# Dual Attentional Higher Order Factorization Machines

Arindam Sarkar
arindamsarkar93@gmail.com
Amazon
India

Dipankar Das
dipankar_95@hotmail.com
Amazon
India

Vivek Sembium
viveksem@amazon.com
Amazon
India

Prakash Mandayam Comar
mc.prakash@gmail.com
Amazon
India

## ABSTRACT

Numerous problems of practical significance such as clickthrough rate (CTR) prediction, forecasting, tagging and so on, involve complex interaction of various user, item and context features. Manual feature engineering has been used in the past to model these combinatorial features but it requires domain expertise and becomes prohibitively expensive as the number of features increases. Feed-forward neural networks alleviate the need for manual feature engineering to a large extent and have shown impressive performance across multiple domains due to their ability to learn arbitrary functions. Despite multiple layers of non-linear projections, neural networks are limited in their ability to efficiently model functions with higher order interaction terms. In recent years, Factorization Machines and its variants have been proposed to explicitly capture higher order combinatorial interactions. However not all feature interactions are equally important, and in sparse data settings, without a suitable suppression mechanism, this might result into noisy terms during inference and hurt model generalization. In this work we present Dual Attentional Higher Order Factorization Machine (DA-HoFM), a *unified* attentional higher order factorization machine which leverages a compositional architecture to compute higher order terms with complexity linear in terms of maximum interaction degree. Equipped with *sparse* dual *attention* mechanism, DA-HoFM summarizes interaction terms at each layer, and is able to efficiently select important higher order terms. We empirically demonstrate effectiveness of our proposed models on the task of CTR prediction, where our model exhibits superior performance compared to the recent *state-of-the-art* models, outperforming them by up to 6.7% on the logloss metric.

## CCS CONCEPTS

• **Computing methodologies** → **Factorization methods**; **Supervised learning**; • **Information systems** → **Personalization**.



*RecSys '22, September 18–23, 2022, Seattle, WA, USA*

ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3546789

## KEYWORDS

recommendation systems, click-through rate prediction, deep factorization machines, higher order interactions.

**ACM Reference Format:**
Arindam Sarkar, Dipankar Das, Vivek Sembium, and Prakash Mandayam Comar. 2022. Dual Attentional Higher Order Factorization Machines. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3523227.3546789

## 1 INTRODUCTION

Supervised learning constitutes a major chunk of problems of practical significance such as click-through rate (CTR) prediction [16], conversion rate (CVR) prediction [46], forecasting, tagging [13, 49] etc., which are common in recommendation systems [12], online advertising [18], social networks [51] and so on. In the general setting, input consists of a set of continuous and discrete variables (features) and the output consists of one or more continuous valued targets for regression and discrete valued labels for classification. Linear models are expressed as a weighted combination of input features, and being fast and interpretable, constitute a simple modeling choice for many of these problems [8]. However, multiple real life scenarios ranging from customer preference for a product being conditioned on both product category and discount in e-commerce platforms (order-2 interactions) to app download behavior of male teenage gamers preferring shooting games/RPGs over other categories (order-3 interactions) [16] provide an intuitive justification for the need of higher order interactions terms in modeling complex functions.

Prior to the rise of neural networks and deep learning, ML practitioners employed feature engineering to include higher order polynomials and cross products of select feature combinations to model important interactions. The success of these models can be in part explained by Weierstrass approximation theorem [35, 39], which states that continuous, bounded functions can be approximated by a polynomial function to an arbitrarily high degree of accuracy. However the problem with manual feature engineering is that it becomes expensive to identify useful feature interactions as the feature space increases. The increased input dimension further complicates learning, especially on domains with sparse data. In our work, we address this problem via an attentional architecture which summarizes higher order interactions of degree $p$ to same

dimensionality as that of the input through a clever attention mechanism. As a result, the additional complexity from higher order interactions is linear w.r.t input size.

Multi-layer feed forward neural networks (FFNN) [4] with a single hidden layer can approximate any continuous function [21] and have been heavily adopted across problems in recommendation systems [52], computer vision [1, 36] and natural language processing [25, 50]. However, neural networks do not effectively approximate higher order combinations of features, and require significant over-parameterization to accurately capture the higher order interactions [5, 17, 43, 44]. Factorization Machines (FM), introduced by [33] are a class of models which explicitly generate feature interactions for all feature combinations, for e.g., a second-order FM can leverage second order feature interactions (pairwise) eliminating the need for extensive feature engineering. On the other hand, being a linear model on features and their combinations, FMs are not as effective in cases where the target function has non-linear dependency on input features. This has led to a popular pattern of combining *explicit* interactions terms using FM and *implicit* feature combinations using a neural network component [16]. We show an empirical justification of this choice via synthetic data experiments (section: 4.6). Despite their success, in all of these models, FMs are applied to learn only up to second order interactions (Neural FM [17], DeepFM [16] etc.). Although the original Factorization Machines paper proposes higher order interactions, it suggests no practical methods to learn the higher order terms. Additionally, since all the interactions in FM get same weight, it is unable to differentiate importance of the feature interactions.

Recent advancements in learning higher order interactions [6, 26, 43, 44] propose computing higher order interactions via multi-layered architecture. However, in all of these approaches, the feature interactions are unnormalized. In a real world scenarios, not all feature combinations attribute equally, and this is problematic in systems with sparse data, where only a few feature combinations might be present during training. Thanks to its useful properties, attention mechanisms have enjoyed tremendous popularity in computer vision and NLP [2, 9, 42, 48, 53]. [47] proposes using attention mechanism to suppress such irrelevant feature interactions. However, it is limited to degree-2 interactions.

To address these shortcomings, in this work we present a *unified* model which simultaneously captures higher order interactions via multiple interaction layers and attends to important combinations by leveraging an attention mechanism. Apart from capturing the combinatorial interactions efficiently, the proposed model leverages a feedforward neural network to model *implicit* higher order features. Our major contributions in this work are as follows:

- We propose a multi-layer, *dual-attention* based deep factorization machine architecture for explicitly modeling bounded degree (all interactions terms up to a degree $p$), higher order feature interactions.
- We further demonstrate that imposing *sparsity* constraints on higher order interactions to allow only useful combinations offers better generalization (on held-out data).
- We establish the efficacy of our model on the task of click prediction via extensive experimentation on three public datasets, on which our approach attains superior performance compared to the *state-of-the-art* models.

## 2 RELATED WORK & BACKGROUND

Leveraging cross feature interactions has been successfully used to achieve state-of-the-art results in recommendation system problems with sparse categorical features such as click-through Rate (CTR) prediction.

