Fashion Recommendations with Operation-aware Neural Networks

Guangyuan Piao[0000−0003−0516−2802]

Department of Computer Science, Maynooth University, Maynooth, Co Kildare, Ireland
guangyuan.piao@mu.ie

Abstract. Recommender systems play a crucial role in improving the quality of service as well as revenue in a wide range of domains such as travel, entertainment, and fashion. In this report, we discuss our solution for the Fashion Recommendations Challenge organized by FARFETCH at ECML/PKDD 2021. Our approach built on top of a state-of-the-art CTR (Click-through rate) prediction model - Operation-aware Neural Networks (ONNs) - for constructing an ensemble model where each ONN is trained on a different subset of training data and a soft voting strategy is adopted for final score prediction. Overall, this ensemble model achieved an MRR score of 0.47013 and ranked 4th in the challenge among 104 teams registered.

Keywords: CTR Prediction · Operation-aware Neural Networks

1 Introduction

Recommendation systems, which aim to recommend a set of items that might be of a user's interest out of a great number of candidate items, play an important role in many services across different domains such as e-commerce[1] [8] and entertainment [2]. For instance, more than 80% of the TV shows people watch on Netflix[2] are discovered through the recommendation system of Netflix[3], and YouTube says these recommendations drive more than 70% of its viewing time[4]. The Fashion Recommendation Challenge [3][5] organized by FARFETCH[6] – an online luxury fashion retail platform – at ECML/PKDD 2021[7] aims to tackle the challenge of fashion item recommendations on the platform in order to provide a tailored, personalized and authoritative fashion shopping experience to their customers.

[1] shorturl.at/rvDX6
[2] https://www.netflix.com
[3] shorturl.at/chjlA
[4] shorturl.at/qACFN
[5] https://www.ffrecschallenge.com/ecmlpkdd2021/
[6] https://www.farfetch.com/
[7] https://2021.ecmlpkdd.org/

1.1 Challenge Description

The challenge provided a dataset consisting of a large sample of FARFETCH's recommendations system impressions and associated click events, captured over a period of two months. The goal of the competition is to build a recommender system to predict which of the products that were shown to a user lead to an actual click, based on known historical labelled interactions. The dataset comprises over 5,000,000 recommendation events, 450,000 products and 230,000 unique users from the FARFETCH platform. According to the organizers, there are 104 teams registered to participate with over 300 submissions during the competition period.

Dataset. The organizers provided a training set (D_{train}), a test set for the 1st phase (D_{test1}), and the final test set for the 2nd phase (D_{test2}) of the challenge. Table 1 provides the statistics about the three datasets. D_{train} consists of 525,287 QIDs (Query IDs) where each QID is associated with six corresponding examples/impressions for the query from a user and the ground truth information, i.e., whether each example is clicked by the target user. Each example consists of a set of given features such as *user_id*, *session_id*, *product_id*, and *page_type*, and the organizers also provided product-related features such as *gender*, *main_colour*, and *season*. There are two testing phases with D_{test1} and D_{test2} being released by the organizers in each phase. Each dataset contains 100,000 QIDs and 600,000 examples with the same information as D_{train} except ground truth information.

Evaluation metric. The evaluation metric of the challenge is Mean Reciprocal Rank (MRR) – the average of the reciprocal ranks of positive items – which can be formally defined as follows:

$$MRR = \frac{1}{N}\sum_{1}^{N}\frac{1}{rank_i} \tag{1}$$

where $rank_i$ refers to the first rank position of one interacted item in the corresponding set of six impressions, and N is the total number of QIDs.

For testing, each team submits the results of ranked examples for each QID in a given test set to the online competition system of FARFETCH where the performance of submission will be evaluated and revealed. For the 1st phase of the challenge, each team is allowed to submit multiple submissions every day while only three submissions are allowed for the 2nd phase.

Table 1. Statistics about the challenge datasets.

Dataset	**# of QIDs**	**# of Examples**	**With groud truth**
D_{train}	584,665	3,507,990	Yes
D_{test1}	100,000	600,000	No
D_{test2}	100,000	600,000	No

Table 2. Statistics about split training dataset for training and validation.

Dataset	**# of QIDs**	**# of Examples**
D_{train}	584,665	3,507,990
$D_{train'}$	525,287	3,151,722
D_{val}	59,378	356,268

Our goal. By participating in this challenge, we are interested in to what extent those state-of-the-art deep CTR (Click-through rate) prediction models from the ready-to-use DeepCTR library[8] perform in the context of the challenge without creating a customized solution from scratch.

