A Sparse Deep Factorization Machine for Efficient CTR prediction

Wei Deng*
Purdue University
West Lafayette, IN, USA
deng106@purdue.edu

Junwei Pan*
Yahoo Research
Sunnyvale, CA, USA
jwpan@verizonmedia.com

Tian Zhou
Yahoo Research
Sunnyvale, CA, USA
tian.zhou@verizonmedia.com

Aaron Flores
Yahoo Research
Sunnyvale, CA, USA
aaron.flores@verizonmedia.com

Guang Lin
Purdue University
West Lafayette, IN, USA
guanglin@purdue.edu

ABSTRACT

Click-through rate (CTR) prediction is a crucial task in online display advertising and the key part is to learn important feature interactions. The mainstream models are embedding-based neural networks which provide the end-to-end training by incorporating hybrid components to model both low-order and high-order feature interactions. These models, however, slow down the prediction inference by at least hundreds of times due to the deep neural network (DNN) component. Considering the challenge of deploying embedding-based neural networks for online advertising, we propose to prune the redundant parameters for the first time to accelerate the inference and reduce the run-time memory usage. Most notably, we can accelerate the inference by 46X on Criteo dataset and 27X on Avazu dataset without loss on the prediction accuracy. In addition, the deep model acceleration makes efficient model ensemble possible with low latency and significant gains on the performance.

KEYWORDS

Deep model acceleration, model compression, factorization machine, field importance, structural pruning

1 INTRODUCTION

Online advertising has grown into a hundred-billion-dollar business since 2018, and the revenue has been increasing by more than 20% per year for 4 consecutive years [2]. CTR prediction is critical in the online advertising industry, and the main goal is to deliver the right ads to the right users at the right time. Therefore, how to predict CTR accurately and efficiently has drawn the attention of both the academic and industry communities.

Generalized linear models, such as [3, 32], are scalable and interpretable and have achieved great successes. However, they are limited in their prediction power due to the lack of mechanisms to learn feature interactions. Meaningful feature interactions, like <Gender=Male, Age=20, Industry=Computer Games, Time=9pm>, are useful to improve the expressiveness of the models but it is impractical to manually construct all of them. There is considerable work done towards learning feature interactions and they can be roughly grouped into two categories, namely, *shallow models* and *embedding-based neural networks*.

*Equal contribution

Shallow models include Factorization Machine (FM) [37], Field-aware Factorization Machine (FFM) [19] and Field-weighted Factorization Machine (FwFM) [34]. Factorization machine (FM) [37] models quadratic feature interactions by matrix decomposition. Although theoretically possible to model any orders of feature interactions, FM is mainly used in modeling linear and quadratic feature interactions in practice. Field-aware Factorization Machine (FFM) [19] identified the importance of fields and proposed to learn several latent vectors for a given feature to model its different interaction effects with other features from different fields. This greatly improves the prediction performance, but the number of parameters is also significantly increased. Field-weighted Factorization Machine (FwFM) [34] was proposed to model different field interactions in a much more memory-efficient way. As a consequence, the prediction performance is as good as FFM while the number of parameters is much smaller.

The embedding-based neural networks provide a more powerful non-linear modeling by using DNNs. Wide & Deep [4] proposed to train a joint network that combines a linear model and a DNN model to learn both low-order and high-order feature interactions. However, the cross features in the linear model still require expertise feature engineering and cannot be easily adapted to new datasets. DeepFM [12] handled this issue by modeling low-order feature interactions through the FM component instead of the linear model. Since then, various embedding-based neural networks have been proposed to learn high-order feature interactions: Deep & Cross Network (DCN) [42] models cross features of bounded degrees in terms of layer depth; Neural Factorization Machines (NFM) [15] divises a bilinear interaction pooling to connect the embedding vectors with the DNN component; eXtreme Deep Factorization Machine (XDeepFM) [26] incorporates a Compressed Interaction Network (CIN) and a DNN to automatically learn high-order feature interactions in both explicit and implicit manners. Other relevant work includes [35, 38, 39].

Despite the advances of DNN components in the embedding-based neural networks, the prediction inference is slowed by hundreds of times compared to the shallow models, leading to unrealistic latency for the real-time ad serving system. To handle this issue, we propose a novel field-weighted embedding-based neural network (DeepFwFM) that is particularly suitable for fast and accurate inference. The model itself combines a FwFM component and a vanilla DNN component into a unified model, which is effective

to learn both low-order and high-order feature interactions. More importantly, DeepFwFM shows the unique advantage in structural pruning to greatly reduce the inference time using such a combination, while the other structures may fail in either deep model accelerations or accurate predictions. We support our statement through extensive pruning experiments, which shows that DeepFwFM obtains the best performance not only in predictions, but also in terms of deep model accelerations. Moreover, we observe that a moderate sparsity improves the state-of-the-art result by applying a compact and sufficient structure. In addition, we can achieve 46X speed-ups on Criteo dataset and 27X speed-ups on Avazu dataset without loss on AUC. The deep model accelerations enables the fast predictions in large-scale ad serving systems and also makes deep model ensemble possible within a limited prediction time. Consequently, we can further improve the performance through integrating the predictions of several sparse DeepFwFM models, which still outperform the corresponding baselines due to the powerful prediction performance and a low latency. We have made code available at https://github.com/WayneDW/sDeepFwFM.

2 PRELIMINARIES

Logistic regression has been widely used in CTR prediction in the online advertising industry. Given a dataset $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}$, where y_i is the label and $\boldsymbol{x}_i$ is a m-dimensional sparse feature vector. We can train a logistic regression model (LR) as follows:

$$\min_{\boldsymbol{w}} \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{|\mathcal{D}|} \log(1 + \exp(-y_i \phi_{\text{LR}}(\boldsymbol{w}, \boldsymbol{x_i}))), \tag{1}$$

where λ is the L_2 penalty, and

$$\phi_{\text{LR}}(\boldsymbol{w}) = w_0 + \sum_{i=1}^{m} x_i w_i. \tag{2}$$

Since feature interactions are important for CTR prediction, a straightforward way to capture them is to use a degree-2 polynomial ϕ_{Poly2} instead of ϕ_{LR}, and the mathematical formula is

$$\phi_{\text{Poly2}}(\boldsymbol{w}, \boldsymbol{W}) = w_0 + \sum_{i=1}^{m} x_i w_i + \sum_{i=1}^{m}\sum_{j=i+1}^{m} x_i x_j W_{i,j}, \tag{3}$$

which introduces m^2 parameters and becomes an issue when m is too large and the feature is sparse. Vowpal Wabbit (VW) [22] alleviated this problem by conducting a feature hashing to reduce the the number of parameters.

