DHEN: A Deep and Hierarchical Ensemble Network for Large-Scale Click-Through Rate Prediction

Buyun Zhang*, Liang Luo*, Xi Liu*, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, Yang Liu, Huayu Li, Yasmine Badr, Jongsoo Park, Jiyan Yang, Dheevatsa Mudigere, Ellie Wen
{buyunz,liangluo,xliu1}@fb.com
Meta Platforms, Inc. 1 Hacker Way, Menlo Park, CA

ABSTRACT

Learning feature interactions is important to the model performance of online advertising services. As a result, extensive efforts have been devoted to designing effective architectures to learn feature interactions. However, we observe that the practical performance of those designs can vary from dataset to dataset, even when the order of interactions claimed to be captured is the same. That indicates different designs may have different advantages and the interactions captured by them have non-overlapping information. Motivated by this observation, we propose DHEN - a deep and hierarchical ensemble architecture that can leverage strengths of heterogeneous interaction modules and learn a hierarchy of the interactions under different orders. To overcome the challenge brought by DHEN's deeper and multi-layer structure in training, we propose a novel co-designed training system that can further improve the training efficiency of DHEN. Experiments of DHEN on large-scale dataset from CTR prediction tasks attained 0.27% improvement on the Normalized Entropy (NE) of prediction and 1.2x better training throughput than state-of-the-art baseline, demonstrating their effectiveness in practice.

1 INTRODUCTION

Online advertising has rapidly grown to be a multi-billion business: for the US alone, it has reached $284.3 billion in the fiscal year 2021, with a 25% growth compared to the fiscal year 2020 [1]. Predicting the probability that a user clicks on a particular advertisement (a.k.a. click-through rate prediction) has played an important role in online advertising. This is because the performance of the click-through rate (CTR) prediction has an impact on the user satisfaction of the business providers.

Due to the importance of CTR prediction, tremendous efforts have been invested in improving the prediction model performance in both academia and industry [14, 25]. In the early stage, logistic regression (LR) [25] was used to model the relationship between features and label. Unfortunately, its assumption on linearity prevents capturing nonlinear input-output relationships, limiting its capability and applicability in complex scenarios. That limitation was partially resolved by the use of decision tree (DT) in [14], where input features are first non-linearly transformed by DT. However, the success of both the LR and the DT requires exhaustive efforts on feature engineering, which in reality is resource-intensive as there is an extensive number of raw features along with feature crossings (interactions). To tackle the challenge, [33] proposes a factorization machine (FMs) that can capture second-order feature interaction through the inner product of latent embeddings. In the meantime, the pre-defined shallow structure of FMs limits its expressive power. Extensions of FM to more expressive formats, such as HOFMs [4], FFMs [16, 17] and AFM [42], suffer from over-fitting along with undesirably high computation cost. All the aforementioned limitations can be properly handled by deep learning based models: the use of deep hidden layers and nonlinear activation functions enables capturing of nonlinear high-degree feature interactions in an end-to-end manner, relieving ranking engineers of the burden of exhaustive manual feature engineering and poor expressiveness imposed by shallow models.

Since 2016, various deep models have been deployed by the business providers to serve their ads ranking services, including Wide&Deep [6] (Google Play), DeepFM [11] (Huawei AppGallery), DIN [50] (Taobao), and FiBiNET [15] (Weibo). Most of those deep models consist of two primary components: feature embedding learning and feature interaction modeling. Feature embeddings are learned via mapping categorical features into embedding vectors. Feature interactions are learned by utilizing functions to model the relationship among the embeddings. Multiple studies [6, 13, 18, 20, 22, 37, 39] have shown that a better design of the interaction part can lift the prediction accuracy significantly over real-world applications. This has motivated a variety of studies that evolves from capturing low-order interactions towards high-order ones [7, 11, 13, 18–20, 22, 23, 29, 30, 34, 40, 43, 43, 48, 51].

Although several prior studies claim their design of feature interaction modules can capture high-order interactions, we notice their practical performance ranking can vary from dataset to dataset, even when they claim to capture the same degree of interaction. This indicates that different interaction modules intended to capture the same order of interaction have different strengths over different datasets, and moreover the root cause might be the information captured by them isn't overlapping. In the meantime, as shown by the experimental parts in those studies, the performance improvement obtained through learning higher-order interaction (often realized by stacking more interaction layers) sometimes has a negative effect (Figure 3 in DCN [39], Figure 7(a) in xDeepFM [20], Figure 4 in InterHAt [19], Table 2 in xDeepInt [43], and Figure 3(a) in GIN [18], to name a few). This is counter-intuitive as according to theory, capturing higher-order interaction should lead to better or at least neutral performance. We hypothesize that this issue originates from the use of a homogeneous interaction module that limits the types of interactions that can be captured. This observation signals the importance of having heterogeneous interaction modules in the model. Motivated by this, we propose DHEN: A Deep and Hierarchical Ensemble Network with a layered structure.

*Three authors contributed equally to this research.

In a DHEN layer, there is a collection of heterogeneous interaction modules and ensemble components. The collection of heterogeneous interaction modules can complement the non-overlapping information that different interaction modules can capture. The ensemble component captures the *correlation* between heterogeneous modules. As such, through recursively stacking DHEN layers, the model learns a hierarchy of the interactions of different orders and captures the correlation of heterogeneous modules, which we find empirically plays an important role in attaining better performance.

