ContextNet: A Click-Through Rate Prediction Framework Using Contextual information to Refine Feature Embedding

Zhiqiang Wang, Qingyun She, PengTao Zhang, Junlin Zhang
Sina Weibo Corp
Beijing, China
roky2813@sina.com,qingyun_she@163.com,{pengtao1,junlin6}@staff.weibo.com

arXiv:2107.12025v1 [cs.IR] 26 Jul 2021

ABSTRACT

Click-through rate (CTR) estimation is a fundamental task in personalized advertising and recommender systems and it's important for ranking models to effectively capture complex high-order features. Inspired by the success of ELMO and Bert in NLP field, which dynamically refine word embedding according to the context sentence information where the word appears, we think it's also important to dynamically refine each feature's embedding layer by layer according to the context information contained in input instance in CTR estimation tasks. We can effectively capture the useful feature interactions for each feature in this way. In this paper, We propose a novel CTR Framework named ContextNet that implicitly models high-order feature interactions by dynamically refining each feature's embedding according to the input context. Specifically, ContextNet consists of two key components: contextual embedding module and ContextNet block. Contextual embedding module aggregates contextual information for each feature from input instance and ContextNet block maintains each feature's embedding layer by layer and dynamically refines its representation by merging contextual high-order interaction information into feature embedding. To make the framework specific, we also propose two models(ContextNet-PFFN and ContextNet-SFFN) under this framework by introducing linear contextual embedding network and two non-linear mapping sub-network in ContextNet block. We conduct extensive experiments on four real-world datasets and the experiment results demonstrate that our proposed ContextNet-PFFN and ContextNet-SFFN model outperform state-of-the-art models such as DeepFM and xDeepFM significantly.

ACM Reference Format:
Zhiqiang Wang, Qingyun She, PengTao Zhang, Junlin Zhang. 2021. ContextNet: A Click-Through Rate Prediction Framework Using Contextual information to Refine Feature Embedding. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Click-through rate (CTR) estimation has become one of the most essential tasks in many real-world applications. Many models have been proposed to resolve this problem such as Logistic Regression (LR) [20], Polynomial-2 (Poly2) [27], tree-based models [13], tensor-based models [16], Bayesian models [9], and Field-aware Factorization Machines (FFMs) [15].

Deep learning techniques have shown promising results in many research fields such as computer vision [12, 17], speech recognition [10, 30] and natural language understanding [7, 22]. As a result, employing DNNs for CTR estimation has also been a research trend in this field [6, 11, 19, 25, 32, 33, 35]. Some deep learning based models have been introduced and achieved success such as Factorisation-Machine Supported Neural Networks(FNN)[35], Attentional Factorization Machine (AFM)[6], wide&deep[33], DeepFM[11], xDeepFM [19], DIN[36] etc.

Feature interaction is critical for CTR tasks and it's important for these ranking models to effectively capture complex features. Most DNN ranking models such as FNN and DeepFM use the shallow MLP layers to model high-order interactions in implicit way which has been proved to be ineffective [3]. Some CTR model like xDeepFM [19] explicitly introduces high-order feature interactions by adding sub-network into the network structure. However, that will significantly increases the computation time and it's hard to deploy it in real-world application.

Inspired by the success of ELMO[24] and Bert[8] in NLP field, which dynamically refine word embedding according to the sentence context where the word appears, we think it's also important to dynamically change the feature embedding according to other contextual features in the same instance it appears in CTR tasks. We can effectively capture the useful feature interactions for each feature by introducing the context aware feature embedding into CTR models.

Though AutoInt[29] and Fi-GNN[18] can also dynamically change feature embedding as our proposed model does, feature representation of these models is kind of weighted summation of pair-wise interaction . These models follow the rule of feature aggregation in summation way after pair-wise interaction, while our proposed model follow the rule of feature interaction in multiplicative way after feature aggregation by a specific network. Alex Beutel et.al[2] have proved that addictive feature interaction is inefficient in capturing common feature crosses. They proposed a simple but effective approach named "latent cross" which is a kind of multiplicative interactions between the context embedding and the neural network hidden states in RNN model. Our work is inspired by both the Bert and "latent cross".

In this work, We propose a new CTR framework named ContextNet which can dynamically refine feature's embedding according to the context it appears and effectively model high-order feature interactions for each feature. Specifically, ContextNet consists

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of two key components: contextual embedding module and ContextNet block. Contextual embedding module aggregates contextual information for each feature from input instance and ContextNet block maintains each feature's embedding layer by layer and dynamically refines its representation by merging contextual high-order interaction information into feature embedding. So the ContextNet provides a flexible mechanism for each feature to dynamically and efficiently filter out the most useful high-order cross information for its own purpose in current context it appears. Another advantage of ContextNet over most DNN models is that it has good model interpretability. Notice that ContextNet is a new CTR framework instead of a specific model, which means we can design various detailed models based on this framework. We also propose two ContextNet based models in this paper in order to make the ContextNet framework specific and experimental results prove its effectiveness.

