Large Scale Recommendation System

Balaji Balasubramanian
1217036457
bbalas10@asu.edu

Rounak Sengupta
1215043206
rsengup7@asu.edu

Shreesh Nayak
1217214310
sdnayak1@asu.edu

***Abstract*—The widespread application of deep learning has yielded immense success on speech recognition, computer vision and natural language processing. While a lot of research and significant efforts have been devoted to performance analysis and optimization for DNNs, a very small amount of focus has been devoted to personalized recommendation systems despite their importance and the amount of compute cycles they consume. In this paper, we will focus on improving a large scale recommendation system. Using the DLRM benchmark, we demonstrate its usefulness for algorithmic experimentation and system co-design. Our performance is characterized and then this is compared with existing recommendation systems.**

***Index Terms*—Recommendation, deep learning, deep neural networks, matrix factorization, click through rate prediction, attention, compositional embeddings.**

I. Introduction

Recommendation systems are used for an assortment of applications everywhere on the web, including providing recommendations, determining click-through rate probabilities for an advertisement and rankings. Many companies use recommendation systems for a variety of different tasks. Facebook uses recommendation engines for recommending pages that we might like or groups that we should join. Netflix uses a recommendation engine to suggest to us, what to watch next. Amazon uses a similar recommendation system to suggest to us what to buy. In spite of the fact that these techniques have had a long history, these methodologies have as of late only recently embraced neural networks. In this work, we are mainly targeting large-scale Recommender Systems for Social Networks as their workloads pose unique challenges in terms of memory capacity, irregular memory accesses, diversity in compute intensive and memory intensive models, and high-throughput and low-latency optimization targets. Furthermore, available implementations of DNN-based recommendation systems are not representative of production scale ones.

Mainly, we'll be tackling problems of reducing latency of inferences with faster training cycles, and improving the accuracy of recommendations by tinkering with our algorithm while also exploring the problem of memory efficiency.

The hope is that this work will help us optimize end-to-end personalized recommendation systems currently running in data centers and motivate additional optimization techniques that address similar challenges.

II. Related Work

A. The Architectural Implications of Facebook's DNN-based Personalized Recommendation [1]

Personalized recommendation for content ranking is largely accomplished due to the widespread application of deep learning. However, despite their widespread success, relatively little research has been done in the context of recommendation systems. This paper presents a set of open source real-world, production-scale DNNs for personalized recommendation coupled with relevant performance metrics for evaluation. It focuses on Intel Haswell, Broadwell, and Skylake servers, and conducts an in depth analysis which underpins broad future system design and optimization for at-scale recommendation. It proposes variations for batching and co-location to drastically improve latency-bounded throughput.

B. Deep Neural Networks for YouTube Recommendations [2]

YouTube represents one of the largest scale and most sophisticated industrial recommendation systems in existence. This paper introduces a deep candidate generation model and a separate deep ranking model. It also provides insights derived from designing, iterating and maintaining a massive recommendation system. A high level overview of the system is provided with a focus on performance improvements brought by deep learning.

C. Scalable Deep Neural Networks Via Low Rank Matrix Factorization [3]

Deep Neural Networks (DNNs) can be extremely large in size, hence it is important to compress them

for real-world applications which operate on resource-constrained devices. Changing the model size once training is completed is a difficult task. This paper proposes a novel method that enables DNNs to flexibly change their size after training. The weight matrices of the DNNs are factorized via single value decomposition (SVD) and their ranks after changed according to the target size. This paper also introduces simple criteria that characterize the importance of each basis and layer, which enables to effectively compress the error and complexity of models as little as possible. This method exhibits favorable performance compared with existing methods.

D. Attentive Collaborative Filtering: MultiMedia Recommendation with Item and Component Level Attention [4]

Multimedia content is dominating today's Web information. The nature of multimedia user-item interactions is 1/0 binary implicit feedback which can be collected at a larger scale with a much lower cost than explicit feedback. However, the majority of existing collaborative filtering (CF) systems are not well-designed for multimedia recommendation, since they ignore the implicitness in users' interactions with multimedia content. This paper introduces a novel attention mechanism in collaborative filtering to address the challenging item-level and component-level implicit feedback in multimedia recommendation. This attention model is a neural network which consists of two attention modules: (1) the component-level attention module, starting from any content feature extraction network which learns to select informative components of multimedia items, and (2) the item-level attention module, which learns to score the item preferences. This approach significantly outperforms state-of-the-art CF methods.