The main contributions of this paper can be summarized as follows:

- We design a novel architecture called Deep and Hierarchical Ensemble Network (DHEN) based on the observations that different interaction modules have different strengths over distinct datasets. Through recursively stacking interaction and ensemble layers, DHEN can learn a hierarchy of the interactions of different orders learned by heterogeneous modules.
- Compared to previous CTR prediction models, DHEN's deeper, multi-layer structure increases training complexity, posing a challenge to practical training. We proposed a series of mechanisms to improve DHEN training performance, including a new distributed training paradigm called Hybrid Sharded Data Parallel that achieves up to 1.2x better throughput than state-of-the-art fully sharded data parallel to support efficient training for large DHEN models.
- A comprehensive evaluation is conducted for DHEN on a large-scale dataset from CTR prediction tasks, demonstrating up to 0.27% Normalized Entropy (NE) gain over the state-of-the-art AdvancedDLRM baseline.

The rest of the paper is organized as follows. Section 2 and Section 3 illustrate the technical details of DHEN modeling and co-designed training systems, respectively. Section 4 provides the empirical evaluations of DHEN over the large-scale dataset. Section 5 briefly introduces related works of CTR prediction and feature interaction modeling. Finally, Section 6 draws the conclusions and discusses the future research directions.

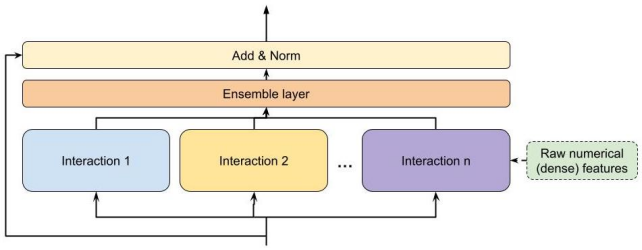

Figure 1: An general hierarchical ensemble building block in DHEN.

2 ARCHITECTURE

Most state-of-the-art high-performance model architectures across multiple prediction tasks use a deep stacking structure (e.g., Resnet, Transformers, Metaformer [12, 38, 46]). Here, the deep stacking structures are usually composed of a repeating block containing the same interaction module. For example, the transformers use self-attention blocks for stacking, and ResNet uses convolution blocks. Each block consumes the output of the previous block or the original embedding tokens as input. Structurally, DHEN follows this overall stacking strategy but at the same time builds a novel hierarchical ensemble framework to capture the correlations of multiple interaction modules. Figure 1 shows a general DHEN building block in which an ensemble of multiple Interaction modules resides. Note that the raw numerical (dense) features can be part of the input to any modules for ensembling in every layer.

2.1 Feature Processing Layer

In CTR prediction tasks, the feature inputs usually contain discrete categorical terms (sparse features) and numerical values (dense features) [8, 11, 28, 35]. In this work, we use the same feature processing layer in DLRM [28], which is shown in the Figure 3. The sparse lookup tables map the categorical terms to a list of numerical embeddings. Specifically, each categorical term is assigned a trainable d-dimensional vector as its feature representation. On the other hand, the numerical values are processed by dense layers. Dense layers compose of several Multi-layer Perceptions (MLPs) from which an output of a d-dimensional vector is computed. After a concatenation of the output from sparse lookup table and dense layer, the final output of the feature processing layer $X_0 \in \mathbb{R}^{d \times m}$ can be expressed as $X_0 = (x_0^1, x_0^2, ..., x_0^m)$, where m is the number of the output embeddings and d is the embedding dimension.

2.2 Hierarchical Ensemble

We propose a novel hierarchical ensemble framework that includes multiple types of interaction modules and their correlations. Conceptually, a deep hierarchical ensemble network can be described as a deep, fully connected interaction module network, which is analogous to a deep neural network with fully connected neurons.

The hierarchical ensemble mechanism provides a framework for an ensemble of multiple interaction modules with a layer and stacking multiple layers. The input of each layer is represented as a list of embeddings, denoted by $X_n \in \mathbb{R}^{d \times m}$. Note that input of the first layer is the list of embeddings X_0 from the feature processing layer, and the n here denotes the n-th stacked layers. Formally, the output of each layer is:

$$Y = Norm(Ensemble_{i=1}^{k} Interaction_i(X_n) + ShortCut(X_n)) \quad (1)$$

$$ShortCut(X_n) = \begin{cases} X_n, & \text{if } len(X_n) == len(Y) \\ W_n X_n, & \text{if } len(X_n)! = len(Y) \end{cases} \quad (2)$$

Where $Norm()$ here denotes a normalization method such as Layer Normalization [2]. The $Ensemble_{i=1}^{k}()$ denotes the ensemble method for $Interaction_i, Interaction_2, ..., Interaction_k$, it can be concatenation, sum, or weighed sum, etc. The $ShortCut()$ function here serves both as residual and a way to match dimensions. If the dimension of the ensemble output and the last layer input X_{n-1} don't match, a linear projection $W_n \in \mathbb{R}^{len(X_n) \times len(Y)}$ will be applied to match the dimensions. Finally, the ensemble result and the shortcut are combined through an element-wise sum, and the output Y will be the input of the next layer X_n.

The main goal of hierarchical ensemble is to capture the correlation of the interaction modules. Figure 2 (left) shows a two-layer two-module hierarchical ensemble component. Unlike traditional