The contributions of our work are summarized as follows:

(1) We propose a novel CTR Framework named ContextNet that implicitly models high-order feature interactions by dynamically refining the feature embedding according to the context information contained in input instance.
(2) To make the ContextNet framework specific, we propose a contextual embedding network and two non-linear mapping sub-network in ContextNet block. So we design two specific ContextNet-based models in our work under the proposed framework, which is named ContextNet-PFFN and ContextNet-SFFN, respectively.
(3) We conduct extensive experiments on four real-world datasets and the experiment results demonstrate that our proposed ContextNet-PFFN and ContextNet-SFFN outperform state-of-the-art models significantly.

The rest of this paper is organized as follows. Section 2 introduces some related works which are relevant with our proposed model. We introduce our proposed ContextNet framework in detail in Section 3. The experimental results on four real world datasets are presented and discussed in Section 4. Section 5 concludes our work in this paper.

2 RELATED WORK

2.1 Context Aware Word Embedding in NLP

Word embedding is a very important concept in NLP and it attempts to map words from a discrete space into a semantic space. In early stage, Word2vec[21] and GloVe[23] learn a constant embedding vector for a word and the embedding is same for a word in different sentences. However, it's obvious that polysemous word should have different embedding in various sentence contexts. To deal with this issue, context information of the sentence is used to predict a dynamic word embedding. For example, ELMO[24] uses the bidirectional RNN to model the context information. The GPT[26] and Bert[8] model leverages the Transformer[31] model to jointly consider both the left and right context information in the sentence. Our work is inspired by these context aware word embedding approaches and we introduce context aware feature embedding into CTR tasks in this work.

2.2 Deep Learning based CTR Models

Many deep learning based CTR models have been proposed in recent years and how to effectively model the feature interactions is the key factor for most of these neural network based models.

Factorization-Machine Supported Neural Networks (FNN)[35] is a feed-forward neural network using FM to pre-train the embedding layer. Wide & Deep Learning[33] jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. However, expertise feature engineering is still needed on the input to the wide part of Wide & Deep model. To alleviate manual efforts in feature engineering, DeepFM[11] replaces the wide part of Wide & Deep model with FM and shares the feature embedding between the FM and deep component. Most DNN CTR models rely on two or three MLP layers to model the high-order interactions in an implicit way and some research[3] has proved MLP is an ineffective way to capture the high-order interactions.

Some works explicitly introduce high-order feature interactions by sub-network. Deep & Cross Network (DCN)[32] efficiently captures feature interactions of bounded degrees in an explicit fashion. Similarly, eXtreme Deep Factorization Machine (xDeepFM) [19] also models the low-order and high-order feature interactions in an explicit way by proposing a novel Compressed Interaction Network (CIN) part. FiBiNET[14] can dynamically learn feature importance via the Squeeze-Excitation network (SENET) mechanism and feature interactions via bilinear function. AutoInt[29] proposes a multi-head self-attentive neural network with residual connections to explicitly model the feature interactions in the low-dimensional space. Fi-GNN[18] represents the multi-field features in a graph structure, where each node corresponds to a feature field and different fields can interact through edges. Though AutoInt[25] and Fi-GNN[18] can also dynamically change feature embedding by proposing a multi-head self-attentive neural network or graph neural network, feature representation of these models is kind of weighted summation of pair-wise interaction. Many research[2, 28] have proved that addictive feature interaction is inefficient in capturing common feature crosses. Our proposed models collect global contextual information by an independent contextual embedding network to change the feature representation in the multiplicative way. Our experimental results show this advantage.

3 OUR PROPOSED MODEL

In this section, we will firstly introduce ContextNet framework and then describe the key components in detail in the following sections.

3.1 ContextNet Framework

As depicted in Figure 1, we propose a novel CTR Framework named ContextNet that implicitly models high-order feature interactions by dynamically refining the feature embedding according to the context information contained in input instance. ContextNet consists of two variable components: contextual embedding module and ContextNet block. Contextual embedding module aggregates contextual information in the same instance for each feature and projects the collected contextual information to the same low-dimensional

