Towards Better Representation Learning for Personalized News Recommendations: A Multi-Channel Deep Fusion Approach

Jianxun Lian[†], Fuzheng Zhang[‡], Xing Xie[‡], Guangzhong Sun[†]
[†]University of Science and Technology of China, Hefei, China
[‡]Microsoft Research, Beijing, China
jianxun.lian@outlook.com, {fuzzhang, xingx}@microsoft.com, gzsun@ustc.edu.cn

Abstract

Millions of news articles emerge every day. How to provide personalized news recommendations has become a critical task for service providers. In the past few decades, latent factor models has been widely used for building recommender systems (RSs). With the remarkable success of deep learning techniques especially in visual computing and natural language understanding, more and more researchers have been trying to leverage deep neural networks to learn latent representations for advanced RSs. Following mainstream deep learning-based RSs, we propose a novel deep fusion model (DFM), which aims to improve the representation learning abilities in deep RSs and can be used for both candidate retrieval and item re-ranking. There are two key components in our DFM approach, namely an inception module and an attention mechanism. The inception module improves the plain multi-layer network via leveraging of various levels of interaction simultaneously, while the attention mechanism merges latent representations learnt from different channels in a customized fashion. We conduct extensive experiments on a commercial news reading dataset, and the results demonstrate that the proposed DFM is superior to several state-of-the-art models.

1 Introduction

In the Internet age, people prefer to read digital news from online services (such as news websites and social media platforms) rather than from printed newspapers [1]. However, millions of news items are streaming from various sources every day, making it impossible and unnecessary for readers to go through all available articles. Thus, articles selection services are indispensable to news content providers, and a recommender system (RS) is the core component to provide personalized services.

Collaborative filtering (CF) is a group of technologies that make automatic recommendations for users by collecting preferences from the community. Some major bottlenecks for CF methods are the data sparsity problem and the cold-start problem, which are especially true for news reading scenarios. News articles are highly time-sensitive and 90% of the articles in our own news service are no longer delivered to users within two days. On the other side, content-based filtering (CBF) is a complementary technology to CF that reduces the cold-start problem. Thus, for news recommendations, leveraging both CF and CBF in a unified model is a better choice than using a pure CF model.

With the remarkable success of deep learning technologies in computer vision, speech recognition, and natural language understanding, in the recent years, many researchers have tried leveraging deep learning techniques on RSs. They find that neural networks can not only be used as a generalized matrix factorization framework, but also be used for representation learning on raw features. For example, CF is formulated as a neural network in [He *et al.*, 2017; Sedhain *et al.*, 2015], and [Covington *et al.*, 2016] combine both CF and CBF in a deep neural network framework, which can be viewed as a non-linear generalization of factorization techniques. [Guo *et al.*, 2017; Cheng *et al.*, 2016] propose to combine deep neural networks with linear models or factorization models to learn diverse levels of feature interaction. In this paper, we take full advantage of some promising deep learning techniques, and further propose two key components, i.e., the **inception module** and the **attentive fusion mechanism**, with the goal of learning better latent representations from various content features.

Fully connected feed-forward neural networks are used in [Covington *et al.*, 2016; Cheng *et al.*, 2016] to learn latent representations and unseen feature combinations. Inspired by the famous architectures of ResNet [He *et al.*, 2016] and GoogleNet [Szegedy *et al.*, 2015], we find that the plain fully connected network can be improved by changing the structure of the multi-layer architecture. Thus, we propose the inception module for learning item representation from contents. Basically, the inception module learns multiple networks with various depths in parallel, feeding the final activation layer with different levels of feature combination in terms of different network structures.

