-display-ad-challenge
B.2 Embedding Layer

For all ctr models, the categorical features and dense features (UserID, ItemID, User Age etc) are hashed to sparse one-hot features as the inputs of embedding layers. We project i-th one-hot feature from sparse high-dimensional space to a lower one via $x_{emb,i} = W_i e_i$, where $e_i \in \{0, 1\}^v$ and $W \in \mathbb{R}^{e \times v}$; $v$ and $e$ means vocab and embedding size respectively. Thus, for feature-based modeling, we take the output of the concatenation of all the embedding vectors: $x_0 = [x_{emb,0}, x_{emb,1}, \ldots, x_{emb,n}]$ as our named features. For field-based modeling, we instead take the embedding vectors of every sparse ID as fields aiming to learning the interaction of pairwise features (e.g. [UserID, ItemID], [UserID, UserAge]).

B.3 Feature-Based and Field-Based Modeling

As mentioned in Eqn. (3), all of our methods induce an extra information features $z$ for boosting generalization. However, we don’t use extra fields as show in Table 8 which represents the effectiveness comes from the combination of fully-connected layer and feature-crossing module. For feature-based and field-based DPN, we stack several feature- and field-based DPO layer followed by classifier layer to build them. The implicit interaction module of feature-based DPN is fully-connected layer with non-linear function, while the implicit part of field-based DPN has two parts: field-interaction module $W_f$ and linear transformation module $W_l$, where $W_f \in \mathbb{R}^{n \times n}$ and $W_l \in \mathbb{R}^{e_l-1 \times e_l}$. Thus the static outputs can be calculate by $W_f X_0 W_l$, where $X_0 \in \mathbb{R}^{n \times e_l-1}$. We name it as Field-DNN.

B.4 User Behavior Modeling on Amazon Dataset

To evaluate the effectiveness of adapting sDPO on various methods, we adopt different strategy to combine sDPO with DIN, DIEN and Transformer. For DIN and DIEN, we produce the heterogeneous DPO outputs followed by element-wise summation with origin attention mechanism outputs. For Transformer, we sum the homogeneous and heterogeneous DPO outputs and self attention outputs in encoder and decoder respectively. Except for the increment benefit over other methods, we test the effectiveness only using DPO, i.e. sequence-based Dynamic Parameterized Network (sDPN). The sDPN simply replaces the self attention counterparts on Transformer framework by DPO layers.

B.5 User Behavior Modeling for Real-world Production Dataset

We follow the setting of KFAtt [Liu et al., 2020] along with the feature-based methods and field-based methods, evaluated only on real-world production dataset. There, 96 multi-modal features are first embedded into 16-dimension vectors and then fed for field-based and feature dynamic operation. The origin implement make use of 4-layer MLP with dimension 1024, 512, 256, 1. When there is a 30 minutes’ time interval between adjacent behaviors, we conduct a session segmentation. For each instance, we use at most 10 sessions and 25 behaviors per session. The learnt hidden user interest, $\hat{v}_q \in \mathbb{R}^{64}$ is concatenated to the output of 1st FC layer together with a 150-dimensional visual feature vector. Our homogeneous behavior-behavior operation are used to fuse the session-based self-attention encoder output in additive way with kernel size = 3. Our heterogeneous query-behavior operation are used to fuse the predicted interest produced by cross-attention. All the weight generate function are a SE-layer [Hu et al., 2018] whose intermediate down ratio is grid searched from [0.125, 0.25, 0.5].
C ANALYSIS MULTIPLICATIVE ATTRIBUTE OF SDPO

C.1 ANALYSIS OF HOMOGENEOUS DPO

\[ y_i = f(x_i; \frac{1}{C(z)} \sum_{j \in t} g(z_j; \theta)) \]

\[ = \frac{1}{C(x)} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} \sum_{j=0}^{t} g_{l,w}(x_j; \theta)x_{i+l} + \sum_{j=0}^{t} g_{l,b}(x_j; \theta) \]

\[ = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} \left( W_{2,l}^T (W_{1,l}^T \sum_{j=0}^{t} x_j + b_{1,l}) + b_{2,l} \right)^T x_{i+l} + g_b(\hat{x}) \]  

\[ = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} (\sum_{j=0}^{t} x_j W_{1,l} W_{2,l} x_{i+l} + W_{s,l}^T x_{i+l}) + g_b(\hat{x}) \quad (9) \]

As shown in Eqn. (9), we show the homogeneous DPO actually learns both multiplicative and additive relation among local neighbor and each behavior features followed by summation aggregate function when we don’t use non-linear function. To further reduce complexity, a common way is to learn the feature interaction between global embedding aggregated firstly and local neighbor, formulated as:

\[ y_i = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} (W_{2,l}^T \sigma(W_{1,l}^T \sum_{j=0}^{t} x_j + b_{1,l}) + b_{2,l})^T x_{i+l} \quad (10) \]

where \( \sigma \) can be sigmoid and softmax function.