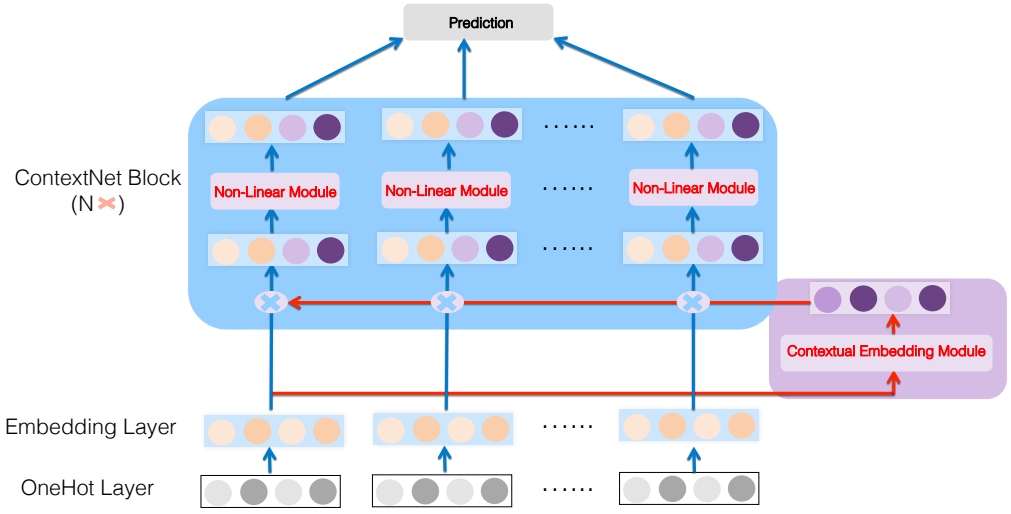

Figure 1: The Neural Structure of ContextNet Framework

space as feature embedding lies in. Notice that the input of contextual embedding module is always from the feature embedding layer. ContextNet block implicitly model high-order interactions by merging contextual information into each feature's feature embedding firstly, and then conduct the non-linear transformation on the merged embedding in order to better capture high-order interactions. We can stack ContextNet block by block to form deeper network and the refined feature's embedding output of the previous block is the input of the next one. The different ContextNet block has the corresponding contextual embedding module to refine each feature's embedding. The final ContextNet block's output is feed into the prediction layer to give the instance's prediction value.

3.2 Feature Embedding

The input data of CTR tasks usually consists of sparse and dense features. Such features are encoded as one-hot vectors which often lead to excessively high-dimensional feature spaces for large vocabularies. The common solution to this problem is to introduce the embedding layer. Generally, the sparse input can be formulated as:

$$x = [x_1, x_2, ..., x_f] \tag{1}$$

where f denotes the number of fields, and $x_i \in \mathbb{R}^n$ denotes a one-hot vector for a categorical field with n features and $x_i \in \mathbb{R}^n$ is vector with only one value for a numerical field. We can obtain feature embedding E_i for one-hot vector x_i via:

$$E_i = W_e x_i \tag{2}$$

where $W_e \in \mathbb{R}^{k \times n}$ is the embedding matrix of n features and k is the dimension of field embedding. The numerical feature x_j can also be converted into the same low-dimensional space by:

$$E_j = V_j x_j \tag{3}$$

where $V_j \in \mathbb{R}^k$ is the corresponding field embedding with size k.

Through the aforementioned method, an embedding layer is applied upon the raw feature input to compress it to a low dimensional, dense real-value vector. The result of embedding layer is a wide concatenated vector:

$$E = concat(E_1, E_2, ..., E_i, ..., E_f) \tag{4}$$

where f denotes the number of fields, and $E_i \in \mathbb{R}^k$ denotes the embedding of one field. Although the feature lengths of instances can be various, their embeddings are of the same length $f \times k$, where k is the dimension of field embedding.

3.3 Contextual Embedding

As discussed in Section 3.1, the contextual embedding module in ContextNet has two objectives: Firstly, ContextNet use this module to aggregate contextual information for each feature from input instance, that is to say, feature embedding layer. Secondly, the collected contextual information for one feature is projected to the same low-dimensional space as feature embedding lies in.

We can formulate this process as follows:

$$CE_i = \mathcal{F}_{project}(\mathcal{F}_{agg}(E_i, E; \Theta_a); \Theta_p) \tag{5}$$

where $CE_i \in \mathbb{R}^k$ denotes the contextual embedding of the i-th feature E_i and k is the dimension of field embedding, $\mathcal{F}_{agg}(E_i, E; \Theta_a)$ is the contextual information aggregation function for the i-th feature field which uses the embedding layer E and feature embedding E_i as input and Θ_a denotes the parameters of aggregation model. $\mathcal{F}_{project}(F_{agg}; \Theta_p)$ is the mapping function to project the contextual information into the same low-dimensional space with feature embedding lies in. Θ_p denotes the parameters of projection model.

To make this module more specific, We propose a two-layer contextual embedding network (TCE) for this module in our paper. That

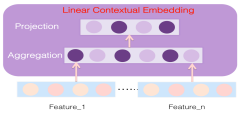

Figure 2: Two Layer Contextual Embedding

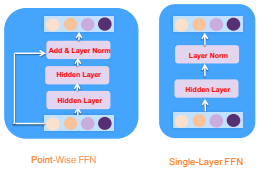

Figure 3: Structure of Non-Linear Transformation

is to say, we adapt the feed forward network as the aggregation function $\mathcal{F}_{agg}(E_i, E; \Theta_a)$ and projection function $\mathcal{F}_{project}(F_{agg}; \Theta_p)$. Notice here that TCE is just a specific solution for this module and there are other options that deserve further exploration. However, the input of the contextual embedding module should be from embedding layer which contains original and global contextual information.

Next we will describe how contextual embedding network works. Suppose we have a feature E_i which belongs to feature field d. As depicted in Figure 2, two fully connected (FC) layers are used in TCE module. The first FC layer is called "aggregation layer" which is a relatively wider layer to collect the contextual information from embedding layer with parameters W_d^a. The second FC layer is "projection layer" which projects the contextual information into the same low-dimension space with feature embedding and reduces dimensionality to the same size as feature embedding has. The projection layer has parameter W_d^p. Formally,

$$CE_i = \mathcal{F}_{project}(\mathcal{F}_{agg}(E_i, E; \Theta_a); \Theta_p) = W_d^p(RELU(W_d^a E)) \quad (6)$$

where $E \in \mathbb{R}^{m=f\times k}$ refers to the embedding layer of input instance, $W_d^a \in \mathbb{R}^{t\times m}$ and $W_d^p \in \mathbb{R}^{k\times t}$ are parameters for aggregation and projection layer in TCE for field d, respectively. t and k respectively denotes the neural number of aggregation and projection layer. Notice here that the aggregation layer is usually wider than the projection layer because the size of the projection layer is required to be equal to the feature embedding size k. The wider aggregation layer will make the model be more expressive.

