DCN, DNN and DeepFM approaches for movie rating prediction

Vaibhav Tiwari[1], Rajeshwari Sah[1], Anuj Ahuja[1], and Shardul Deshpande[1]

[1]University of California, San Diego

December 10, 2021

Abstract

In this project, we have done comparative analysis of various rating prediction methods on MovieLens100k[1] dataset. Initially, extensive Exploratory Data Analysis (EDA) is done to visualise feature distributions and pick informative characteristics for implementation. While implementing, we have considered DeepFM, Deep & Cross Networks (DCNs) with stacked, parallel architecture along with Low-rank techniques and Deep Neural Networks (DNNs). Furthermore, variations to RMSE for each model based on different values for hyperparameters are visualised.

Keywords: Movie Rating Prediction, Deep Neural Networks, Deep & Cross Networks, DeepFM, Movielens

1 Dataset

MovieLens Datasets were generated as a result of users interacting with the MovieLens online recommender system for number of years. It originated at the University of Minnesota in the summer of 1997 and since that time growth of MovieLens system has been stable especially without any marketing efforts. MovieLens has 4 datasets differentiated using their sampling methods but they share several common characteristics. Each has tuples of the form <user, item, rating, timestamp>. Ratings and other data are attributed to anonymous user ids. Movies are listed along with their titles, along with their one or more genres. The 100k and 1m datasets include users' demographic data (age, gender, occupation, zipcode). According to experience-based guidelines for making more effective use of MovieLens datasets, much of the power lies in comparing results with prior research papers' algorithms and analysis which we have discussed in following sections.

There are several other datasets that are frequently used with MovieLens. These differ from MovieLens in terms of size, shape and context of interaction. Table 1 shows most frequently cited datasets along with MovieLens ordered by number of ratings.

Name	**Date Range**	**Domain**	**# Ratings**
Book-Crossing	2001-2004	books	1.1m
EachMovie	1995-1997	movies	2.7m
Jester	1999-2003	jokes	4.1m
Amazon	1996-2004	many	82.8m
Netflix Prize	1998-2005	movies	100.5m
Yahoo Music	1999-2009	music	262.8m

Table 1: Prominent Alternative Datasets

1.1 Exploratory Data Analysis

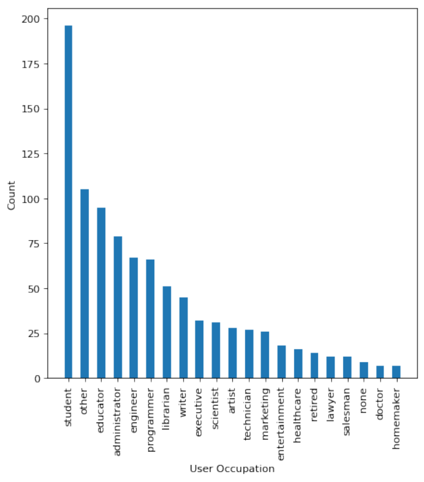
Core sections of of MovieLens100k[1] are explained below:

1. User Ids - MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between ratings.csv and tags.csv (i.e., the same id refers to the same user across the two files).

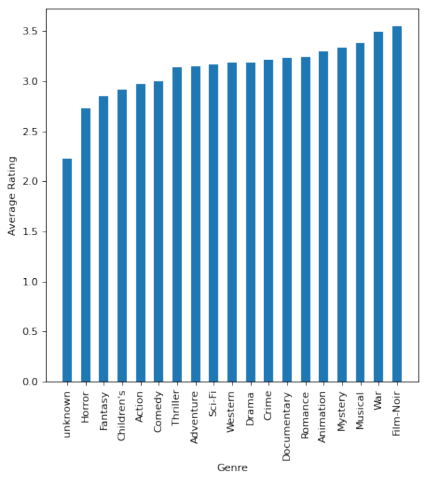
2. Movie Ids - Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id 1 corresponds to the URL https://movielens.org/movies/1). Movie ids are consistent between ratings.csv, tags.csv, movies.csv, and links.csv (i.e., the same id refers to the same movie across these four data files).