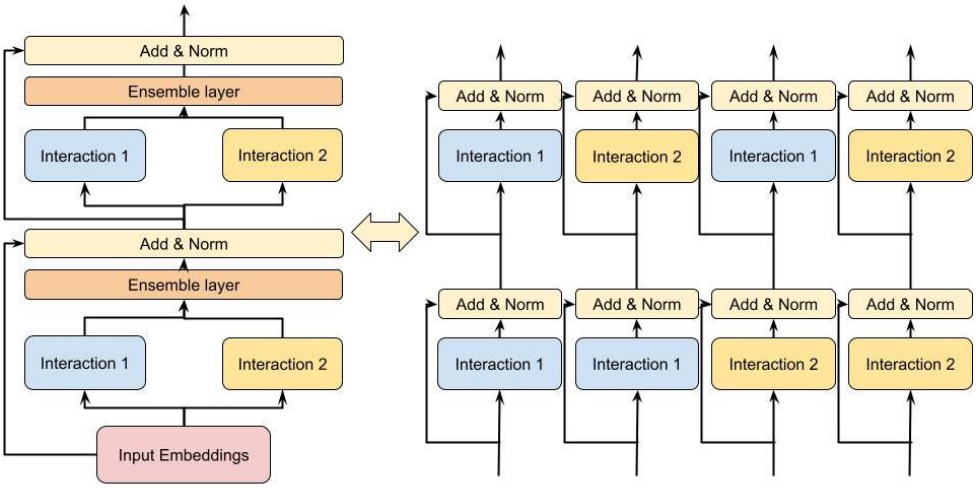

Figure 2: A Two-layer two-module hierarchical ensemble (left) and its expanded details (right). A general DHEN can be expressed as a mixture of multiple high-order interactions. We omit the potential dense feature input for the interaction modules in this figure for clarity.

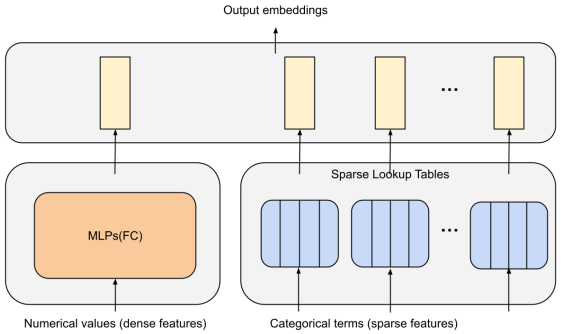

Figure 3: The feature processing layer in DHEN

stacking model structures that only capture one type of higher-order interaction, hierarchical ensembles can capture a mixture of high-order interactions. As shown in the right part of Figure 2, the design of DHEN enjoys the benefit of capturing complicated higher-order interactions between the features by letting each module consume outputs of various interaction modules (e.g., $Interaction_1(Interaction_1)$, $Interaction_1(Interaction_2)$, $Interaction_2(Interaction_1)$, and $Interaction_2(Interaction_2)$). Thus, this mixture of interaction modules can leverage multiple feature interaction types to achieve better model prediction accuracy.

2.3 Interaction Modules

We applied five types of interaction modules in our model: AdvancedDLRM, self-attention, Linear, Deep Cross Net, and Convolution. In practice, the interaction modules that can be included in DHEN are not limited to the five options above. Also, note that if an interaction module outputs a single tensor $v \in \mathbb{R}^{1\times h}$, such as a tensor output from MLPs, we will apply a $W_m \in \mathbb{R}^{h\times(d*l)}$ to map it to a list of embeddings with dimension d. The l here represents the number of output embeddings from the interaction module. We now provide a brief introduction of these interaction modules used in DHEN.

2.3.1 AdvancedDLRM. We use a DLRM style interaction module to capture the feature interactions. We call it AdvancedDLRM in our following experiments. Given an input of embeddings X_n, a new list of embeddings $u \in \mathbb{R}^{d\times l}$ is output by the AdvancedDLRM:

$$u = W_m \cdot AdvancedDLRM(X_n) \tag{3}$$

2.3.2 Self-attention. We consider self-attention, widely used in transformer networks for its superior performance in text understanding, as an interaction module in this paper. Self-attention was also adopted in CTR prediction tasks before [19]. A typical transformer includes multiple stacking encoder/decoder layers, with the self-attention mechanism at their core. In this paper, given an input of embeddings X_n, we apply a transformer encoder layer as:

$$u = W \cdot TransformerEncoderLayer(X_n) \tag{4}$$

where $W \in \mathbb{R}^{m\times l}$ is used to match and unify the output dimensions from all interaction modules.

2.3.3 Convolution. Convolution layers are widely used in computer vision tasks. It was also adopted in NLP, and CTR tasks [10, 21]. In this paper, we adopt convolution as one of the interaction modules. Given an input of embeddings X_n, a list of embeddings u is obtained from:

$$u = W \cdot Conv2d(X_n) \tag{5}$$

Similarly, $W \in \mathbb{R}^{m\times l}$ is also used to match and unify the output dimensions from all interaction modules.

2.3.4 Linear. Linear layers is one of the most straightforward modules to capture the raw information from the original feature embeddings. In this paper, we use the linear layer as one of the interaction modules to condense the information in each layer. Given an input of embeddings X_n, a list of embeddings u is obtained from:

$$u = W \cdot (X_n) \tag{6}$$

Where $W \in \mathbb{R}^{m \times l}$ here is used as a linear module weight to match the dimensions.

2.3.5 Deep Cross Net. Deep Cross Net (DCN) is a widely used feature interaction Module in CTR prediction tasks [39]. It introduces a cross network that is efficient in learning certain bounded-degree feature interactions. In this paper, we adopt the DCN module as one of the interaction modules in each layer. Given an input of embeddings X_n, a list of embeddings u is obtained from:

$$u = W \cdot (X_n \cdot X_n^T) + b \tag{7}$$

Where W and b denote the weight and bias metric in the DCN modules. We omit the skip connection process from the original paper in the equation above because we already used skip connection to flow information across the stacked layers.

3 TRAINING SYSTEM

Intuitively, the depth of the stacked hierarchical ensemble layers contributes to the expressiveness of DHEN, but they also create a challenge in practical training of DHEN. This section highlights how we enable efficient and scalable training of DHEN in our cluster.