For better user profiling, a common practice in commercial systems is to collect data from multiple channels/services to enrich user features. We argue that simply treating features

[1]http://www.digitalnewsreport.org/survey/2017/overview-key-findings-2017/

from different channels equally and acquiring user vectors via the same static neural model for all users is not the optimal solution. On one side, different users have different engagement distribution over different channels/services. When a user is more active in the news reading channel, we are more confident in inferring his news reading interests merely from the news reading channel, thus we can reduce the weights of features from other channels; correspondingly, for new-joint users, we rely more on the data from auxiliary channels. On the other side, domain discrepancy may exist [Lian *et al.*, 2017], and we need to select useful signals according to the feature contents. Thus, we propose learning different user representations on different channels using the above-mentioned inception module, and dynamically fuse them via an attention network. We name the proposed framework with the deep fusion model (DFM). DFM can be used for item retrieval from millions of candidates and is also effective in precise re-ranking when equipped with some modern components, such as the deep&wide component [Cheng *et al.*, 2016]. To summarize, the key contributions of this paper are as follows:

- We propose a novel deep fusion model (DFM) for feature-aware representation learning. The method can be applied to both item retrieval and item re-ranking.
- We design two key components in the DFM. The inception module boosts representation learning ability by combining various network structures, while the attention mechanism is designed to solve the data diversity problem.
- We evaluate the proposed model comprehensively on a commercial news reading dataset, and experimental results demonstrate that our model outperforms state-of-the-art methods.

2 Our Proposed Model

A typical architecture for commercial recommendation systems contains a candidate retrieval layer and an item re-ranking layer [Covington *et al.*, 2016; Okura *et al.*, 2017]. In this section, we will introduce our Deep Fusion Model (DFM) and its applications in both candidate generation and item ranking. Let U and V denote the set of users and news, respectively. The training set is $\{(u_i, v_i), y_i\}_{i=1}^{n}$, where the pairs (u_i, v_i) are user-news dyads, and y_i are binary labels indicating whether user u_i has read news v_i. The goal is to predict the label of unobserved dyads (u_k, v_k). Both u_i and v_i are associated with sparse feature vectors, $\mathbf{x}_u$ and $\mathbf{z}_v$.

2.1 Key Components for Representation Learning

A simple but effective latent factor model for collaborative filtering is SVD [Koren *et al.*, 2009]. It associates each user u and item v with a latent factor vector $\mathbf{p}_u, \mathbf{q}_v \in \mathbb{R}^D$. Prediction is given by:

$$\hat{y}_{uv} = b_{uv} + \mathbf{p}_u^T \mathbf{q}_v \tag{1}$$

where b_{uv} denotes the bias scalars. [Chen *et al.*, 2012] proposes a unified framework called SVDFeature to summarize various feature-based matrix factorization models into a single model. Prediction is given by:

$$\hat{y}_{uv} = \mathbf{w}_s^T \mathbf{s}(u,v) + (\sum_{j=1}^{|X_u|} X_{uj}\mathbf{p}_j)^T (\sum_{j=1}^{|Z_v|} Z_{vj}\mathbf{q}_j) \tag{2}$$

where $\mathbf{s}$(u,v) denotes the union set of user features, item features and context features. $\mathbf{w}_s$ denotes the linear weights. By omitting the non-dot product part from Eq.(2), the representation for users and items from the SVDFeature model is $\tilde{\mathbf{p}}_u = \sum_{j=1}^{|X_u|} X_{uj}\mathbf{p}_j$ and $\tilde{\mathbf{q}}_v = \sum_{j=1}^{|Z_v|} Z_{vj}\mathbf{q}_j$.

Inception Module

Recently, researchers have used neural networks to learn advanced representation to replace $\tilde{\mathbf{p}}_u$ and $\tilde{\mathbf{q}}_v$. Following mainstream methods in deep neural RSs, we adopt the field-wise format to store the users' and items' raw features. Let m denote the number of feature fields. We have the input feature :

$$\mathbf{x}_u = [x_{fd_0}, x_{fd_1}, ..., x_{fd_j}, ..., x_{fd_{m-1}}]$$

Where a feature field x_{fd_j} can be continuous or categorical. Categorical features can be further split into two types: univalent feature (such as user ID, or bin index of a continuous feature), and multivalent feature (such as the set of topics the user has tracked). Each field is mapped to a D dimension vector which we call field embedding. For the continuous field, the embedding is acquired via a fully connected layer; for the univalent field, the embedding is acquired via a dictionary look-up operation; and for the multivalent field, we use an average pooling after looking up each individual variable's embedding so that the length of field embedding is fixed. The field embedding process is illustrated in Figure 1.