**Factorization Machines (FM)**: First introduced in [33] FMs are well known for modelling higher order interactions. While FMs can be theoretically extended to arbitrarily high order, the original paper only shows how to do it efficiently for order-2 (pairwise) interactions. Given data samples $\mathrm{X}_{i=1}^{N}$ with $F$ features, the FM model is expressed as

$$f_{FM}(X) = w_0 + \sum_{i=1}^{F} w_i x_i + \sum_{i=1}^{F} \sum_{j=i+1}^{F} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i x_j \tag{1}$$

where, $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^F$, $\mathbf{V} \in \mathbb{R}^{F \times d}$ are parameters to be estimated, where $d$ is the latent factor/feature embedding size. A similar approach *Poly2* was proposed in [7]. In last few years, several variants and extensions of Factorization Machines have been proposed. Field Aware FM (FFM) [23] extended the idea of learning single latent vector per feature to latent vector per 'field' pair (a field is equivalent of a categorical feature which can assume multiple values). However, FMs of this form are still linear models of individual and pairwise features.

**Combining Interactions and Neural Nets**: Wide & Deep [10] was one of the early works to report performance gains with use of cross features and combining it with a *deep* neural network. However, the cross features were hand-crafted. Product based Neural Networks (PNN) [31] proposed inner/outer product based operators to automatically compute pairwise interactions terms. This was followed by architectures such as DeepFM [16] which automatically learned the implicit and explicit feature interactions, by combining a FM and a neural network while sharing the latent embeddings of features between the FM and the neural network component. On similar lines, Neural FM [17] proposed 'Bi-Interaction' layer, which essentially creates a (sum) pooled representation of pairwise interaction terms, and leverages a deep neural network on top of the pooled representation. However, a major issue with the above-mentioned architectures is that only single interaction layer is present, which restricts the class of polynomials these models can explicitly capture.

**Higher Order Interactions:** Higher Order Factorization Machines [6] proposed one of the first algorithms for efficiently training arbitrarily high order FMs (HOFM) which used a dynamic programming based approach. However, due to lack of any non-linear projections, HOFM is still limited to linear combination of interaction terms. Deep & Cross Network (DCN) [43] proposed a multi-layer architecture with 'cross' layers in which each layer learns a higher order representation ($\mathbf{v}^{(l)}$) of input feature space ($\mathbf{v}^{(0)}$) by taking a cross-product of last layer's representation with first-order features and leverages residual connections to include lower level interaction/cross terms. Recently, DCNv2 [44], an improvement over the original Deep & Cross Network was proposed. Both the

**Table 1: Summary of representational capabilities of various higher order interaction models.**

| Model | (Explicit) Order-2 interactions | (Explicit) Efficient higher order interactions | Interaction level attention | (Implicit) higher order interactions |
|---|---|---|---|---|
| Wide & Deep [10] | No | No | No | Yes |
| FM [34] | Yes | No | No | No |
| HOFM [6] | Yes | Yes | No | No |
| DeepFM [16] | Yes | No | No | Yes |
| AFM [47] | Yes | No | Yes | No |
| AoAFM [45] | Yes | No | Yes | No |
| InterHAT [26] | Yes | Yes | No | Yes |
| AFN+ [11] | Yes | Yes | No | Yes |
| DCNv2 [44] | Yes | Yes | No | Yes |
| **DA-HoFM (Proposed)** | **Yes** | **Yes** | **Yes** | **Yes** |

architectures can be summarized as

$$\mathbf{v}^{(l+1)} = \mathbf{v}^{(0)} \odot (\mathbf{W}^{(l)}\mathbf{v}^{(l)} + \mathbf{b}^{(l)}) + \mathbf{v}^{(l)} \tag{2}$$

where in case of DCN, $\mathbf{W}^{(l)} \in \mathbb{R}^d$ is vector, whereas it is a square matrix in case of DCNv2; $\mathbf{b}^{(l)}$ is a bias term, $\odot$ represents the elementwise *Hadamard* product. Another related model is InterHAT [26] which utilizes a transformer encoder [42] to learn higher order feature representations, followed by hierarchical cross layers with residual connections to capture higher order terms. It differs from Deep & Cross Networks in its usage of attentional aggregation to compute final representation used for prediction. Apart from these *enumerative* approaches to compute higher order interactions, approaches like Adaptive Factorization Network (AFN) [11] take a slightly different route to computing higher order interactions. Instead of enumerating and aggregating factors, leveraging ideas from Logarithmic Neural Networks [19], it introduces a logarithmic transformation layer which implicitly captures arbitrary order terms.

**Attentional Interaction (single layer):** Despite the improvements over plain neural networks and order-2 FMs, all higher order terms get equal weightage in these architectures, which can be problematic in problems with sparse data domains like CTR prediction, where only few of the feature combinations co-occur frequently. Attentional Factorization Machines (AFM) [47] was one the first papers to address this issue and proposed assigning an attention weight to each of the interaction terms to lower the effect of unimportant feature interactions. Recently, another attentive model, AoAFM [45] was proposed which improves upon AFM by introducing an interaction level attention network which helps in cases where there are redundant features causing high attention weights on interaction of these features. However both of these model only up to order-2 interactions. Representational capabilities of representative models are summarized in Table 1.

We extend and enhance the prior work by proposing an unified attentional higher order factorization machine model, which captures *explicit* higher order terms by using a *contextual* attention mechanism and leverages a feed forward neural network to capture *implicit* higher order representation of feature space. Additionally, we propose various methods to induce *sparsity* in attention mechanism to further suppress irrelevant higher order interaction terms.

# 3 OUR APPROACH

Here we describe our proposed model which leverages a multi-layer dual-attention mechanism to model higher order interactions. Fig. 1 illustrates the architecture of our model.

## 3.1 Feature Representation

An instance of supervised learning problem consists of categorical valued and numerical valued input features. Categorical features may take thousands of unique values, and directly using the one-hot encoding might blow up the feature space. Thus given a sample $\mathbf{x}$ with $F$ features, an embedding layer is used to encode unique categorical feature values to a dense $d$-dimensional representational space $\mathbf{v}_i^{(0)} \in \mathbb{R}^d$ ($d$ is the latent factor size in FM terminology). Additionally, similar to [17, 26], for the $j^{th}$ numerical feature, an embedding $\mathbf{v}_j \in \mathbb{R}^d$ is learned, and embedding for a particular value of numerical feature $x_j$ is given as $\mathbf{v}_j^{(0)} = x_j.\mathbf{v}_j$ (where '.' represents the scaling operation of a scalar/vector by scalar). The final feature representation of a data sample is given as $\mathbf{v}^{(0)} = [\mathbf{v}_1^{(0)}, \mathbf{v}_2^{(0)}, ..., \mathbf{v}_F^{(0)}]$. Note that, the latent embeddings corresponding to a feature are the counterpart of low-dimensional latent-factors in original Factorization Machines.