We can see from formula (6) that each feature field d maintains its own parameters W_d^a and W_d^p for aggregation and projection layer, respectively. Suppose we have f different feature fields in embedding layer, the parameter number of TCE will be $f * (W_d^a + W_d^p)$. We can reduce parameter number by sharing W_d^a in aggregation layer among all fields. The parameter number of TCE will be reduce to $W_d^a + f * W_d^p$. In order to reduce the model parameters, we adopt the following strategy: we share the parameters of aggregation layer among all feature fields while keep the parameters of projection layer private for each feature field. This strategy effectively balances the model complexity and the model's expressive ability because the private projection layer will make each feature extract useful contextual information for its purpose independently. This makes the TCE module look like the "share-bottom" structure in multi-task learning where the bottom hidden layers are shared across tasks as works [5] and [4] did.

3.4 ContextNet Block

As discussed in Section 3.1, ContextNet block is used to dynamically refine each feature's embedding by merging the contextual embedding produced for that feature to implicitly capture the high-order feature interactions. To achieve this goal, there are two consequential procedures in ContenxtNet block as shown in Figure 1: embedding merging and a following non-linear transformation. We can stack ContextNet block by block to form deep network and the output of the previous block is the input of the next block.

Next we will describe how ContextNet block works. We use E_i^l to denote the output feature embedding of the l-th block, that is to say, E_i^l is the input embedding of the i-th feature for the $(l+1)$-th block. CE_i^{l+1} denotes the corresponding contextual embedding computed by TCE for the i-th feature field in the $(l+1)$-th block.

We can describe this process for the i-th feature as follows:

$$E_i^{l+1} = \mathcal{F}_{non-linear}(\mathcal{F}_{merge}(E_i^l, CE_i^{l+1}; \Theta_m); \Theta_n) \quad (7)$$

where $E_i^{l+1} \in \mathbb{R}^k$ denotes the fine-tuned feature embedding outputted by the $(l+1)$-th ContextNet block for the i-th feature E_i^l and k is the dimension of field embedding, $\mathcal{F}_{merge}(E_i^l, CE_i^{l+1}; \Theta_m)$ is the merging function for the i-th feature which uses the previous block's output feature embedding E_i^l and contextual embedding CE_i^{l+1} in current block as input. Θ_m denotes the parameters pf merging function. $\mathcal{F}_{non-linear}(F_{merge}; \Theta_n)$ is the mapping function to conduct the non-linear transformation on the merged embedding in order to further capture high-order interactions for the i-th feature. Θ_n denotes the parameters of non-linear transformation function.

As for the merging function $F_{merge}(E_i^l, CE_i^{l+1}; \Theta_m)$, Hadamard product is used in this work to merge the feature embedding E_i^l and the corresponding contextual embedding CE_i^{l+1} as follows:

$$E_i^l \otimes CE_i^{l+1} = \left[E_{i1}^l \cdot CE_{i1}^{l+1}, ..., E_{ij}^l \cdot CE_{ij}^{l+1}, ..., E_{ik}^l \cdot CE_{ik}^{l+1}\right] \quad (8)$$

where k is the size of embedding vector E_i^l and contextual embedding CE_i^{l+1} for the i-th feature. Hadamard product is a kind of element-wise production operation without parameters.

As for the non-linear function $F_{non-linear}(F_{merge}; \Theta_n)$, we propose two neural networks which are shown in Figure 3 in this paper: point-wise feed-forward network and single-layer feed-forward network.

Point-Wise Feed-Forward Network:

Though the ContextNet uses the contextual embedding module to aggregate all other feature's embedding and then project them to a fixed embedding size, ultimately it is still a linear model. For endowing the model with nonlinearity in order to capture high-order interactions better, we can apply a point-wise two-layer feed-forward network to all E_i identically (sharing parameters among all feature field):

$$\mathcal{F}_i = PFFN(E_i; \Theta_n) = LN(RELU(E_i W^1)W^2 + E_i) \quad (9)$$

where W^1, W^2 are $k \times k$ matrices and k is the dimension of field embedding. We also adapt the residual connection and layer normalization (LN) [1]. The extra parameter number introduced by FFN is $W^1 + W^2$ which is not large because of the parameter sharing in FFN. ContextNet with this version FFN is called "ContextNet-PFFN" in the following part of this paper.

Single-Layer Feed-Forward Network:

We propose another much simpler one-layer feed-forward network by reducing the residual connection and ReLU nonlinearity. We can apply this transformation to all E_i identically with sharing parameters as follows:

$$\mathcal{F}_i = FFN(E_i; \Theta_n) = LN(E_i W^1) \tag{10}$$

where W^1 is $k \times k$ matrix and the bias is also discarded. The layer normalization will bring the nonlinearity to high-order feature interactions though the ReLU is removed from the mapping function. The extra parameter number introduced by FFN is W^1 which is small because of the parameter sharing in FFN. We call ContextNet with this version FFN "ContextNet-SFFN" in the following part of this paper. Though much simpler in the mapping form compared with ContextNet-PFFN, ContextNet-SFFN has comparable or even better performance in many datasets and we will discuss this in detail in Section 4.2.