3. Ratings Data File Structure - All ratings are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user, and has the following format: userId,movieId,rating,timestamp. The lines within this file are ordered first by userId, then, within user, by movieId. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

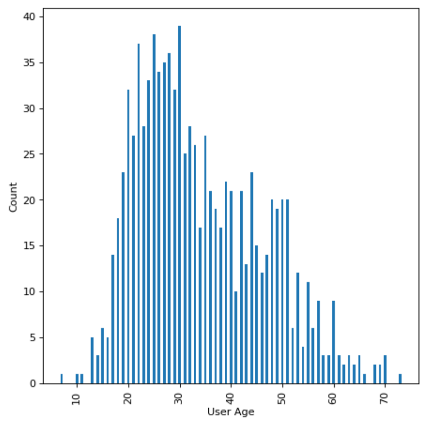
4. Tags Data File Structure - All tags are contained in the file tags.csv. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format: userId,movieId,tag,timestamp The lines within this



(a) User Occupation Frequency



(b) Genre Frequency



(c) User Age Frequency

Figure 1: Data Distributions

file are ordered first by userId, then, within user, by movieId. Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.

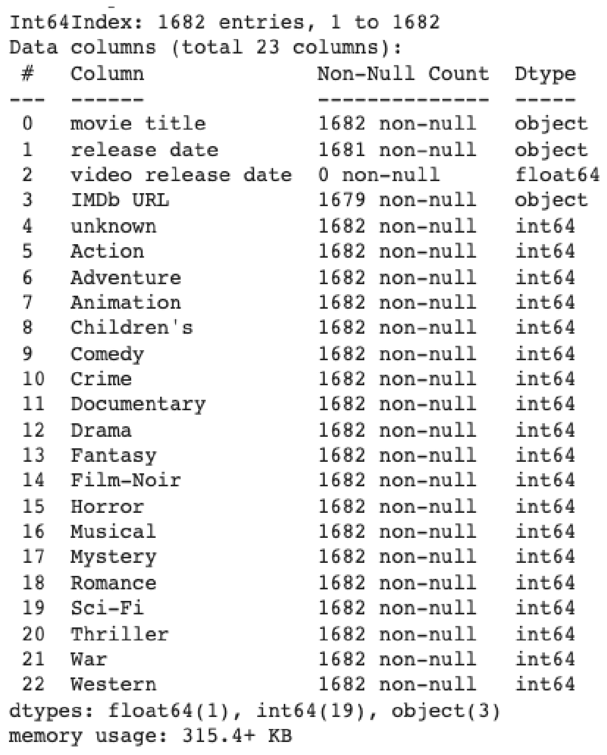
5. Movies Data File Structure - Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie, and has the following format: movieId,title,genres Movie titles are entered manually or imported from https://www.themoviedb.org/, and include the year of release in parentheses.

We have incorporated various movie metadata to significantly improve on the model's accuracy. Movie metadata mainly consists of Release Date, Genres, Name. (Note that Genre has sparse representation and a movie can have multiple genres). To better understand how ratings vary based on various features of movie and users, visualisations for same are given in Fig(1). In Fig(1b), we observe change in average rating based on genre of the movies, Horror is the least rated whereas Film-noir is the highest rated genre in the given dataset. We also see that most movies are rated 3 or 4 (near average) which makes intuitive sense for analyzing. Fig(1C), age distribution for users is visualized, we can see that most of the users who rated are aged between 20-30.

In Fig.2 & Fig.3, various features for movies and users are given. Note that number of unique values for users and movies are as given in the figure but entire dataset may contain a movie ranked by multiple users and a user ranking multiple movies.

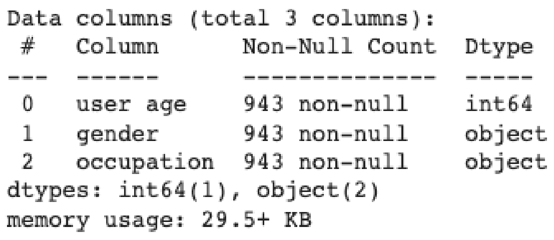
2 Rating Prediction

For the movielens dataset we have studied the rating prediction task using the features mentioned in the dataset section. To evaluate the performace of the Deep & Cross Network (DCN)[2] model in this task we have used root



```
Int64Index: 1682 entries, 1 to 1682
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   movie title         1682 non-null   object
 1   release date        1681 non-null   object
 2   video release date  0 non-null      float64
 3   IMDb URL            1679 non-null   object
 4   unknown             1682 non-null   int64
 5   Action              1682 non-null   int64
 6   Adventure           1682 non-null   int64
 7   Animation           1682 non-null   int64
 8   Children's          1682 non-null   int64
 9   Comedy              1682 non-null   int64
 10  Crime               1682 non-null   int64
 11  Documentary         1682 non-null   int64
 12  Drama               1682 non-null   int64
 13  Fantasy             1682 non-null   int64
 14  Film-Noir           1682 non-null   int64
 15  Horror              1682 non-null   int64
 16  Musical             1682 non-null   int64
 17  Mystery             1682 non-null   int64
 18  Romance             1682 non-null   int64
 19  Sci-Fi              1682 non-null   int64
 20  Thriller            1682 non-null   int64
 21  War                 1682 non-null   int64
 22  Western             1682 non-null   int64
dtypes: float64(1), int64(19), object(3)
memory usage: 315.4+ KB
```