Estimating the matrix $\boldsymbol{W}$ in (3) given insufficient data is not easy, thus FM [37] proposed to use matrix decomposition to learn the k-dimensional embedding vectors $\{\boldsymbol{e_i}\}_{i=1}^m$ and models the feature interaction $W_{i,j}$ through the inner product $\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle$:

$$\phi_{\text{FM}}(\boldsymbol{w}, \boldsymbol{e}) = w_0 + \sum_{i=1}^{m} x_i w_i + \sum_{i=1}^{m}\sum_{j=i+1}^{m} x_i x_j \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle. \tag{4}$$

A large k can approximate $\boldsymbol{W}$ accurately with enough data; nevertheless, a small k leads to better generalization given insufficient data and thus improves the parameter estimation of $\boldsymbol{W}$ under sparsity. In addition, the use of low dimensional embedding vectors $\{e_i\}_{i=1}^m$ reduces the training time complexity from $O(m^2)$ to $O(mk)$. However, the drawback of FM is that it assumes that each feature has only one latent vector and ignores the field importance on the feature interactions. FFM [19] handled this issue by explicitly training embedding vectors depending on the field of other features:

$$\phi_{\text{FFM}}(\boldsymbol{w}, \boldsymbol{e}) = w_0 + \sum_{i=1}^{m} x_i w_i + \sum_{i=1}^{m}\sum_{j=i+1}^{m} x_i x_j \langle \boldsymbol{e}_{i,f_j}, \boldsymbol{e}_{j,f_j} \rangle, \tag{5}$$

where $f_i \in \{1, 2, ..., n\}$ and n is the number of fields. The inclusion of field importance on the feature interactions enhances the prediction performance, which, however, significantly increases the training time complexity to $O(mnk)$ and consumes too much memory in the online ad serving system.

Considering the high complexity and notable empirical performance of FFM, FwFM [34] proposed to extend FM to model the interaction strengths of different field pairs explicitly via a field matrix $\boldsymbol{R}$. Mathematically, the model is formulated as follows:

$$\phi_{\text{FwFM}}(\boldsymbol{v}, \boldsymbol{e}, \mathbf{R}) = w_0 + \sum_{i=1}^{m} x_i \langle \boldsymbol{e_i}, \boldsymbol{v}_{f_i} \rangle + \sum_{i=1}^{m}\sum_{j=i+1}^{m} x_i x_j \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle \boldsymbol{R}_{f_i, f_j}, \tag{6}$$

where $\boldsymbol{v} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{R}$ is a symmetric matrix of dimension n. The updates on the linear units and quadratic units reduces its complexity significantly through including the field importance information. In addition, FwFM obtains comparable performance as FFM with only hundredths of parameters of FFM and is much more efficient in training.

3 OUR MODEL

Our goal is to model both low-order and high-order feature interactions efficiently using a compact yet effective structure for fast online advertising, with the potential to provide the best deep model accelerations. We propose a field-weighted embedding-based neural network which contains hybrid components as follows:

$$\phi_{\text{DeepFwFM}}(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{e}, \mathbf{R}) = \phi_{\text{Deep}}(\boldsymbol{w}, \boldsymbol{e}) + \phi_{\text{FwFM}}(\boldsymbol{v}, \boldsymbol{e}, \mathbf{R}), \tag{7}$$

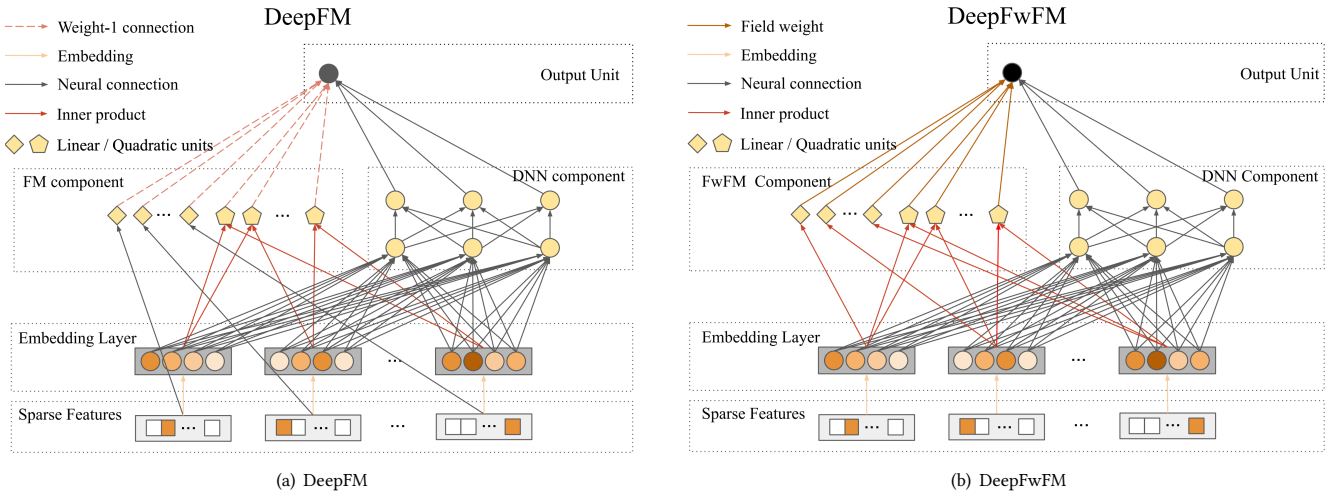
where ϕ_{Deep} is a non-linear transformation of the embeddings through a Multilayer Perceptron (MLP) to learn high-order feature interactions and $\boldsymbol{w}$ is the DNN parameter that contains the weights and bias of MLP.