E. Neural Network Matrix Factorization [5]

Most computer systems view data in the form of an array or matrix. Matrix factorization techniques attempt to recover missing or corrupted entries. This is done by assuming that the matrix can be written as the product of two low-rank matrices, this is known as the inner product. This paper introduces an arbitrary function which is learned from data at the same time as we learn the latent feature vectors. This function replaces the inner product which is used for matrix factorization. Particularly, the inner product is replaced by a multi-layer feed-forward neural network. This paper also proposes learning by alternating between optimizing the network for fixed latent features, and optimizing the latent features for a fixed network. This approach dominates standard low-rank techniques on a suite of benchmark but is dominated by some recent proposals that take advantage of the graph features.

F. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction [6]

In order to maximize CTR for recommender systems, it is essential to learn practical feature interactions behind user behaviors. Existing models seem to have a strong bias towards low-order or high-order interactions, or they require expert feature engineering. This paper derives an end-to-end learning model that emphasizes both low- and high-order feature interactions. The proposed model, DeepFM, combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. DeepFM has a shared input to its wide and deep parts, with no need of feature engineering besides raw features. This approach has proven to be effective and efficient over the existing models for CTR prediction on both benchmark data and commercial data.

G. Deep Context-Aware Recommender System Utilizing Sequential Latent Context [7]

Context-aware recommender systems (CARSs) apply sensing and analysis of user context in order to provide highly personalized services. The addition of context to a recommendation model can be extremely challenging, because this addition increases both the dimensionality and sparsity of the model. This paper proposes new context-aware recommendation models that extend the neural collaborative filtering approach and learn nonlinear interactions between latent features of users, items, and contexts which take into account the sequential latent context representation as part of the recommendation process. This sequential latent context-aware model (SLCM) suprasses state of the art context-aware recommender system models.

H. Compositional Embeddings Using Complementary Partitions for Memory-Efficient Recommendation Systems [8]

Modern deep learning-based recommendation systems exploit hundreds to thousands of different categorical features, each with millions of different categories ranging from clicks to posts. Categorical data is highly diverse, hence embeddings are required in order to map each category to a unique dense representation within

an embedded space. Creating these embedding tables constitute the primary memory bottleneck during both training and inference. This paper proposes a novel approach for reducing the embedding size in an end-to-end fashion by exploiting complementary partitions of the category set to produce a unique embedding vector for each category without explicit definition. It introduces the concept of storing multiple smaller embedding tables based on each complementary partition and combining embeddings from each table to define a unique embedding for each category at smaller cost. This approach has proven to be effective over the hashing trick for reducing the size of the embedding tables in terms of model loss and accuracy, while retaining a similar reduction in the number of parameters.

I. Deep Matrix Factorization Models for Recommender Systems [9]

Recommender systems make personalized recommendation with user-item interaction ratings, implicit feedback and auxiliary information. Matrix factorization is the basic idea to predict a personalized ranking over a set of items for an individual user with the similarities among users and items. This paper proposes a novel matrix factorization model with a neural network architecture. Through this architecture, users and items are projected into low dimensional vectors which allows for full utilization of both explicit ratings and implicit feedback in two ways. This method is highly effective on the benchmarks on which it has been tested.

J. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks [10]

Factorization Machines (FMs) are a supervised learning approach that enhance the linear regression model by incorporating the second-order feature interactions. However, FM can be hindered by its modelling of all feature interactions with the same weight, as not every single feature interaction is equally useful and predictive. This paper proposes a novel model called Attentional Factorization Machine (AFM) which learns the importance of each feature interaction from data via a neural attention network. This approach provides an 8.6% relative improvement over a FM while also improving the representation ability and interpretability of a FM model.

K. Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets [11]

Deep Neural Networks (DNNs) have achieved tremendous success for large vocabulary continuous speech recognition (LVCSR) however the time required to train such DNNs is quite slow. A contributing factor to the slow training times is the large number of training parameters (i.e., 10-50 million). This paper explores a low-rank matrix factorization on the final weight layer in a DNN for acoustic modelling on three different LVCSR tasks. This approach allows for the reduction of the parameters by 30-50% which significantly reduces the training time for the DNN with minimal loss in accuracy.