Figure 2: Movie data description



```
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   user age     943 non-null    int64
 1   gender       943 non-null    object
 2   occupation   943 non-null    object
dtypes: int64(1), object(2)
memory usage: 29.5+ KB
```

Figure 3: User data description

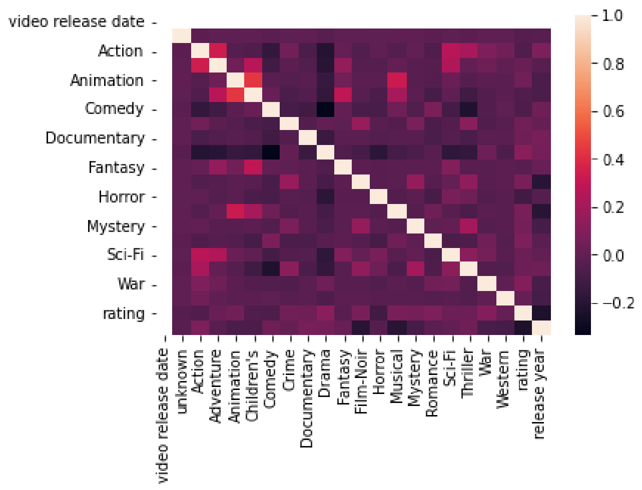


Figure 4: Pearson Correlation for Movie Data

mean square error (RMSE). We compare it's root mean square error with the root mean square error of the baseline model. We have selected DeepFM as the baseline model. Results indicate an improvement in the RSME for the DCN model when compared with DeepFM which validates the performace of the model.

2.1 Date Pre-Processing

The MovieLens dataset contains the movie id, user id, user zip code, user occupation, user gender, genre, and bucketized user age. All these features are collated from the CSV files mentioned in the Dataset section. Movie id, user id, user zip code, user occupation are textual features while user gender, genre, and bucketized user age are integer features.

We then build a vocabulary for these features by determining the unique values present in each of the features. The build vocabulary is then used to create the encoded input for the embedding layer in the models used in following sections. This method is to avoid the high-dimensional features and thus, leads to less computation and feature engineering on text.

We also considered vectorizing the categorical features using One Hot Encoding and feeding it to the model. Although this approach can lead to fast and simple feature generation, but it makes the dimensionality high (spare vector) by creating a new dimension for each category. Embedding on the other hand results in a dense vector. Further, One-Hot Encoding tells us nothing about the semantics of the item in the feature. Each vectorization is an orthogonal representation in another dimension. Embeddings will group commonly co-occurring items in the feature together in the representation space.

3 Model

For the task of rating prediction task we have experimented with neural network and factorization machine models like Deep & Cross Network(DCN), Deep Neural Networks(DNN) and DeepFM.

All the models were trained on Google Colab GPUs for faster model train times.[3]

3.1 Deep & Cross Network (DCN)

In this section, we describe the architecture of the DCN model [4] and justify why it is a good fit for the rating prediction problem statement. DCN consists of the following layers:

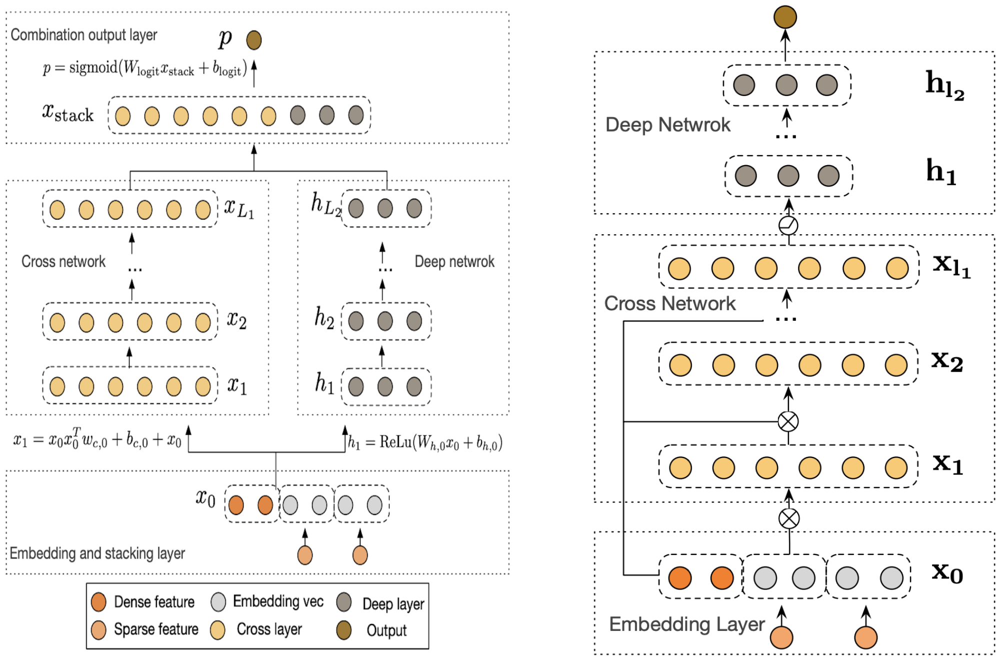
1. **Embedding & Stacking Layer**: Embedding layer is used after Input layer to convert categorical features into low-dimensional features. This helps in avoiding task specific feature engineering in subsequent layers. For example, the input includes categorical feature user occupation. Such features are often encoded as one hot vectors eg. [1,0,0,0..] ; however, this often leads to excessively high-dimensional feature spaces for large vocabularies. The embedding layer helps in reducing the dimensionality of such features. Finally, the processed continuous features and embeddings of categorical features are stacked together to form the input to the deep layers.