## 3.2 Attentional Interaction for Higher Order terms

The proposed higher order interaction component is composed of multiple layers of attentional interaction, wherein each layer $l$ generates $(l+1)^{th}$ order representation of features. Starting with initial feature representation $\mathbf{v}^{(0)} \in \mathbb{R}^{F \times d}$, for each layer $l > 0$, higher order term corresponding to feature $i$ is generated as:

$$\mathbf{v}_i^{(l+1)} = \sum_{j=1}^{F} \alpha_{ij}^{(l)} (\mathbf{v}_i^{(l)} \odot \mathbf{v}_j^{(0)}) \tag{3}$$

$$\text{where, } \alpha_{ij}^{(l)} = \text{softmax}_j(\alpha_i'^{(l)}) = \frac{\exp(\alpha_{ij}'^{(l)})}{\sum_{q=1}^{F} \exp(\alpha_{iq}'^{(l)})} \tag{4}$$

$$\text{and,} \quad \alpha_{ij}'^{(l)} = (\mathbf{v}_i^{(l)} \odot \mathbf{v}_j^{(0)})^T \mathbf{W}_a^{(l)} + \mathbf{b}_a^{(l)} \tag{5}$$

Here $\odot$ represents the Hadamard product, and $\text{softmax}_j$ denotes softmax(.) transform across $j^{th}$ dimension. As discussed before, not all higher order terms are equally important, and thus we use a

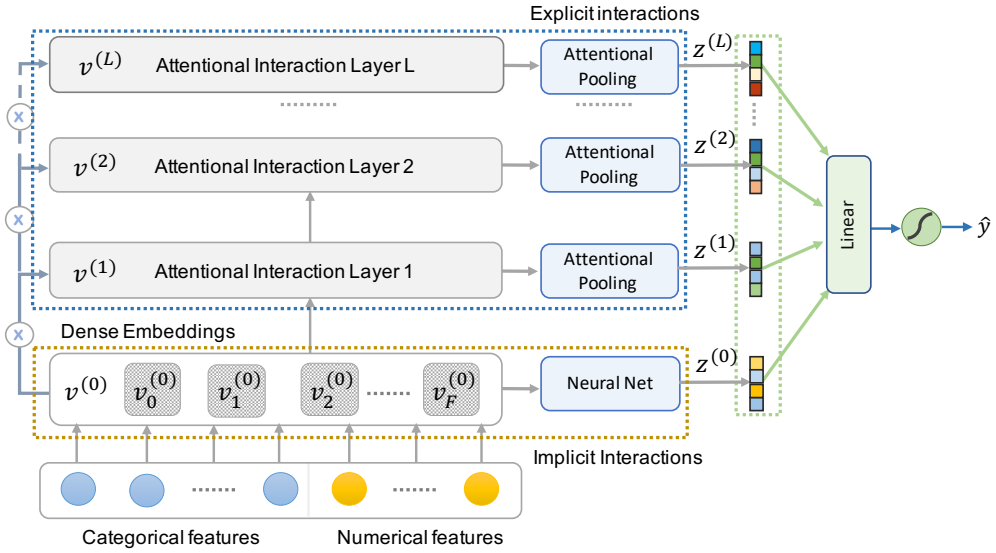


**Figure 1: Visualization of the proposed model composing multiple layers of attentional interaction to capture *explicit* bounded degree feature interactions, and non-linear projections of input features to model *implicit* higher order feature representations.**

*contextual* attention mechanism which is a function of both the interacting terms, to downscale the low valued interactions (Eq. (5)). $\mathbf{W}_a^{(l)} \in \mathbb{R}^d$ and $\mathbf{b}_a^{(l)}$ represent the *contextual* weights and biases at a given layer $l$, respectively.

One potential issue with use of softmax(.) function to compute attention-weights in Eq. (4) is that outputs of softmax consist of non-zero terms, and thus the generated probability distribution is not sparse. In our case, this implies that all the higher order terms become part of the final representation after layer $L$. This might result in overfitting. One way to circumvent this is by using sparsity inducing normalizing functions, for instance, Sparsemax [28], which is similar to softmax, but can output sparse probabilities. However, in our experience, the commonly available implementations being $O(F\ log\ F)$ add significant latency to train/inference time. Instead, we used the temperature scaled version of the softmax function [20]

$$\alpha_{ij} = \text{softmax}_j(\alpha'_i/\tau) \tag{6}$$

The $\tau$ hyperparameter can be used to tune the sharpness of softmax distribution. As $\tau \to \infty$, the output distribution will tend to uniform distribution, whereas, for $\tau \to 0$, the distribution will peak around the maximum value in input vector, driving other values to 0, thus sparsifying the output. Suitably choosing a value of $\tau$ ensures that only selective higher order terms are part of the final representation.

## 3.3 Attentional Pooling

The output of an attentional interaction layer ($l$) for a data sample $\mathbf{x}$ is the higher order representation $\mathbf{v}^{(l)} = [\mathbf{v}_1^{(l)}, \mathbf{v}_2^{(l)}, .., \mathbf{v}_F^{(l)}]$ ($\mathbf{v}^{(l)} \in \mathbb{R}^{F \times d}$), where $\mathbf{v}_i^{(l)} \in \mathbb{R}^d$ summarizes all the higher order terms of order $l$ involving feature $i$. To include the lower order terms, broadly two major approaches have been proposed in prior work: first involves using residual connections [40] at each layer $l$ from layer $l-1$ [26, 43, 44], and second involves sum pooling of outputs from each layer [27]. For our model, we found pooling as a summarization mechanism performed better compared to the residual architecture.

We leverage *attentional* pooling to compute the layer representation $\mathbf{z}^{(l)}$ ($l > 0$) as

$$\mathbf{z}^{(l)} = \sum_{j=1}^{F} \beta_j^{(l)} \mathbf{v}_j^{(l)} \quad \text{where,} \tag{7}$$

$$\beta_j^{(l)} = \frac{\exp(\beta_j'^{(l)})}{\sum_{q=1}^{F} \exp(\beta_q'^{(l)})} \quad \text{and,} \quad \beta_j'^{(l)} = \mathbf{v}_j^{(l)^T} \mathbf{W}_r^{(l)} + \mathbf{b}_r^{(l)} \tag{8}$$

where $\mathbf{W}_r^{(l)} \in \mathbb{R}^d$ and $\mathbf{b}_r^{(l)}$ represent the learnable parameters for attentional sum at layer $l$.

While the described dual attention mechanism enables our model to select *explicit* higher order terms, to capture *implicit* higher order interactions [17, 44], we further augment our model with a neural network component.

## 3.4 Implicit Interactions via Neural Networks

Neural networks are known to be very effective in modelling functions with a compositional structure, which can be approximated by series of non-linear projections. Different from the Factorization Machine variants which utilize a linear layer to capture first order terms [33, 47], we leverage a feed-forward neural network (FFNN) to learn implicit higher order representations of the raw features similar to [16, 43]. The network consists of $M$ hidden layers, with ReLU (rectified linear unit) [29] as non-linearity, and embeds the input features into $d$ dimensional space (which is chosen to be same as the factor size) represented by $\mathbf{z}^{(0)}$. We use the standard neural network recipe of using Batch Normalization [22] between

the layers, as it is known to speed up the convergence of neural networks by alleviating the problem of covariate shift. We also leverage Dropout [38] and $L_2$ regularization to prevent overfitting.

