Journal of Signal Processing Systems (2023) 95:271–280
https://doi.org/10.1007/s11265-022-01773-4

Object-aware Policy Network in Deep Recommender Systems

Guoqiang Zhou[1] · Zhangxian Xu[1] · Jiayin Lin[2] · Shudi Bao[3] · Liliang Zhou[4] · Jun Shen[2]

Received: 24 October 2021 / Revised: 26 March 2022 / Accepted: 5 May 2022 / Published online: 13 May 2022
© The Author(s) 2022

Abstract
Deep learning has been successfully applied in the recommender system. Low-dimensional dense embedding is typically used to represent the feature of users and items. To optimize the model, some models propose to dynamically search the embedding size based on the popularity of different users and items. However, these models ignore the interaction between the user and the item which will hinder the optimization of the features in embedding. In this paper, we propose Object-aware Policy Network (OPN) and introduces an object-aware method that is used for optimizing the features in embedding. We evaluate our model on the two real-world benchmark datasets. With less than 10% increased time consumption in all experiments, the results show that our proposed model is able to improve the performance of binary classification task by a margin of 0.30 and multiclass classification task by a margin of 0.35 compared to the best accuracies achieced by baselines on different datasests.

Keywords Recommender system · Deep learning · Embedding · Object-aware · Feature optimizing

1 Introduction

Deep learning techniques have been applied to many recommender systems [1–3] for its excellent performance in nonlinear transformation and representation learning. Deep recommender systems commonly consist of a representation layer and an inference layer. The former learns to map discrete user and item identifiers into real-value embedding representations and the latter takes this as input to calculate it with MLP (Multi-Layer Perception) or FM (Factorization Machine) [4] to get the prediction. Lots of works improve the learning models' performance by capturing high-order feature interactions [5], applying the attention mechanism [6] or combining the factorization machines with the MLP [5], but many interesting researches [1, 7, 8] have shown that the representation layer is also an important factor for the performance improvement of the model.

Recently, research on the representation layer mainly focused on searching the embedding size for different users and items. ESPAN (Embedding Size Adjustment Policy Network) [9], NIS (Neural Input Search) [10], and AutoEmb (AutoML Based End-to-end Framework) [11] uphold that the embedding size should be dynamically changed according to the popularity of a user-item pair. Applying high-dimensional embeddings to the user/item with low frequency will lead to overfitting due to over-parameterization

✉ Jun Shen
jshen@uow.edu.au

Guoqiang Zhou
zhougq@njupt.edu.cn

Zhangxian Xu
xzhangxian@foxmail.com

Jiayin Lin
jl461@uowmail.edu.au

Shudi Bao
shudi.bao@nbut.edu.cn

Liliang Zhou
levin_1221@163.com

[1] College of Computer Science, Nanjing University of Posts and Telecommunications, Xianlin, Jiangsu 210046 Nanjing, China

[2] School of Computing and Information Technology, University of Wollongong, Street, Wollongong 2522, NSW, Australia

[3] School of Electronic and Information Engineering, Ningbo University of Technology, Kongpu, Ningbo 315200, Zhejiang, China

[4] Southwest Institute of Electronic Technology, Jinniu, Chengdu 610036, Sichuan, China

[12]. And such a model cannot be trained effectively [11] when applying low-dimensional embeddings to the user/item with high frequency. However, the models mentioned above improve the model only in the aspect of the embedding size without optimizing features in the embeddings, which will impact the accuracy of the model. To address the above problem, in this work, we present an Object-aware Policy Network (OPN) on the basis of the prior work of ESPAN [9]. 'Object' means the interaction target of the user and item. 'Policy Network' is a model proposed in the ESPAN and the 'Policy' defines how to adjust embedding size of the user and item. The proposed model introduces the object-aware method to optimize the features by learning user preference when user interacts with different items. The design of the OPN has taken the following considerations.

Firstly, features play a central role in the success of many predictive systems. For many real-world tasks, the lack of the high-quality features often affects the effectiveness of a recommender system. Hence, many recommender systems involve operations of feature interaction. Secondly, to improve the FM [4], FFMs (Field-aware Factorization Machines) [13] saves several embeddings for a single feature field, which shows that, comparing to an independent the static manner, dynamic features in embeddings can yield better performance. Thirdly, [14] believes that different users show different preferences when interacting with different items, which means that we can optimize the features in embeddings with the interactions between the user and the item.

OPN optimizes the features in embeddings by capturing the interactions between the user and the item with object-aware method. Moreover, we have conducted extensive experiments on two public recommendation datasets and the results have indicated that our proposed model could outperform several baselines by a margin of 0.30 with barely little extra time consumption being introduced as overhead. Our main contributions are summarized as follows:

1. We propose an object-aware method which optimizes the features in embeddings by capturing the interactions between the user and the item.
2. In order to find the appropriate concatenated embedding size, we carried out a large number of experiments, and finally gave the most suitable size for the model.
3. We evaluate OPN on benchmark data, which shows consistent improvement over several competitive baselines. The remainder of this paper is organized as follows. First, we review related work in Sect. 2. Then, we introduce some preliminary knowledge for better understanding of our method in Sect. 3. Next, we detail the proposed method in Sect. 4. The experiment detail is introduced and analyzed in Sect. 5. Finally, this paper is concluded in Sect. 6.