2. **Cross Network Layer**: The cross network layer is used to generating synthetic features. Synthetic features is the combination of non-linear features and linear features. Hence, this process is to merge categorical and continuous features which simplifies and avoids the feature engineering process.

3. **Deep Network Layer**: The deep network is a fully-connected feed-forward neural network with ReLU activation function.

4. **Combination Output Layer**: Outputs of Deep and Cross Networks are concatenated and fed into a standard logit layer

Deep and cross network provides an more efficient learning of certain bounded-degree feature interactions. In particular, DCN explicitly applies feature crossing at each layer, requires no manual feature engineering, and adds negligible extra complexity to the DNN model. Our experimental results have demonstrated its superiority over the state-of-art algorithms on the Movielens dataset, in terms of both model accuracy. DCN also provides a generalization for FMs. The cross network shares the spirit of parameter sharing as the FM model and further extends it to a deeper structure. In particular, Movielens dataset is a set of complex feature interaction ideal for cross network layer. For feature like genre which is the one hot encoding of all 19 genres, could lead to a wide sparse feature vectors. As stated as one of the benefits of DCN, it is able to automatically apply feature crossing at each layer, which generates deeper interaction amongst the user items overall. With DCN we achieved an RMSE of 0.92 appx.



(a) Parallel

(b) Stacked

Figure 5: Deep & Cross Network Architectures

3.2 Deep Neural Network (DNN)

Deep Neural Network was the first algorithm for training MLP-NNs with many layers. This algorithm marked the commencement of the Deep Learning (DL). The Deep and cross networks are also based on DNN models. As the name suggests DNNs provide deep network for learning. Deep neural networks (DNN) are able to learn non-trivial high-degree feature interactions due to embedding vectors and nonlinear activation functions. Movielens containing a list of dense feature constitute as an ideal candidate for neural net training with deep layers. One of the drawbacks of using DNN . The reasons for choosing DNN model as one of our experiment is to create a baseline for DCN. With DNN we achived an RMSE of 0.94 appx. Our experimental results have demonstrated that with a cross network, DCN has lower logloss than a DNN.

3.3 DeepFM

DeepFM, combines the power of factorization machines and deep learning for feature learning in a new neural network architecture. Compared to the Wide & Deep model, DeepFM has a shared input to its "wide" and "deep" parts, with no need of feature engineering besides raw features. For rating predication, the userid and movieid interactions provides an ideal candidate for choosing FM models like DeepFM. As mentioned in the DCN paper, DCN provides a generalisation for FMs. Both models have each feature learned some param- k k=0 eters independent from other features, and the weight of a cross term is a certain combination of corresponding parameters. Parameter sharing not only makes the model more efficient, but also enables the model to generalize to unseen feature interactions and be more robust to noise. The reason for running DeepFM expermient on movielens dataset was to provide a baseline for SOTS FMs model like DeepFM and compare it against DCN and evaluate the metrics. With DeepFM using similar set of feature vectors, we observed that DeepFM had an MSE of 0.876 appx (RMSE 0.935 appx.), slightly more than DCN.