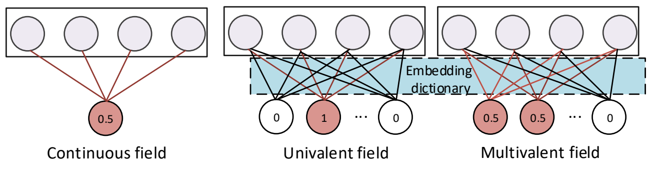


Figure 1: The field embedding layer. For univalent and multivalent fields, network weights (edges) are the feature embedding dictionary $\mathbf{E}$, and highlighted edges are corresponding latent vectors from dictionary look-up operations.

Let $\boldsymbol{l}_0$ denote the output of the embedding layer. $\boldsymbol{l}_0$ is a concatenated wide vector in length of $m \times D$:

$$\boldsymbol{l}_0 = [\boldsymbol{l}_{fd_0}, \boldsymbol{l}_{fd_1}, ..., \boldsymbol{l}_{fd_j}, ..., \boldsymbol{l}_{fd_{m-1}}]$$

Where $\boldsymbol{l}_{fd_j} \in \mathbb{R}^D$ denotes one field embedding. In [Covington *et al.*, 2016], $\boldsymbol{l}_0$ is fed to several layers of fully connected Rectified Linear Units (ReLU):

$$\boldsymbol{l}_{h+1} = \sigma(\mathbf{W}_h \boldsymbol{l}_h + \mathbf{b}_h) \tag{3}$$

Where $h \in [0, H]$ denotes the hidden layer index, and σ is ReLU. This type of multi-layer fully connected network is commonly used in recent studies [Covington *et al.*, 2016; Zhang *et al.*, 2016; Guo *et al.*, 2017]. Here we argue that it can be improved by more advanced structures. [Szegedy *et*

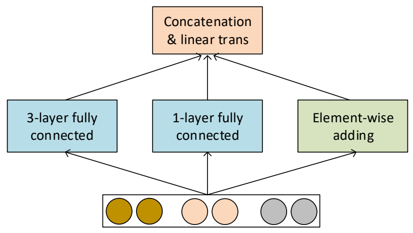


Figure 2: The proposed inception module.

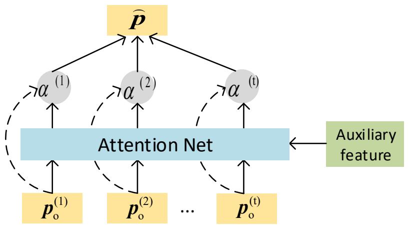


Figure 3: Attention mechanism for multi-channel fusion.

al., 2015] introduces the inception module in which a variety of convolutions are used in parallel and then the resulting feature maps are concatenated before going to the next layer. By doing so, the model can pick the best convolution size automatically and recover both local features with smaller convolutions and high abstracted features with larger convolutions. Besides convolution size, network depth is also of crucial importance for learning latent features. However, stacking more layers may result in an accuracy degradation problem, which is caused by the difficulty of network optimization rather than overfitting. Thus ResNet [He *et al.*, 2016] is proposed to let some layers fit a residual mapping instead of fitting a desired mapping directly. Inspired by the inception and ResNet framework, we propose our inception module for learning latent representations, with the structures depicted in Figure 2.

Different from visual computing, our input features are usually sparse and have no spatial proximity relationship, thus convolution filter is not feasible. The previous field embedding process can to some extent be regarded as the convolution operation which maps all features within the same field into one space. Here we exploit the network-in-network spirit [Lin *et al.*, 2013] via combining subnetworks at various depths. The element-wise addition block adds the field embedding vectors element-wisely and return a vector with the same length equal to the field embedding dimension. It serves as the SVDFeature component $\tilde{\mathbf{p}}_u = \sum_{j=1}^{|X_u|} X_{uj}\mathbf{p}_j$, and makes the other subnetworks fit a residual mapping. The depth of none-empty subnetworks can be arbitrary, for illustration we use two networks with 3 hidden layers and 1 hidden layer in Figure 2. Outputs from subnetworks are first concatenated horizontally, denoted as $\boldsymbol{l}_c$, and followed by a linear projection process:

$$\mathbf{p}_o = \mathbf{W}_o \boldsymbol{l}_c \tag{4}$$

where $\mathbf{p}_o$ denotes the output of our inception module. Note that the benefits of using the inception module are two-fold: first, we allow the model to recover latent features from both shallower and deeper interactions of raw features; second, we endow the model with the ability of picking the most suitable depth of network automatically.