2 Related Work

Recently, deep learning is gradually applied in recommender systems. Many models capture the latent feature interactions between the user and the item with deep learning's ability of non-linear transformation. [1] proposed Wide&Deep, which jointly trained wide linear models and deep neural networks to exploit the correlation available in the historical data and to explore new feature combinations. In order to handle the parameter explosion caused by models like FM [4] in the high-order feature interactions, [5] proposed a DCN (Deep & Cross Network) model that used a novel cross-network to efficiently learn certain bounded-degree feature interactions. [15] proposed NFM (Neural Factorization Machine) model, which enhanced the expression ability of FM by modelling high-order and non-linear feature interactions with novel Bilinear Interaction pooling. This operation could encode more informative feature interactions and facilitate the following deep layers to learn meaningful information. Attention is an important technique in recommendation system. [16] proposed AutoInt which automatically learns the high-order feature interaction of input features with multi-head self-attentive neaural network which could model different orders of feature combinations of input features. [17] proposed to learn the importance of each feature interaction from data with attention technique which is implemented by the attention net.

Deep recommender systems consist of two key components: a representation layer and an inference layer. All the works discussed above focus on the inference layer. After analyzing many earlier studies [7, 9–11, 13, 18, 19], we can conclude that the improvements of the representation layer are mainly summarized into two aspects: searching embedding size and optimizing the features in embeddings.

The appropriate size of the embeddings could either avoid the overfitting or effectively decrease the number of the parameters. [20] proposed the NAS (Neural Architecture Search) that leveraged reinforcement learning (RL) with RNN (Recurrent Neural Network) to train lots of candidate components to convergence. Inspired by the NAS, [10] proposed a NIS (Neural Input Search) model which searched for appropriate embeddings size in a collection of Embedding Blocks with a RL algorithm like ENAS (Efficient Neural Architecture Search) [21]. [11] proposed AutoEmb model, which was an AutoML based end-to-end framework. This model can automatically and dynamically leverage embedding with various dimensions. [9] proposed ESPAN model, which overcome the drawbacks of soft selection with hard selection.

It has been proved to be efficient to improve the model performance by optimizing the features in the embeddings

[18, 22, 23]. [13] proposed a FFMs model, which saved several embeddings for a single field for interactions with different fields, but the number of parameters in FFMs was in the order of the multiplication of the feature number and field number. To solve this problem, the FWFMS (Field-weighted Factorization Machines) model proposed by [7] had learned a field pair weight matrix and could effectively capture the heterogeneity of field pair interactions, thus greatly reducing the number of parameters. Work reported in [1] was a hybrid network structure that combined a wide model and a deep model, using feature engineering in the wide component to enhance the learning of deep component. However, feature engineering can be expensive and requires domain knowledge. [18] proposed a FGCNN (Feature Generation by Convolutional Neural Network) model, which generated sophisticated feature interactions automatically through machine learning models to avoid human intervention and obtain more useful feature interactions. [19] and [24] proposed to optimize the features in the embeddings with attention mechanism, which could learn the context-aware latent features based on text information. In order to capture user's diverse interests from historical behaviors, [25]proposed to adaptively learn the representation of user interests with a locl activation unit which will calculate the distribution of the user's interests with attention technique. As user's trend of interest will change all the time, [26] proposed to capture interest elvolving process with interest elvolving layer which embeds attention mechanism into the sequential structure novelly. [27] uses a hierarchical structure which incorporates character-level and word-level information and applies an attention mechanism to both levels to differentiate more important information from less important information.

Previous works ignored the possibility that the features in the embeddings could be affected by the interacted object which is important to the model [28–30]. In this paper, based on ESPAN [9], we propose an Object-aware Policy Network (OPN), which dynamically optimizes the features in the embeddings.

3 Preliminaries

3.1 Embedding

One-hot is a coding method commonly used in NLP (Natural Language Processing) and recommender systems. Though it can convert the categorical data to the form that can be easily used by machine learning, there are still some drawbacks in the practice. In natural language processing and understanding, the words are encoded as one-hot vectors, such as "programmer = [0, 0, 1, 0, 0, 0]" and "salesman = [0, 0, 0, 0, 1, 0]". However, this treats individual words as unique symbols and can't reflect characteristics of them.