Our motivation behind choosing all the above models for our experiments research and analyse DCN and the SOTA models of the concepts/idea on which DCN. As stated, DCN is widely based on neural network models like DNN which is able to provide a generalisation for FM models. Thus, along with DCN we decided to also experment with SOTA NN & FM models like DNN and DeepFM. From our experiments we observed that DCN performed significantly better than DNN(given the small scale of data params) and performed slightly better than DeepFM. Since Deep Neural Networks (DNNs) are inefficient to even approximately model 2nd or 3rd-order feature crosses, it makes sense to use Cross Networks that can efficiently capture these higher order feature crosses. This eliminates the need of determining efficient feature crosses through manual feature engineering or exhaustive search. To avoid over fitting we have incorporated early stopping. One of the issues that we face during feature engineering was to curate the sparse feature vector like movie genre. Coming to DCN vs DeepFM, we see both the models combines the power of factorization machines for and DL for feature learning. Even though DCN was able to perform slightly better than DeepFM, run of DCN was comparatively much higher than DeepFM. With scaling to even bigger dataset, DCN was very slow on the infrastructure set up. This can be attributed to the wide variety of cross netwrok layer used in DCN.

4 Literature Review

4.1 Dataset source

This dataset was prepared by GroupLens Research group at University of Minnesota. GroupLens Research group operates a movie recommender based on collaborative filtering, MovieLens which is the source of these data.

4.2 Past similar datasets

Table 1 describes past datasets used for research on similar research areas. Each of them is explained below

1. **Book-Crossing**: It is a snapshot of online book rating community. It contains 100,000 users, 340,000 books, and 1.1million ratings. The snapshot was taken back in 2004 but ratings are not associated with timestamps. It's important to not that dataset reveals users' locations and ages.

2. **EachMovie**: This dataset was heavily used in research before it was made unavailable due to legal concers about potential reidentifiability of users. It contains 2.7 million rating from 59,000 users across 1,500 movies. Ratings take 26 possible values in range of 0-14, reflecting use and rescaling of approaches to collect ratings.

3. **Jester**: This dataset contains explicit rating of jokes on -10 to 10 scale. The largest of the dataset contains 4.1million ratings from 73,421 users between 1999 to 2003. It's important to not that dataset has only 100 jokes which extremely small compared to number of ratings and hence very dense compared to other options.

4. **Amazon**: The ratings in this dataset are associated with textual reviews, span 18 years and encompass a wide variety of products. Largest of these dataset contains 82million ratings from 21 million users across 10 million items. Ratings are between 1-5 with 5 possible values.

5. **Netflix**: Netflix dataset was part of the competition for rating prediction on movie datasets. It was made available in 2006 but subsequently taken down in 2009 for legal reasons. It had 480,000 users, 17,000 items and 100 million ratings.

6. **Yahoo Music**: Yahoo! Labs provided number of music datasets which were extracted from their music product. Their "C15" which was released for the KDD Cup provided user ratings on songs, albums, artists and music genres. It contains 1 million users, 620,000 music items and 262 million ratings.

4.3 Approaches considered

We explored various approaches (eg: Deep & Wide learning [5] and Deep & Cross Network v2 [6]) for tackling the problem at hand before finalizing DeepFM [7] and Deep & Cross Network [4] models. For DCN model, we've chosen the deep neural network pipeline as our baseline since that depicts the neural network's ability to learn by itself, without any FM or cross layers.

1. **DeepFM:** The factorization-machine based neural network consists of two componenets, FM component and deep component that share the same input. Besides a linear (order-1) interactions among features, FM models pairwise (order-2) feature inter actions as inner product of respective feature latent vectors. While, the deep component is a feed-forward neural network, which is used to learn high-order feature interactions.

2. **DCN Stacked architecture:** The stacked DCN architecture 5 allows the processed output from the final cross network layer to be directly fed into the deep network.

3. **DCN Parallel architecture:** The parallel architecture 5 keeps the processing from cross network layer and deep network layers separate and the final outputs from both networks are combined in the combination layer.

4. **Low-Rank DCN:** To reduce the training and serving cost, we leverage low-rank techniques to approximate the DCN weight matrices. To reduce the training and serving cost, we leverage low-rank techniques to approximate the DCN weight matrices

5. **DNN method:** This model has the cross network layer removed to test the baseline ability of a normal neural network to learn the interactions between various dataset features.

5 Experiments & Results

For movie rating prediction for Movielens dataset, we have conducted experiments on four main models, Deep and Cross network, Deep Neural Network and DeepFM.

5.1 Experiment Set Up

5.1.1 Feature Representation

As mentioned in the dataset section, following features from the MovieLens data set were considered for generating the Feature Vector:

1. User Id - int

2. Movie ID - int

3. User gender - category

4. User occupation - string

5. User location - int

6. Bucketized user age

7. Movie Genre, 19 in total (one hot encoded)

5.1.2 Feature selection

MovieLens also had other data entries like Movie Title, IMDB URL etc. We did not consider these as a feature as this was a redundant information as movie id is able to capture this information.