Attentive Fusion Layer

As mentioned in Section 1, for better user modeling we leverage several domains' data to enrich the raw feature. A common approach is that we treat all features from different domains equally and feed them directly to the model to learn a final user embedding. We argue that it lacks flexibility in representation learning. Activity distribution among domains differs from user to user. For cold-start users in the news domain, it is of great significance to borrow knowledge from other domains; however, for heavy users, we already accumulate a certain amount of data from the news domain, thus it is not necessary to rely on other domains' data too much. In addition, content deviation should be taken into consideration. Incorporating domain-dependent knowledge may generate noise for the target domain [Lian *et al.*, 2017]. We leverage an attention mechanism to alleviate the above-mentioned factors and let the model dynamically combine knowledge from different domains. The structure is depicted in Figure 3.

We distinguish latent features from different channels[2] with superscript like $^{(t)}$. For instance, $\mathbf{p}_o^{(t)}$ denotes the output vector of the inception module of channel t. We use a two-layer neural network to compute the attentive score α, as:

$$a(i) = \mathbf{W}_i^2 \sigma\left(\sum_{j=1}^{t} \mathbf{W}_j^1 \mathbf{p}_o^{(j)} + \mathbf{W}_{aux}^1 \mathbf{x}_{aux} + \mathbf{b}^1\right) + \mathbf{b}_i^2 \tag{5}$$

where the superscript indicates the layer index, and $\mathbf{x}_{aux}$ are auxiliary features including users' activity distribution among different domains. More auxiliary features such as context information will be introduced in the next section. The attention scores are obtained by softmax normalization:

$$\alpha(i) = \frac{exp(\tau a(i))}{\sum_{j=1}^{t} exp(\tau a(j))} \tag{6}$$

where τ is a smoothing factor which is set empirically according to the validation set. The final representation for the user is calculated by $\widehat{\mathbf{p}} = \sum_{i=1}^{t} \alpha(i)\mathbf{p}_o^{(t)}$. A small trick is that we merge data from all channels together and name this fake channel as the *merged channel*. Different from user modeling, for items we do not have multiple domain's data. This means there is no attentive fusion process for items, and we use the output vector of the inception module directly.

2.2 Deep Fusion Model for Item Retrieval

The entire corpus of fresh news is exceptionally large (millions), which makes it infeasible to rank all items for each user on the fly. The candidate retrieval layer aims to narrow down the relevant news to a small subset (hundreds). There are several ways to retrieve relevant news, such as generating by story freshness, customized topics selected by the user, co-visited stories [Das *et al.*, 2007], and latent factor models

[2]In this paper, "domain" and "channel" are used interchangeably.

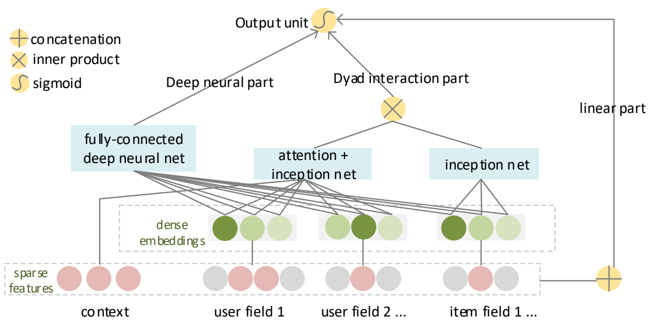


Figure 4: Deep fusion model for ranking.

[Covington *et al.*, 2016]. In this paper we use the latent factor method, which retrieves relevant news via collaborative filtering. The user-item relevance score is calculated by

$$s(u, v) = \widehat{\mathbf{p}}_u \cdot \widehat{\mathbf{q}}_v \qquad (7)$$

Which is exactly the dot product of two representation vectors. Thus item retrieval can be done through the approximate nearest neighbor searching (ANN) [Liu *et al.*, 2005].