The different ContextNet block has the corresponding contextual embedding module to refine each feature's embedding when we stack multi-block to form deeper network. We can further reduce the model parameter by sharing the parameters of aggregation layer or projection layer among TCE modules for each ContextNet block. The experiments are conducted about these three different parameter sharing strategies and we will discuss this in detail in Section 4.3.

3.5 Prediction Layer

To summarize, we give the overall formulation of our proposed model's output as:

$$\hat{y} = \delta(w_0 + \sum_{i=1}^{f*k} w_i x_i) \tag{11}$$

where $\hat{y} \in (0, 1)$ is the predicted value of CTR, δ is the sigmoid function, f is the feature filed number, k is the feature embedding size, x_i is the bit value of all feature's embedding vectors outputted by the last ContextNet block and w_i is the learned weight for each bit value.

For binary classifications, the loss function is the log loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{12}$$

where N is the total number of training instances, y_i is the ground truth of i-th instance and $\hat{y}_i$ is the predicted CTR. The optimization process is to minimize the following objective function:

$$\mathfrak{L} = \mathcal{L} + \lambda \|\Theta\| \tag{13}$$

where λ denotes the regularization term and Θ denotes the set of parameters.

3.6 Interpretability of the ContextNet

Compared with simple models such as LR[20], DNN models are notorious for lack of interpretability because of the non-linearity introduced by widely used MLP layers. One advantage of ContextNet over most DNN models is that it has good model interpretability.

The last ContextBlock's outputs maintain each feature's embedding which have merged useful high-order information and the prediction layer of ContextNet is actually a LR model. So we can easily compute each feature's weight and contribution to the final prediction score given an input instance as follows:

$$FW_j = \sum_{i=1}^{k} w_i E_{ji} \tag{14}$$

where $FW_j \in \mathbb{R}$ is the weight score of the j-th feature in an instance. $E_j \in \mathbb{R}^k$ is the embedding of the j-th feature outputted by last ContextBlock and w_i is the learned weight in prediction layer for bit E_{ji}. k is the size of feature embedding. Positive score leads to positive label and negative score leads to negative label.

For a specific instance, we can detect important features which can explain why the instance has the final prediction label according to formula (14). We can also compute the feature importance on the whole training set level by accumulating or averaging scores as follows:

$$FW_j = \sum_{i=1}^{m} |FW_j^i| \tag{15}$$

$$FW_j = (\sum_{i=1}^{m} |FW_j^i|)/(n + \alpha) \tag{16}$$

where $FW_j \in \mathbb{R}$ is the weight score of the j-th feature in the whole set. FM_j^i is the score for the j-th feature in instance i and the size of the training set is m. The absolute value is adopted here because the score is either positive or negative. n is the size of instance set where a feature appears and α is a norm number to reduce the influence of low frequent features. We can find out the important features according to formula (16) or discarding unimportant features with small scores to compress model according to formula (15).

4 EXPERIMENTAL RESULT

We evaluate the proposed approaches on four real-world datasets and answer the following research questions:

- **RQ1** Does the proposed method performs better than existing state-of-the-art deep learning based CTR models?
- **RQ2** What is the training efficiency of ContextNet?
- **RQ3** What is the influence of various components in the ContextNet architecture?
- **RQ4** How does the hyper-parameters of networks influence the performance of ContextNet?
- **RQ5** Can ContextNet really gradually refine the feature embedding to capture feature interactions? How about the feature importance computation?

In the following, we will first describe the experimental settings, followed by answering the above research questions.

4.1 Experiment Setup

4.1.1 Datasets. The following four data sets are used in our experiments:

(1) **Criteo**[1] **Dataset:** As a very famous public real world display ad dataset with each ad display information and corresponding user click feedback, Criteo data set is widely used in

[1]Criteo http://labs.criteo.com/downloads/download-terabyte-click-logs/

Table 1: Statistics of the evaluation datasets

Datasets	#Instances	#fields	#features
Criteo	45M	39	30M
Movielens	1M	7	7478
Malware	8.92M	82	0.97M
Avazu	40.43M	24	9.5M

many CTR model evaluation. There are 26 anonymous categorical fields and 13 continuous feature fields in Criteo data set.

(2) **MovieLens**[2] **Dataset:** MovieLens is a popular benchmark dataset for evaluating recommendation algorithms. We adopt the well-established MovieLens 1m(ML-1m) as our evaluation dataset in this work, which contains 1 million ratings from 6000 users on 4000 movies.

(3) **Malware** [3] **Dataset:** Malware is a dataset to predict a Windows machine's probability of getting infected. The malware prediction task can be formulated as a binary classification problem like a typical CTR estimation task does.

(4) **Avazu**[4] **Dataset:** The Avazu dataset consists of several days of ad click-through data which is ordered chronologically. For each click data, there are 24 fields which indicate elements of a single ad impression.

We randomly split instances by 8:1:1 for training , validation and test while Table 1 lists the statistics of the evaluation datasets.