3.1 Training Strategy

Each DHEN training sample contains both categorical and numerical features, and all features must be converted to dense representations before they can be consumed by the DHEN layers. One immediate challenge of this process is the sheer size and complexity of DHEN completely overshadows the capacity of a single standard datacenter server, typically hosting 8 GPUs with an aggregate high bandwidth memory of a few hundred gigabytes and compute capability of up to tens of petaflops. Thus a single server is drastically underpowered compared to the typical number of parameters (up to trillions) and flops needed (up to giga flops) to train a single sample in DHEN.

To allow efficient training of DHEN, we take advantage of the ZionEX fully synchronous training system detailed in [27]. At a high level, the ZionEX system groups 16 hosts into a "supernode", called a pod, which contains 128 A100 GPUs (8 per host) with a total HBM capacity of 5TB and 40PF/s BF16 compute capability. Within a host, each GPU is connected through NVLink, and each host in a pod is then connected with a high bandwidth network of up to 200GB/s, shared with 8 GPUs.

With ZionEX, we solve the compute and memory capacity issue with the following distributed training strategy. We first distribute the embedding tables across a pod. To provide better load balancing and deal with oversized embedding tables, instead of placing whole tables to different GPUs, we proactively slice oversized embedding tables into equal column shards, and place these columns based on an empirical cost function. Our cost function captures both compute and communication overhead of such placement. We distribute the table shards in a load balanced manner dictated by the cost function, using the LPT [41] approximate set partition algorithm. On the other hand, for dense modules including the DHEN layers, we replicate them on each GPU and train them in a data parallel (DP) fashion. This choice is based on the observation that the activation of the dense DHEN layers can be much larger than the weights themselves, and thus synchronizing the weights has a lower cost than sending the activations through the network. This training strategy thus induces a hybrid training paradigm, where each batch starts with DP, enters model parallelism for distributed embedding lookup, and ends with DP for the dense layers.

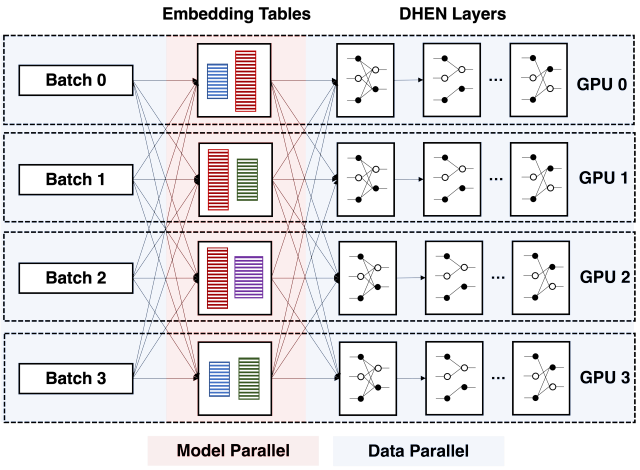

Figure 4: Training strategy for DHEN (4 GPUs shown).

Training dense modules using DP imposes a parameter size ceiling equal to per-GPU HBM capacity on the stacked DHEN layers, which hinders our exploration into the limits of DHEN scalability. To solve this issue, we use fully sharded data parallel (FSDP [3, 32]) to remove memory redundancy in traditional data parallelism by further sharding weights to different GPUs, activation checkpointing to trade more compute for less peak memory usage [5], cpu offloading [31] to further reduce GPU memory usage by aggressively storing parameters and gradients to CPU, and bring them back to GPU right before needed. Since all of these techniques hurt training efficiency, we carefully tune the system to turn on a minimum set of them for our training needs based on the number of DHEN layers.

The described training strategy of DHEN is summarized in Figure 4.

3.2 Training Optimizations

We provide additional optimizations to further speedup training.

3.2.1 Common Optimizations. We enable a set of widely-used optimizations including large batch training [9] to reduce synchronization frequency, FP16 embedding with stochastic rounding, BF16 optimizer, and quantized all to all and allreduce collectives [44, 47] to further reduce memory footprint, help with numeric stability, leverage specialized accelerator hardware such as Tensor Core [24] and to reduce communication overhead.

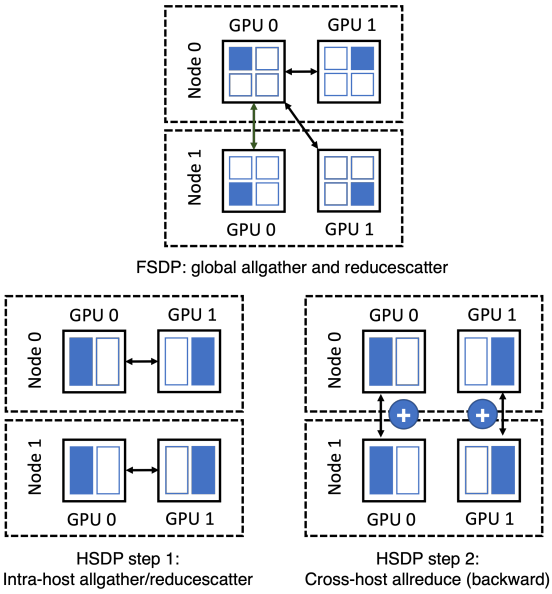

Figure 5: Comparison between FSDP (top) vs HSDP (bottom). Shown with 2 hosts with 2 GPUs each. We train large DHENs using FSDP, and medium-sized DHENs using HSDP.