### 3.5 Loss Function

Final representation of a sample $x$ is computed by concatenating fixed sized ($d$-dimensional) representations from all the intermediate higher order terms and assigning importance to individual terms via a linear projection

$$\hat{y}(x) = \text{sigmoid}([\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, .., \mathbf{z}^{(L)}]^T \mathbf{W}_f + \mathbf{b}_f) \tag{9}$$

where $\mathbf{W}_f \in \mathbb{R}^{(L+1)\times d}$ and $\mathbf{b}_f$ are learnable parameters for the final projection. Note that this projection can be replaced by a more complex component like a neural network, however in this work, we simply utilize a linear layer.

Since all the benchmark datasets we use are datasets with binary response, we use binary cross-entropy loss to learn the model parameters.

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{s \in \mathcal{D}_{tr}} y_s \log(\hat{y}_s) + (1 - y_s) \log(1 - \hat{y}_s) + \lambda_1 \|\theta\|_2 + \lambda_2 \mathcal{L}_{sp}(\theta) \tag{10}$$

Here $N$ is the number of data samples in the training set, $\theta$ represents all the model parameter and $\lambda_1$ denotes decay term for $L_2$ regularization. Apart from using scaled-softmax, to further enforce sparsity in attention layers, we add per-sample *entropy* loss [15] with weight $\lambda_2$ on both the attention layers described in Eq. (5, 7)

$$\mathcal{L}_{sp}(\theta) = \frac{1}{NL} \sum_{x \in \mathcal{D}_{tr}} \sum_{l=1}^{L} L_{ent}(\alpha^{(l)}) + L_{ent}(\beta^{(l)}) \text{ where,}$$

$$\mathcal{L}_{ent}(\alpha^{(l)}) = -\frac{1}{F} \sum_{i=1}^{F} \alpha_i^{(l)} \log(\alpha_i^{(l)}) \tag{11}$$

where, $L_{ent}(.)$ denotes the entropy of a given distribution and is similarly defined for $\beta^{(l)}$. We optimize Eq. (10) using the AdaGrad [14] optimizer.

## 4 EXPERIMENTS

In this section, we evaluate our proposed model **DA-HoFM** on real world datasets to answer the following research questions:

- **RQ1** How does the proposed model compare against the *state-of-the-art* models?
- **RQ2** What is the incremental benefit of different components of the model – *contextual* attention, *attentional* pooling and *sparsity* enforcing regularization?
- **RQ3** What is the incremental benefit of (explicit) multiplicative interactions *vs.* (implicit) non-linear additive interactions?
- **RQ4** How does the model performance vary with change in specific hyperparameters like factor size, size of hidden layers and number of layers?

**Table 2: Dataset statistics for Criteo, Avazu and Frappe.**

| Dataset | Criteo | Avazu | Frappe |
|---|---|---|---|
| Feature fields (C + N) | 20 + 14 | 21 + 0 | 10 + 0 |
| Total records | 45.8M | 40.4M | 288K |
| Unique tokens | 526.97K | 32.38K | 5.38K |

### 4.1 Datasets

We evaluate our model on three publicly available datasets:
**Criteo**[1]**:** It is a popular benchmark dataset for CTR prediction, containing chronologically ordered click-through records. The first 80% of data is used for training, and the rest is randomly split in 1:1 ratio to form the validation and test sets.
**Avazu**[2]**:** Avazu is another large scale CTR dataset of users' mobile behavior. Data is randomly split in 8:1:1 ratio for form the train, validation and test sets respectively. We follow data pre-processing methodology suggested in [26] for Criteo and Avazu.
**Frappe**[3] **[3]:** Frappe is a dataset containing context-aware app usage. It contains count values for usages. Similar to prior work [17, 47], we use negative sampling in ratio 2:1, binarize the count values and split the data into 7:2:1 to form the train, validation and test sets.

Table 2 shows the statistics of the three datasets, where C and N denote the categorical and numerical features. Various works in literature have used different subset of features and training data, and various pre-processing strategies for the Criteo and Avazu datasets. In this work, we have followed the feature subsetting strategy followed in InterHAT [26], and have used the entire labeled dataset for creating the data splits for training and evaluation. For Frappe, we have used the standardized dataset proposed in [17]. We have not used any specialized pre-processing on the raw data.

### 4.2 Baselines

The choice of baselines has been carefully made to cover the best performing models covering a wide range of higher order interaction models. We compare our model with the following baselines:

- **Wide & Deep** [10]: Combines a linear model with a deep neural network. Note that the authors had used hand-crafted cross-features, which we omit for a fair comparison with other models. It is a representative model for vanilla neural network based approaches.
- **Factorization Machine (FM)** [34]: We implemented order-2 FM in PyTorch [30] and used it for our experiments.
- **DeepFM** [16]: It combines a deep neural network with order-2 FM, and feature embeddings are shared between the two components. We use it as a representative architecture for models combining FM with neural networks.
- **Attentional Factorization Machines (AFM)** [47]: Enhances order-2 FM by using an attention mechanism to give

[1] ailab.criteo.com/kaggle-contest-dataset-now-available-academic-use/
[2] kaggle.com/c/avazu-ctr-prediction/
[3] baltrunas.info/research-menu/frappe

higher weight to more relevant interactions. We use the publicly available implementation[4].

- **Attention Over Attention Factorization Machines (AoAFM)** [45]: AoAFM extends AFM by augmenting the interaction attention mechanism with an additional parameter to handle redundant features. We implemented the non field Aware version of the model in PyTorch for fairness of comparison.
- **Adaptive Factorization Network (AFN+)** [11]: AFN is a recently proposed model which computes higher order interactions via a logarithmic transformation layer. We use the AFN+ model proposed in the paper which also augments AFN with a parallel deep neural network. We use the publicly available implementation[5].
- **Deep & Cross Network v2 (DCNv2)** [44]: It is a state-of-the-art model for CTR tasks. We implemented the model in PyTorch and used the optimal parameters mentioned by the authors in the paper for each dataset.
- **InterHAT** [26]: A recently proposed model for interpretable predictions on CTR tasks. We implemented InterHAT in PyTorch for our experiments.
- **Stacked Transformer Encoders (Trans-Enc)** Owing to the popularity of Transformer [32, 42] architectures in recent years, and due to the similarity of self-attention mechanism to the interaction strength calculated in FM style architectures, we also include a custom baseline consisting of 3 transformer encoder blocks stacked over each other. It also generalizes models like AutoInt [37] which uses transformer encoder followed by a linear projection to capture complex non-linear functions.

## 4.3 Experimental Setup

**Metrics:** To establish the efficacy of our proposed models on CTR datasets with binary response, we report logloss (binary cross-entropy) and ROC-AUC (Area under the ROC curve). Note that even an improvement of 0.1 basis-points (bps) in logloss is considered to be *significant* improvement for the CTR tasks, since the improvement gets amplified due to millions of impressions on daily basis [10, 44, 47].