4.1.2 Evaluation Metrics. AUC (Area Under ROC) is used in our experiments as the evaluation metric. AUC's upper bound is 1 and larger value indicates a better performance.

RelaImp is also adopted as work [34] does to measure the relative AUC improvements over the corresponding baseline model as another evaluation metric. Since AUC is 0.5 from a random strategy, we can remove the constant part of the AUC score and formalize the RelaImp as:

$$RelaImp = \frac{AUC(Measured\ Model) - 0.5}{AUC(Base\ Model) - 0.5} - 1 \qquad (17)$$

Log loss is another widely used metric in binary classification, measuring the distance between two distributions. The log loss results of our experiments show similar trends with AUC, so we didn't present performances in this metric because of the limited space of the paper.

4.1.3 Models for Comparisons. We compare the performance of the following models with our proposed approaches: FM, DNN, DeepFM, Deep&Cross Network(DCN), xDeepFM, Transformer and AutoInt Model, all of which are discussed in Section 2. For DCN, xDeepFM, Transformer and AutoInt, 3-order feature interaction network structure is adopted as default setting as the original papers use. FM is considered as the base model in evaluation.

4.1.4 Implementation Details. We implement all the models with Tensorflow in our experiments. For optimization method, we use the Adam with a mini-batch size of 1024 and a learning rate is set to 0.0001. Focusing on neural networks structures in our paper, we make the dimension of field embedding for all models to be a fixed value of 10. For models with DNN part, the depth of hidden layers is set to 3, the number of neurons per layer is 400, all activation function are ReLU. Except for special mention, we have the default settings as follows for ContextNet: feature embedding is 10, default model has 3 ContextBlocks and hidden layer size of TCE module is 20. We conduct our experiments with 2 Tesla $K40$ GPUs.

4.2 Performance Comparison (RQ1)

The overall performance for CTR prediction of different models on four evaluation datasets is shown in Table 2. We have the following key observations:

(1) ContextNet achieves the best performance on all four datasets and obtains significant improvements over the state-of-the-art methods. It can boost the accuracy over the baseline FM by 2.80% to 11.17%, baseline DeepFM by 1.06% to 5.02%, as well as the best of DNN baselines by 0.78% to 4.24%. We also conduct a significance test to verify that our proposed models outperforms baselines with the significance level $\alpha = 0.01$. It explicitly proves the proposed ContextNet indeed yields strong learning capacity by modeling high-order interactions implicitly using the feature contextual embedding.

(2) As for the comparison of ContextNet-PFFN and ContextNet-SFFN, we can see from Table 2 that ContextNet-SFFN consistently outperforms ContextNet-PFFN model on all four datasets with the same settings, though it's much simpler in model structure and has less parameters. This means ContextNet-SFFN is a more applicable model in real-world applications.

(3) For models which explicitly introduce high-order feature interactions by sub-network such as DCN, xDeepFM , Transformer and AutoInt, xDeepFM outperforms other two models on three datasets while DCN's performance is best on ML-1m dataset. Compared with DCN and xDeepFM, Both AutoInt and Transformer model show no advantage on any dataset. That means feature interaction in multiplicative way after feature aggregation by a specific network indeed has an advantage over feature aggregation in summation way after pair-wise interaction as AutoInt and Transformer did.

4.3 Model Efficiency (RQ2)

As mentioned in Section 3.4, In order to reduce the parameters of the linear contextual embedding module, we can share the parameters in TCE module for different ContextNet block. We have the following three strategies: 1) share the parameters in aggregation layer among corresponding TCE modules for each block(Share-A); 2) share the parameters in both aggregation layer and projection layer(Share-A&P); 3) we don't share parameters(Share-Nothing), which is a default setting for all other experiments if not specially mentioned.

[2]MovieLens 1m. https://grouplens.org/datasets/movielens/1m/
[3]Malware https://www.kaggle.com/c/microsoft-malware-prediction
[4]Avazu http://www.kaggle.com/c/avazu-ctr-prediction

Table 2: Overall performance (AUC) of different models on four datasets(CNet-PFFN means ContextNet-PFFN while CNet-SFFN means ContextNet-SFFN)

	Criteo		ML-1m		Malware		Avazu	
	AUC	RelaImp	AUC	RelaImp	AUC	RelaImp	AUC	RelaImp
FM	0.7895	+0.00%	0.8446	+0.00%	0.7166	+0.00%	0.7785	+0.00%
DNN	0.8054	+5.49%	0.8527	+2.35%	0.7246	+3.70%	0.7820	+1.26%
DeepFM	0.8057	+5.60%	0.8537	+2.64%	0.7293	+5.86%	0.7833	+1.72%
DCN	0.8058	+5.63%	0.8595	+4.32%	0.7300	+6.19%	0.7830	+1.62%
xDeepFM	0.8064	+5.84%	0.8561	+3.34%	0.7310	+6.65%	0.7841	+2.01%
Transformer	0.8037	+4.90%	0.8578	+3.83%	0.7267	+4.66%	0.7819	+1.125%
AutoInt	0.8051	+5.39%	0.8569	+3.57%	0.7282	+5.36%	0.7824	+1.40%
CNet-PFFN	0.8104	+7.22%	0.8641	+5.66%	0.7399	+10.76%	0.7862	+2.76%
CNet-SFFN	**0.8107**	**+7.32%**	**0.8681**	**+6.82%**	**0.7408**	**+11.17%**	**0.7863**	**+2.80%**