Our model is a direct improvement of DeepFM [12], where the latter has the following formulation:

$$\phi_{\text{DeepFM}}(\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{e}) = \phi_{\text{Deep}}(\boldsymbol{w}, \boldsymbol{e}) + \phi_{\text{FM}}(\boldsymbol{v}, \boldsymbol{e}). \tag{8}$$

Clearly, the FM component is replaced with the FwFM component in DeepFwFM. DeepFM attempts to model the low-order feature interactions through the weight-1 connections, which is however inefficient due to the incapability to adapt to the local geometry. The inclusion of the field matrix $\boldsymbol{R}$ in DeepFwFM resembles an adaptive preconditioned algorithm to speed up the convergence by approximating the Hessian matrix [24]. In addition, it also shows the potential for further structural pruning. With respect to the updates in the linear terms, the joint structure reduces the number of parameters from m to nk, which speeds up the training and increases the robustness of the model.

Regarding the modeling of low-order feature interactions, the state-of-the-art XDeepFM model proposes to include a compressed



(a) DeepFM (b) DeepFwFM

Figure 1: Model architectures of DeepFM and DeepFwFM. The inner products in the linear part of DeepFwFM are simplified.

interaction network (CIN) to improve the learning power by explicitly approximating a fixed-order polynomial in high dimensions. However, the major drawback is that the CIN has an even larger time complexity than the DNN component as discussed in [26], resulting in expensive computations in large-scale ad systems. Moreover, inspired from the successful second-order optimization algorithms, such as Newton's method, Quasi-Newton method and L-BFGS [27], we argue that it is sufficient to model low-order feature interactions using the second-order FwFM. Further advancements in the learning of low-order terms in the embedding-based neural networks may lead to too much cost and only show marginal improvements.

In summary, our model is simple and efficient to learn both low-order and high-order feature interactions. This model has a joint structure which contains a shared embedding layer, a powerful FwFM component and a vanilla DNN component and achieves a good balance between low-order and high-order feature interactions. The additional field matrix $\boldsymbol{R}$ accelerates the learning of low-order feature interactions via an adaptively preconditioned process and shows the potential for further pruning to yield an even more compact structure.

3.1 Complexity Analysis

The embedding-based neural networks, such as DeepFwFM, DeepFM, NFM, and XDeepFM have similar computational complexity (inference time) and space complexity (number of parameters). However, they are much more computationally intensive compared to shallows models such as FM, FFM and FwFM. We will discuss the computation and space complexity of DeepFwFM in this section.

Computational complexity. The computational cost of DeepFwFM is dominated by the DNN component during the inference time. The embedding layer only has n lookups and therefore leads to little computational cost. Given the embedding size k, the number of layers l, and the number of nodes in each layer h, the number of floating point operations (FLOPs) of the DNN component and the FwFM component are $O(lh^2 + nkh)$ and $O(n^2k)$, respectively. Since h is usually in the order of hundreds while n is in the order of tens, the number of FLOPs in DNN much larger than that of FwFM. Nevertheless, DeepFwFM is slightly slower than DeepFM and NFM due to the use of FwFM [1], but this minor issue can be avoided without affecting the prediction performance when we remove the redundant parameters in the field matrix $\boldsymbol{R}$. In distinction to XDeepFM, the field matrix $\boldsymbol{R}$ shows the computational advantage because a l-layer CIN in XDeepFM takes $O(nkh^2l)$ time [26], which is even slower than the DNN component.

Space complexity. The embedding layer dominates the total number of parameters in DeepFwFM. The number of parameters in the embedding layer, FwFM component and DNN component is $O(mk)$, $O(n^2+nk)$ and $O(lh^2+lnk)$, respectively. m is usually in the order of millions while n and h are in the order of tens or hundreds, therefore the embedding layer has the most number of parameters. For example, the embedding layer accounts for more than 96% of all the parameters in DeepFwFM on Criteo dataset and Avazu dataset.

4 STRUCTURAL PRUNING

Despite successes of applying DNN to model high order interactions [4, 12, 26, 38, 42], the costly computations in DNN bring further challenges in efficient CTR predictions. In real-world online ad serving systems, only a few tenths of a millisecond is acceptable for the inference of each sample (bid request). However, the vanilla DNN component costs milliseconds of latency and fails to meet the online requirement. Therefore, a proper deep model acceleration method is on demand to speed up the predictions. Deep model acceleration consists of three main methods: network pruning [14, 25], low-rank approximation [10, 18], and weight quantization [6, 36], among which network pruning methods have received wide attentions [6, 9, 13, 14, 17, 25] due to their remarkable performance and compatibility. However, network pruning with a uniform sparse

[1] Although the computational cost of FwFM is n-times larger than FM, the inference time is still much smaller than the cost from the DNN component

rate to all the components of a model may not yield the most accelerations. Therefore, structural pruning [23, 33, 43] is often used to introduce component-wise sparsity and have shown better accelerations.

4.1 Structural Pruning on DeepFwFM

Structural pruning has achieved great popularity in computer vision [7, 11, 13, 14, 25, 28, 43]. However, to the best of our knowledge, it has not been applied to embedding-based neural networks to accelerate the CTR predictions. For the first time, we study structural pruning for embedding-based neural networks, specifically for DeepFwFM. In particular, we prune the weights (excluding bias) of the DNN component to remove the neural connections and prune the field matrix $\boldsymbol{R}$ to remove field interactions. In addition, we prune the parameters of the embedding vectors, leading to sparse embedding vectors. We follow a standard pruning algorithm in [9] and present it in Alg.1. Consequently, the resulting architecture of DeepFwFM is shown in Fig. 2. Structural pruning requires an iterative process. In each step, we enumerate the candidate component and prune the redundant parameters based on the corresponding sparse rate. The algorithm tends to prune faster in the early phase when the network is stable and slower in the late phase when the network becomes sensitive to the remaining weights.

Algorithm 1 Structural pruning for a target model, the target sparse rate $S\% = 99\%$ means 99% of the parameters are pruned.