C.2 ANALYSIS OF HETEROGENEOUS DPO

\[ y = f(x; \frac{1}{C(z)} \sum_{j \in t} g(z_j; \theta)) \]

\[ = g_w(\frac{1}{C(z)} \sum_{j \in t} z_j; \theta)^T x + g_b(\frac{1}{C(z)} \sum_{j \in t} z_j; \theta) \]

\[ = (W_2^T (W_1^T \sum_{j=0}^{t} z_j + b_1) + b_2)^T x + g_b(\hat{z}) \]  

\[ = (W_2^T (W_1^T \hat{z}))^T x + (W_2^T b_1 + b_2)^T x + g_b(\hat{z}) \]

\[ = \hat{z}^T W_1 W_2 x + W_s^T x \quad \text{query-whole} = \frac{1}{C(z)} \sum_{j=0}^{t} (z_j^T W_1 W_2 x) + W_s^T x + g_b(\hat{z}) \quad \text{query-each} \]

As shown in Eqn. (11) We decompose the simplest formulation of Eqn. (8) where \( W_1 \in \mathbb{R}^{d \times n}, W_2 \in \mathbb{R}^{n \times (m+c)}, b_1 \in \mathbb{R}^n, b_2 \in \mathbb{R}^{m+c}; x \in \mathbb{R}^m, z \in \mathbb{R}^n \). The equation omits the reshape operation. We can see that Heterogeneous DPO captures multiplicative and additive between each behavior and query different to the self attention mechanism. However, Eqn. (11) can’t be enhanced by non-linear function. Empirically, a better way is to learn the relation of the embedding representing for the whole behavior sequence and query features, which further reduce the computations and also have strong compatibility with complex weight-generation function \( g \) such as SE-layer [Hu et al., 2018].

Under review as a conference paper at ICLR 2022
We perform ablations on variants of feature-based and field-based DPN. Unless specified otherwise, which demonstrate the robustness of the method. However, the atomic field-based DPN still lack the capacity to compete on par with MLP baseline. Furthermore, we find the atomic fields’ interaction is hard to inert into feature-based DPO due to the extra fields dimension.

Varying depth and width for feature-based and field-based DPN [Table 9] presents the impact of different depth and width on performance. Our experiments indicate that Feature-based DPN can achieve better performance with similar time consuming over a wide range hyperparameters, which demonstrate the robustness of the method. However, the atomic field-based DPN still lack the capacity to compete on par with MLP baseline. Furthermore, we find the atomic fields’ interaction is hard to inert into feature-based DPO due to the extra fields dimension.

MD, MH vs Plain [Table 10] presents the performance of matrix decomposition and multi-head mechanism on feature-based DPN. Despite their success when applied in other domains (Vaswani et al., 2017; Li et al., 2021; Wu et al., 2019), we find these methods don’t work in traditional CTR prediction. We conjecture the multi-head mechanism hurts the expressivity of fully-connected layer due to the separable interaction among channels. For matrix decomposition, we aim to generate instance-wise kernels by \( P\phi(z)Q \), where \( P \in \mathbb{R}^{n \times r} \), \( \phi(z) \in \mathbb{R}^{r \times r} \), \( Q \in \mathbb{R}^{r \times m} \). We set \( r = 32 \) in [Table 10]. Thus, we may solve the low-rank completion problem for better performance. We conjecture simply utilization of dynamic parameterized fully-connected layer by matrix decomposition leads to optimization difficulty. We leave it into future study.

D ADDITIONAL EXPERIMENTS ON MOVIELENS-TAG

D.1 ABLATION STUDY

We perform ablations on variants of feature-based and field-based DPN. Unless specified otherwise, all experimental results in this sections report MovieLens AUC and logloss by training a DPN architecture that replace weighted fully-connected layer.

Table 9: Comparison of different depth and width over 5-runs results on MovieLens-tag.