Table 3: Overall performance (AUC) of different parameter sharing strategies of Linear Contextual Module in ContextNet-PFFN)

	Criteo	Malware
Share-Nothing	0.8104	0.7399
Share-A	0.8094	0.7400
Share-A&P	0.7926	0.7117

We conduct some experiments to explore the influence of the different parameter-sharing strategies on the model performance and Table 3 shows the results. We can see from the results that the performances of Share-Nothing and Share-A strategies are comparable on two datasets. However, the performance will degrade greatly if we share the parameters both in aggregation layer and projection layer. This indicates that it's very critical for the model's good performance for TCE module to extract different high-order interaction information for each refined feature of different ContextNet block. Compared with Share-Nothing strategy, Share-A is maybe a better choice because it has less parameter and can maintain good model performance.

To compare the model efficiency of different models, we use the runtime per epoch as evaluation metric. DNN and DeepFM are regarded as efficiency baseline because they are relatively simple in network structure and are widely used in many real life applications. The comparison is conducted on Criteo dataset and the results are shown in Figure 4. xDeepFM is much more time-consuming compared with all the other models and this implies xDeepFM is hard to be applied in many real life scenarios. As for the training efficiency of our proposed ContextNet models, we can see that both the ContextNet-PFFN and ContextNet-SFFN have faster training speed compared with AutoInt and xDeepFM. If we share the parameters in aggregation layer of two proposed ContextNet models, the training speed can be further increased. The ContextNet-SFFN model with Share-A strategy can run just slightly slower than baseline model, which means that ContextNet is sufficiently efficient for real world applications.

Table 4: Overall performance (AUC) of models removing different components of ContextNet-PFFN)

	Criteo	
ContextNet-PFFN	0.8104	0.7399
-w/o TCE	0.7923	0.7125
-w/o FFN	0.8043	0.7354
-w/o LN	0.8069	0.7373
-w/o RC	0.8098	0.7380

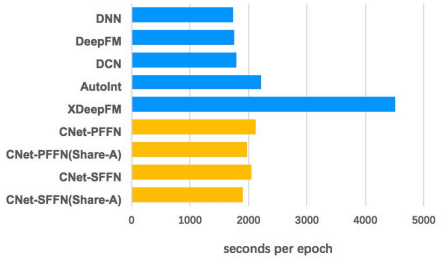

Figure 4: Efficiency comparison of different models in terms of run time per epoch on Criteo dataset

4.4 Ablation Study (RQ3)

In this section, we perform ablation experiments over key components of ContextNet in order to better understand their impacts on Criteo and Malware datasets(the other two datasets show similar trend), including linear contextual embedding (TCE), feed-forward network (FFN), layer normalization (LN), and residual connection (RC) . Table 4 shows the results of our default version (Block = 3), and its variants on two datasets.

(1) Remove TCE: The performance of ContextNet-PFFN dramatically degrades on both datasets without TCE module. It tells us that the contextual information gathered by TCE module is critical for the ContextNet and we deem TCE module extracts different high-order interactions information for various features.
(2) Remove FFN: Without FFN, the model performance also degrades obviously and that may indicate the non-linear transformation on the result of element-wise product of feature embedding and contextual information is also important for ContextNet.
(3) Remove LN or RC. From the results in Table 4, we can see that removing either LN or RC also decreases model performance, though the performance degradation is not as much as that of removing ICE and FFN.

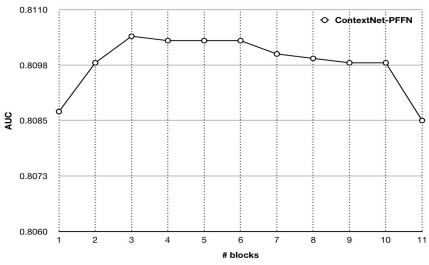

Figure 5: Effect of Different Blocks on Model Performance

We can see the usefulness of the different components of ContextNet from the above-mentioned ablation studies.

4.5 Hyper-Parameter Study(RQ4)

In this section, we study the impact of hyper-parameters on ContextNet, including (1) the number of ContextNet blocks; (2) the number of feature embedding size. The experiments are conducted on Criteo and Malware datasets via changing one hyper-parameter while holding the other settings. Other two datasets show the similar trends and we didn't present them because of the limited space.

Number of ContextNet Blocks. To explore the influence of the number of ContextNet's blocks on model's performance, we conduct some experiments on Criteo dataset to stack blocks of ContextNet-PFFN model from 1 block to 11 blocks. Figure 5 shows the experimental results. It can be observe that the performance increases with more blocks at the beginning and the performance can be maintained until the number is set greater than 10. Considering there are 4 hidden layers in one block of Context-PFFN model, we can see that ContextNet-PFFN has the depth of nearly 40 hidden layers in the model while keeping good performance. This may indicate that contextual embedding module helps the trainability of very deep network in CTR tasks.

Number of Feature Embedding Size. The results in Table 5 show the impact of the number of feature embedding size on model performance. We can observe that the performance of ContextNet-PFFN increases with the increase of embedding size at the beginning. However, model performance degrades when the embedding size is set greater than 50. Over-fitting of deep network is maybe the reason for this. Compared with the performance of the model shown in Table 2 which has a embedding size of 10, the bigger embedding size further increases model's performance on two datasets.