3.2.2 Hybrid Sharded Data Parallel. To find the best DHEN configuration within the training cost budget, we must probe both the configuration of token mixers and the number of layers. In practice, we must frequently experiment with a candidate DHEN model whose memory footprint just exceeds the per-GPU HBM capacity (we call these models "embarrassingly-sized"), which renders data parallel infeasible and necessitates FSDP. However, in these cases, we find that applying FSDP directly does not result in optimal efficiency, especially when training at a production scale (hundreds of GPUs). This is because the allgather operations that brings different shards of weights from each GPU, required on the critical path of both forward and backward passes, can take long to finish as (1) it needs to communicate with all GPUs in the cluster, and (2) each shard would be too small to efficiently leverage network bandwidth. Although techniques such as prefetching help alleviate this problem, they add memory pressure to the system, and thus defeats the purpose of using FSDP in the first place.

To that end, we propose a novel training paradigm for these embarrassingly-sized networks, called *hybrid sharded data parallel* (HSDP), co-designed with the DHEN model for our training cluster. HSDP recognizes the 24X bandwidth difference between the GPU interconnect (NVLink, 600GB/s) and host interconnect (RoCEv2, 25GB/s), and work as follows: (1) HSDP shards the entire model within a single host, so that the reducescatter operation in the backward pass, and the allgather operation in both the forward and backward pass in FSDP operate completely within a host; (2) when the reducescatter operation finishes in the backward pass, we hook concurrent allreduce operations, each one involving all GPUs on each host with the same local host ID, to compute an average gradient for its local shard, in an asynchronous manner to avoid blocking computation of the backward pass. (3) at last, we register a backward hook to wait on the handles of the pending allreduce operations, so the semantic is left intact in HSDP compared to FSDP and DDP from a training perspective. We contrast FSDP and our HSDP in Figure 5.

Qualitatively, compared to FSDP, HSDP provides a few benefits: (1) the latency of allgather operation on the critical pass of both forward and backward pass is significantly reduced because and it can enjoy the high speed NVLink interconnect without needing to communicate cross host through the slower RoCE link; (2) the scale at which the communication collectives operates scales with the number of host, instead of number of GPUs, promoting efficiency. The tradeoff for HSDP, however, is that it supports models at most 8x (number of GPUs per host) the size supported by pure data parallel, and incurs an 1.125x communication overhead in terms of bytes on wire due to the added allreduce operation. These are acceptable for these embarrassingly-sized models, because the allreduce overhead can be hidden with computation of next layers and allreduce is a highly optimized operator.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of the Deep Hierarchical Ensemble Network for CTR prediction tasks. Our experiments has the following objectives: (1) assess the effectiveness of hierarchical ensemble and identify a good set of interaction modules for ensemble; (2) evaluate the end-to-end accuracy gain of DHEN compared to a state-of-the-art DLRM model, and demonstrate the effectiveness of our system-level optimizations on the training throughput of DHEN.

4.1 Experiment Setup

We use an industrial dataset for all experiments. As previously mentioned, the in-house AdvancedDLRM model was considered as one of the interaction modules. We omit the detailed descriptions for the AdvancedDLRM and the dataset settings for simplicity. Training hyper-parameters, including learning rate and optimizer configurations are the same for all the experiments. All models are trained with hundreds of sparse (categorical) features and thousands of dense (numerical) features. The full-sync training scheme ensures both model performance and training throughput can be reproduced [26]. We use Normalized Entropy loss to evaluate the CTR prediction accuracy [14].

4.2 Model Variation with Different Interaction Modules

The purpose of this section is to understand the behavior of DHEN with commonly used modules mentioned in Section 2.3. Specifically, we used the four interaction modules (DCN, Self-attention, CNN, and Linear) to test the performance of hierarchical ensemble (we compare with AdvancedDLRM in the next section in depth). In each experiment, we include different types of interaction modules in each layer. Table 1 shows the performance of different model settings, where N denotes the number of stacked layers. To facilitate comparison, we use DCN as baseline and use relative Normalized Entropy loss difference at different training steps (training examples) to evaluate the model performance.

Model id	Interaction type(s)	NE diff @10B examples	NE diff @20B examples	NE diff @35B examples	N
1	DCN(baseline)	NA	NA	NA	5
2	Self-attention	0.036%	0.026%	-1.044%	5
3	CNN	-1.441%	-1.535%	-1.534%	5
4	Linear	**-1.461%**	**-1.546%**	-1.538%	5
5	DCN + Linear	-0.002%	-0.004%	-0.004%	5
6	Self-attetion + Linear	-1.363%	-1.537%	**-1.576%**	5
7	Self-attetion + CNN	0.024%	-1.270%	-1.508%	5

Table 1: Model variations on the hierarchical ensemble architecture. The hierarchical ensemble of Self-attention and Linear interaction modules shows the best prediction accuracy, and significantly helps the learning of self-attention interaction converge. Here, NE denotes the Normalized Entropy [14]

Model id	Interaction type(s)	N	NE diff @5B examples	NE diff @15B examples	NE diff @25B examples
1	AdvancedDLRM(baseline)	1	NA	NA	NA
2	AdvancedDLRM + Linear	2	-0.0315%	-0.134%	-0.176%
3	AdvancedDLRM + Linear	4	-0.071%	-0.197%	-0.255%
4	AdvancedDLRM + Linear	8	**-0.068%**	**-0.208%**	**-0.273%**

Table 2: DHEN performance vs. Industrial AdvancedDLRM model. Deeper DHEN captures higher order interactions and shows better prediction accuracy.

.

Model id	Scaling method	Training FLOPs	NE diff @50B example
1	AdvancedDLRM (baseline)	0.06G	NA
2	4 expert MoE	1.3G	-0.06%
3	2 layer DHEN	1.44G	**-0.11%**
4	8 expert MoE	3.3G	-0.09%
5	4 layer DHEN	3G	**-0.21%**
6	16 expert MoE	6G	-0.10%
7	6 layer DHEN	4.6G	**-0.26%**

Table 3: DHEN achieves better scaling efficiency than MLP scaling by MoE.