To solve the above problems, it is suggested to apply an embedding layer upon the raw feature input to compress it to a low dimensional, dense real-value vector:

$$e_i = W_{embed,i} x_i \tag{1}$$

where e_i is the embedding vector, x_i is the one-hot vector for an item, and $W_{embed,i} \in \mathbb{R}^{n_e \times n_v}$ is the corresponding embedding matrix and n_e, n_v are the embedding size and number of the item, respectively.

As shown in Fig. 1, embedding layer maps the words to a joint latent factor space of dimensionality f. "programmer" and "salesman" are associated with embedding vectors $p, q \in \mathbb{R}^f$ respectively, and embeddings could measure the extent to which the word possesses those factors.

3.2 Field-aware

Inspired by FFMs [13], this section will introduce it for the convenience of understanding the method proposed in this paper. Introduction to the FM [4] will be shown first before the FFMs.

FM is the first model that computes the interactions between different field features in linear time with a linear number of parameters. It models the 2-order interaction between features as following:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j \tag{2}$$

where n denotes the number of fields, $w_0 \in \mathbb{R}$,$w \in \mathbb{R}^n$, $V \in \mathbb{R}^{n \times k}$. $\langle v_i, v_j \rangle$ is the dot product of two vectors, used to capture the 2-order interaction between features x_i and x_j.

However, FM models feature interactions with static embeddings while features from one field often interact differently with features from different other fields. Thus, FFMs proposed to model such difference explicitly by saving

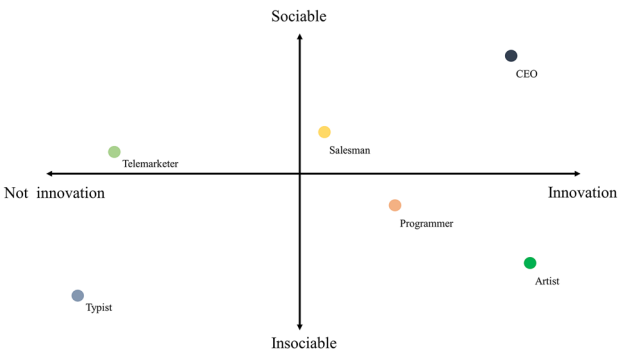


Figure 1 A simplified illustration of the embedding which map words to latent factor.

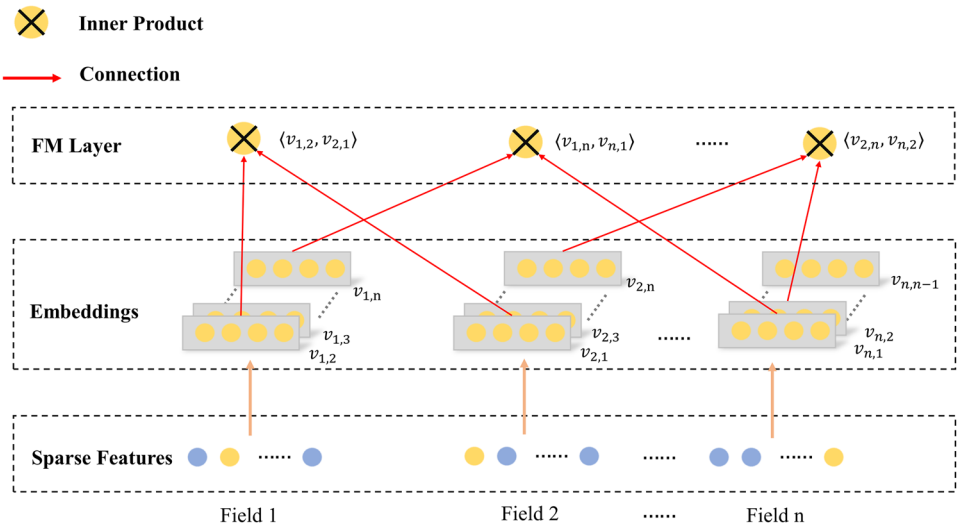


Figure 2 FFMs mode different features with corresponding embeddings.

several embeddings for a single field and choosing different embeddings when interacting with different other fields. It models 2-order interaction between feature:

$$\Phi_{FM}(\omega, x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \omega_{j_1,f_2} \cdot \omega_{j_2,f_1} \rangle x_{j_1} x_{j_2} \tag{3}$$

where f_1 and f_2 are respectively the fields of j_1 and j_2, $w \in \mathbb{R}^{n \times n}$ and n is the number of fields. As depicted in Fig. 2, FFMs save n - 1 embedding vectors for each feature. When modeling different features, FFMs only use the corresponding one $v_{i,j}$ to interact with another feature j.

4 Our Approach

In this section, we will present the object-aware method which effectively tackles the challenges mentioned in Sect. 1 via capturing the interaction between user and item with MLP layers. We will first provide the overview of the framework; next detail the feature optimizing model and recommendation model.

4.1 Overview

We aim to optimize the features in embeddings by capturing interactions between users and items. To this end, we propose the object-aware method to tackle this challenge based on ESPAN [9]. As depicted in Fig. 3, our framework consists of three core components: the policy network, feature optimizing model and the deep recommendation model.