<table>
<thead>
<tr>
<th>Model</th>
<th>Head</th>
<th>Auc</th>
<th>Logloss</th>
<th>Auc(infreq)</th>
<th>Logloss(infreq)</th>
<th>Params</th>
<th>CPU Time/epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN, 100-100</td>
<td>0.9498 ± 6e-4</td>
<td>0.2607 ± 3e-3</td>
<td>0.9358 ± 5e-4</td>
<td>0.2986 ± 3e-4</td>
<td>13k</td>
<td>35s</td>
<td></td>
</tr>
<tr>
<td>DNN, 100-100-100</td>
<td>0.9468 ± 4e-4</td>
<td>0.2648 ± 1e-3</td>
<td>0.9524 ± 9e-4</td>
<td>0.2983 ± 2e-4</td>
<td>24k</td>
<td>37s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 100-100</td>
<td>0.9516 ± 3e-4</td>
<td>0.2581 ± 4e-3</td>
<td>0.9389 ± 8e-4</td>
<td>0.2951 ± 1e-2</td>
<td>53k</td>
<td>39s</td>
<td></td>
</tr>
<tr>
<td>DNN, 200-200</td>
<td>0.9507 ± 9e-4</td>
<td>0.2575 ± 2e-3</td>
<td>0.9370 ± 1e-3</td>
<td>0.2945 ± 4e-3</td>
<td>47k</td>
<td>38s</td>
<td></td>
</tr>
<tr>
<td>DNN, 200-200-200</td>
<td>0.9476 ± 5e-4</td>
<td>0.2633 ± 2e-3</td>
<td>0.9336 ± 1e-3</td>
<td>0.2962 ± 4e-3</td>
<td>8k</td>
<td>41s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 200-200</td>
<td>0.9533 ± 9e-4</td>
<td>0.2527 ± 8e-4</td>
<td>0.9412 ± 9e-4</td>
<td>0.2897 ± 2e-3</td>
<td>187k</td>
<td>48s</td>
<td></td>
</tr>
<tr>
<td>DNN, 64-64</td>
<td>0.9487 ± 2e-4</td>
<td>0.2619 ± 1e-3</td>
<td>0.9341 ± 1e-4</td>
<td>0.2984 ± 3e-3</td>
<td>6k</td>
<td>35s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 64-64</td>
<td>0.9502 ± 1e-3</td>
<td>0.2593 ± 3e-3</td>
<td>0.9370 ± 2e-3</td>
<td>0.2959 ± 5e-3</td>
<td>25k</td>
<td>37s</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Variants of Feature-based DPN over 5-runs results on MovieLens-tag.

<table>
<thead>
<tr>
<th>Model</th>
<th>Head</th>
<th>Auc</th>
<th>Logloss</th>
<th>Auc(infreq)</th>
<th>Logloss(infreq)</th>
<th>Params</th>
<th>CPU Time/epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN, 200-200</td>
<td>0.9507 ± 9e-4</td>
<td>0.2575 ± 2e-3</td>
<td>0.9370 ± 1e-3</td>
<td>0.2945 ± 4e-3</td>
<td>47k</td>
<td>38s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 100-100</td>
<td>0.9516 ± 3e-4</td>
<td>0.2581 ± 4e-3</td>
<td>0.9389 ± 8e-4</td>
<td>0.2951 ± 1e-2</td>
<td>53k</td>
<td>39s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 100-200</td>
<td>0.9498 ± 4e-4</td>
<td>0.2618 ± 2e-3</td>
<td>0.9379 ± 1e-3</td>
<td>0.2913 ± 2e-3</td>
<td>44k</td>
<td>43s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 100-100</td>
<td>1-1</td>
<td>0.9513 ± 5e-4</td>
<td>0.2531 ± 2e-3</td>
<td>0.9382 ± 5e-4</td>
<td>23k</td>
<td>40s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 100-200</td>
<td>1-2</td>
<td>0.9511 ± 7e-4</td>
<td>0.2591 ± 2e-3</td>
<td>0.9385 ± 6e-4</td>
<td>34k</td>
<td>42s</td>
<td></td>
</tr>
<tr>
<td>Fe DPN, MoK, 200-200</td>
<td>2-2</td>
<td>0.9479 ± 9e-4</td>
<td>0.2657 ± 2e-3</td>
<td>0.9353 ± 1e-3</td>
<td>2974 ± 4e-3</td>
<td>14k</td>
<td>39s</td>
</tr>
<tr>
<td>Fe DPN, MoK, 200-200</td>
<td>2-2</td>
<td>0.9491 ± 1e-3</td>
<td>0.2719 ± 1e-2</td>
<td>0.9380 ± 1e-3</td>
<td>2994 ± 1e-2</td>
<td>48k</td>
<td>43s</td>
</tr>
<tr>
<td>Fe DPN, MD, 100-100</td>
<td>1-1</td>
<td>0.9467 ± 1e-3</td>
<td>0.2708 ± 4e-3</td>
<td>0.9330 ± 1e-3</td>
<td>3007 ± 5e-3</td>
<td>20k</td>
<td>41s</td>
</tr>
<tr>
<td>Fe DPN, MD, 200-200</td>
<td>1-1</td>
<td>0.9494 ± 5e-4</td>
<td>0.2628 ± 2e-3</td>
<td>0.9362 ± 6e-4</td>
<td>2975 ± 4e-4</td>
<td>30k</td>
<td>46s</td>
</tr>
</tbody>
</table>