1: **Input** Set the target sparse rate S, damping ratios $\mathbb{D}$ and $\mho$.
2: **Warm up** Initialize a neural network by training i epochs.
3: **Iterative Pruning**
For $k = 1, 2, \ldots$ do
Train the network for one iteration.
Enumerate the candidate component X in a model
Update the current sparse rate $s_X \leftarrow S_X(1 - \mathbb{D}^{k/\mho})$.
Prune the bottom-$s_X\%$ lowest magnitude weights.
4: **Online Prediction**
Transform the sparse model to efficient structure.

Efficient network pruning also rests on the right regularization. The L_0 penalty regularization is theoretically ideal for sparsity detection but is computationally NP-hard in general. Many methods have been proposed to approximate that problem, such as penalized likelihood approaches [40, 44], greedy methods [8, 31], and Bayesian methods [29, 30]. Penalized likelihood approaches, such as Lasso [40], has been widely used to induce sparsity due to its convexity. However, Lasso tends to overshrink large coefficients, which often leads to unsatisfactory solutions [8, 14]. To promote sparse solutions and obtain robust parameter estimates, greedy methods aims to optimize a L_2 penalized problem with an adaptive L_0 constraint to yield sparse solutions. Bayesian methods, such as the stochastic gate approach [30], instead, model the probability of each weight being 0 through continuous stochastic variables to handle the non-differentiable problem of L_0 penalty and further use L_2 to regularize the model. Inspired by these methods, we adopt the L_2 penalty with an implicit L_0 penalty to conduct iterative pruning.

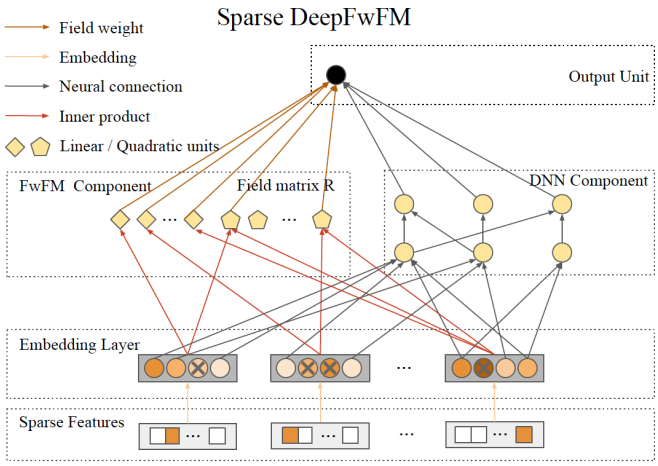


Figure 2: Sparse DeepFwFM: a pruned architecture of DeepFwFM. The inner products in the linear part are simplified.

4.2 Reduction of computational complexity

The DNN component is the bottleneck that causes the high inference time. After the pruning of the DNN component, the FwFM component becomes the limitations, which requires further pruning on the field matrix $\boldsymbol{R}$. The pruning of the embedding layer has no significant speed-ups on the computations. With a medium $S_{\text{dnn}}\%$ sparsity on the weights in the DNN component (excluding the bias), the corresponding speed-ups can be close to the ideal $1/(1 - S_{\text{dnn}}\%)$ times. However, when the sparsity $S_{\text{dnn}}\%$ is higher than 95%, we may not achieve the ideal rate because of the computations in the biases and the overhead of sparse structures, such as the compressed row storage (CRS). As to the pruning of the field matrix $\boldsymbol{R}$, the speed-ups becomes more significant as we increase the sparsity $S_{\text{dnn}}\%$ in the DNN component.

4.3 Reduction of space complexity

Pruning also dramatically reduces the number of parameters in DeepFwFM and therefore save lots of memory. In the embedding layer, a $S_{\text{emb}}\%$ sparsity reduces the number of parameters from mk to $(1 - S_{\text{emb}}\%)mk$. While in the DNN component, the number of weights (excluding the bias) can be reduced from $O(nkh + lh^2)$ to $O\left((nhk + lh^2)(1 - S_{\text{dnn}}\%)\right)$ by storing the sparse weight matrix through the CRS. Similarly, a $S_{\text{R}}\%$ sparsity on the field matrix $\boldsymbol{R}$ reduces the number of parameters proportionally. Since the parameters in the embedding vectors dominate the total parameters in DeepFwFM, a $S_{\text{emb}}\%$ sparsity on the embedding vectors leads to the total memory reduction by roughly $1/(1 - S_{\text{emb}}\%)$ times.

5 EXPERIMENTS

5.1 Experimental setup

5.1.1 Data sets. 1. Criteo Dataset: It is a well-known benchmark dataset for CTR prediction [21]. It contains 45 million samples and each sample has 13 numerical features (counting) and 26 categorical features. Regarding the numerical features, there are methods such as log transformation, binning and GBDT feature transformation

Table 1: Statistics of datasets

Data	Training set	# Fields	# Numerical	# Features
Criteo	41.3M	39	13	1.33M
Avazu	32.3M	23	0	1.54M

[16] to transform them to categorical features. We adopt the log transformation of $\log(x)^2$ if $x > 2$ proposed by the winner of Criteo Competition [2] to normalize the numerical features. We count the frequency of categorical features and treat all the features with a frequency less than 8 as unknown features. We randomly split the datasets into two parts: 90% is used for training and the rest is left for testing. 2. Avazu Dataset: We use the 10 days of click-through log on users' mobile behaviors and randomly split 80% of the samples for training and leave the rest for testing. We treat the features with frequency less than 5 as unknown and replace them by a field-wise default feature. A description of the two datasets is shown in Table.1.

5.1.2 Evaluation metrics. To evaluate the prediction performance on Criteo dataset and Avazu dataset, we use Logloss and AUC where Logloss is the cross-entropy loss to evaluate the performance of a classification model and AUC is the area under the ROC curve to measure the probability that a random positive sample is ranked higher than a random negative sample.

5.1.3 Baselines. Among the popular embedding-based neural networks such as PNN [35], Deep & Wide [4], Deep Crossing [38], Deep Cross [42], DeepFM [12], NFM [15] and XDeepFM [26], we choose the last three because they have similar architectures to DeepFwFM and they are also the state-of-the-art models for CTR prediction. As a result, the 6 baseline models to evaluate DeepFwFM are LR (Logistic regression), FM [37], FwFM [34], DeepFM [12], NFM [15], XDeepFM [26].