2.3 Deep Fusion Model for Ranking

After the candidate generation process, items relevant to the target user are shrunk to a small set. The purpose of the ranking component is to provide elaborate personalization with more sophisticated models as well as more features, such as context-aware features. Some promising deep models such as Deep&Wide [Cheng *et al.*, 2016] reveal that combining memorization and generalization significantly improves recommendation systems. In the item retrieval section, we make recommendation only via the user-item interaction model. A straightforward way to improve it is to add heterogeneous components such as a linear model and a deep neural model. The new structure is illustrated in Figure 4. The linear model has the form $y_{linear} = \mathbf{w}_{li}^T \mathbf{s} + b$, and the deep neural model is a fully-connected neural network with each hidden layer in the form of $\mathbf{a}^{l+1} = f(\mathbf{W}^l \mathbf{a}^l + \mathbf{b}^l)$. Our model then makes prediction through:

$$\hat{y} = \sigma(\mathbf{w}_{li}^T \mathbf{s} + \mathbf{w}_{deep}^T \mathbf{a}^{l+1} + w_0(\widehat{\mathbf{p}}_u \cdot \widehat{\mathbf{q}}_v) + b) \qquad (8)$$

where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{w}_{deep}$ is the weights vector applied to the final hidden layer $\mathbf{a}^{l+1}$, and b is the bias term. The learning process aims to minimize the following objective function:

$$\sum_{i=1}^{n} f_{loss}(y_i, \hat{y}_i) + \lambda_\star ||\Theta|| \qquad (9)$$

where f_{loss} denotes the loss function. We take recommendation as a binary classification problem, so we use the logloss function in our experiments. We leave the extension for pairwise ranking framework as a future work. $\lambda_\star$ represents regularization constants.

3 Experiments

We aim to improve the recommendation system for Bing News, which is part of Microsoft's Bing search engine. We collect impression logs from the news reading domain over four consecutive days, and randomly sample about 35,000 users who have at least one click behavior for experiments. This subset consists of 84560 news articles and 1394085 impressions. To enrich the user features, we collect users' behavior over the last 90 days on three different channels, including news reading, general web browsing, and web searching. For item features we use the url (item ID), domain of the url, title length, news categories, and some other fields extracted from the title, such as topics and entities. For user features, we use the user ID, gender, age, locations, and the corresponding collection of item features from his/her reading histories. In item ranking experiments, additional context-aware features are used, including positions, temporal messages, and location information.

Parameter Settings. We implement our model based on TensorFlow. Parameters are optimized with mini-batch Adam. The hyper-parameters are tuned with grid-search on a hold-out validation set, with the best settings as follows: λ=0.001 for neural network parameters and λ=0.002 for parameters in embedding dictionary; τ=2.0; $learning\_rate$=0.001; activations=*tanh*; depths for inception module=[3,1,0]; dimension of embedding and hidden layers=32.

3.1 Experiments on Candidate Retrieval

We adopt the *leave-one-out* evaluation method to evaluate candidate generation models. We use the impressions in the first 3 days for training and validation, while on the fourth day, for each user we keep only the first clicked news as a positive instance. Negative instances in training set are dynamically sampling according to item popularity. For evaluation, since it is too time-consuming to rank all items, for each user we sample 100 items according to item popularity which are not interacted by the user, and rank the test item among the 100 items. This strategy is commonly used in previous studies such as [He *et al.*, 2017; Elkahky *et al.*, 2015; Koren, 2008]. The performance is judged by two popular metrics, i.e., **HR** (hit ratio) and **MAP** (mean average precision). HR measures whether the positive item is present on the top-K list, while MAP accounts for the position of the hit.

We report results of three models which can demonstrate the effectiveness of the attention mechanism and the inception module. We omit some baselines, such as random selection and recommending by item popularity, due to that they are weak and irrelevant to our focus.

YoutubeNet. The model introduced in [Covington *et al.*, 2016] for candidate generation. YoutubeNet treats features from all channels equally and leverages a deep neural network for learning representations.