III. System Architecture & Algorithms

Recently, Facebook open sourced a state-of-the-art deep learning recommendation and personalization model (DLRM) [1]. We have noted a few key takeaways from their model as part of our research. This model is divided into three parts to allow parallel execution of the algorithm on heterogeneous compute instances. The first two parts are a dense feature extraction network module and a sparse feature extraction network module. The latter uses embedding tables to extract relevant features from sparse categorical input. These two models are then combined into a final Multi Layer Perceptron module that provides recommendations from the interactions between the two feature extraction modules mentioned earlier. The separation of the two modules allows for tweaking of each module separately to produce interesting results. This tool also provides benchmark results which include amount of memory consumed, time taken for testing and training, accuracy metrics, etc.

A. Facebook DLRM

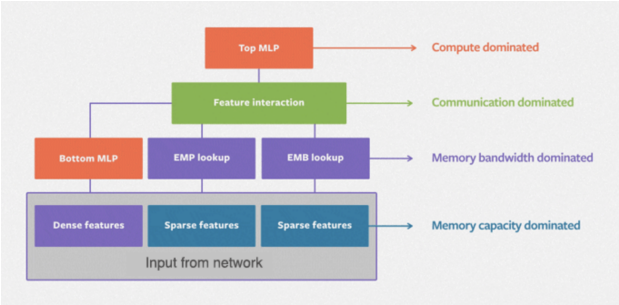

Fig. 1. Architecture of the DLRM

This section explains the high-level overview of DLRM architecture. The architecture can be broken down into four main components/techniques. [12]

1) Embeddings: An embedding is a mapping of a discrete — straight out — variable to a vector of continuous numbers. With regards to neural network systems, embeddings are low-dimensional, learned continuous vector portrayals of discrete variables. Embeddings in a neural network are valuable since they can diminish the dimensionality of categorical factors.

2) Matrix Factorization: Matrix factorization is a class of CF-algorithms (Collaborative Filtering) utilized in recommender frameworks. These algorithms work by deconstructing the user-item matrix into the result of two lower dimensionality rectangular matrices.

3) Factorization Machine: A factorization machine is a universally useful supervised machine learning algorithm that can be used for both regression and classification. It is an expansion of a linear model that is intended to retrieve interactions between features inside high dimensional sparse datasets feasibly.

4) Multilayer Perceptron: A multilayer perceptron (MLP) is a class of feedforward artificial neural network systems. An MLP comprises of a minimum of three layers of nodes: an input layer, a hidden layer, and an output layer. Each node excluding the input layer is a neuron that utilizes a nonlinear activation function. MLP uses a supervised machine learning method called backpropagation for training.

In the DLRM model, categorical features are prepared utilizing embeddings, whereas continuous features are handled with a base bottom multilayer perceptron (MLP). The second-order interactions between these two are processed by a top MLP and are taken care of in a sigmoid function so as to provide the likelihood of a click on an advertisement.

IV. Our Proposed Solution

Our proposed solution consists of two steps. The first step comprises of matrix factorization [11]. Using matrix factorization allows us to compress dense networks. This in turn decreases the memory utilized and thus decreases the inference time as well by reducing the amount of memory needed to be allocated for all parameters. We have experimented with two approaches to the application of said matrix factorization; the first of which involves changing the structure of the model before training using low rank matrix factorization, while the second approach involves changing the model's structure after training using non-negative matrix factorization. The second step in our proposed solution is an Attention based CTR Prediction [10]. Attention Factorization Machines learn the importance of feature interactions from data via a neural attention network. This attention network ensures that a Factorization Machine can discriminate between the importance of different feature interactions. This in turn increases the accuracy when compared to traditional regression and factorization models.

A. Matrix Factorization

Fig. 2 depicts a typical neural network architecture with a few hidden layers. Specifically, let us consider a layer A which is of dimension m x n. If A has a rank r, then there exists a factorization A = B x C where B is a full rank matrix or size m x r and C is also a full rank matrix of size r x n. Thus, we replace matrix A by two small matrices B and C. By doing so, we can reduce the number of parameters of the system so long as the number of parameters in B (i.e., mr) and C(i.e., rn) is less than A (i.e.,mn). It is also possible to reduce the number of parameters in A by a fraction p, this can be done by considering the following equation

$$mr + rn < pmn \tag{1}$$

solving for r in Equation 1 gives the following requirement needed to reduce overall parameters by fraction p.