From Table 1, we observed that the model (4) with linear interaction module shows the best performance among the models with single interaction module per layer, and the model (2) with self-attention interaction module needs large training data to converge. We further observed that, after the hierarchical ensemble of the self-attention and linear interaction module, model (6) starts to show the best performance among all the variants. This result also suggested that the hierarchical ensemble architecture captured the correlation of the self-attention and linear interaction module, and significantly helped the self-attention interaction module to converge. Further, we noticed that it's not always beneficial to have more interaction modules for ensemble in each layer: for example, model (5) has an hierarchical ensemble of DCN and linear layers, and its performance is better than stacking the DCN layer but worse than using linear layer alone; a similar situation can be found for model (7) with an hierarchical ensemble of self-attention and CNN interaction modules. These observations confirm our initial hypothesis that different interaction modules capture non-overlapping interactions, and hence the hierarchical ensemble mechanism is key to the performance of DHEN.

4.3 Experiments On Industrial AdvancedDLRM

4.3.1 Model Prediction Performance. After examining the effectiveness of DHEN with various interaction modules, we take one step further to evaluate its effectiveness on top of the industrial module, AdvancedDLRM. Specifically, we leverage the state-of-the-art AdvancedDLRM [28] and the Linear module as two interaction modules for hierarchical ensemble. Table 2 shows the DHEN performance compared to the AdvancedDLRM model.

We trained 4 DHEN models with a hierarchical ensemble of AdvancedDLRM and Linear interaction modules in each layer. Each of them has N layers. From Table 2, we observed that all the DHEN models outperform the industrial AdvancedDLRM model, and deeper DHEN models achieve larger Normalized Entropy improvement. In both cases, we observed that the gain keeps enlarging as more training examples are processed. This suggests that both higher order interactions and the correlations of various interactions play an important role in the CTR prediction tasks, and the enlarging Normalized Entropy improvement with larger datasets indicates DHEN performance is both consistent and generalizable.

4.3.2 Scaling Efficiency. To evaluate DHEN model scaling efficiency, we summarize our results with a comparison between two scaling methods using MoE [36] and stacking DHEN layers, in Table 3. Both scaling methods were implemented based on the same industrial AdvancedDLRM [28]. We use the MoE architecture to scale up the MLP layers in AdvancedDLRM while using an ensemble of AdvancedDLRM and Linear modules to stack DHEN layers. The results show that DHEN consistently beats AdvancedDLRM MoE with similar training complexity (measured in FLOPs) in terms of accuracy, and thus stacking DHEN layers stands out as an effective mechanism for scaling with superior return of investment.

4.3.3 Training Throughput. This section assesses the throughput of training DHEN model on a 256-GPU cluster based on the ZionEX design introduced in Section 3.1. We start by demonstrating the effectiveness of our system level optimizations with a 8-layer DHEN model trained using DP. Overall, we see 1.08x end to end speedup from applying FP16 embedding, AMP, quantized BF16 allreduce and all to all collectives. The improvement comes from the reduced exposed communication latency with the quantized collectives, and the remaining bottleneck lies in the optimizer cost and the all to all collectives calls that cannot be fully overlapped with the compute of dense layers.

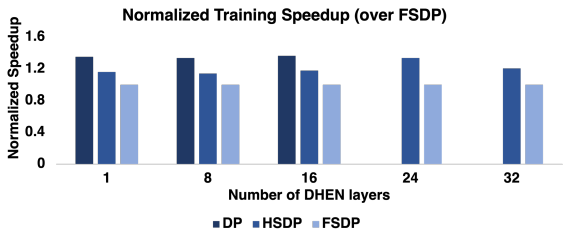

Figure 6: HSDP supports larger model sizes compared to DP and higher training throughput compared to FSDP.

While DHEN scales well with more layers, training deeper DHEN models require paradigm shift as DP can only support up to 22 layers of DHEN in our cluster. We now quantify how our proposed training paradigm codesigned with DHEN, HSDP, can help bridge both the throughput gap of training with FSDP compared to DP, as well as close the memory gap of training DHEN with DP. We conduct training throughput experiments on various DHEN models with different the number of DHEN layers. We train each model using all three paradigms and compare the resulting throughput. As shown in Figure 6, DP performs well up to 16 layers, and further increase in layers results in errors; on the other hand, both FSDP and HSDP support larger layer counts, and HSDP consistently outperforms FSDP by up to 1.2x. Our trace analysis shows the throughput gain from HSDP indeed comes from drastically reduced allgather latency on the training critical path, thanks to the fast NVLink connection within the single host.