AFM. Our proposed model without the inception module, which we refer to as the attentive fusion model (AFM). Besides treating multiple domains' feature equally, we also learn a separate user embedding for each domain, and the final user embedding is combined by an attention network.

DFM[3] (Deep Fusion Model). Based on AFM, we exploit the inception module to learn the latent representation.

[3]The source code is available at `https://github.com/Leavingseason/dfm`

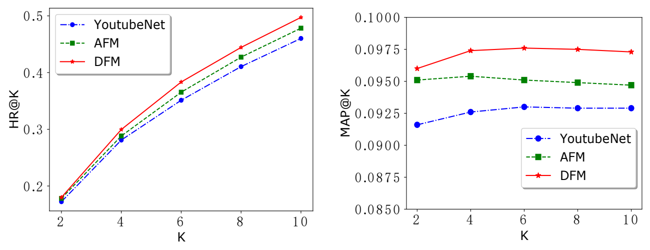
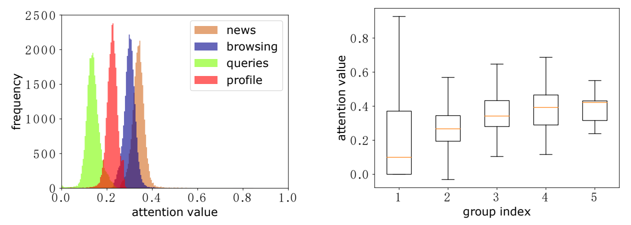


Figure 5: Performance in terms of HR (left) and MAP (right).

The performance comparison is shown in Figure 5. We observe that AFM consistently outperforms YoutubeNet with various settings of k, which demonstrates that domain behavior patterns vary among users and our attention mechanism can capture diversity effectively. In addition, DFM further improves AFM, which shows that learning neural networks with various depths simultaneously is superior to learning via the plain deep neural network. When $k = 10$, our DFM outperforms YoutubeNet by 7.4% in terms of HR and 4.7% in terms of MAP.



(a) Distributions of attention weights for different channels. (b) Attention weights of the news channel among different groups.

Figure 6: An exploration of attentive weights.

To study how the attention weights vary with different channels, we disable the merged channel and regard users' demographic information as a separated channel named *profile*. Figure 6a depicts the distribution of attentive weights for different channels, from which we can observe the channel importance to some extent. Behavior data from the news channel itself is weighted the heaviest, which accords with intuition. Queries (search behavior) relatively gain the least weights, which may due to that search behaviors are often triggered by some instant demands, and they contain more irrelevant messages for news recommendation. We are also interested in how attentive weights for one certain channel change among different users. As the easiest case, we study the relation between the news channel's attentive weights and users' engagement for the news channel. User's engagement for the news channel is defined as the relative ratio of behavior data for the news channel versus other channels. For instance, if a user is more active in the news channel and less active in other channels, his/her engagement for the news channel is high. We split the user into five groups according to the news channel engagement value, and the engagement values increase from group 1 to group 5. Figure 6b demonstrates that attentive weights are positively correlative with the engagement value.

comments	method	AUC	LogLoss
linear	LR(-)	0.7604	0.4673
	LR	0.8005	0.4422
non-parametric	GBDT(-)	0.8015	0.4417
	GBDT	0.8306	0.3825
factorization	FM	0.8228	0.4220
	SVDFeature	0.8271	0.4081
deep factorization	DeepFM	0.8283	0.3879
	Deep&Wide	0.8331	0.3620
	YoutubeRank	0.8242	0.3677
our approaches	AFM	0.8342	0.3593
	DFM	**0.8386***	**0.3553***

Table 1: Performance comparison among different models.

3.2 Experiments on Ranking

Next, we investigate the item re-ranking case. Again, we use the first 3 days of the impression log for training and validation, and leave the fourth day's impression log as the test set. None-clicked impressions are used as negative instances. Since we take the ranking as a binary classification problem, we use **AUC** (area under the ROC curve) and **LogLoss** (the negative log likelihood) as the evaluation metrics.