The policy network in OPN serves as an RL agent that dynamically adjusts the embeddings size of users and items and provides appropriate embeddings for feature optimizing model. Next, the feature optimizing model will optimize the features in embeddings with object-aware method and unify the sizes of the input embeddings. Afterward, two unified embeddings will be concatenated and fed into the deep recommendation model to calculate the prediction results. Finally, the recommendation model and the feature optimizing model will be updated with the backward-propagation.

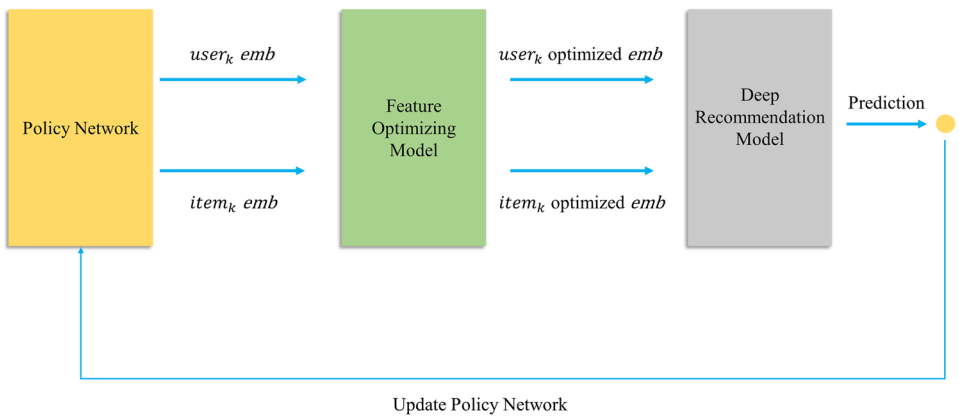


Figure 3 Overview of the OPN.

4.2 Object-aware Policy Network

4.2.1 Feature Optimizing Model

Commonly, embeddings generated from users and items will be fed into recommendation model directly. However, raw features in embeddings can't represent the users or items appropriately. So, we propose to capture the interactions between users and items to optimize the features by object-aware method.

As shown in Fig. 4, after fed into feature optimizing model, embeddings of the user $e^{(u)}$ and the item $e^{(i)}$ will be concatenated with a fixed size embedding of the item $e_f^{(i)}$ or the user $e_f^{(u)}$ as the following:

$$\begin{aligned} e'^{(u)} &= [e^{(u)} : e_f^{(i)}] \\ e'^{(i)} &= [e^{(i)} : e_f^{(u)}] \end{aligned} \tag{4}$$

Suppose we have n candidate embedding sizes for both of users and items $D = [d_1, d_2, \cdots, d_n]$, where $d_1 < d_2 < \cdots < d_n$. For simplicity, we denote $e'^{(u)}$ and $e'^{(i)}$ as e' which has embedding size $D' = [d'_1, d'_2, \cdots, d'_n]$, where $d'_i = d_i + D_f (i = 1, 2, \ldots, n)$. D_f is the fixed embedding size of $e_f^{(u/i)}$. Next, to capture the interactions between users and items, we send e, into the transformation map as following:

$$\begin{aligned} e'_2 &= W_{1\to 2} e'_1 + b_{1\to 2} \\ e'_3 &= W_{2\to 3} e'_2 + b_{2\to 3} \\ &\cdots \cdots \\ \hat{e} &= W_{n-1\to n} e'_{n-1} + b_{n-1\to n} \end{aligned} \tag{5}$$

where e'_k is any embedding with dimension $d_k (k = 1, 2, \cdots, n-1)$, $\hat{e}$ is embedding with dimension d_n. $W_{k-1\to k}$ and $b_{k-1\to k}$ are learnable weight and bias parameters. Furthermore, $e'$with d'_n embedding size will be transformed as the following:

$$\hat{e} = W_n e'_n + b_n \tag{6}$$

where W_n and b_n are learnable weight and bias parameters. Thus, we can ensure that embedding e'_j with dimension $d_j (j = 1, 2, \cdots, n)$ will always be transformed into the embedding with dimension d_n.