5.1.4 Implementation details. We train our model using PyTorch. To make a fair comparison on Criteo dataset, we follow the parameter settings in [12, 26] and set the learning rate to 0.001. The embedding size is set to 10. The default settings for the DNN components of DeepFM, NFM, and XDeepFM are: (1) the network architecture is $400 \times 400 \times 400$; (2) the dropout rate is 0.5. Specifically for XDeepFM, the cross layer in the CIN architecture is $100 \times 100 \times 50$. We fine-tuned the L_2 penalty and set it to 3e-7. We use the Adam optimizer [20] for all of the experiments and the minibatch is chosen as 2048. Regarding Avazu dataset, we keep the same settings except that the embedding size is 20, the L_2 penalty is 6e-7, and the DNN network structure is $300 \times 300 \times 300$. Regarding the training time in practice, all the models don't differ each other too much. FwFM and DeepFwFM are slightly faster than DeepFM and xDeepFM due to the innovations in the linear terms.

5.2 Dense model evaluations

The evaluations of dense models without pruning show the maximum potential that the over-parameterized models perform. From Table 2, we observe that LR underperforms the other methods by at least 0.7% on Criteo dataset and 1.7% on Avazu dataset in terms of AUC, which shows that feature interactions are critical to improving the CTR prediction. Most of the embedding-based neural networks outperform the low-order methods such as LR and FM, implying the importance of modeling high-order feature interactions. However, the low-order FwFM still wins over NFM and DeepFM, showing the strength of field matrix $\boldsymbol{R}$ to learn second-order feature interactions to adapt to the local geometry. NFM utilizes a black-box DNN to implicitly learn the low-order and high-order feature interactions, which may potentially over-fit the datasets due to the lack of mechanism to identify the low-order feature interactions explicitly. Among all the embedding-based neural network models, XDeepFM and DeepFwFM achieves the best result on Criteo dataset and Avazu dataset and outperform the other models by roughly 0.7% on Criteo dataset and 0.4% on Avazu dataset in terms of AUC. However, the inference time of XDeepFM is almost ten times longer than DeepFwFM, showing the inefficiency in real-time predictions for large-scale ad serving systems.

5.3 Sparse model evaluations

Uncovering the right sub-networks rests on good initiazations [11, 14]. We first train the network by 2 epochs, and then run 8 epochs for the pruning experiments. The damping ratios are set to $\mathcal{D} = 0.99$ and $\Omega = 100$ on Criteo dataset and Avazu dataset to iteratively increase the sparse rates. We prune the network every 10 iterations to reduce the computational cost.

5.3.1 DNN pruning. When we prune the DNN component, only the weights of the DNN component are pruned. The biases of DNN, the field matrix $\boldsymbol{R}$ and the parameters in the embedding layer is treated as usual. We try different pruning rates to study the prediction performance and deep model accelerations. To show the superiority of network pruning on a large network over training from a smaller one, we compare the networks with different sparse rates to the networks of smaller structures. As shown in Table.3, we see that the DeepFwFMs with sparse DNN components outperforms the dense DeepFwFMs even when the sparse rate is as high as 95% on Criteo dataset. This phenomenon remains the same until we increase the sparsity to 99% on Criteo dataset. By contrast, the corresponding small networks with similar number of parameters such as N-25 [3] and N-15 obtain much worse results than the original N-400, *showing the power of pruning an over-parametrized network over training from a smaller one*. On Avazu dataset, we obtain the same conclusions. The sparse model obtains the best prediction performance with 90% sparsity and only starts to perform worse when the sparsity is larger than 99.5%. Regarding the deep model acceleration, we see from Fig.3(a) and Fig.4(a) that a larger sparsity always brings a lower latency and when the sparsity is 98% on Criteo dataset and 99% on Avazu dataset, we realize the performance is still surprisingly better than the original dense network and we achieve 16X speed-ups on both datasets.

5.3.2 Pruning of the field matrix $\boldsymbol{R}$. After applying a high sparsity on the DNN component, we already obtain significant speed-ups which is close to 20X. To further boost the accelerations, increasing the sparsity on the DNN may risk in decreasing the performance and

[2] https://www.csie.ntu.edu.tw/ r01922136/kaggle-2014-criteo.pdf

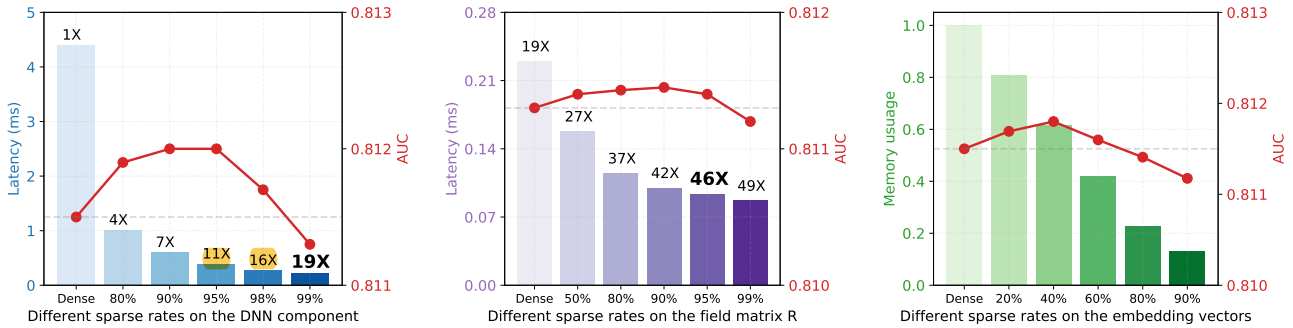
[3] A model with 25 nodes in each DNN layer is referred to as N-25.

Table 2: Model comparison on the Criteo and Avazu datasets.