Effectiveness Comparison

We compare a variety of models in our experiments, including **LR** (logistic regression), **GBDT** (gradient boosting decision trees), **FM** (factorization machine), **SVDFeature** [Chen *et al.*, 2012], **DeepFM** [Guo *et al.*, 2017], **YoutubeRank** (the deep ranking network in [Covington *et al.*, 2016]), **Deep&Wide** [Cheng *et al.*, 2016], and our **DFM** as well as its variant **AFM**. Let **LR(-)** denote the LR model trained on data with the news reading channel only and the same goes for GBDT. For the other models we use all channels' data. The performance comparison is shown in Table **??**, from which we can observe that:

- GBDT and LR is significantly better than GBDT(-) and LR(-) respectively, which means that including more channels' data for user modeling is quite beneficial to user modeling.
- All factorization models greatly outperform the LR model; meanwhile, deep factorization models (those leverage neural networks) are superior to traditional factorization models (FM and SVDFeature). This observation demonstrates that advanced representation learning is of immense importance for sparse data.
- Our proposed deep fusion model (DFM) perform best among the baselines, and the relative improvement over the best baseline is 0.66% and 1.8% in terms of AUC and LogLoss, respectively. Specifically, AFM improves the Deep&Wide model with only a small gain, which may due to the fact that for the ranking task, we have already included a linear and deep component, which can learn the domain behavior diversity to some degree, so that our attention mechanism will not further benefit too

much. However, DFM further improves AFM significantly, which demonstrates the power of representation learning from the inception module.

Regularization Study

λ controls the strength of L_2 regularization in our DFM to prevent overfitting. Figure 7 depicts the trends of performance evolution with different settings of λ. We can see that DFM evidently overfits on the training set when λ is set too small (1e-4), while underfits the data with large λ (1e-2) . 1e-3 turns out to be a proper magnitude of value for λ.

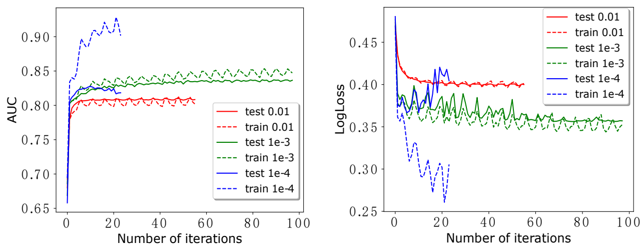


Figure 7: Training and test performance of DFM *w.r.t.* the number of iterations (mini-batch) in terms of AUC (left) and LogLoss (right).

Activation Functions

Next, we investigate the influence of the activation function on neural models. Figure 8 show the results. We observe that on our dataset, *tanh* is more suitable while *sigmoid* performs the worst.

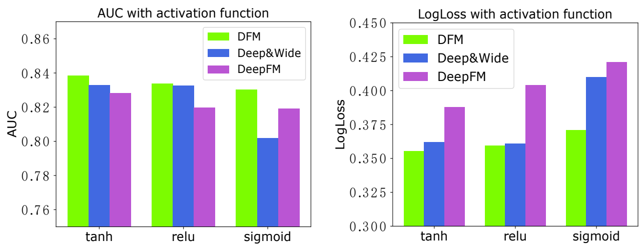


Figure 8: AUC (left) and LogLoss (right) with activation functions.

Number of Hidden Layers

Figure 9 depicts the influence of network depths on AUC and logloss. We find two interesting observations: (1) for all three models, adding a first hidden layer achieves a big improvement on both AUC and logloss metrics, and subsequent improvements brought by adding more hidden layers are relatively small; (2) the optimal network structure for DFM (3 layers) is of one layer deeper than that of DeepWide and DeepFM (2 layers), which demonstrates that the inception module can make the deep model easier to optimize.

4 Related Work

There is a vast amount of research in recommender systems, ranging from content-based filtering methods [Lops *et al.*, 2011] to collaborative filtering methods, from explicit feedback [Koren, 2008] to implicit feedback [Hu *et al.*, 2008].