4.2.2 Deep Recommendation Model

As depicted in Fig. 5, two embeddings are fed into the deep recommendation model. Since the model adopts two pathways to model users and items, we combine the features of two pathways by concatenating them

$$\alpha_0 = [\hat{e}^{(u)} : \hat{e}^{(i)}] \tag{7}$$

where $\hat{e}^{(u)}$ and $\hat{e}^{(i)}$ are embeddings of users and items. This design has been widely adopted in many deep learning works [31, 32]. To capture the interactions between user and item latent features we feed them into inference layer, which consists of MLP with m layers:

$$\begin{aligned} \alpha_1 &= \sigma(W_1 \alpha_0 + b_1), \\ \alpha_2 &= \sigma(W_2 \alpha_1 + b_2), \\ &\cdots \cdots \\ \alpha_l &= \sigma(W_l \alpha_{1-1} + b_l) \end{aligned} \tag{8}$$

where $W_l \in \mathbb{R}^{k_l \times k_{l-1}}$ and $b_l \in \mathbb{R}^{k_l}$ represent the model weight and bias of the l-th layer. k_l is the number of neurons in the l-th layer perception. l is the layer depth and the σ is an

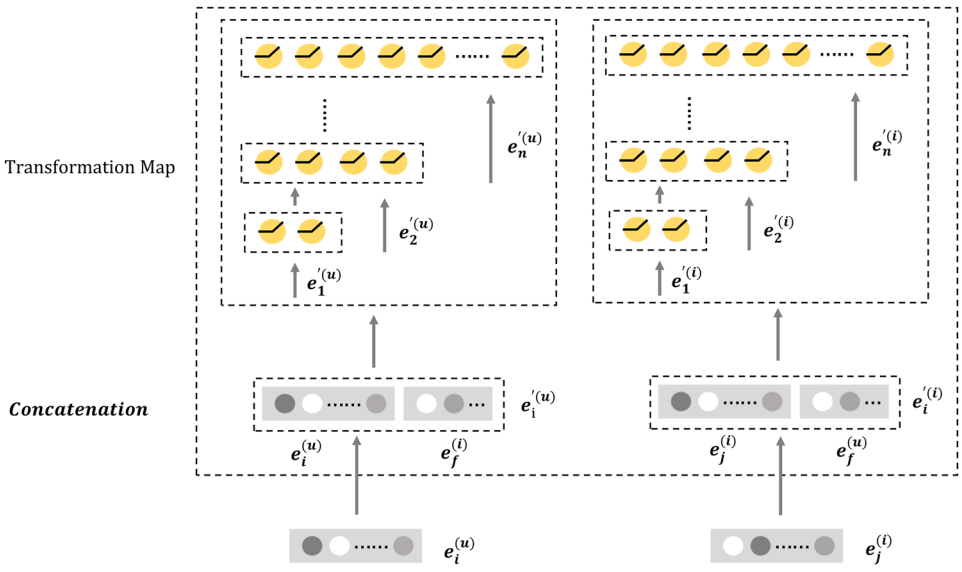


Figure 4 Overview of the feature optimizing model.

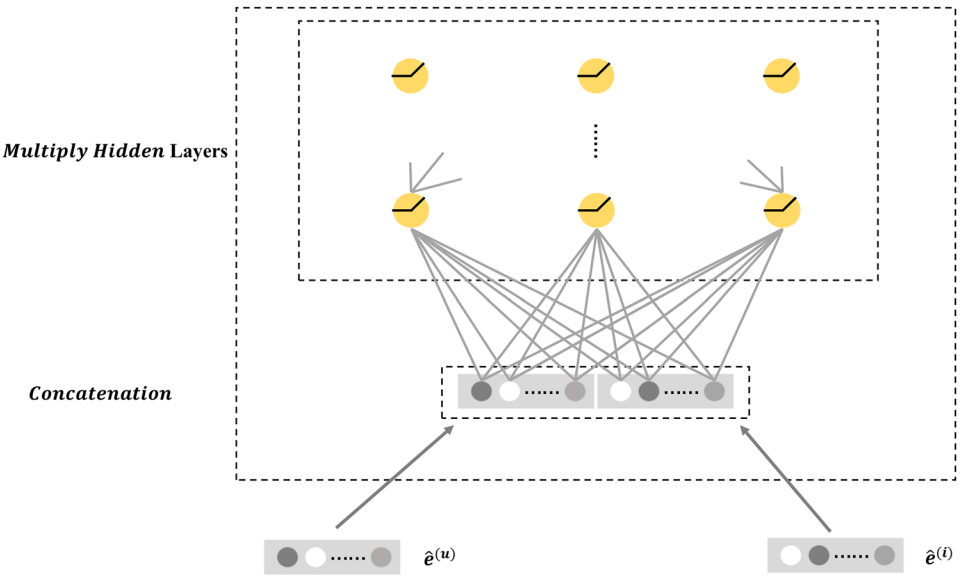


Figure 5 Overview of the deep recommendation model.

activation function. After that, α_l is then used to calculate the prediction results through an activation function.

4.3 Model Analysis

Suppose we have n candidate embedding sizes for both of users and items $D = [d_1, d_2, \cdots, d_n]$, where $d_1 < d_2 < \cdots < d_n$. The number of parameters in ESPAN is $d_1^2 + d_2^2 + \cdots + d_n^2$ and in OPN is $(d_1 + K)^2 + (d_2 + K)^2 + \cdots + (d_n + K)^2$, where K is the size of the concatenated embedding.OPN use $2K(d_1 + d_2 + \cdots + d_n) + nK^2$ additional parameters in the whole model and the number of parameters of OPN is not significantly more than that of ESPAN. However, it's hard to select the size of the concatenated embedding and we have to carry out a large number of experiments to try every possible size to alleviate this problem.