Models	Criteo				Avazu			
	Test		# Parameters	Latency (ms)	Test		# Parameters	Latency (ms)
	LogLoss	AUC			LogLoss	AUC		
LR	0.4615	0.7881	1,326,056	0.001	0.3904	0.7617	1,544,393	0.001
FM	0.4565	0.7949	14,586,606	0.005	0.3816	0.7782	32,432,233	0.009
FwFM	0.4466	0.8049	13,261,682	0.145	0.3764	0.7866	30,888,853	0.105
DeepFM	0.4495	0.8036	15,064,206	4.181	0.3780	0.7852	32,751,433	2.719
NFM	0.4497	0.8030	15,204,206	4.091	0.3777	0.7854	32,689,033	2.704
XDeepFM	0.4420	0.8102	15,508,958	40.85	0.3749	0.7894	32,689,033	7.129
DeepFwFM	0.4403	0.8116	13,739,321	4.271	0.3751	0.7893	31,208,053	2.824

Table 3: DeepFwFMs with sparse DNN components v.s. DeepFwFMs with smaller DNN components. The DeepFwFM model with X nodes in each DNN layer is referred to as N-X. The dense alternatives with smaller DNN components are chosen to have the same level of parameters of the sparse network, e.g. in Criteo dataset, the sparse model D-99% & R-0% & F-0% has 6361 parameters and the dense alternative N-15 has 6360 parameters.

Criteo						Avazu					
Sparse model	Test		Model	Test		Sparse Model	Test		Model	Test	
	Logloss	AUC		Logloss	AUC		LogLoss	AUC		LogLoss	AUC
No Pruning	0.4403	0.8115	N-400	0.4403	0.8115	No Pruning	0.3751	0.7893	N-300	0.3751	0.7893
D-90% & R-0% & F-0%	**0.4398**	**0.8120**	N-87	0.4414	0.8104	D-90% & R-0% & F-0%	**0.3747**	**0.7898**	N-57	0.3757	0.7883
D-95% & R-0% & F-0%	0.4399	**0.8120**	N-51	0.4421	0.8098	D-95% & R-0% & F-0%	0.3747	0.7896	N-32	0.3762	0.7875
D-98% & R-0% & F-0%	0.4401	0.8117	N-25	0.4429	0.8089	D-99% & R-0% & F-0%	0.3748	0.7895	N-9	0.3769	0.7864
D-99% & R-0% & F-0%	0.4405	0.8113	N-15	0.4438	0.8078	D-99.5% & R-0% & F-0%	0.3749	0.7892	N-6	0.3771	0.7862



(a) DNN pruning with no pruning on the field matrix $\boldsymbol{R}$ and embedding vectors.

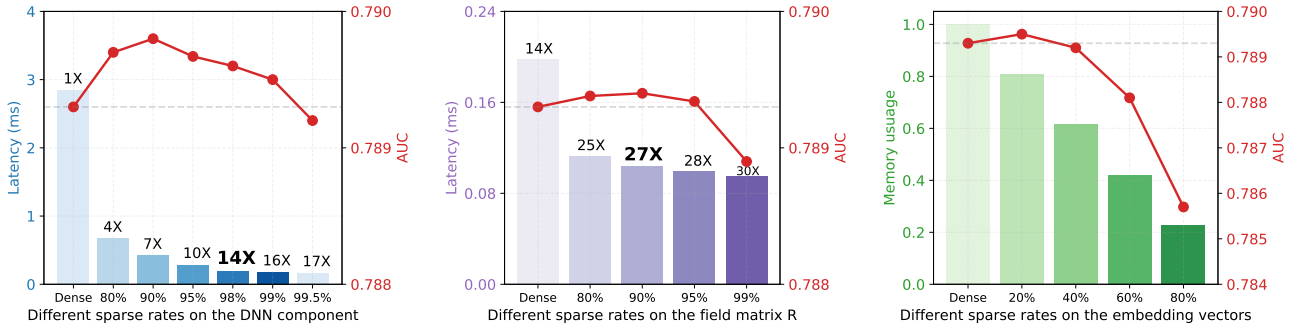
(b) Field matrix R pruning with 99% sparsity on the DNN component and 40% sparsity on embedding vectors.

(c) Embedding vector pruning with no pruning on the field matrix $\boldsymbol{R}$ and the DNN component.

Figure 3: Criteo: Structural pruning for accelerations and memory savings.

doesn't yield obvious accelerations. Instead, we propose to prune the field matrix $\boldsymbol{R}$ given a 99% sparsity for the DNN component on the Criteo dataset (98% sparsity on the Avazu dataset). From Fig.3(b) and Fig.4(b), we observe that we can adopt 95% sparsity on the field matrix $\boldsymbol{R}$ without sacrificing the performance. Additionally, the predictions can be further accelerated by two to three times and obtain 46X and 27X speed-ups without sacrificing the performance.

5.3.3 Embedding pruning. As to the pruning of embeddings, we find that setting a global threshold for the embeddings of all fields obtains a slightly better performance than setting individual thresholds for the embedding vector from each field. Therefore, we conduct all the following experiments based on a global threshold. As shown in Fig.3(c) and Fig.4(c), Criteo can adopt a high sparse rate, such 80%, to remain the same performance on Criteo dataset; by contrast, the model is sensitive on Avazu dataset and starts to decrease the performance when a 60% sparsity is applied. From Table. 4, we see most of the models outperforms the baseline models (referred to as E-X) with a smaller embedding size, which sheds



(a) DNN pruning with no pruning on the field matrix $\boldsymbol{R}$ and embedding vectors.

(b) Field matrix R pruning with 98% sparsity on the DNN component and 0% sparsity on embedding vectors.

(c) Embedding vector pruning with no pruning on the field matrix $\boldsymbol{R}$ and the DNN component.

Figure 4: Avazu: Structural pruning for accelerations and memory savings.

light on the use of large embedding sizes and pruning techniques to over-parameterize the network while avoiding over-fitting.