E ADDITIONAL EXPERIMENTS ON MOVIELENS-1M

Besides experiments on three public datasets, We extraly conduct feature-based and field-based ablation studies on widely-used MovieLens-1M.
Table 11: Hyper-parameters of Dynamic Parameterized Network in MovieLens-1M.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>feature-based</th>
<th>field-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td>MovieID, zipcode, Age, Occupation, Gender, Timestamp</td>
<td></td>
</tr>
<tr>
<td>embedding size</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>hidden size</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>$g$</td>
<td>2-layer MLP</td>
<td>2-layer MLP</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>sigmoid</td>
<td>sigmoid</td>
</tr>
<tr>
<td>residual</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>LN</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>non-linear</td>
<td>ReLU</td>
<td>ReLU</td>
</tr>
<tr>
<td>initializer</td>
<td>glorot</td>
<td>glorot</td>
</tr>
<tr>
<td>context</td>
<td>$x_{l-1}$</td>
<td>$x_{l-1}$</td>
</tr>
<tr>
<td>Layers</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Heads</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Learning rate</td>
<td>3e-4</td>
<td>1e-3</td>
</tr>
<tr>
<td>Adam $\epsilon$</td>
<td>1e-8</td>
<td>1e-8</td>
</tr>
<tr>
<td>Adam $\beta_1$</td>
<td>0.09</td>
<td>0.9</td>
</tr>
<tr>
<td>Adam $\beta_2$</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>Batch size</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Params</td>
<td>86k</td>
<td>30k</td>
</tr>
</tbody>
</table>

**MovieLens-1M** is a relative small dataset [Harper & Konstan, 2015], contains users’ rating on movies with 7 features, e.g. user features and movie features and rating. We follow the experiment setting of AutoInt [Song et al., 2019], which randomly split 80% of dataset into train data, 10% into valid data and the rest 10% into test data. And we set the ratings below 1s and 2s as 0s, above 4s and 5s as 1s. All rating 3s should be removed.

**Training** For ablation study on MovieLens-1M, our baseline models follow the settings in AutoInt [Song et al., 2019] with random initialized weights, implemented by Tensorflow [Abadi et al., 2016] with the tricks [Srivastava et al., 2014; Nair & Hinton, 2010] but we don’t use BN [Ioffe & Szegedy, 2015] due to the poor performance. The feature-based model take the same setting as DeepCrossing (Shan et al., 2016) introduced in AutoInt with 4-layer MLP of size 100. The function $g$ is 2-layer MLP. The field-based models use three dynamic layer and set hidden states as 64 the same as their public code [5], (e.g. replace self-attentive layer with our field-based dynamic operation). After that, we use grid search to select hyperparameters, (i.e. dropout rate, learning rate). For fair comparison, all methods set embedding dimension as 16 and use Adam [Kingma & Ba, 2014] as default optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size is 1024. However, we only replace feature-based operation on the final two layers before classifier, due to the memory limitation and insert 1-layer field-based operation firstly.