5 Experiments

To validate the effectiveness of our proposed method, we carry out comprehensive experiments on two real-world datasets for recommendation benchmark purpose. In the experiment, we want to validate and testify:

- **(Q1)** How does the proposed model perform compared to the traditional models and ESPAN?
- **(Q2)** How to choose the size of the concatenated embeddings for the object-aware method?
- **(Q3)** How does object-aware method affect the training time of the model?

In this section, we will first introduce the experimental settings, then present the overall performance comparison and time consumption with discussion.

5.1 Dataset

The following two datasets are used to evaluate the proposed model:

- **MovieLens 20M Dataset (ml-20m)** [33]: This is a public movie rating dataset from grouplen. It has 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users and it is guaranteed that every user has rated at least 20 movies.
- **MovieLens Latest Datasets (ml-latest)** [34]: This is the newest public movie rating dataset from grouplen. It has 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users and every user has rated at least 1 movie.

5.2 Parameter Settings

First, we set six candidate embedding sizes $D = \{2, 4, 8, 16, 64, 128\}$ for both the users and items. The recommender system and the policy network are trained upon the prior work of ESPAN [9]. Adam [35] optimizers with an initial learning rate of 0.003 and 0.0001 are used for about two networks, respectively. Dimension of the concatenated embeddings is set as 8. The whole framework is trained on mini-batches with a size of 500.

5.3 Platform

All experiments are conducted on a Windows PC with CPU Intel(R) Core(TM) i5-9300H @2.40GHz, 16GM memory and GTX 1650 with 4GB memory.

Table 1 Overall performance comparison on baselines and OPN.

ml-20m				
Method	Binary		Multiclass	
	MSE Loss	Accuracy	Accuracy	CE Loss
FIXED	72.13	0.1845	49.45	1.1517
DARTS	72.18	0.1836	49.85	1.1423
AutoEmb	72.27	0.1828	49.94	1.1399
ESPAN	72.98	0.1785	50.98	1.1155
OPN	**73.30**	**0.1766**	**51.21**	**1.1101**
ml-latest				
Method	Binary		Multiclass	
	MSE Loss	Accuracy	Accuracy	CE Loss
FIXED	72.13	0.1845	50.01	1.1414
DARTS	72.22	0.1834	50.46	1.1304
AutoEmb	72.36	0.1823	50.47	1.1311
ESPAN	72.88	0.1790	51.11	1.1147
OPN	**73.18**	**0.1773**	**51.46**	**1.1072**

5.4 Baseline

We compare our model with the following four baseline methods:

- **FIXED**: The original recommendation model with fixed embedding size and we choose this model as baseline to show that multi-size embedding could improve the performance of the model.
- **DARTS** [36]: This is a classic model based on multi-size embedding.
- **AutoEmb** [11]: A DARTS-based method assigns weights based on the frequency of a given user and item. The whole framework is end-to-end differentiable with soft selection. We set it as the baseline model to show that hard selection could improve prediction accuracy more than soft selection.
- **ESPAN** [9]: We propose the OPN on the basis of this model and set it as the baseline model to prove the effectiveness of the object-aware method.

5.5 Evaluation Metric

We evaluate the baselines and our model on the following tasks:

- **Binary Classification Task:** Users' five ratings are divided into positive and negative labels. Positive label includes 4-star and 5-star, and other ratings are assigned to the negative label. Predicting the correct label is the recommendation model's task and we will evaluate the model by classification accuracy and mean-squared-error loss.
- **Multiclass Classification Task:** Users' ratings are divided into 5 levels and the model should predict the correct rating. Evaluation of the model will be evaluated by classification accuracy and cross-entropy loss.

5.6 Performance Comparison (Q1)

The target of the recommendation model is to promote the prediction accuracy, so we set the prediction accuracy as the evaluation indicator. We carry out experiments for baseline models on different datasets and compare their results with the OPN. The result shows that OPN model improves the prediction accuracy on ml-20m and ml-latest datasets.