The training:validation:testing dataset ratio considered was 75:10:15.

5.2 Result and Analysis

The experiments conducted for **Deep & Cross Network** architectures largely concentrated on identifying the optimal hyper-parameters for each of the proposed approaches. A grid search was carried out to perform the hyper parameter tuning for each of the DCN model architectures. The main difference between the experiments conducted in terms of finding hyper parameters of optimum models in the literature compared to our approach is that we did not limit our search to the number of layers and nodes for each architecture but also included the learning rate and epochs for convergence in our experiments.

The various hyperparameters that were tuned and their respective search spaces are given in Table 3. The nodes in each layer have been chosen to allow the models to have a modest number of parameters to train on allowing large number of experiments to be conducted while having exposure to a range of different layer configurations. The result of these experiments are shown in figure ??, ??, ?? and ??. Following each line gives us the hyper parameter config that was run and the corresponding RMSE that was achieved by the model in that run.

The experiment conducted for DeepFM used similar set of feature vecture as described above in this section. I performed a grid search on the hyperparameters. The best set of hyperparameter for DeepFM was 15 epochs with bacth size of 256. We achieved a MSE of 0.93 appx for deepFM.

The best hyper parameter configurations that were identified in our experiments are shown in Table 4. The RMSE and loss vs epochs graphs of the models with most optimum set of hyper parameter configuration identified above is shown in figures 7, 8, 9, and 10. It can be seen that the best models have converged fairly well within the limits.

Finally, Table 5 shows a comparison of different models and their average time per experiment in seconds. Using this information in conjunction with Table 4 it can be concluded that there exists a trade-off between performance and cost of the full stacked/parallel models and low-cost models.

Layer Layout	**Mapping**
[256, 128]	0
[192, 192]	1
[512, 256, 128]	2

Table 2: Layer layout to deep layer type mapping for hyper parameter tuning figures ?? to ??

Hyperparameter	**Search Space**
Learning rate	0.001, 0.005, 0.01, 0.05, 0.1
Epochs	5, 10, 15, 20
Layers & Nodes	192,192|256,128|512, 256, 128

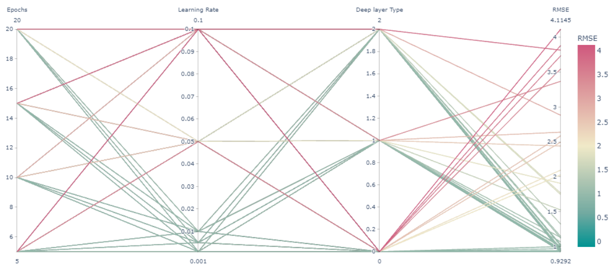
Table 3: Hyperparameters considered for DCN models and their search space

Arch.	**LR**	**Layer layout**	**Epoch**	**RMSE**
DCN St	0.01	512,256,128	20	0.9292
DCN Par	0.05	256, 128	15	0.9308
DNN	0.05	256, 128	15	0.9326
DCN LC	0.01	256, 128	10	0.9331
DeepFM	-	-	15	0.935

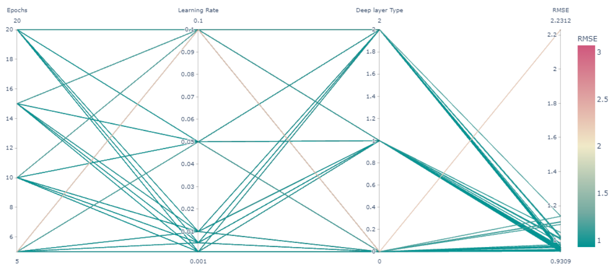
Table 4: Best hyper parameter config for each DCN architecture

Architecture	**Avg. time per experiment**
DCN Stacked	27.2
DCN Parallel	27.9
DCN Low Cost	26.4
DNN	22.9

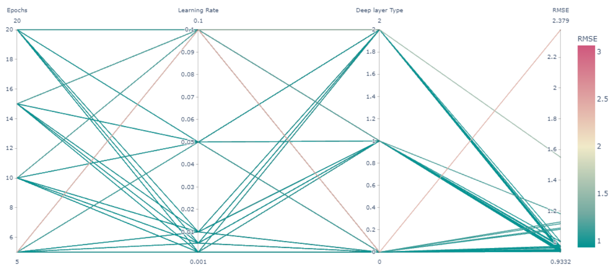
Table 5: Average time per experiment between different DCN models in sec