$$r < \frac{pmn}{m+n} \tag{2}$$

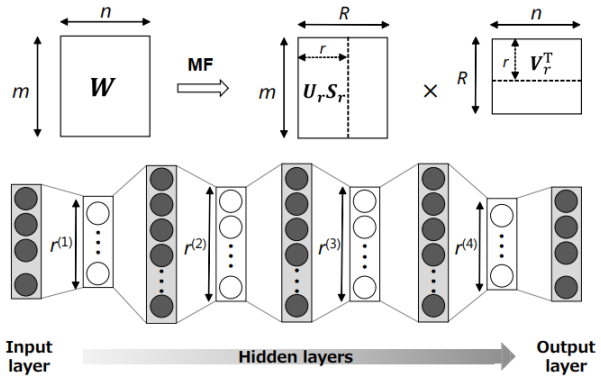

Fig. 2. Matrix Factorization

The example mentioned above was for a single layer alone, however in our proposed solution we consider this matrix factorization of all layers. This is done through two approaches as stated previously (i.e., applying matrix factorization before training, and applying matrix factorization after training).

B. Attention CTR

The Attention CTR model constitutes the second step in our proposed solution which involves the porting of the Attention Factorization Machine (AFM) model mentioned in [10] to the Facebook DLRM to improve performance while maintaining scalability. In this section, we briefly recapitulate the key components/techniques of the AFM introduced earlier.

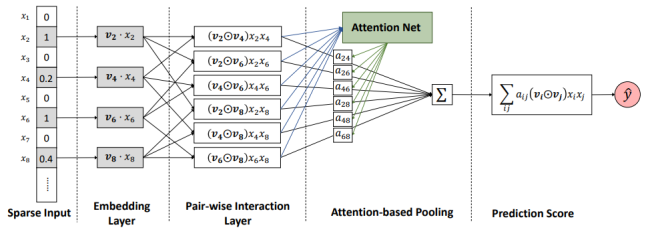

Figure 1: The neural network architecture of our proposed Attentional Factorization Machine model.

Fig. 3. Attention Factorization Machine

1) Model: Figure 3 illustrates the neural network architecture of the AFM model proposed [10]. The input layer and embedding layer are the same with FM, which adopts a sparse representation for input features and embeds each non-zero feature into a dense vector.

2) Pair-wise Interaction Layer: One of the main contributions of [10] is a new Pairwise Interaction Layer in neural network modelling. It expands m vectors to m(m - 1)/2 interacted vectors, where each interacted vector is the element-wise product of two distinct vectors to encode their interaction. This layer's output can be represented as a set of vectors:

$$f_{PI}(\mathcal{E}) = \{(\mathbf{v}_i \odot \mathbf{v}_j)\, x_i x_j\}_{(i,j)\in\mathcal{R}_x} \tag{3}$$

where $\mathcal{X}$ represents the set of non-zero features in the feature vector $\mathbf{x}$, and $\mathcal{E} = \{\mathbf{v}_i x_i\}_{i\in\mathcal{X}}$ represents the output of the embedding layer, $\odot$ denotes the element-wise product of two vectors, and $\mathcal{R}_x = \{(i,j)\}_{i\in\mathcal{X}, j\in\mathcal{X}, j>i}$.

Following this, the FM is expressed under the neural network architecture by defining the pair-wise interaction layer. This is represented, by first compressing $f_{PI}(\mathcal{E})$ with a sum pooling, after which a fully connected layer is used to project it to the prediction score.

$$\hat{y} = \mathbf{p}^T \sum_{(i,j)\in\mathcal{R}_x} (\mathbf{v}_i \odot \mathbf{v}_j)\, x_i x_j + b \tag{4}$$

where $\mathbf{p} \in \mathbb{R}^k$ and $b \in \mathbb{R}$ denote the weights and bias for the prediction layer, respectively.

3) Attention-based Pooling Layer: The Attention-based Pooling layer is another one of the main contributions of [10]. It was proposed with the idea of allowing different parts to contribute differently when compressing them to a single representation. The attention mechanism on feature interactions is employed by performing a weighted sum on the interacted vectors:

$$f_{Att}\left(f_{PI}(\mathcal{E})\right) = \sum_{(i,j)\in\mathcal{R}_x} a_{ij}\left(\mathbf{v}_i \odot \mathbf{v}_j\right) x_i x_j \tag{5}$$

where a_{ij} is the attention score for feature interaction $\hat{w}_{ij}$, which can be interpreted as the importance of $\hat{w}_{ij}$ in predicting the target. a_{ij} is estimated by directly learning it by minimizing the prediction loss. However, the problem is that, for features that have never co-occurred in the training data, the attention scores of their interactions cannot be estimated. This was addressed by further parameterizing the attention score with a multi-layer perceptron (MLP) which is called the attention network. The input to the attention network is the interacted vector of two features, which encodes their interaction information in the embedding space.