**Comparison with state-of-the-art** Table 12 shows the results from AutoInt [Song et al., 2019]. Due to the random data splits, we rerun their experiments for comparison. We note the 0.001-level fluctuation on evaluation metric compared to the cited’ ones is acceptable when trained and evaluated on randomly split setting. Nevertheless, our feature-based method surpasses the other existing high-order based methods [Wang et al., 2017; Lian et al., 2018; He & Chua, 2017] by a good margin, even the proposed field-based operation by ourselves. However, what surprise us is the heavily tuned DNN model still excels high-order parts in contrast to the results reported in the AutoInt. We infer that MLP-based DNN model can be a good universal approximate function to attain a suitable local minimum with good hyperparameters. In this way, there is no doubt feature-based solution is better than others, because of its capacity for learning both implicit and explicit feature interaction. As far as we know, our proposed method is state-of-the-art solution in the same setting, even without the DNN component.

https://github.com/DeepGraphLearning/RecommenderSystems/tree/master/featureRec
Table 12: Comparison with different algorithms over 10-runs results. Std≈1e-3. ↓ means below while ↑ means above. Throughput means training time of one epoch. We also cite the results from AutoInt (Song et al., 2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>Auc</th>
<th>Logloss</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP(ours)</td>
<td>0.8475</td>
<td>0.3785</td>
<td>53k</td>
</tr>
<tr>
<td>DeepCrossing</td>
<td>0.8448</td>
<td>0.3814</td>
<td>N/A</td>
</tr>
<tr>
<td>AutoInt(ours)</td>
<td>0.8448</td>
<td>0.3812</td>
<td>39k</td>
</tr>
<tr>
<td>NFM</td>
<td>0.8357</td>
<td>0.3883</td>
<td>N/A</td>
</tr>
<tr>
<td>CrossNet</td>
<td>0.7968</td>
<td>0.4266</td>
<td>N/A</td>
</tr>
<tr>
<td>CIN</td>
<td>0.8286</td>
<td>0.4108</td>
<td>N/A</td>
</tr>
<tr>
<td>Feature-based</td>
<td><strong>0.8522</strong></td>
<td><strong>0.3726</strong></td>
<td>86k</td>
</tr>
<tr>
<td>Field-based</td>
<td>0.8452</td>
<td>0.3806</td>
<td>30k</td>
</tr>
</tbody>
</table>

Table 13: Ablation study of feature-based dynamic operation

(a) Comparison with different head number. \(l=4\), \(\sigma\) is sigmoid, context is genre.

<table>
<thead>
<tr>
<th>head</th>
<th>Auc</th>
<th>Logloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8483</td>
<td>0.3781</td>
</tr>
<tr>
<td>2</td>
<td>0.8504</td>
<td>0.3738</td>
</tr>
<tr>
<td>4</td>
<td>0.8136</td>
<td>0.4126</td>
</tr>
</tbody>
</table>

(b) Comparison with non-linear function. \(l=4, h=2\), context is genre.

<table>
<thead>
<tr>
<th>(\sigma)</th>
<th>Auc</th>
<th>Logloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity</td>
<td>0.8494</td>
<td>0.3781</td>
</tr>
<tr>
<td>sigmoid</td>
<td>0.8504</td>
<td>0.3738</td>
</tr>
<tr>
<td>softmax</td>
<td>0.8506</td>
<td>0.3745</td>
</tr>
</tbody>
</table>

(c) Comparison with different context. \(l=8, h=2\), \(\sigma\) is softmax.

<table>
<thead>
<tr>
<th>context</th>
<th>Auc</th>
<th>Logloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>genre</td>
<td>0.8511</td>
<td>0.3741</td>
</tr>
<tr>
<td>gender</td>
<td>0.8457</td>
<td>0.3782</td>
</tr>
<tr>
<td>genre, gender</td>
<td>0.8452</td>
<td>0.3800</td>
</tr>
<tr>
<td>all</td>
<td><strong>0.8522</strong></td>
<td><strong>0.3726</strong></td>
</tr>
</tbody>
</table>

Comparison with different heads in feature-based operation [Table 13a] compares different head numbers of 4 dynamic parameterized layers with context \(z\) as genre. We find a good head number can effectively improve the performance, while larger lead to bad results. We derive this large gap is mainly due to the reason our baseline model use relative smaller embedding size and hidden state with increasing head numbers. When head numbers equal to 4, the size of total parameters is about 11k, which seriously downgrades the performance. However, a properly head number trades off the extent of implicit and explicit high-order interaction, inducing larger variance of selection on dynamic weights/bias.