Table 1 shows the overall performance of all models on two datasets and we can have the following observations:

- All the models with dynamic changing embedding size outperform the baseline FIXED on all the datasets and tasks. According to this observation, we conclude that the performance of a recommender system can be boosted by utilizing dynamic embedding strategy.
- DARTS, ESPAN and OPN outperform the DARTS and AutoEmb on all datasets and tasks because they avoid the interference of the redundant information from the embeddings of other sizes with hard selection [9]. Therefore, we can boost the model performance by applying hard selection instead of soft selection.
- Our model outperforms ESPAN on all datasets and tasks. The main difference between our proposed model and

Table 2 Overall performance comparison between ESPAN and OPN with different selected concatenated embeddings.

ml-20m					
Method	Embedding size	Binary		Multiclass	
		MSE Loss	Accuracy	Accuracy	CE Loss
ESPAN	-	72.98	0.1785	50.98	1.1155
OPN	**2**	72.67	50.94	72.74	50.88
	4	73.16	51.06	73.16	51.29
	8	73.30	**51.21**	**73.18**	**51.46**
	16	**73.35**	51.04	72.98	51.22
ml-latest					
Method	Embedding size	Binary		Multiclass	
		MSE Loss	Accuracy	Accuracy	CE Loss
ESPAN	-	72.13	0.1845	50.01	1.1414
OPN	**2**	0	0.1802	1.1162	0.1796
	4	0	0.1776	1.1135	0.1776
	8	0	0.1766	**1.1102**	**0.1773**
	16	0	**0.1762**	1.1141	0.1785

ESPAN is whether involving object-aware mechanism or not. Such experimental result demonstrates the effectiveness of our proposed object-aware mechanism.

5.7 Selection of the Concatenated Embedding Size (Q2)

In object-aware method, there is a fixed dimension embedding that is concatenated to the user or item embedding. We need to know how to choose size of this concatenated embedding and carry out a large number of experiments by recording the result of the model with different dimensions of the concatenated embedding. The result shows that the concatenated embeddings with 8 dimension could improve the prediction accuracy the most.

As presented in the Table 2, increasing size of the concatenated embeddings improves the performance of the model at the beginning, however their performance begins to degrade when dimension size is greater than 8, except the binary task on ml-20m dataset.

Therefore, we can see that: **(i)** only concatenated embeddings with 8 dimension could significantly outperform the ESPAN on all datasets and tasks. **(ii)** concatenated embeddings with 2 dimensions showed the worst performance on most of the dataset and tasks. It is easy to explain this since inefficient features contained in the embeddings could hardly provide sufficient information for the deep learning models. **(iii)** Appropriate size of the concatenated embeddings was a key factor for the performance of the OPN, some models with improper embeddings size performed even worse than the ESPAN.

5.8 Efficiency of the Model (Q3)

The efficiency of the deep learning model is very important for real-world production systems. FFMs introduces large amounts of parameters for the field-aware method, which increases the complexity of the model and requires more time to train the model. Therefore, we want to explore how the object-aware method will affect the training time of the model by recording the training time of these two models.

We measured the training time of the model on MovieLen 20M and MovieLen Latest, respectively. As the Fig. 6 shows, except of the multiclass task on MovieLens Latest dataset, the training time of OPN increased by less

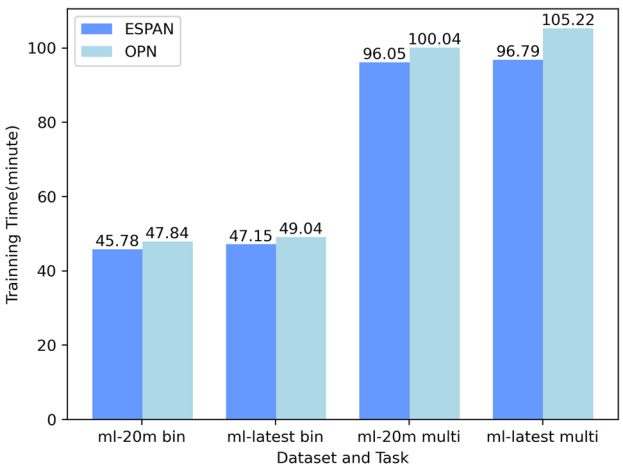


Figure 6 Comparison of the time consumption.

than 10% in all experiments, indicating that OPN achieved better performance with little time consumption overhead introduced.

6 Conclusion

The well-known ESPAN unifies different embedding sizes by sending embeddings to a series of linear transformation without consideration of the interaction between the user and the item. Inspired by FFMs, we believe that the features in embeddings should be different when a user interacts with different items. Hence, an object-aware method is proposed to further optimize the ESPAN. However, the concatenated embedding size is hard to choose in object-aware method, and we will attempt to implement the this method in a new way.

We conducted comprehensive experiments on two public real-world datasets and the result shows that our model outperforms the baseline models on all datasets, while introducing little extra time consumption.

Author Contributions Conception and design of study: Guoqiang Zhou. Aquisition of data: Zhangxian Xu. Analysis and interpretation of data: Zhangxian Xu. Drafting the manuscript: Zhangxian Xu, Jiayin Lin, Shudi Bao. Revising the manuscript critically for important intellectual content: Jun Shen, Shudi Bao. Approval of the version of the manuscript to be published: Guoqiang Zhou, Zhangxian Xu, Linjia Yin, Shudi Bao, Jun Shen

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. Jun Shen is partially supported by ARC DP180101051 and this work was supported by National Key Research and Development Program of China (grants 2020YFA0713600).