Table 5: Overall performance (AUC) of different feature embedding size of ContextNet-PFFN

	10	20	30	50	80
Criteo	0.8104	0.8109	0.8111	0.8113	0.8097
Malware	0.7399	0.7416	0.7417	0.7414	0.7405

Field Name	gender	age	occupation	zip code	movie name	movie year	movie genres
Feature Value	M	35	0	98503	Cat Ballou	1965	Comedy|Western
Weight	0.772	0.785	0.383	0.501	0.379	0.704	0.137

Figure 6: Example Instance from ML-1m

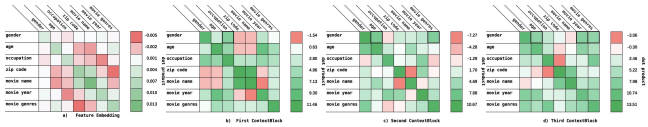

Figure 7: Analysis of Dynamic Feature Embedding

a) top 10 features

index	field & feature value	weight
1	movie year=1959	0.994
2	movie year=1946	0.984
3	movie year=1935	0.981
4	movie year=1948	0.943
5	movie year=1944	0.936
6	movie year=1931	0.908
7	movie year=1957	0.900
8	movie year=1949	0.896
9	movie year=1942	0.870
10	movie year=1950	0.868

b) top 10 features(except for movie years)

index	field & feature value	weight
1	age=50	0.606
2	age=56	0.587
3	gender=M	0.575
4	movie name=Usual Suspects, The	0.567
5	movie name=Raiders of the Lost Ark	0.564
6	movie name=GoodFellas	0.544
7	movie name=North by Northwest	0.539
8	movie name=American Beauty	0.536
9	movie name=Pulp Fiction	0.535
10	movie name=Princess Bride, The	0.531

Figure 8: Top 10 Features of ML-1m Dataset

4.6 Analysis of Dynamic Feature Embedding (RQ5)

To verify that ContextNet indeed dynamically changes each feature's embedding block by block to capture feature interactions, we input a randomly sampled instance (Figure 6, the instance has positive label with estimated CTR score 0.975) from ML-1m dataset into trained ContextNet-PFFN. Then we compute the dot product for each feature pair using each ContextBlock's outputted feature embedding, including the feature embedding layer. The high positive dot product value means the two feature have similar embedding content and are highly correlated interactions. The Figure 7 shows the result. The deeper green color means bigger positive value while deeper red color denotes bigger negative dot product value. The elements on the diagonal of the matrix can be ignored because that's each feature's self dot product. We can see from the Figure 7 that:

For features in feature embedding layer, most dot product values are small numbers near zero, which indicates the embedding values are very small and there is no correlation between different fields of input features. (Fig. a of Figure 7). However, each feature's embedding is dynamically changed by ContextBlock to gradually find the most useful feature interactions for its own purpose: more bigger dot product values begin to appear which means correlations between features. Take the field "gender" as an example, if we adopt 5.0 as the threshold, the highly correlated features change from "occupation" (Fig. b of Figure 7) to "age" and "movie genres" (Fig. c of Figure 7). The final useful feature interactions focus on the features from field "age", "movie year" and "movie genres" (Fig. d of Figure 7).

As for the feature importance analysis, we provide examples shown in Figure 6 and Figure 8. The feature contributes most to the final prediction score is "age"=35 in this instance(Figure 6). Figure 8 shows the most important features in ML-1m dataset according to formula (16) in Section 3.6.

5 CONCLUSION

In this paper, We firstly propose a novel CTR Framework named ContextNet that implicitly models high-order feature interactions by dynamically refining the feature embedding. We also propose two specific models(ContextNet-PFFN and ContextNet-SFFN) under this framework. We conduct extensive experiments on four real-world datasets and the experiment results demonstrate that our proposed ContextNet-PFFN and ContextNet-SFFN model outperform state-of-the-art models such as DeepFM and xDeepFM significantly.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] A. Beutel, P. Covington, S. Jain, C. Xu, and E. H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *the Eleventh ACM International Conference*.

[3] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[4] Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias ICML. *Google Scholar Google Scholar Digital Library Digital Library* (1993).

[5] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[13] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[14] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

[15] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 43–50.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[18] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. (2019).

[19] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1754–1763.

[20] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, and et al. 2013. Ad Click Prediction: A View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 1222–1230. https://doi.org/10.1145/2487575.2488200

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[22] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[23] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[24] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[25] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

[27] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[28] S. Rendle, W. Krichene, Z. Li, and J. Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. (2020).

[29] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[30] Hao Tang, Liang Lu, Lingpeng Kong, Kevin Gimpel, Karen Livescu, Chris Dyer, Noah A Smith, and Steve Renals. 2017. End-to-end neural segmental models for speech recognition. *IEEE Journal of Selected Topics in Signal Processing* 11, 8 (2017), 1254–1264.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[32] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[33] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[34] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2020. Deep Feedback Network for Recommendation. 2491–2497. https://doi.org/10.24963/ijcai.2020/345

[35] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[36] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.