**Implementation details:** All our model variants as well as the baselines are implemented in PyTorch, except AFN for which the publicly available implementation is used. For other models, hyperparameter selection is performed using a mix of grid search (where a group of hyperparameter values are pre-specified) and manual tuning. Factor/Embedding size is chosen to be constant 16, and batch size 512 for all the baselines for fair comparison. The best set (model version) is selected using (minimum) logloss on validation set. In choice of optimizers, we experiment with ADAM [24] and AdaGrad [14] algorithms with multiple learning rate and $L_2$ decay parameters and final results are reported with the best optimizer setting. For learning rate and $L_2$ decay, search parameters ranged from $\{10^{-1}, 10^{-2}, 10^{-3}\}$, and $\{0, 10^{-2}, 10^{-4}, 10^{-5}, 10^{-6}\}$ respectively. For the models with a neural network component, we start with the optimal hyperparameters reported by the authors wherever applicable (for instance DCNv2, InterHAT, AoAFM), and tuned the parameters like number of hidden layers ($\{2, 3\}$), hidden layer size ($\{64, 100, 200, 300, 512, 768\}$) and dropout probability ($\{0, 0.2, 0.5\}$).

**Our Configuration**: We report results for our models with AdaGrad learning-rate of $10^{-1}$ and $L_2$ decay of $10^{-4}$, $10^{-6}$, 0 for Frappe, Avazu and Criteo respectively. Number of interaction layers $L$ and number of hidden layers in neural network are set to 2. Size of the hidden layers is set to 512. Scaling parameter $\tau$ (softmax) used is set to 0.8.

**Hardware:** All the experiments are run on a 64 CPU machine with single Nvidia V100 GPU per model.

## 4.4 RQ1: Performance Comparison

Table (3) shows performance metrics of our proposed model **DA-HoFM** along with the other baseline models. It can be observed that our model outperforms the *state-of-the-art* DCNv2 and AFN+ models on Criteo and Frappe datasets, and on Avazu it is competitive to the DCNv2 model. Our model outperforms more complex models transformer encoder based models such as InterHAT and stacked transformer encoder. Note that all of these are multilayer models and DCNv2 and AFN+ are additionally augmented with a deep neural network, but lack an attentive mechanism to suppress irrelevant interactions. For other models, it can be observed that Wide & Deep model without pairwise interaction terms, underperforms FM with much lesser parameters on Avazu and Frappe datasets, however on Criteo dataset it outperforms the FM. Note that in contrast to the prior works which use LibFM [34] based implementation of FM, we use a PyTorch based implementation which leverages embedding layer. Similar to [47], we observe that this implementation performs better compared to the libFM implementation, and and yields competitive results on all the datasets. As expected DeepFM which combines a neural network and order-2 FM, outperforms both the Wide & Deep and FM models. Equipped with the attention mechanism, AFM outperforms the plain FM. AoAFM which extends AFM by learning interaction level parameters to further suppress redundant features outperforms AFM. The multilayer (higher order) interaction model DCNv2 and DA-HoFM which leverage neural networks for modelling *implicit* interactions outperform all the models which only consider up to order-2 interactions, which demonstrates the importance of higher order interactions.

## 4.5 RQ2: Model Ablation

To illustrate the incremental value of each component of our model, we benchmark four model variants: (1) Model with only contextual attention, which uses sum pooling instead of attentional pooling. We call it **SA-HoFM**. (2) Model with neural network component replaced by a single linear layer. We refer to it as **DA-HoFM-lin**, (3) Model without sparsity regularization, which we indicate as **DA-HoFM-NS**. Table (4) summarizes the results. We can observe from the table that the full model with *dual* attention outperforms the model variant with only *contextual* attention and *sparsity* regularization further improves the model. Similarly, on both the large scale datasets, neural network component results in performance improvement being able to capture non-linearities which are not efficiently captured by FMs.

[4] github.com/hexiangnan/attentional_factorization_machine
[5] github.com/WeiyuCheng/AFN-AAAI-20

**Table 3: Performance comparison of DA-HoFM with the baselines (Logloss: *lower* is better; AUC: *higher* is better).**

| | | Criteo | | Avazu | | Frappe | |
|---|---|---|---|---|---|---|---|
| **Interaction Type** | **Model** | Logloss ↓ | AUC ↑ | Logloss ↓ | AUC ↑ | Logloss ↓ | AUC ↑ |
| (Implicit) Interactions | Wide & Deep | 0.4651 | 0.7860 | 0.3892 | 0.7619 | 0.1946 | 0.9771 |
| (Explicit) Order-2 Interactions | FM | 0.4665 | 0.7829 | 0.3880 | 0.7644 | 0.1772 | 0.9757 |
| | AFM | 0.4641 | 0.78506 | 0.3949 | 0.7512 | 0.17313 | 0.9753 |
| | AoAFM | 0.4621 | 0.7882 | 0.3879 | 0.7644 | 0.1597 | 0.9804 |
| (Explicit) Order-2 + Implicit | DeepFM | 0.4590 | 0.7926 | 0.3879 | 0.7647 | 0.1701 | 0.9793 |
| (Explicit) Higher Order + Implicit | InterHAT | 0.4593 | 0.7908 | 0.3898 | 0.7613 | 0.2164 | 0.9635 |
| | Trans-Enc. | 0.4585 | 0.7917 | 0.3878 | 0.7646 | 0.1894 | 0.9758 |
| | AFN+ | 0.4538 | 0.7980 | 0.39016 | 0.76136 | 0.1580 | 0.9784 |
| | DCNv2 | 0.4551 | 0.7963 | **0.3865** | 0.7666 | 0.1688 | 0.9805 |
| | **DA-HoFM** | **0.4503** | **0.8014** | **0.3865** | **0.7668** | **0.1474** | **0.9821** |

**Table 4: Performance comparison of the model variants.**

| **Dataset** | **Criteo** | | **Avazu** | | **Frappe** | |
|---|---|---|---|---|---|---|
| **Metrics** | Logloss ↓ | AUC ↑ | Logloss ↓ | AUC ↑ | Logloss ↓ | AUC ↑ |
| SA-HoFM | 0.4505 | 0.8012 | 0.3867 | 0.7665 | 0.1595 | 0.9806 |
| DA-HoFM-lin | 0.4656 | 0.7838 | 0.3885 | 0.7632 | 0.1546 | 0.9810 |
| DA-HoFM-NS | 0.4508 | 0.8008 | 0.3866 | 0.7667 | 0.1484 | 0.9817 |
| DA-HoFM | **0.4503** | **0.8014** | **0.38657** | **0.7668** | **0.1474** | **0.9821** |

## 4.6 RQ3: Explicit vs. Implicit interactions

Factorization Machine based architectures explicitly model the combinatorial terms, whereas neural network based architectures are well suited to model functions with a compositional structure involving non-linear projections. This is one of the central arguments to support combining the two architectures for superior model performance [16, 17, 26, 41, 44]. Here, we conduct synthetic experiments to empirically establish that for certain classes of functions, for instance polynomials with product terms, FMs are more parameter efficient, while for non-linear compositional functions, neural networks are better suited. In a nutshell, first input (X)/output (y) data is generated by a probabilistic process, and then, neural network and vanilla factorization machine models are used to approximate $f : X \rightarrow y$. In this experiment, 70% of data is used for training, while the remaining 30% is split into validation and test sets.