The attention network is defined as:

$$\begin{aligned} a'_{ij} &= \mathbf{h}^T \operatorname{Re} LU\left(\mathbf{W}\left(\mathbf{v}_i \odot \mathbf{v}_j\right) x_i x_j + \mathbf{b}\right), \\ a_{ij} &= \frac{\exp\left(a'_{ij}\right)}{\sum_{(i,j)\in R_z} \exp\left(a'_{ij}\right)} \end{aligned} \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{t\times k}, \mathbf{b} \in \mathbb{R}^t, \mathbf{h} \in \mathbb{R}^t$ are model parameters, and t denotes the hidden layer size of the attention network, which is called attention factor. The attention scores are normalized through the softmax function. A rectifier is used as the activation function. The output of the attention-based pooling layer is a k dimensional vector, which compresses all feature interactions in the embedding space by distinguishing their importance. This is then projected to the prediction score. To summarize, the overall formulation of the AFM model introduced in [10] is:

$$\hat{y}_{AFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{p}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}\left(\mathbf{v}_i \odot \mathbf{v}_j\right) x_i x_j \tag{7}$$

where a_{ij} has been defined in Equation (6). The model parameters are

$$\Theta = \{w_0, \{w_i\}_{i=1}^n, \{\mathbf{v}_i\}_{i=1}^n, \mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{h}\}$$

.

This AFM is ported to the Facebook DLRM where we term it as Attention CTR. The Attention CTR replaces top layer of the DLRM.

V. Evaluations

In this section, we describe our experimental setup and the various experiments conducted by us. The corresponding results for each of those experiments have been noted down below.

A. Experimental Setup

The experimental setup consists of the Criteo Dataset, a Google Colab instance, and the default DLRM configuration. The Criteo Dataset consists of rows which corresponds to a display ad served by Criteo. We experimented with dataset of size 100K and 500K. The dataset contained 40 fields; Label: 0 (ad not clicked) or 1 (ad clicked), I1-I13 which are integer count features (dense features) and C1-C26 which represent 26 columns of categorical features (sparse features). The Google Colab instance was configured with a single-core, hyper-threaded Xeon processor, a Tesla K80 with 12 GB of GDDR5 VRAM, 13 GB of RAM and 33 GB of storage. The default DLRM configuration consists of a Bottom Layer MLP structured as follows: "13-512-256-64-16" and a Top Layer MLP structured as follows: "512-256-1".

B. Metrics

Accuracy and Speed are the fundamental metrics that are reported for each approach. Accuracy, essentially, is aggregating the number of true positive and true negatives over the total number of predictions made, which can be represented as an equation

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Predictions}} \tag{8}$$

True positives and true negatives mean that both the predicted and actual outputs are true and false respectively. On the other hand, false positives indicate that the predicted output is true while the actual output is false. Similarly, false negatives indicates that the predicted output is false, while the actual output is true.

Speed, essentially, is used to capture the amount of time required to train the model and/or perform inference. We calculated the speed using the inbuilt PyTorch autograd profiler. The profiler lets us inspect the cost of different operators inside our model - both on the CPU and the GPU.

C. Findings

In this section we present evaluation metrics for few of the previously mentioned experiments with the best results. The presented results are evaluations of the above approaches on the dataset.

1) 100K Dataset: For the four approaches that have been tested (Baseline, Matrix Factorization (MF) Approach 1, Matrix Factorization Approach 2 (Inference) and Attention with Matrix Factorization), the following tables highlights the number of epochs, accuracy, and speed for the 100K dataset.

Baseline		
Epochs	**Accuracy**	**Time**
10	80.469	2.120
20	80.469	4.234
30	80.469	6.558

MF Approach 1		
Epochs	**Accuracy**	**Time**
10	80.469	2.172
20	80.469	3.755
30	81.250	5.947

MF Approach 2 (Inference)		
Epochs	**Accuracy**	**Time**
10	78.208	2.101
20	80.143	3.612
30	80.398	5.749

Attention with MF		
Epochs	**Accuracy**	**Time**
10	86.722	3.652
20	88.635	5.787
30	88.976	7.834

From the above tables, it can be seen that both the Matrix Factorization approaches achieved similar accuracy compared to the baseline while being completed in a shorter amount of time. The Attention with Matrix Factorization approach however took a longer amount of time but achieved a higher accuracy than the baseline. This was mainly due to the need of training an Attention model separately.