5 RELATED WORK AND DISCUSSION

CTR Prediction Models. In the early stage, logistic regression (LR) [25] was adopted as the model for CTR prediction task. Unfortunately, it is a linear model and thus unable to capture the nonlinear relationship between the features and the outcome. That limitation was later resolved by decision tree (DT) [14] based feature transformation. Yet, both LR and DT require exhaustive efforts on feature engineering to succeed, which is difficult especially when the number of raw features is large and high-order feature interaction is present. To attack the problem, factorization machine (FMs) [33] was proposed to model second-order feature interactions through the inner product of their latent embeddings. However, the predefined shallow structure of FMs limits its expressiveness. Later, FMs was extended to higher order in [4], but due to its need to enumerate all high-order interactions without differentiating their importance, the large number of parameters to be learned for their latent vectors causes impractically high computational cost in the real-world systems. FFMs [16, 17] extends FM, by assigning each feature with multiple vectors to interact with features from other fields. Despite the significant performance improvement, FFMs introduce many more parameters and suffer from over-fitting issues. AFM [42] extends FMs with an "attention net" that improves not only the performance but also interpretability. The authors argue that the feature salience provided by the attention network greatly enhances the transparency of FMs. That said, AFM can only learn up to the second-order attention-based salience due to the inherent architectural limit of FMs. Those concerns can be properly addressed by deep learning based models: the use of deep hidden layers and nonlinear activation functions enable those models to capture nonlinear high-degree feature interactions in an end-to-end manner, getting rid of the exhaustive manual efforts of feature engineering and the limitations in expressiveness by the predefined formula. Since 2016, various deep models have been deployed by the business providers on their ads ranking services, such as Wide&Deep [6] on Google Play, DeepFM [11] on Huawei AppGallery, DIN [50] on Taobao, and FiBiNET on Weibo [15]. Most deep CTR prediction models consist of two primary components: feature embedding learning and feature interaction modeling. Feature embeddings are learned via mapping categorical features into low-dimensional embedding vectors. Feature interactions are learned by utilizing some functions to model the relationship among the embeddings. Multiple studies [6, 13, 18, 20, 22, 37, 39] have shown that a better design of the interaction part can generate significant lift on the prediction accuracy, which motivates a variety of designs from different perspectives.

Feature Interaction. In FNN [48], fully connected layers are used to learn higher-order interaction on top of the second-order interaction from pre-trained FMs. As a result, its performance is largely limited by the capability of the pre-trained FMs. PNN [30] mitigates the limitation through replacing the pre-trained FMs by a product layer. NFM [13] extends FMs by replacing the inner-product with a Hadamard product. Similarly like PNN and FNN, NFM stacks deep neural networks (DNNs) on top of the second-order feature interactions to model higher-order features. The major downside of FNN and PNN is that they both focus more on high-order feature interactions while capturing little low-order interactions, which are also essential for CTR prediction. To model both low-and-high-order feature interactions, Wide& Deep [6] proposes a hybrid network structure that combines a shallow part for artificial cross-features and a deep part for raw features. The limitation is that the input of the shallow part still relies on manual efforts of feature engineering.

To mitigate that, DeepFM [11] imposes a FMs as the shallow part of Wide&Deep to capture the cross-features (second-order interaction). DCN [39, 40] replaces the shallow part of Wide&Deep with a cross-product transformation that can efficiently capture feature interactions of bounded degrees. Unfortunately, as pointed out by [20] its output is constrained to be a very specific format and its captured interactions are solely at the bit-wise level. xDeepFM [20] improves DeepFM through introducing higher-order interactions and improves DCN through introducing replacing the cross network in DCN with compressed interaction network - a more general interaction module that can capture feature-wise interactions. It has been noted by [43] that the aforementioned methods ignore the potential benefits of combining the feature-wise and bit-wise interactions or rely on deep and resource-intensive structure to realize the latter. To overcome the limitation, xDeepInt [43] proposes to utilize subspace crossing between individual interaction layers to learn high-order feature-wise and bit-wise interactions recursively, dispensing with jointly-trained DNN and nonlinear activation functions.

In recent years, explainability has attracted more attention due to an increasing desire in reliability and security. Although more than one technique such as DCN and xDeepFM could capture high-order interactions, they suffer from poor explainability. Inspired by the success of Transformer [38] in mining feature relevance as well as the attention mechanism's capability in locating features greatly affecting predictions [7, 23, 29], AutoInt [37] proposes to use multi-layer, multi-head, and self-attentive neural networks with residual connection to learn different orders of feature interactions explicitly and meanwhile offer good model explainability. Similarly like AutoInt, InterHAt [19] proposes to capture high-order feature interactions through an attentional aggregation strategy that provides good explainability and also proves to have lower computational cost than Wide&Deep and DCN. This is by no means to enumerate all studies on interaction modules but provide representative examples. A more comprehensive review can be found in [45, 49]. Although multiple aforementioned studies have shown the benefits of high-order interaction in prediction accuracy, a more in-depth look reveals that those high-order interactions are not equally important. In fact, useless interactions may bring unnecessary noise that impairs the learning process and leads to worse performance [34]. However, it is challenging to identify their importance manually due to exponentially growing combination space for interactions. To automatically learn which feature interactions are essential, AutoFIS [22] introduces a gate (in open or closed status) for each feature interaction to control whether its output should be passed to the next layer. Similarly, GIN [18] relies on a second-stage re-training to prune unimportant feature interactions. AIM [51] further improves AutoFIS through relaxing discrete selection of open gates to be continuous parameters that can be jointly trained with model parameters and allowing for dedicated embedding size for features of different importance. Although several above studies claim their design of interaction part can capture high-order interaction, we notice their practical performance can vary from dataset to dataset, even when their bounded degrees of the interaction are the same. That indicates, the information captured by different interaction modules still has a difference, thus demonstrating different strengths over different datasets. In the meantime, as shown by those studies, the performance improvement obtained through learning higher-order interaction (often realized by stacking more interaction layers) sometimes have an opposite effect. This motivates our study on DHEN.

In the end, it is worth mentioning the architectures closest to DHEN from prior studies are GIN [18] and AutoInt [37]. Compared to them, our study has salient differences: although GIN and AutoInt also uses more than one interaction module (the multi-head attention structure in GIN and the multi-branch structure in AutoInt), each head is still homogeneous, which cannot complement non-overlapping information and capture the correlation of different modules as well as the hierarchy. Further, AutoInt also needs a second-stage re-training, limiting its applicability whereas DHEN follows an end-to-end manner and only needs to be trained once, making it more feasible in different applications.