Comparison with different non-linear function in feature-based operation [Table 13b] shows the performance of different non-linear function in Eqn. (5). Here, we set \(l=4\) and head numbers equal to 2, for simplicity. We find even using plant low-rank approximation without non-linear function, the result still better than high-order part reported in Table 1. This gives evidence that non-restrict multiplicative interaction among hidden states can be a complementary component for further improving the capacity of MLP. When using non-linear function sigmoid and softmax, we found the performance moves a significant step about 0.001-level. Constrained non-linear function controls the outputs of selection layer between \((0, 1)\), which can be seen as the attention over kernels (e.g. mixture of expert layers). We note that the best non-linear function depends on the dataset respectively. Such methods for kernels aggregation are friendly for optimization, allowing us to learn diversity of weights while consume nearly same time as plain fully-connected layer.

Comparison with different context in feature-based operation [Table 13c] compares the different context of feature-based dynamic operation. We can see that different combination of fields as context \(z\) will lead to variant results, where the worst only improve slightly than AutoInt, but less than DNN model. Interestingly, we select separately the user feature gender and movie feature genre as context. The gender typically consist two classes of male and female. Thus, the multiplicative interaction can be understood as how the gender influences whether a user rates a movie. Contrast to human intuition on gender, the results reveal that gender is not main reason for different rating on movie, e.g. explicit modeling gender interaction with other features leads to unnecessary bias. However, when modeling with genre of movies, we get large improvement on performance compared to the genders’. We conjecture the genre plays important role on rating of users. Moreover, when combined with all field features, we can achieve best results, representing the given features can be classified effectively in high dimension.
Table 14: Ablation study of field-based dynamic operation

(a) Comparison with different aggregation methods.

<table>
<thead>
<tr>
<th>Agg</th>
<th>Auc</th>
<th>Logloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summation</td>
<td>0.8372</td>
<td>0.3868</td>
</tr>
<tr>
<td>Self</td>
<td>0.7917</td>
<td>0.4264</td>
</tr>
<tr>
<td>Attention</td>
<td>0.8452</td>
<td>0.3806</td>
</tr>
<tr>
<td>Concat</td>
<td>0.8443</td>
<td>0.3808</td>
</tr>
</tbody>
</table>

(b) Comparison with different combination methods of field-based and feature-based operation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>structure</th>
<th>Auc</th>
<th>Logloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self + Feature</td>
<td>stacked</td>
<td>0.8471</td>
<td>0.3787</td>
</tr>
<tr>
<td>Self (One layer) + Feature</td>
<td>stacked</td>
<td>0.8520</td>
<td>0.3738</td>
</tr>
<tr>
<td>Self + Feature</td>
<td>parallel</td>
<td>0.8532</td>
<td>0.3725</td>
</tr>
<tr>
<td>Weighted Sum + Feature</td>
<td>stacked</td>
<td>0.8527</td>
<td>0.3746</td>
</tr>
<tr>
<td>Weighted Sum + Feature</td>
<td>parallel</td>
<td>0.8536</td>
<td>0.3729</td>
</tr>
</tbody>
</table>

Comparison with different aggregation function in field-based operation

The feature-based methods give empirical evidence that how to select head numbers, non-linear function and context. For simplicity, we choose the same best setting in ablation study of field-base dynamic operation. Table 14a compares four methods of how to aggregate different field information as mentioned in Section 2.4. When only using the field to correspond itself, we find it perform worst due to the scarcity of field-wise feature interaction compared to the simple summation version. When using a weighted-sum layer and concatenation-MLP layer, we find such methods improve almost 0.005-level compared to summation-based solution. The main difference is using a learned linear combination of all field features, which implicit models low-order feature interaction. Thus, we can conclude implicit linear transformation overall fields is beneficial for modeling high-order representation. However, we notice the DNN result overwhelm the field-wise operations’, meaning the multiplicative interaction brought by high-order component is not the main course in CTR prediction.

Does field-wise component contribute to feature-wise component?

Despite the low capacity of field-wise dynamic operation, we explore the contribution when combining field-wise component with feature-wise’s. Table 14b shows there are no obvious benefit when we use both operation at same time no matter how we stack them, while facilitating the performance when composed of parallel structure. The stack structure directly exploit the field-based features as the inputs of feature-based layers. Once the fundamental representation has low diversity and expressivity, subsequent fancy modules can’t perform their proper ability, e.g. three layers stacked Self-based operation dominates the effectiveness. However, we find the aggregation methods of weighted summation can effectively boost the performance sightly no matter which structure we used. Thus, we confirm when combined those two methods in a suitable way, field-wise component can contribute to the feature-wise component for further advancement of performance.