5.3.4 Structural Pruning. From the above three sets of experiments, we see that the DNN component and the field matrix $\boldsymbol{R}$ accept a much higher sparse rate to remain the same prediction performance, which inspires us to *apply different pruning rates on the hybrid components*. We denote a sparse DeepFwFM model as sDeepFwFM; similarly, we have sFwFM, sDeepFM, sNFM and sXDeepFM. As shown in Table.5 and Table.6, for the *performance-driven tasks*, we can improve the state-of-the-art AUC from 0.8116 to 0.8223 on Criteo dataset via a sparse DeepFwFM where 90% of the parameters in both the DNN component and the field matrix $\boldsymbol{R}$ and 40% of the parameters in the embedding vectors are pruned, and such model is denoted by D-90% & R-90% & F-40%. On Avazu dataset, a sparse DeepFwFM with structure D-90% & R-90% & F-20% improves the AUC from 0.7893 to 0.7897. For the *memory-driven tasks*, the memory savings are up to 10X and 2.5X on Criteo dataset and Avazu dataset, respectively. Compare to the pure embedding pruning in Table.4, we notice that hybrid pruning allows a higher sparse rate. For the latency-driven tasks, we achieve 46X speed-ups on Criteo dataset using a sDeepFwFM with the structure D-99% & R-95% & F-40% and 27X speed-ups on Avazu dataset using the structure D-98% & R-90% & F-0% without loss of accuracy.

For the other models, we also try the corresponding best structure for accelerating the predictions without sacrificing the performance. For the particular CIN component in XDeepFM, we denote the 99% sparsity on the CIN component by C-99%. We report the results in Table.7 and observe that all the embedding based neural networks adopt high sparse rates to maintain the performance. Moreover, sDeepFwFM is comparable to sDeepFM and sNFM in terms of prediction time but improves the AUC by at least 0.8% on Criteo dataset and 0.4% on Avazu dataset. sDeepFwFM obtains a similar prediction performance as sXDeepFM but is almost 10X faster than sXDeepFM. This shows the superiority of sDeepFwFM over sXDeepFM in large-scale online ad serving systems.

5.4 Efficient model ensemble

Model ensemble is a popular technique in computer vision [1, 5, 41] and Kaggle competitions, which enhances the predictions of non-linear systems significantly but slows down the inference. The acceleration on each sparse DeepFwFM model makes model ensemble much more efficient. In this section, we evaluate the performance of integrating different number of testing models based on different sparse rates. In addition, we study the predictions under the constraint of limiting the latency within 0.5ms. We see from Table.8 that the model ensemble of several sDeepFwFMs boosts the AUC by as large as 0.3% and 0.5% on Criteo dataset and Avazu dataset, respectively. On the contrary, sXDeepFM is acceptable in this case only if we apply an extremely sparse structure D-99.8% & C-99.8% & F-40%, which leads to a worse performance due to the risk in destroying the ideal structure for predictions. As to the ensemble of sNFMs and sDeepFMs, the AUCs are lower than that of sDeepFwFMs by roughly 0.4% and 0.3% on Criteo dataset and Avazu dataset, respectively. In summary, the compact and sufficient structure of DeepFwFM shows great potential in large-scale ad serving systems to yield fast and accurate predictions.

The above experiments are tested based on sequentially implementing all the models without considering parallel strategies. The model acceleration performance can be further improved using parallel techniques.

6 CONCLUSIONS

In this paper, we propose a field-weighted embedding-based neural network, DeepFwFM, for structural pruning to learn a compact and sufficient structure for efficient CTR predictions. To the best of our knowledge, this is the first work of network pruning applied to the area of CTR prediction in online advertising to solve the high-latency issues of embedding-based neural networks. We observe that network pruning is not only powerful to prune redundant parameters to alleviate over-fitting but also achieves significant acceleration on the inference time and shows a pronounced reduction on the memory usage with little impact on the prediction performance. This strategy overcomes the challenge of efficient

Table 4: DeepFwFM with sparse embedding vectorss v.s. DeepFwFM with smaller embedding sizes. The DeepFwFM model with embedding size X is referred to as E-X. On Criteo, D-0% & R-0% & F-20% and E-8 have a similar level of parameters.

Criteo						Avazu					
Sparse model	Test		Model	Test		Sparse Model	Test		Model	Test	
	Logloss	AUC		Logloss	AUC		Logloss	AUC		Logloss	AUC
No Pruning	0.4403	0.8115	E-10	0.4403	0.8115	No Pruning	0.3751	0.7893	E-20	0.3751	0.7893
D-0% & R-0% & F-20%	0.4402	0.8116	E-8	0.4404	0.8115	D-0% & R-0% & F-20%	**0.3750**	**0.7895**	E-16	0.3752	0.7890
D-0% & R-0% & F-40%	**0.4401**	**0.8118**	E-6	0.4407	0.8113	D-0% & R-0% & F-40%	0.3751	0.7891	E-12	0.3750	0.7889
D-0% & R-0% & F-60%	0.4404	0.8116	E-4	0.4412	0.8106	D-0% & R-0% & F-60%	0.3762	0.7881	E-8	0.3765	0.7874
D-0% & R-0% & F-80%	0.4406	0.8114	E-2	0.4423	0.8094	D-0% & R-0% & F-80%	0.3773	0.7857	E-4	0.3770	0.7859

Table 5: Structural pruning of DeepFwFM on Criteo dataset. D-90% & R-90% & F-40% is short for the sparse DeepFwFM which has 90% sparse rate on the DNN component and the field matrix R and a 40% sparse rate on the embedding vectors.

Dataset	Goal	Structural Pruning	Test		# Parameters	Latency (ms)
			Logloss	AUC		
Criteo	None	No Pruning	0.4403	0.8116	13,739,321	4.271
	High performance	D-90% & R-90% & F-40%	**0.4395**	**0.8123**	8,012,094	0.469
	Low memory	D-90% & R-90% & F-90%	0.4404	0.8114	**1,376,431**	0.472
	Low latency	D-99% & R-95% & F-40%	0.4405	0.8114	7,413,578	**0.093**

Table 6: Structural pruning of DeepFwFM on Avazu dataset.

Dataset	Goal	Structural Pruning	Test		# Parameters	Latency (ms)
			Logloss	AUC		
Avazu	None	No Pruning	0.3751	0.7893	31,208,053	2.824
	High performance	D-90% & R-90% & F-20%	**0.3748**	**0.7897**	24,808,262	0.422
	Low memory	D-90% & R-90% & F-60%	0.3753	0.7892	**9,322,791**	0.318
	Low latency	D-98% & R-90% & F-0%	0.3753	0.7894	30,859,675	**0.104**

Table 7: Evaluation of sparse models on Criteo and Avazu datasets. For each individual model, we only report the most efficient structure that yields the best accelerations with almost no sacrifice on the prediction performance.