2 Proposed Approach

At the beginning of the participation, we first investigated several ready-to-use models provided by the DeepCTR library such as DeepFM [4], xDeepFM [6], and ONN [11]. One initial observation is that those models can be overfitted easily with the given training set D_{train}. To get rid of the overfitting problem and choose a main model for fine-tuning hyper-parameters and build our ensemble approach, we decided to use around 10% of D_{train} as our validation set – D_{val}, and the rest (denoted as $D_{train'}$ afterwards) for training as shown in Table 2.

Comparison of different DeepCTR models. Figure 1 illustrates the performance of those investigated methods from the DeepCTR library where the best MRR scores on the validation set (D_{val}) of each method are reported with reasonable effort of tuning. We observe that DeepFM, xDeepFM, AutoInt [9], FiBiNet [5], and FLEN [1] have similar performance, and outperform AFN [10]. Overall, ONN provides the best performance (0.458) compared to other methods. Therefore, we ended up using ONN as the main model for constructing our ensemble approach. We give a brief introduction about ONN in the following before delving into the ensemble approach.

Overview of ONN. ONN was proposed by Yang et al. in the context of a Tencent Social Advertising College Algorithm Competition [11] where the authors used ONN to win the first prize of the competition. Figure 2 shows an overview of the ONN architecture. The main innovation of ONN is that it learns different representations of a feature for different operations while existing models usually learn one representation for each feature which will be shared in all operations. ONN also distinguishes operations of the same type performed on different features. For instance, the "inner-product" between feature i and j and i and p are considered as different operations. The embedding features and interaction features are then fed into a MLP (multi-layer perceptron) network

[8] https://deepctr-doc.readthedocs.io/en/latest/

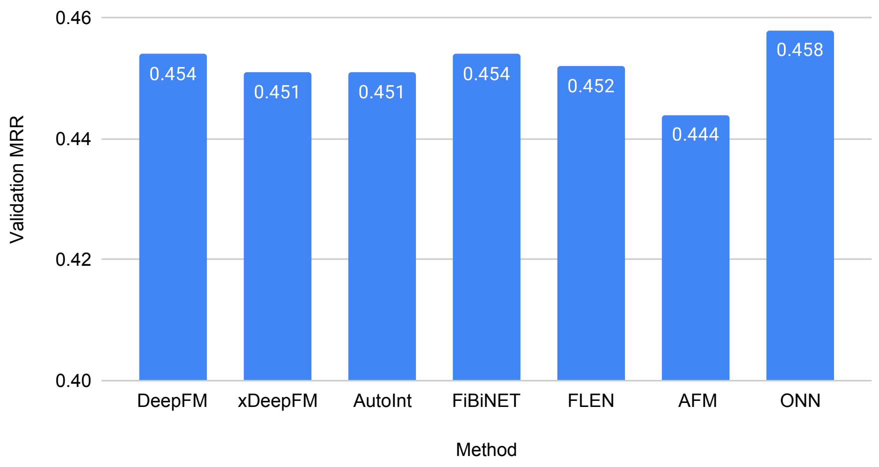

Fig. 1. Performance of different DeepCTR models in terms of MRR on D_{val}.

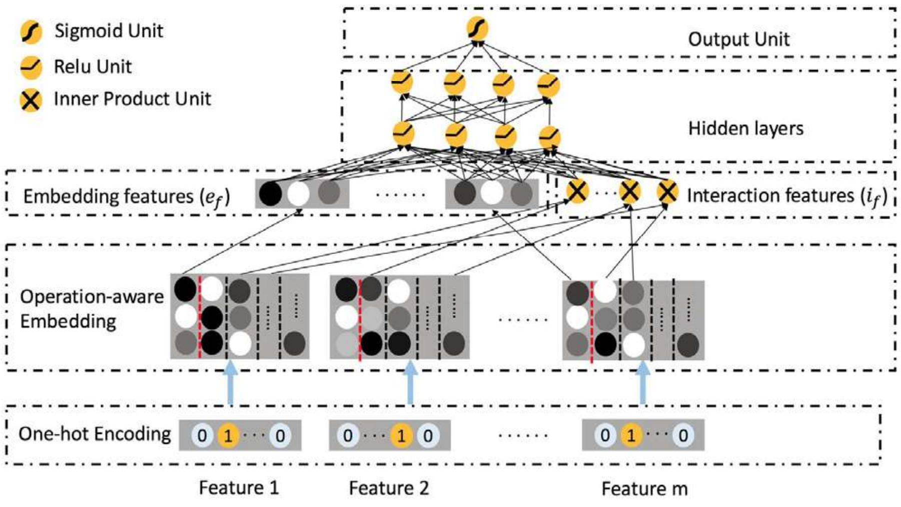

Fig. 2. Overview of the ONN architecture [11].

followed by the *sigmoid* function for predicting the preference score $\hat{y} \in (0, 1)$ of a given item with respect to the target user. We used the implementation of ONN from DeepCTR library where the ONN is made flexible by also allowing dense/numerical features as an input.