F Additional experiments on real-world production dataset for user behavior modeling

F.1 Incorporating dynamic operation into online ranking system

In previous section, we mainly discuss the dynamic mechanism among different inputs and context. However, real-world industry system scales behavior-based and feature-based modeling containing many techniques to precisely extract user interest and maintain low latency at the same time. We now introduce the whole behavior modeling module and feature modeling module deployed in our CTR prediction system, which consists of three parts: a MLP-based deep neural network, a transformer-based (Vaswani et al., 2017) encoder aims to learn long-range dependency of session-based behaviors, and a KFAtt-based (Liu et al., 2020) decoder to predict user interest along with total user actions. We term the encoder-decoder part as DyTrans, the MLP part as DyMLP and the whole module as DyJoint, followed by:

- Replace MLP with feature-based dynamic operation
- Insert field-based dynamic operation before MLP
- Combine Behavior-Behavior dynamic operation with Self-Attention
- Combine Query-Behavior dynamic operation with KFAtt
Table 15: Experiment Results on Real-world Production dataset

<table>
<thead>
<tr>
<th>DyMLP</th>
<th>SelfAtt</th>
<th>KFAtt</th>
<th>Encoder (k = 3)</th>
<th>Decoder</th>
<th>Auc</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Sep + SE + Homo</td>
<td>Sep + SE + Heter</td>
<td>0.7633</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Sep + SE + Homo, k = 5</td>
<td>Sep + SE + Heter</td>
<td>0.7626</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>SE + Homo</td>
<td>SE + Heter</td>
<td>0.7622</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Conv1D</td>
<td>Sep + SE + Heter</td>
<td>0.7624</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Conv1D, k = 5</td>
<td>Sep + SE + Heter</td>
<td>0.7624</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Sep + SE + Homo</td>
<td>Sep + SE + Heter</td>
<td>0.7617</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Self + MLPs + Homo, k=1</td>
<td>Heter</td>
<td>0.7604</td>
</tr>
</tbody>
</table>

F.2 IMPLEMENT DETAILS

Encoder: Within Session Interest Extractor KFAtt (Liu et al., 2020) mainly adopt the multi-head self-attention used in Transformer (Vaswani et al., 2017). While this self-attention is only conduct in session-based behavior for efficiency. Typically, we divide the behavior sequence into Sessions according to their occurring time. The self-attention can be formulated as:

$$\text{MultiHead}(K_s, K_s, V_s) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

(12)

$$\text{head}_i = \text{Attention}(K_sW_i^Q, K_sW_i^K, V_sW_i^V) = \text{softmax}(K_sW_i^QW_i^K^T K_s^T / \sqrt{d_k})V_sW_i^V$$

(13)

Incorporated our proposed methods, the module can be formulated as:

$$\text{MultiHead}(K_s, K_s, V_s) + \text{DynamicOp}(K_{s,k}, g_s(K_s))$$

(14)

Decoder: Query-specific Interest Aggregator KFAtt acts as the decoder to aggregate interest from all sessions for query-specific interest prediction. Incorporated with query-behavior decoder, it can be formulated as:

$$v^q = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O + \text{DynamicOp}(\text{query}, g(K))$$

(15)

F.3 ABLATION STUDY

Effect of varying dynamic kernels in encoder In session-based behavior modeling, we investigate the kernel size of homogeneous behavior operation which influences the temporal receptive fields. The larger kernel size means the larger locality information. Table 15 shows when we use larger dynamic kernel size, there exists marginally performance downgrading. The main reason is the session-based behaviors are not longer than 10, i.e. the locality has been learned by self-attention (Vaswani et al., 2017).

Effect of convolution generation function Table 15 investigates the benefits of convolution-based weight-generate function. The separated convolution with k=3 shows further advancement in user behavior modeling. In this experiment, the weight-generate function and aggregation function both captures neighborhood representation, in order to learn short-term behaviors to short-term behaviors relationships, besides the dimension-behavior association.

Effect of dynamic compared to static Table 15 also shows dynamic convolution with plain static convolution. As we can see, the dynamic operation generalize better than static one. But the locality brought by convolution still works in self-attention mechanism. When we decrease the kernel size as 1 without locality information and global context, we find it hardly improve the performance. We conjecture that such method can act as plain MoE over time-steps, demonstrating interaction among neighbor behaviors facilitates global connected features while features based on behaviors themselves only highlight the diagonal of attention matrix.