Criteo					Avazu			
Models	Test		Structure	Latency (ms)	Test		Structure	Latency (ms)
	LogLoss	AUC			LogLoss	AUC		
sDeepFM	0.4496	0.8032	D-98% & F-40%	0.114	0.3782	0.7851	D-98% & F-20%	**0.102**
sNFM	0.4494	0.8031	D-98% & F-40%	0.114	0.3778	0.7854	D-98% & F-20%	**0.102**
sXDeepFM	0.4421	0.8102	D-99% & C-99% & F-40%	0.907	0.3750	0.7893	D-98% & C-98% & F-0%	0.927
sDeepFwFM	**0.4405**	**0.8114**	D-99% & R-95% & F-40%	**0.093**	**0.3753**	**0.7894**	D-98% & R-90% & F-0%	0.104

Table 8: Prediction performance when we limit the prediction time within 0.5 ms on Criteo dataset and Avazu dataset. The term "4 sDeepFMs" denotes the model ensemble via four sparse DeepFM models.

Criteo				Avazu			
Model ensemble	Structure	Test		Model ensemble	Structure	Test	
		Logloss	AUC			Logloss	AUC
4 sDeepFMs	D-98% & F-40%	0.4419	0.8104	4 sDeepFMs	D-98% & F-20%	0.3731	0.7915
4 sNFMs	D-98% & F-40%	0.4417	0.8109	4 sNFMs	D-98% & F-20%	0.3733	0.7912
1 sXDeepFM	D-99.8% & C-99.8% & F-40%	0.4445	0.8069	1 sXDeepFM	D-99.6% & C-99.6% & F-0%	0.3761	0.7882
5 sDeepFwFMs	D-99% & R-95% & F-40%	**0.4372**	**0.8151**	4 sDeepFwFMs	D-98% & R-90% & F-0%	**0.3712**	**0.7945**

online advertising with the embedding-based neural networks. Furthermore, the deep model acceleration on sparse DeepFwFMs also makes efficient model ensemble desirable for online settings with significant gains on the performance and a limited impact on the prediction latency.

REFERENCES

[1] William H. Beluch, Tim Genewein, Andreas NÃijrnberger, and Jan M. KÃűhler. 2018. The Power of Ensembles for Active Learning in Image Classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Interactive Advertising Bureau. 2019. IAB internet advertising revenue report. In *https://www.iab.com/wp-content/uploads/2019/05/Full-Year-2018-IAB-Internet-Advertising-Revenue-Report.pdf*.

[3] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2014. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 61 (Dec. 2014), 34 pages. https://doi.org/10.1145/2532128

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[5] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*. 3642–3649.

[6] Matthieu Courbariaux and Yoshua Bengio. 2016. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR* (2016). arXiv:1602.02830

[7] Yann Le Cun, John S. Denker, and Sara A. Solla. 1990. Optimal Brain Damage. In *Advances in Neural Information Processing Systems 5 (NIPS)*. Morgan Kaufmann, 598–605.

[8] G. Davis, Stéphane Georges Mallat, and Marco Avellaneda. 1997. Adaptive greedy approximations. *Constructive Approximation* 13 (1997), 57–98.

[9] Wei Deng, Xiao Zhang, Faming Liang, and Guang Lin. 2019. An Adaptive Empirical Bayesian Method for Sparse Deep Learning. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.*

[10] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. In *Advances in Neural Information Processing Systems 27 (NIPS)*. 1269–1277.

[11] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

[13] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations 2016 (ICLR)*.

[14] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems 28 (NIPS)*. 1135–1143.

[15] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. *CoRR* abs/1708.05027 (2017). arXiv:1708.05027 http://arxiv.org/abs/1708.05027

[16] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (ADKDD'14)*. ACM, New York, NY, USA, Article 5, 9 pages. https://doi.org/10.1145/2648584.2648589

[17] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531

[18] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up Convolutional Neural Networks with Low Rank Expansions. *CoRR* abs/1405.3866 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1405.html#JaderbergVZ14

[19] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 43–50. https://doi.org/10.1145/2959100.2959134

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations 2015 (ICLR)*.

[21] Criteo Labs. 2014. Display Advertising Challenge. In *https://www.kaggle.com/c/criteo-display-ad-challenge*.

[22] John Langford, Lihong Li, and Alex Strehl. 2007. Vowpal Wabbit. In *https://github.com/VowpalWabbit*.

[23] Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. 2019. Structured Pruning of Neural Networks With Budget-Aware Regularization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. 2016. Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 1788–1794.

[25] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations 2017 (ICLR)*.

[26] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *CoRR* abs/1803.05170 (2018). arXiv:1803.05170 http://arxiv.org/abs/1803.05170

[27] Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *MATHEMATICAL PROGRAMMING* 45 (1989), 503–528.

[28] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).

[29] Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian Compression for Deep Learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*. 3288–3298.

[30] Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *International Conference on Learning Representations 2018 (ICLR)*.

[31] S.G. Mallat and Zhifeng Zhang. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* 41 (1993), 3397–3415.

[32] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[33] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance Estimation for Neural Network Pruning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[34] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. *CoRR* abs/1806.03514 (2018). arXiv:1806.03514 http://arxiv.org/abs/1806.03514

[35] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based Neural Networks for User Response Prediction. *CoRR* abs/1611.00144 (2016). arXiv:1611.00144 http://arxiv.org/abs/1611.00144

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. http://arxiv.org/abs/1603.05279 cite arxiv:1603.05279.

[37] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[38] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. 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 (KDD '16)*. ACM, New York, NY, USA, 255–262. https://doi.org/10.1145/2939672.2939704

[39] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2018. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. *CoRR* abs/1810.11921 (2018). arXiv:1810.11921 http://arxiv.org/abs/1810.11921

[40] Robert Tibshirani. 1994. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B* 58 (1994), 267–288.

[41] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*.

[42] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. *CoRR* abs/1708.05123 (2017). arXiv:1708.05123 http://arxiv.org/abs/1708.05123

[43] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

[44] Hui Zou. 2006. The Adaptive Lasso and Its Oracle Properties. *J. Amer. Statist. Assoc.* 101 (2006), 1418–1429.