2.1 Our Approach

In this section, we discuss our ensemble approach illustrated in Figure 3. As shown in the figure, our approach consists of six ONNs where each ONN is trained on a different subset of D_{train}, and the predictions of those ONNs are aggregated with a soft voting strategy.

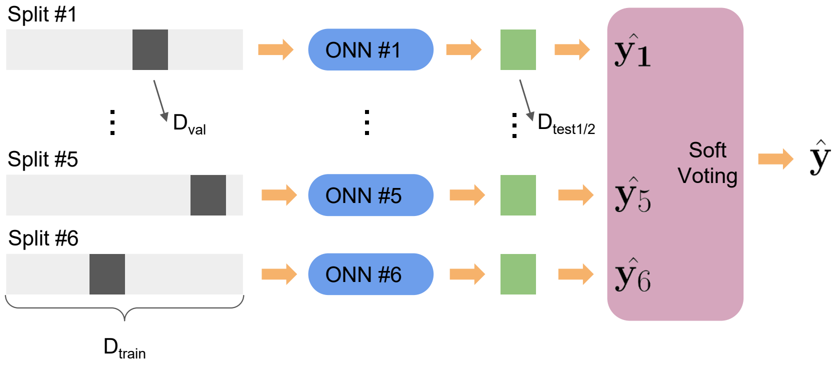

Fig. 3. Overview of our ensemble approach.

Training six ONNs. We construct $n = 6$ different versions of $D_{train'}$ and D_{val} by splitting D_{train} differently[9]. Intuitively, different ONNs trained based on different subsets of the dataset should improve the generalization of the final ensemble model, and provide better prediction results on a test set. The splitting strategy is based on an observation that the users in the test set consist of observed users in the training set as well as unseen users (i.e., no interaction history in the training set for those users), and consists of three main steps.

- First, for a given test set such as D_{test1}, we construct U_{test1} which contains users who are in both D_{train} and D_{test1} and having more than one QID associated with each user.
- Secondly, we sample one QID and its corresponding six examples for each user $u \in U_{test1}$ for constructing our validation set – D_{val}.
- Finally, we randomly sample P users from D_{train} who only have a single QID to construct a set of unknown users $U_{unknown}$, and use their QIDs and associated examples together with those samples in step 2 to construct D_{val}. $P = 5,517$ is determined based on the proportion of the unknown users in D_{test1}.

[9] We observe there is no performance improvement with a larger value of n.

For each $D_{train'}$ and D_{val}, we train an ONN by minimizing MSE (Mean Squared Error) for examples in $D_{train'}$, and adopt an early stopping strategy by observing the MRR score on D_{val} if there is no improvement with three consecutive epochs. Table 3 summarizes the set of features used for training ONNs where all of them are given by the challenge organizers. By removing each feature at a time, we confirmed that those features are all contributing to the performance, i.e., the performance is decreasing by removing any of those features. We treat the majority of features as sparse ones, i.e., categorical features, and consider *product_price* as a dense/numerical feature to be fed into an ONN. The categorical features are encoded with Ordinal Encoders[10], and we mask 20% of examples to better work with unseen categories. In addition, we also observed that setting sample weights higher for the examples of users who are not in the target test set further reduces overfitting and improves the MRR performance on the test set empirically. The hyper-parameters of ONNs are set as follows: *embedding_size*=4, *batch_size*=2048, *learning_rate*=0.001, *dnn_hidden_units*=(512, 128, 128) where the best MRR score can be achieved on the validation set.

Table 3. The set of features used in our approach with ONN.