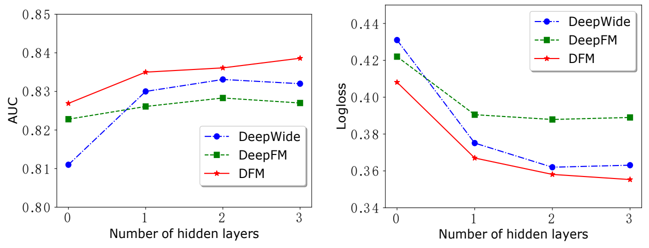


Figure 9: AUC and LogLoss with number of hidden layers.

The proposed model in this paper is based on deep learning techniques, thus in this section we mainly discuss related works in deep-learning based RSs.

Feature learning is the most straightforward application of deep learning in RSs. [Van den Oord *et al.*, 2013] use deep neural networks to learn factors from music audio, [Elkahky *et al.*, 2015] use deep neural networks for content-based recommendation, and [He and McAuley, 2016] use pre-trained deep networks to extract visual features for personalized ranking. Besides content learning, researchers are also interested in connecting collaborative filtering with deep learning. Collaborative filtering is formulated as autoencoders in [Sedhain *et al.*, 2015], and is further formulated as a generalized matrix factorization in [He *et al.*, 2017; Covington *et al.*, 2016]. [Wang *et al.*, 2015] utilize autoencoders to jointly perform deep representation learning for content information and the ratings matrix. Recently, the attention network has been used to improve the tradition model, such as [Chen *et al.*, 2017] for improving SVD++ and [He and Chua, 2017] for improving FM. Inspired by those works, we leverage the attention mechanism to tackle the data diversity challenge. To learn better latent representations, we further propose the inception module.

Autoencoders are widely used when the side information is given in terms of articles or images. However, in web data mining, input features are usually the multi-field type and are mostly categorical. In this thread, deep neural works are used not only for representation learning, but also for exploring feature interactions [Zhang *et al.*, 2016; Qu *et al.*, 2016]. [Cheng *et al.*, 2016] suggest combining combine memorization and generalization in one recommender engine. To do so, they incorporate a linear regression component and a deep neural network component in their final model. Similarly, [Guo *et al.*, 2017] incorporate a deep neural network component to extend the factorization machine. Inspired by them, we further incorporate our deep fusion component with the wide and deep part, and the resulting model perform best among existing models.

5 Conclusions

In this paper, we propose the deep fusion model for personalized news recommendations. There are two key components in our proposed model, i.e. the inception module which aims to learn better latent representations via leveraging various subnetworks in parallel, and the attention mechanism which contextually fuses diverse data across multiple channels. Through comprehensive experiments we have demonstrated that, on one hand, DFM learns good representations for users/items which are effective for candidate retrieval; on the other hand, when incorporated with a wide and deep part, the new model can provide state-of-the-art performance for item ranking.

References

[Chen *et al.*, 2012] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, 13(Dec):3619–3622, 2012.

[Chen *et al.*, 2017] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 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*, pages 335–344. ACM, 2017.

[Cheng *et al.*, 2016] 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. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, pages 7–10. ACM, 2016.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

[Das *et al.*, 2007] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280. ACM, 2007.

[Elkahky *et al.*, 2015] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1725–1731, 2017.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 355–364, 2017.

[He and McAuley, 2016] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150, 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272. IEEE, 2008.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 2008.

[Lian *et al.*, 2017] Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. A multifaceted model for cross domain recommendation systems. In *International Conference on Knowledge Science, Engineering and Management*, pages 322–333. Springer, 2017.

[Lin *et al.*, 2013] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[Liu *et al.*, 2005] Ting Liu, Andrew W Moore, Ke Yang, and Alexander G Gray. An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems*, pages 825–832, 2005.

[Lops *et al.*, 2011] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[Okura *et al.*, 2017] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1933–1942. ACM, 2017.

[Qu *et al.*, 2016] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1149–1154. IEEE, 2016.

[Sedhain *et al.*, 2015] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 111–112. ACM, 2015.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[Van den Oord *et al.*, 2013] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[Wang *et al.*, 2015] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

[Zhang *et al.*, 2016] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. In *European conference on information retrieval*, pages 45–57. Springer, 2016.