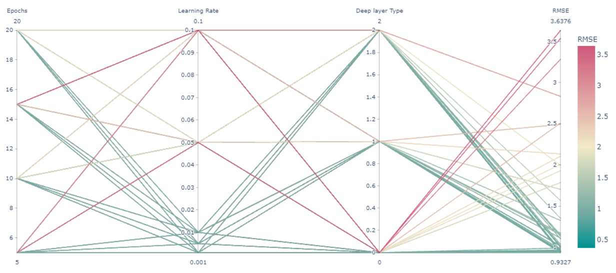
(a) DCN (Stacked) hyperparameter tuning



(b) DCN (Parallel) hyperparameter tuning



(c) DCN Stacked low-cost architecture hyperparameter tuning



(d) DNN Architecture hyperparameter tuning

Figure 6: Data Distributions

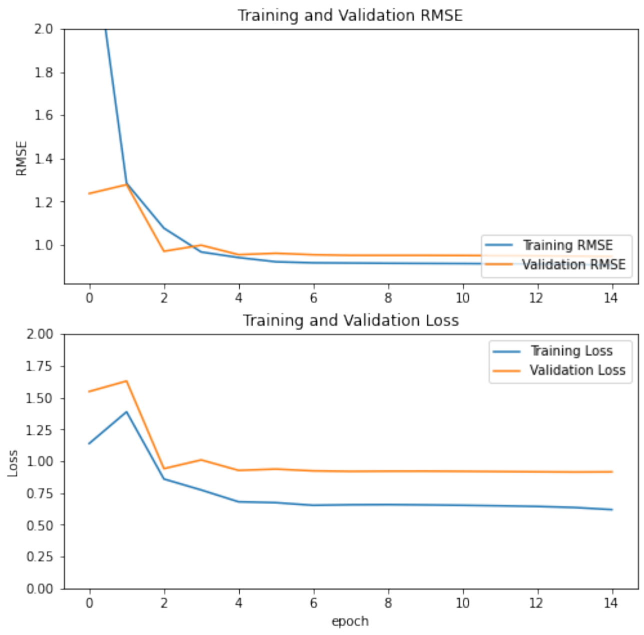


Figure 7: DCN parallel RMSE & loss vs epoch

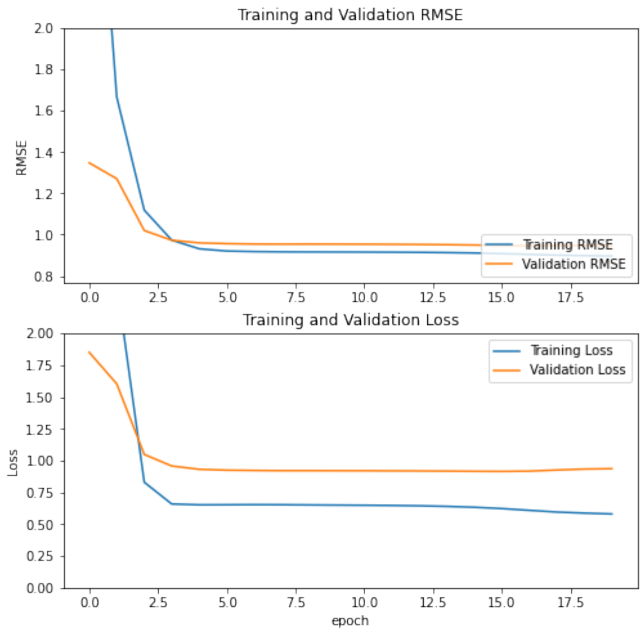


Figure 8: DCN stacked RMSE & loss vs epoch

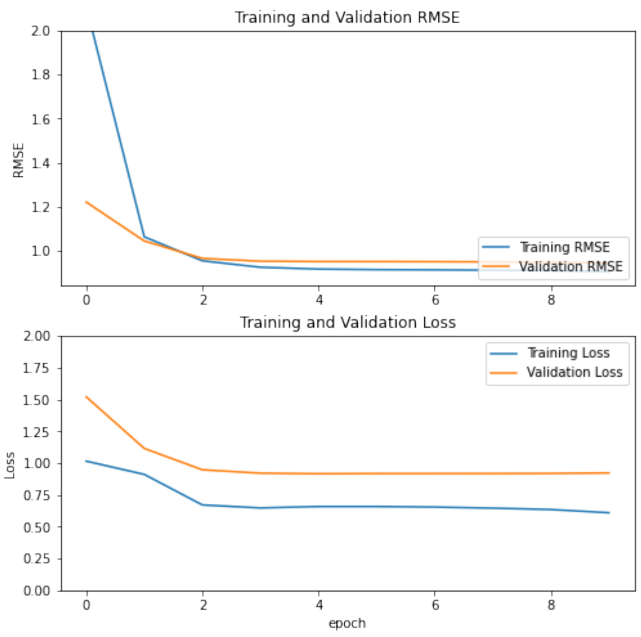


Figure 9: DCN low-cost model RMSE & loss vs epoch

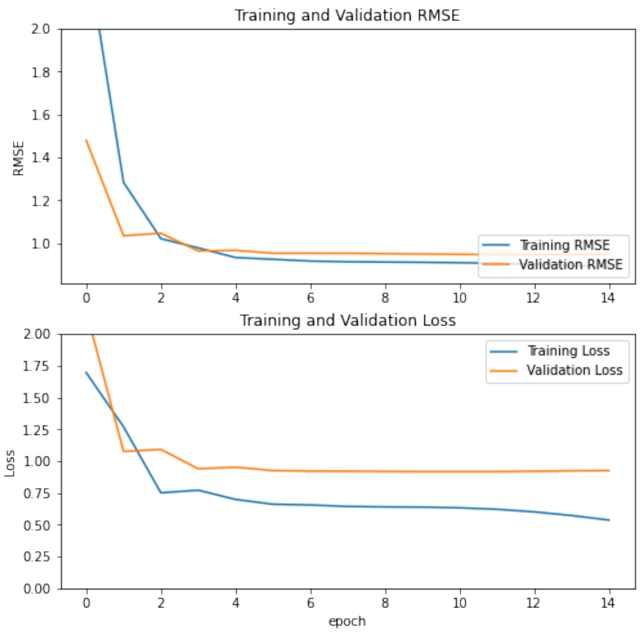


Figure 10: DNN RMSE & loss vs epoch

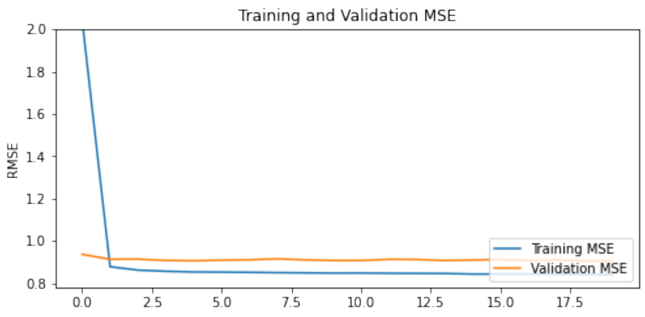


Figure 11: DeepFM RMSE & loss vs epoch

6 Conclusions

The key factor in the success of many prediction models is efficient identification of feature interactions. Factorization machine is one such concept that provides learning for user item interaction dataset used in predictive tasks. Moreover, to avoid manual feature learning, Deep Neural Networks are popular choice for automatic. The FM models presents a shallow structure limiting the representation of cross terms. In contrast, DCN, is able to construct all the cross terms with more depth. Learning DNN network can get unnecessarily large and inefficient in learning certain features. Both DeepFM and the Deep Cross Network can handle a large set of sparse and dense features, and learns explicit cross features of bounded degree jointly with traditional deep representations. For DCN, The degree of cross features increases by one at each cross layer.

The authors in [6] have advised to choose between full stacked and parallel DCN models according to the characteristics of the features used in the dataset. The stacked DCN model performs slightly better than parallel DCN model but there exists a trade off with the low-cost model where the overall run-time of the low-cost model is slightly lower than both full models. Furthermore, significantly less explicit feature engineering was required for experimenting with these models. This significantly reduces effort to perform manual feature engineering on large production size datasets further advancing the argument to use deep learning models in production environments.

Acknowledgements

References

[1] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. acm transactions on interactive intelligent systems (tiis) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872.

[2] Naonori Ueda Mathieu Blondel, Akinori Fujino and Masakazu Ishihata. Higher-order factorization machines. in advances in neural information processing systems.

[3] Bisong E. Google colaboratory. in: Building machine learning and deep learning models on google cloud platform. apress, berkeley, ca. https://doi.org/10.1007/978-1-4842-4470-8_7.

[4] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. *CoRR*, abs/1708.05123, 2017.

[5] 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. Wide & deep learning for recommender systems. *CoRR*, abs/1606.07792, 2016.

[6] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. DCN-M: improved deep & cross network for feature cross learning in web-scale learning to rank systems. *CoRR*, abs/2008.13535, 2020.

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. *CoRR*, abs/1703.04247, 2017.