*4.6.1 Polynomial function:* Input data $[x, z] \in \mathbb{R}^2$ and output $y \in \mathbb{R}$ are generated as follows

$$x \sim \mathcal{U}(x_{min}, x_{max});\ z \sim \mathcal{U}(z_{min}, z_{max});$$
$$y_i \sim \mathcal{N}(f(x_i, z_i), 0.01); \text{where, } f(x_i, z_i) = ax_i^2 + bz_i^2 + cx_iz_i \quad (12)$$

where $\mathcal{U}(,)$ and $\mathcal{N}(,)$ represent uniform and normal distributions respectively; $a, b, c$ are constants.

*4.6.2 Non-linear function:* Input data $[x, z] \in \mathbb{R}^2$ (generated as in Eq. 12) and output $y \in \mathbb{R}$ generated as follows

$$y_i \sim \mathcal{N}(f(x_i, z_i), 0.01);\ f(x_i, z_i) = \tanh(c \tanh(ax + bz + d)) \quad (13)$$

where *tanh* represents the hyperbolic-tangent function, a popular activation function used in neural networks; $a, b, c, d$ are constants.

For both the experiments, Stochastic Gradient Descent based optimization with learning-rate of $1e^{-3}$ is used. Training is run for 500 epochs, and the best model is selected using mean squared error on the validation set. Non-linear *tanh* is used as an activation in the neural network model for the hidden layers.

It should be noted that the first data generating process naturally matches the inductive bias in factorization machine models whereas the second favors the neural networks. Figure 2 summarizes results of the experiments. It can be observed that FMs are much more efficient (w.r.t to the parameter space) on the task of learning the polynomial function with product terms, whereas neural network is more effective on the non-linear function. This provides an empirical justification for our approach where we use multiple interaction layers to capture *explicit* higher order terms and use a feedforward neural network to learn *implicit* higher order representation of input space.

## 4.7 RQ4: Sensitivity to Hyperparameters

Hyperparameters can profoundly impact a model's performance. In this section, we analyze the effect of some of the important hyperparameters on the Frappe dataset. The other two datasets, Avazu and Criteo follow a similar pattern.

**Factor size:** One interpretation of the interaction terms ($\mathbf{V}^{(0)}\mathbf{V}^{(0)^T} = \sum_i \sum_j \mathbf{v}_i^{(0)^T} \mathbf{v}_j^{(0)}$) is a low rank factorization of the actual feature interaction matrix $W_I \in \mathbb{R}^{d \times d}$. Thus latent factor size determines the ability to approximate interaction information. Intuitively larger factor sizes will incur lower approximation loss. This explains the trend observed in figure (3.a) – performance improves with increase in factor size up to a latent factor size of 512, after which a dip in both logloss and AUC metrics can be observed, which can be attributed to overfitting on data.

**Number of interaction layers:** As more layers of interaction are added, DA-HoFM will in effect, capture higher order/degree interactions. For this experiment, we constrain the hidden layer size of neural network component to be the same as latent factor size (16), so that effect of interactions is more pronounced. Figure 3.b) shows performance trends as number of interaction layers increases. It

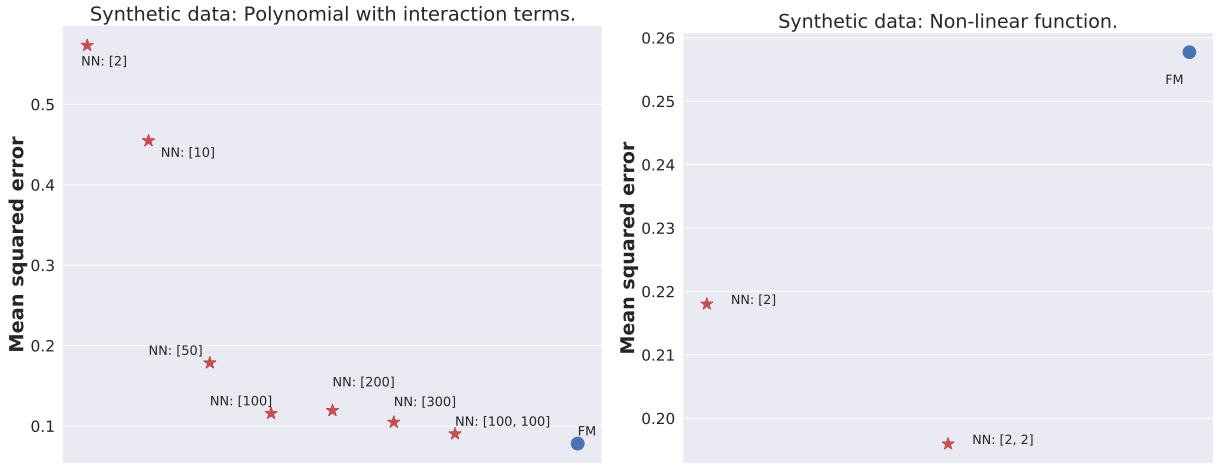


**Figure 2: Test-set mean squared error on approximation of simple functions with feedfoward neural networks (marked as NN, sizes of hidden layers are indicated inside [..]), and factorization machines (marked as FM, with factor size $d = 1$). It can be observed that for polynomial function with interaction/combinatorial terms, FMs with parameter size of 4 is more efficient to learn (w.r.t parameter space), compared to NN, which requires 300 parameters to attain equivalent performance. Whereas, for the non-linear function, NN with parameter size of 4 attains lower mean squared error compared to FMs with similar number of parameters.**

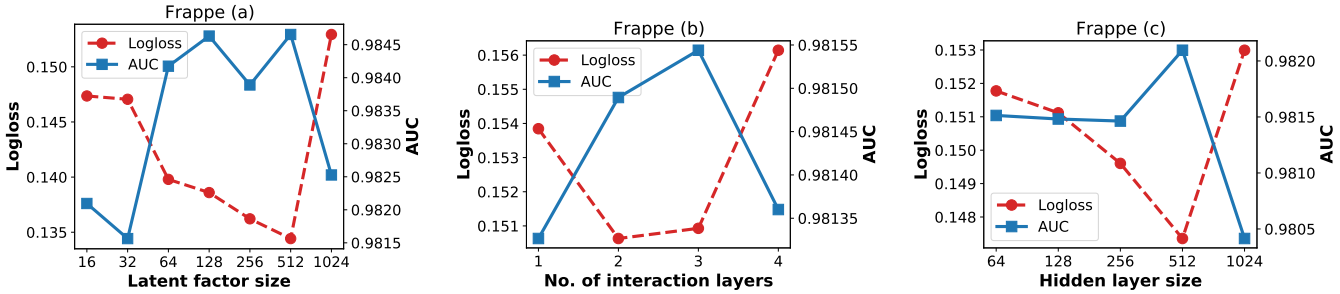


**Figure 3: Effect of various hyperparameters on the performance of DA-HoFM (Frappe)**

can be observed that, there is steady improvement in logloss as number of interaction layers increase from 1 (order = 2) to 3 (order = 4), however, there is slight increase in logloss possibly due to overfitting when order-5 interactions are also added.