Feature type	**Feature name**
Sparse	*user_id*
	session_id
	product_id
	page_type
	previous_page_type
	device_category
	device_platform
	user_tier
	user_country
	context_type
	context_value
	week
	week_day
Dense	*product_price*

Soft voting. After training those ONNs, we can predict the preference score of an example in a given test set such as D_{test1} or D_{test2}, which can be formalized as: $\hat{\mathbf{y}}_i = ONN_i(\mathbf{X_{test1}})$ where $\mathbf{X_{test1}}$ indicates the feature representation of the test set D_{test1}, ONN_i is i-th trained ONN, and $\hat{\mathbf{y}}_i$ is a vector referring to the preference score prediction of each example in the test set. Given the predictions of different ONNs, the final prediction score of an example $\hat{y} \in \hat{\mathbf{y}}$ in D_{test1} can

[10] shorturl.at/bjtGO

be obtained as follows:

$$\hat{y} = \sigma(\sum_{i=1}^{6} g(\hat{y_i})) \tag{2}$$

where $\sigma(\cdot)$ is the *sigmoid* function, and $g(\cdot)$ is the inverse function of the *sigmoid*. As the final prediction requires a ranked list, soft voting is better than hard voting using the predicted labels of those models for ranking. More sophisticated ensemble approaches such as training an additional model based on those predictions or using non-constant weighting functions [7] can be explored in the future but beyond our scope here.

3 Results

Table 4 shows the MRR results on D_{test1} during the 1st phase with different variants including a single ONN with sparse features only and that with the full set of features described in Table 3, and the ensemble approach. The results show that the ensemble approach improves the MRR score by 1.3% compared to a single ONN using sparse features.

Table 4. The MRR results on D_{test1} with different variants of our approach.

Method	**MRR**	**Impv.(%)**
ONN#1 (sparse features only)	0.450	-
ONN#1 (all features)	0.452	+0.4
Ensemble with six ONNs	0.456	+1.3

Table 5 shows the top-10 teams ranked in terms of MRR on the final test set provided by the organizers in the 2nd phase. *Baseline* is a baseline approach from the organizers, and we can observe that the proposed approaches from different teams can improve the MRR score by 0.9%∼19.8% compared to the baseline. As we can see from the table, our approach (user) achieves an MRR score of 0.47013 with 7.2% improvement over the baseline. However, the results also indicate that there is a significant room for improvement compared to the Second Life team which achieves the best-performing score of MRR (0.52571).

4 Conclusions

In this report, we presented our ensemble approach using ONNs for the Fashion Recommendation Challenge. Initial investigation of different DeepCTR models indicates that ONN outperforms other alternatives. In addition, the ensemble approach based on six ONNs trained with different subsets of the training dataset and a soft voting strategy shows further improvement for the challenge, and leads to the final MRR score of 0.47013 on the final test set. Despite of the promising results, the best performance from Second Life team indicates the

Table 5. The final ranking and MRR results on the final test set – D_{test2} – provided by the challenge organizers where our approach ranked 4th. *Baseline* indicates a baseline approach provided by the organizers.

Rank	**Team**	**MRR**
1	Second Life	0.52571
2	DeepBlue	0.49031
3	densityRecs	0.47407
4	user	0.47013
5	Data Clube	0.46793
6	allmightyML	0.46585
7	POLSL Team	0.44691
8	UFSJ	0.44681
9	Space Red Squirrel	0.44262
10	*Baseline*	0.43864

performance can be further improved, e.g., by considering the product-related attributes provided by the organizers, which we did not use in this work.

References

1. Chen, W., Zhan, L., Ci, Y., Yang, M., Lin, C., Liu, D.: Flen: leveraging field for scalable ctr prediction. arXiv preprint arXiv:1911.04690 (2019)
2. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 191–198 (2016)
3. Gonçalves, D., Chaves, I., Gomes, J., Nogueira, P., Otto, T., Marinho, V.: Farfetch fashion recommendations challenge – ecml-pkdd discovery challenge 2021. Proceedings of ECML-PKDD 2021 Discovery Challenge (2021)
4. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247 (2017)
5. Huang, T., Zhang, Z., Zhang, J.: Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 169–177 (2019)
6. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD. pp. 1754–1763 (2018)
7. Mendes-Moreira, J., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: A survey. Acm computing surveys (csur) **45**(1), 1–40 (2012)
8. Smith, B., Linden, G.: Two decades of recommender systems at amazon. com. Ieee internet computing **21**(3), 12–18 (2017)
9. Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., Tang, J.: Autoint: Automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM CIKM. pp. 1161–1170 (2019)
10. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)
11. Yang, Y., Xu, B., Shen, S., Shen, F., Zhao, J.: Operation-aware neural networks for user response prediction. Neural Networks **121**, 161–168 (2020)