Declarations

Ethics Approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendents or comparable ethical standards.

Informed Consent Informed consent was obtained from all individual participants included in the study.

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems.
2. Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction.
3. Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks.
4. Rendle, S. (2010). Factorization machines.
5. Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions.
6. Chen, Q., Zhao, H., Li, W., Huang, P., & Ou, W. (2019). Behavior sequence transformer for e-commerce recommendation in alibaba.
7. Pan, J., Xu, J., Ruiz, A. L., Zhao, W., Pan, S., Sun, Y., & Lu, Q. (2018). Field-weighted factorization machines for click-through rate prediction in display advertising.
8. Qu, Y., Fang, B., Zhang, W., Tang, R., Niu, M., Guo, H., Yu, Y., & He, X. (2018). Product-based neural networks for user response prediction over multi-field categorical data.
9. Liu, H., Zhao, X., Wang, C., Liu, X., & Tang, J. (2020). Automated embedding size search in deep recommender systems.
10. Joglekar, M. R., Li, C., Chen, M., Xu, T., Wang, X., Adams, J. K., Khaitan, P., Liu, J., & Le, Q. V. (2020). Neural input search for large scale recommendation models.
11. Zhao, X., Wang, C., Chen, M., Zheng, X., Liu, X., & Tang, J. (2020). Autoemb: Automated embedding dimensionality search in streaming recommendations.
12. Kang, W.-C., Cheng, D. Z., Chen, T., Yi, X., Lin, D., Hong, L., & Chi, E. H. (2020). Learning multi-granular quantized embeddings for large-vocab categorical features in recommender systems.
13. Juan, Y., Zhuang, Y., Chin, W.-S., & Lin, C.-J. (2016). Field-aware factorization machines for ctr prediction.
14. Tu, C., Liu, H., Liu, Z., & Sun, M. (2017). Cane: Context-aware network embedding for relation modeling.
15. He, X., & Chua, T.-S. (2017). Neural factorization machines for sparse predictive analytics.
16. Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., & Tang, J. (2019). Autoint: Automatic feature interaction learning via self-attentive neural networks.
17. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., & Chua, T.-S. (2017). Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*.
18. Liu, B., Tang, R., Chen, Y., Yu, J., Guo, H., & Zhang, Y. (2019). Feature generation by convolutional neural network for click-through rate prediction.
19. Santos, C. D., Tan, M., Xiang, B., & Zhou, B. (2016). Attentive pooling networks.
20. Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning.
21. Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018). Efficient neural architecture search via parameters sharing.
22. Belhadi, A., Djenouri, Y., Srivastava, G., Djenouri, D., Lin, J. C.-W., & Fortino, G. (2021). Deep learning for pedestrian collective behavior analysis in smart cities: A model of group trajectory outlier detection. *65*, 13–20.
23. Belhadi, A., Djenouri, Y., Djenouri, D., Michalak, T., & Lin, J. C.-W. (2020). Deep learning versus traditional solutions for group trajectory outliers.

24. Shao, Y., Lin, J. C.-W., Srivastava, G., Jolfaei, A., Guo, D., & Hu, Y. (2021). Self-attention-based conditional random fields latent variables model for sequence labeling. *145*, 157–164.
25. Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., & Gai, K. (2018). Deep interest network for click-through rate prediction. (pp. 1059–1068).
26. Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., & Gai, K. (2019). Deep interest evolution network for click-through rate prediction. (pp. 5941–5948). volume 33.
27. Lin, J. C.-W., Shao, Y., Djenouri, Y., & Yun, U. (2021). Asrnn: a recurrent neural network with an attention model for sequence labeling. *212*, 106548.
28. Kossen, J., Stelzner, K., Hussing, M., Voelcker, C., & Kersting, K. (2019). Structured object-aware physics prediction for video modeling and planning. *arXiv preprint arXiv:1910.02425*.
29. Ho, D., Rao, K., Xu, Z., Jang, E., Khansari, M., & Bai, Y. (2021). Retinagan: An object-aware approach to sim-to-real transfer. (pp. 10920–10926).
30. Gao, C., Chen, J., Liu, S., Wang, L., Zhang, Q., & Wu, Q. (2021). Room-and-object aware knowledge reasoning for remote embodied referring expression. (pp. 3064–3073).
31. Srivastava, N., Salakhutdinov, R., et al. (2012). Multimodal learning with deep boltzmann machines.
32. Zhang, H., Yang, Y., Luan, H., Yang, S., & Chua, T.-S. (2014). Start from scratch: Towards automatically identifying, modeling, and naming visual attributes.
33. grouplens. (2016). Movielens 20m dataset. https://grouplens.org/datasets/movielens/20m/
34. grouplens. (2018). Movielens latest dataset. https://grouplens.org/datasets/movielens/latest/
35. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.
36. Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.