**Size of hidden layer** in neural network: Our model uses a neural network to model *implicit* higher order representation of features. A larger hidden layer increases representational capacity of the neural network. This can be observed in Fig. 3.c) where with increase in hidden layer size (from 64 to 1024), model logloss decreases, even as AUC remains flat, up to hidden layer size of 512.

## 4.8 Sparse Feature Interactions

One of the key differences of our proposed model with other higher order interaction models like DCN is use of sparse attention. In this section, we conduct experiments on a synthetic dataset with sparse higher order interaction terms.

The data contains 100k synthesised records ($X \in \mathbb{R}^m$) which are generated via the following generative process

$$\mathbf{X}_i \sim \mathcal{U}(0,1); \mathbf{X} \in \mathbb{R}^{10} \tag{14}$$

$$\mathbf{W}_i \sim \mathcal{U}(1,2); \mathbf{W} \in \mathbb{R}^t; \quad M_i \sim \mathcal{B}(0.01) \tag{15}$$

The corresponding targets $y \in \{0, 1\}$ are generated as

$$y_i^{(lin)} = 1 + \sum_{j=0}^{m} W_j X_{ij}, \tag{16}$$

$$y_i^{(ho)} = \sum_{j=0}^{m}\sum_{k=0}^{m} W_j M_i X_{ij} X_{ik} + \sum_{j=0}^{m}\sum_{k=0}^{m}\sum_{l=0}^{m} W_j M_i X_{ij} X_{ik} X_{il} \tag{17}$$

$$y_i^{(val)} = y_i^{(lin)} + y_i^{(ho)}; \; y_i = 1, \text{if } y_i^{(val)} > y^{(th)} \text{ else } 0 \tag{18}$$

where, $t$ denotes the number of terms in the multivariate polynomial $P(x)$ of degree $d$. $\mathcal{U}(,)$ denotes a uniform distribution and $\mathcal{B}(.)$ represents a Bernoulli distribution. For purpose of our experimentation, we constrain the number of variables $m = 10$ and degree

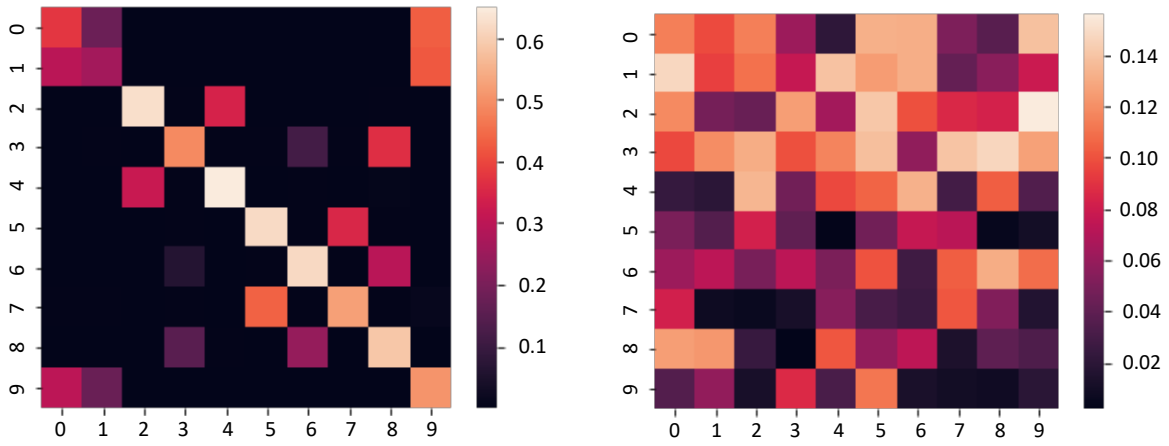


**Figure 4: Visualization of feature-feature interaction for (*left*) DA-HoFM and (*right*) DCNv2 models on synthetic dataset. For DA-HoFM, values are derived by averaging over $\alpha^{(0)}$ over the evaluation set. For DCNv2, the interactions are visualized Frobenius norm of each matrix block in the learned cross-interaction matrix.**

of $P(x)$ to 3. By construction, this dataset has all the linear terms, while the higher order interactions are extremely sparse. $y^{(th)}$ is set such that less than 5% of data is labeled as positive, to simulate response in CTR systems.

We train DA-HoFM and DCNv2 models on this dataset, where we remove the neural network components from both the models. The dataset is split into train, validation and evaluation sets in 8:1:1 ratio, where validation set is used for model selection (logloss metric). Figure 4 shows the feature-feature interaction learned by the two models. For DA-HoFM, feature attention at each layer $\alpha^{(l)}(x)$ (Eq. 4) corresponding to each data sample $x$ provides a straightforward way to visualize normalized feature interaction strengths. We average the $\alpha$ values over all the samples in evaluation set for visualization purpose. For DCNv2, blockwise view of the learned cross-interaction matrix corresponds to the the feature interactions learned by the model [44]. It can be observed that DA-HoFM learns a sparser interaction structure compared to DCNv2. Note that the dense diagonal patch in DA-HoFM output is result of self-interactions. Evaluation logloss for DA-HOFM is 0.0227, compared to 0.0238 for DCNv2, which is consistent with our experiments on the public datasets where the proposed model consistently outperforms DCNv2.

## 4.9 Training & Inference Latency

Apart from satisfactory performance, in production systems (e.g., recommendation system serving Ads), training cost and inference latency are two important factors. Table (5) displays training time per epoch (averaged over 5 epochs) and inference latency (per a batch of 100 samples) for Criteo and Avazu datasets. We omit reporting results for Frappe as it is a relatively small dataset. Even as the proposed **DA-HoFM** model has increased complexity due to the computation of attentional interactions at every layer, resulting into higher training latency compared to simpler models such as FM/AFM, it can be observed that prediction latency is permissible for deployment in real time systems.