2) 500K Dataset: Similarly, the following tables highlights the number of epochs, accuracy, and speed for the four tested approaches for the 500K dataset.

Baseline		
Epochs	**Accuracy**	**Time**
10	72.656	2.176
20	75.781	4.518
30	75.781	6.572

MF Approach 1		
Epochs	**Accuracy**	**Time**
10	72.656	2.130
20	73.483	4.013
30	74.219	6.119

MF Approach 2 (Inference)		
Epochs	**Accuracy**	**Time**
10	72.656	2.259
20	73.871	3.932
30	75.147	5.902

Attention with MF		
Epochs	**Accuracy**	**Time**
10	80.231	3.652
20	83.479	5.087
30	84.644	7.234

As compared to the 100K dataset, we were able to achieve similar results for the 500k dataset. Both the Matrix Factorization approaches achieved similar accuracy yet again while being trained in a shorter

amount of time. Additionally, the Attention with Matrix Factorization approach achieved a higher accuracy once again.

VI. Exploratory Work

In addition to the provided experiments, our group has made an effort to do additional research on the impact of using different types of embeddings on the Facebook DLRM to determine whether a boost in performance and efficiency can be achieved. Mainly, we experimented with compositional embeddings which use complementary partitions for better memory-efficiency [8]. We found that while there was no improvement in the model's performance, there was however a reduction in the model's size. This can be attributed to the method in which the embeddings are being created for the model. Using the default method would result in a model of size 17.4 MB for the 100K dataset, and using the second method for creating embeddings resulted in a model of size 2.3 MB. This is significant decrease in size with similar performance.

VII. Conclusion

The key takeaways from the project are that matrix factorization can be implemented on a deep neural network in order to reduce the complexity of its operations while preserving accuracy and reducing training time. Furthermore, an attention model can also be coupled with a DNN recommendation model to significantly boost accuracy albeit at the cost of training time.

VIII. Future Work

Although the improvements achieved by this project were significant, it can be further improved by the use of dynamic matrix factorization through which we determine the smallest possible rank for a weight matrix, and subsequently factorize it according to the target size in order to preserve accuracy and reduce complexity of the model. Furthermore, we can optimize the Attention model in terms of reducing the time required to train the model. We can also utilize a more current state-of-the-art CTR model as the one implemented is slightly dated.

References

[1] Gupta, U.; Wu, C.-J.; Wang, X.; Naumov, M.; Reagen, B.;Brooks, D.; Cottel, B.; Hazelwood, K.; Jia, B.; Lee, H.-H. S.;Malevich, A.; Mudigere, D.; Smelyanskiy, M.; Xiong, L.;and Zhang, X. 2019. The architectural implications of facebook's dnn-based personalized recommendation.

[2] Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems.

[3] Yaguchi, A.; Suzuki, T.; Nitta, S.; Sakata, Y.; and Tanizawa,A. 2019. Scalable deep neural networks via low-rank matrix factorization.

[4] Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; and Chua, T.-S. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 335–344. New York, NY, USA: Association for Computing Machinery.

[5] Dziugaite, G. K., and Roy, D. M. 2015. Neural network matrix factorization.

[6] Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: A factorization-machine based neural network for ctr prediction.

[7] Livne, A.; Unger, M.; Shapira, B.; and Rokach, L. 2019.Deep context-aware recommender system utilizing sequential latent context. ArXiv abs/1909.03999.

[8] Shi, H.-J. M.; Mudigere, D.; Naumov, M.; and Yang, J.2019. Compositional embeddings using complementary partitions for memory-efficient recommendation systems.

[9] Hong-Jian Xue; Xin-Yu Dai; Jianbing Zhang; Shujian Huang; and Jiajun Chen, 2017 Deep Matrix Factorization Models for Recommender Systems.

[10] 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.

[11] Tara N. Sainath; Brian Kingsbury; Vikas Sindhwani; Ebru Arisoy; and Bhuvana Ramabhadran, 2013 Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets.

[12] https://medium.com/@rehan_ahmad/deep-learning-recommendation-machines-dlrm-92e37733d07f