Future Work. DHEN provides a solid foundation that opens up greater opportunities for even more accurate CTR prediction. For instance, we can apply a dedicated gating activation to individual layers of DHEN so that different examples can harness different order of interaction in the hierarchy; we can also introduce Mixture-of-Experts structure to each layer of DHEN to allocate dedicated hierarchy for individual examples; finally, we can also enable shared and dedicated layers in DHEN for different tasks to enable DHEN for multi-task scenarios.

6 CONCLUDING REMARKS

In this paper, motivated by the observation that existing interaction modules may possess different advantages on different datasets, we propose DHEN - a hierarchical ensemble architecture that can leverage strengths of heterogeneous interaction modules and learn a hierarchy of the interactions of different orders. To attack the challenge from training the deeper and multi-layer structure of DHEN, we propose a co-designed training system that improves training efficiency of training hierarchical architectures. Evaluation over large-scale dataset from CTR prediction tasks demonstrates 0.27% improvement on the NE of CTR prediction and 1.2x better training throughput.

REFERENCES

[1] Brad Adgate. Agencies agree; 2021 was a record year for ad spending, with more growth expected in 2022, Dec 2021.

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

[3] Mandeep Baines, Shruti Bhosale, Vittorio Caggiano, Naman Goyal, Siddharth Goyal, Myle Ott, Benjamin Lefaudeux, Vitaliy Liptchinsky, Mike Rabbat, Sam Sheiffer, Anjali Sridhar, and Min Xu. Fairscale: A general purpose modular pytorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale, 2021.

[4] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*, pages 3351–3359, 2016.

[5] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

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

[7] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems*, 29, 2016.

[8] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender*

systems, pages 191–198, 2016.

[9] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[10] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 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. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364, 2017.

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

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

[16] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 680–688, 2017.

[17] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM conference on recommender systems*, pages 43–50, 2016.

[18] Lang Lang, Zhenlong Zhu, Xuanye Liu, Jianxin Zhao, Jixing Xu, and Minghui Shan. Architecture and operation adaptive network for online recommendations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3139–3149, 2021.

[19] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 313–321, 2020.

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

[21] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. Feature generation by convolutional neural network for click-through rate prediction. In *The World Wide Web Conference*, pages 1119–1129, 2019.

[22] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2636–2645, 2020.

[23] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems*, 29, 2016.

[24] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. Nvidia tensor core programmability, performance amp; precision. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 522–531, 2018.

[25] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[26] Dheevatsa Mudigere, Yuchen Hao, Jianyu Huang, Zhihao Jia, Andrew Tulloch, Srinivas Sridharan, Xing Liu, Mustafa Ozdal, Jade Nie, Jongsoo Park, et al. Software-hardware co-design for fast and scalable training of deep learning recommendation models. *arXiv preprint arXiv:2104.05158*, 2021.

[27] Dheevatsa Mudigere, Yuchen Hao, Jianyu Huang, Zhihao Jia, Andrew Tulloch, Srinivas Sridharan, Xing Liu, Mustafa Ozdal, Jade Nie, Jongsoo Park, Liang Luo, Jie Amy Yang, Leon Gao, Dmytro Ivchenko, Aarti Basant, Yuxi Hu, Jiyan Yang, Ehsan K. Ardestani, Xiaodong Wang, Rakesh Komuravelli, Ching-Hsiang Chu, Serhat Yilmaz, Huayu Li, Jiyuan Qian, Zhuobo Feng, Yinbin Ma, Junjie Yang, Ellie Wen, Hong Li, Lin Yang, Chonglin Sun, Whitney Zhao, Dimitry Melts, Krishna Dhulipala, KR Kishore, Tyler Graf, Assaf Eisenman, Kiran Kumar Matam, Adi Gangidi, Guoqiang Jerry Chen, Manoj Krishnan, Avinash Nayak, Krishnakumar Nair, Bharath Muthiah, Mahmoud khorashadi, Pallab Bhattacharya, Petr Lapukhov, Maxim Naumov, Ajit Mathews, Lin Qiao, Mikhail Smelyanskiy, Bill Jia, and Vijay Rao. Software-hardware co-design for fast and scalable training of deep learning recommendation models, 2021.

[28] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[29] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788, 2018.

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

[31] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning, 2021.

[32] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. *DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters*, page 3505–3506. Association for Computing Machinery, New York, NY, USA, 2020.

[33] Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

[34] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.

[35] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 255–262, 2016.

[36] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[39] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7, 2017.

[40] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, pages 1785–1797, 2021.

[41] Wikipedia. Longest-processing-time-first scheduling. https://en.wikipedia.org/wiki/Longest-processing-time-first_scheduling. (Accessed on 01/19/2022).

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

[43] Yachen Yan and Liubo Li. xdeepint: a hybrid architecture for modeling the vector-wise and bit-wise feature interactions. In *Proceedings of the 3rd international workshop on deep learning practice for high-dimensional sparse data*, 2021.

[44] Jie Amy Yang, Jongsoo Park, Srinivas Sridharan, and Ping Tak Peter Tang. Training deep learning recommendation model with quantized collective communications. In *Proceedings of the 3rd international workshop on deep learning practice for high-dimensional sparse data*, 2021.

[45] Yanwu Yang and Panyu Zhai. Click-through rate prediction in online advertising: A literature review. *Information Processing & Management*, 59(2):102853, 2022.

[46] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021.

[47] Jian Zhang, Jiyan Yang, and Hector Yuen. Training with low-precision embedding tables. In *Systems for Machine Learning Workshop at NeurIPS*, volume 2018, 2018.

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

[49] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. Deep learning for click-through rate estimation. In *30th International Joint Conference on Artificial Intelligence (IJCAI) Survey Track*, 2021.

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

[51] Chenxu Zhu, Bo Chen, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Aim: Automatic interaction machine for click-through rate prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2021.