**Table 5: Model latency comparison.**

| Dataset | Criteo | | Avazu | |
|---|---|---|---|---|
| | Train/Epoch (s) | Inference (ms) | Tr./Ep. (s) | Inf. (ms) ↓ |
| FM | 660.86 | 3.42 | 299.62 | 2.59 |
| AFM | 866.81 | 3.25 | 432.23 | 2.84 |
| Wide&Deep | 801.5 | 3.35 | 431.89 | 2.76 |
| DeepFM | 827.38 | 3.36 | 417.31 | 2.60 |
| DCNv2 | 788.17 | 3.30 | 432.16 | 2.93 |
| AoAFM | 946.87 | 3.39 | 494.01 | 2.61 |
| **DA-HoFM-NS** | 1450.72 | 3.80 | 902.28 | 3.23 |
| **DA-HoFM** | 1625.82 | 4.01 | 996.81 | 3.12 |
| AFN+ | 1672.71 | 4.38 | 1183.78 | 4.12 |
| InterHAT | 2021.36 | 5.03 | 1338.49 | 3.79 |
| Trans-Enc. | 2217.35 | 5.46 | 1469.39 | 4.14 |

## 5 CONCLUSION & FUTURE DIRECTIONS

In this work, we propose **DA-HoFM**, an *unified* model which combines benefits of multiple layers of *attentional* interaction to compute bounded degree, higher order interaction terms, with representational capacity of neural networks. At the same time, thanks to the *sparse* attention mechanism, it is capable of pruning out the non-important higher order interaction terms. Our model outperforms the state-of-the-art models on benchmark CTR datasets. Additionally, we demonstrate incremental benefits of the individual components of our model via a granular ablation study. Although the proposed model scales linearly in maximum order of interaction terms, one issue with the proposed model is that computation complexity at each layer is quadratic w.r.t the number of features F, and thus one key direction will be to optimize this to sub-quadratic complexity. While we demonstrate the efficacy of our model on CTR datasets, it is generic in nature and can be applied to other supervised learning problems as well.

## ACKNOWLEDGMENTS

We are immensely grateful to Vineet Chaoji for the insightful discussions and reviews during the course of this work.

## REFERENCES

[1] Md. Zahangir Alom, T. Taha, C. Yakopcic, Stefan Westberg, P. Sidike, M. S. Nasrin, B. V. Essen, A. Awwal, and V. Asari. 2018. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *ArXiv* abs/1803.01164 (2018).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2015).

[3] L. Baltrunas, K. Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the Usage and Perception of Mobile App Recommendations In-The-Wild. *ArXiv* abs/1505.03014 (2015).

[4] G. Bebis and M. Georgiopoulos. 1994. Feed-forward neural networks. *IEEE Potentials* 13, 4 (1994), 27–31. https://doi.org/10.1109/45.329294

[5] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, V. Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018).

[6] Mathieu Blondel, Akinori Fujino, N. Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In *NIPS*.

[7] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and C. Lin. 2010. Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. *J. Mach. Learn. Res.* 11 (2010), 1471–1490.

[8] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 61 (Dec. 2015), 34 pages. https://doi.org/10.1145/2532128

[9] S. Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. 2019. An Attentive Survey of Attention Models. *ArXiv* abs/1904.02874 (2019).

[10] Heng-Tze Cheng, L. Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, H. Aradhye, Glen Anderson, G. Corrado, Wei Chai, M. Ispir, Rohan Anil, Zakaria Haque, L. Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (2016).

[11] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. *ArXiv* abs/1909.03276 (2020).

[12] A. Da'u and N. Salim. 2019. Recommendation system based on deep learning methods: a systematic review and new directions. *Artificial Intelligence Review* 53 (2019), 2709–2748.

[13] Emily L. Denton, J. Weston, Manohar Paluri, Lubomir D. Bourdev, and R. Fergus. 2015. User Conditional Hashtag Prediction for Images. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015).

[14] John C. Duchi, Elad Hazan, and Y. Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *J. Mach. Learn. Res.*

[15] Yves Grandvalet and Yoshua Bengio. 2004. Semi-Supervised Learning by Entropy Minimization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'04)*. MIT Press, Cambridge, MA, USA, 529–536.

[16] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI*.

[17] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017).

[18] Xinran He, Junfeng Pan, Ou Jin, T. Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD'14*.

[19] J. Hines. 1996. A logarithmic neural network architecture for unbounded non-linear function approximation. *Proceedings of International Conference on Neural Networks (ICNN'96)* 2 (1996), 1245–1250 vol.2.

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML]

[21] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* 2, 5 (July 1989), 359–366.

[22] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 448–456. https://proceedings.mlr.press/v37/ioffe15.html

[23] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and C. Lin. 2016. Field-aware Factorization Machines for CTR Prediction. *Proceedings of the 10th ACM Conference on Recommender Systems* (2016).

[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[25] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *AAAI*.

[26] Zeyu Li, Wei Cheng, Yang Chen, H. Chen, and W. Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020).

[27] Jianxun Lian, Xiaohua Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guang zhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018).

[28] André F. T. Martins and Ramón Fernández Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *ICML*.

[29] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA, 807–814.

[30] Adam Paszke, S. Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.

[31] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Transactions on Information Systems (TOIS)* 37 (2019), 1 – 35.

[32] Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

[33] S. Rendle. 2010. Factorization Machines. *2010 IEEE International Conference on Data Mining* (2010), 995–1000.

[34] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages.

[35] ANTON R. Schep. 2007. weierstrass.pdf. https://people.math.sc.edu/schep/weierstrass.pdf.

[36] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 815–823.

[37] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, M. Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).

[38] Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15 (2014), 1929–1958.

[39] M. H. Stone. 1948. The Generalized Weierstrass Approximation Theorem. *Mathematics Magazine* 21, 4 (1948), 167–184. http://www.jstor.org/stable/3029750

[40] Christian Szegedy, S. Ioffe, V. Vanhoucke, and Alexander Amir Alemi. 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI*.

[41] Zhulin Tao, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. HoAFM: A High-order Attentive Factorization Machine for CTR Prediction. *Information Processing & Management* 57, 6 (2020), 102076. https://doi.org/10.1016/j.ipm.2019.102076

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[43] Ruoxi Wang, Bin Fu, G. Fu, and M. Wang. 2017. Deep & Cross Network for Ad Click Predictions. *Proceedings of the ADKDD'17* (2017).

[44] Ruoxi Wang, Rakesh Shivanna, D. Cheng, Sagar Jain, Dong Lin, L. Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. *Proceedings of the Web Conference 2021* (2021).

[45] Zhibo Wang, Jinxin Ma, Yongquan Zhang, Q. Wang, Ju Ren, and Peng Sun. 2020. Attention-over-Attention Field-Aware Factorization Machine. In *AAAI*.

[46] Hong Wen, Jing Zhang, Yuan Wang, Fuyu Lv, Wentian Bao, Quan Lin, and Keping Yang. 2020. Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[47] 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. In *IJCAI*.

[48] Ke Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, R. Salakhutdinov, R. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*.

[49] Dawei Yin, Zhenzhen Xue, L. Hong, and Brian D. Davison. 2010. A probabilistic model for personalized tag prediction. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010).

[50] Tom Young, Devamanyu Hazarika, Soujanya Poria, and E. Cambria. 2018. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine* 13 (2018), 55–75.

[51] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: a heterogeneous information network approach. *Proceedings of the 7th ACM international conference on Web search and data mining* (2014).

[52] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2018. Deep Learning based Recommender System: A Survey and New Perspectives.

[53] P. Zhou, Wei Shi, Jun Tian, Zhenyu Qi, B. Li, Hongwei Hao, and Bo Xu. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *ACL*.