# On the Transferability of Deep Neural Networks for Recommender System

Duc Nguyen[1], Hao Niu[1], Kei Yonekawa[1], Mori Kurokawa[1], Chihiro Ono[1], Daichi Amagata[2], Takuya Maekawa[2], and Takahiro Hara[2]

[1] KDDI Research Inc., Japan
{du-nguyen,ha-niu, ke-yonekawa, mo-kurokawa, ono}@kddi-research.jp
[2] Osaka University, Japan
{amagata.daichi, maekawa, hara}@ist.osaka-u.ac.jp

**Abstract.** Recommender system is an essential component in many practical applications and services. Recently, significant progress has been made to improve performance of recommender system utilizing deep learning. However, current recommender systems suffers from the long-standing data sparsity problem, especially in domains with little data. With the ability to transfer knowledge across domains, transfer learning is a potential approach to deal with the data sparsity problem in recommender system. In this paper, we carry out an investigation on the transferability of deep neural networks for recommender system. We show that network-based transfer learning can improve recommendation performance on target domains by up to 20%. In addition, our investigation reveals that transferring the layers close to the output leads to better transfer performance. The transfer performance is also found to be dependent on the similarities between data distributions of the source and target domains. Meanwhile, target domain characteristics such as size and sparsity have little impacts on the transfer performance.

**Keywords:** Transfer Learning, Recommender System, Neural networks

## 1 Introduction

With the explosive growth of information available on the Internet, it is challenging for users to find their desired products/services. Thus, recommender systems (RSs) play a central role in enhancing user experience, especially in online news services, E-commerce websites, and online advertising [24]. The main task of RSs is to provide suggestions for items (e.g., news, books, movies, event tickets, etc.) to individual users. RSs enable the so-called *personalized experience*, which is the key to the successes of many Internet companies like Amazon [28], Netflix [8].

Starting with the Netflix Prize [3], significant progress has been made in recommender system research [33]. The past few years have also witnessed the great success of deep learning in many application domains, especially in computer vision and natural language processing [15]. In this trend, in the past few years, deep learning has been studied extensively for recommender system such

as in [1, 4, 10, 13, 25, 26, 34]. Although these deep learning-based methods are effective in improving the performance of recommender system, they are mostly based on information (e.g., ratings, reviews) in a single domain. As a result, these methods inevitably suffer from the data sparsity problem because each item is usually rated or reviewed by a few users [24]. Moreover, current applications should be able to react quickly to new situations such as new products or new users. Therefore, techniques to reuse knowledge across times, domains, and tasks are highly desirable.

Transfer learning is a machine learning technique capable of transferring knowledge learned in a domain (*source domain*) to another related domain (*target domain*) [22]. Thus, it can be used to deal with the data sparsity problem in recommender system as well as to increase system's ability to adapt to new situations. Existing works on transfer learning for recommender system apply either *instance-based* [5, 7, 16, 23] or *feature-based* [19, 35] approaches, in which data samples/features from one or more source domains are transferred to a target domain. One of the main problems of instance-based and feature-based transfers is that they require access to data of other source domains. In other words, data sharing between domains is necessary. Nevertheless, inter-domain data sharing has become more and more difficult nowadays due to data regulations such as GDPR [29], especially if the shared data contains user-relevant information.

To improve the performance of recommender systems, it is still desirable to be able to transfer knowledge across domains even if shared data is not available. In such circumstances, *network-based transfer learning*, which transfers features of model (e.g., parameters, structure, etc.) learned on a source domain, is a potential approach. Although network-based transfer has been studied extensively in the literature, previous works mainly focus either on computer vision [9,14,21,27,30,32] or natural language processing [6,11,12,18,31]. In context of recommender systems, despite the fact that deep neural network-based models have shown their superiority, there is still no existing work on the transferability of those deep neural networks.

In this paper, we focus on answering the following three questions in order to understand the transferability of neural network for recommender system.

- Q1: Does network-based transfer learning lead to better recommendation performance on the target domain?
- Q2: How to transfer a neural network for the best transfer performance?
- Q3: What are the factors affecting the transfer performance?

Although network-based transfer learning has been found to be effective in many computer vision and natural language processing(NLP) tasks, there is still a lack of understanding on the transferability of neural networks for recommender tasks (i.e., Q1). In computer vision and NLP tasks, those layers close to the input are found to be highly transferable, whereas those close to the output are task-specific [21]. Yet, it is still unknown which layers can be effectively transferred in recommendation tasks (i.e., Q2). It is also important to understand how different factors affect the transfer performance (i.e., Q3).

In this paper, we investigate the transferability of deep neural networks for recommender system, focusing on *top-N item recommendation task*. For that purpose, a recommender system built on Multi-layer Perceptron (MLP) neural network is used as the base network. The base network consists of an embedding layer and an interaction function consisting of multiple fully connected layers. Then, we examine various options to transfer the knowledge of the base network to a target domain. Extensive evaluation with eighteen real-world datasets demonstrate that transferring the interaction function layers can improve recommendation performance on the target domain by up to 20%. Especially, our evaluation reveals that, unlike deep neural networks for computer visions and NLP tasks, those layers close to the output are more transferable than those close to the input in deep neural networks for recommender system. To the best of our knowledge, this is the first work on transferablity of deep neural networks for recommender system.

The remaining of the paper is organized as follows. Section 2 surveys related works. The base network and transfer options are described in Section 3. The evaluation is given in Section 4. Finally, the paper is concluded in Section 5.

## 2 Related Work

In recent years, deep learning-based methods have been studied extensively for recommender systems. These methods mainly focus on replacing one or more components in conventional methods by deep neural networks. For instance, in [10], instead of using the dot product as in traditional matrix factorization [2], the interaction function is learned by a MLP network. In [25, 36], Autoencoder is utilized to learn the user/item embeddings. In [1], Gate Recurrent Unit is used to exploit the order of words in sentences, which is shown to outperform a simple average of word embeddings for text recommendation. Other deep neural network architectures such as Generative Adversarial Network (GAN) [4] and Attention Model [26] have also been used in recommender system. A comprehensive survey of deep learning-based methods can be found in [33]. In this paper, we adopt the MLP as the base network due to its simplicity. Other deep neural networks will be studied in our future work.

In the literature, transfer learning has been used to tackle the data sparsity problem in recommender system. Most transfer learning methods in previous studies are either *instance-based* or *feature-based*. In [5], training samples of a source domain are directly used to train the recommendation model at the target domain. In [16], users/items in a source domain are clustered to construct a codebook, which is then transferred to a target domain. In [19,35], the user/item feature vectors learned on a source domain are transferred to the target domain by means of a mapping function. Some other studies leverage *multi-task learning* to enable dual knowledge transfer across domains such as [13, 34]. However, instance-/feature-based transfers and multi-task learning require sharing data between domains. In contrast, our work focuses on *network-based transfer*, and thus does not require data sharing across domains. Such a property is especially

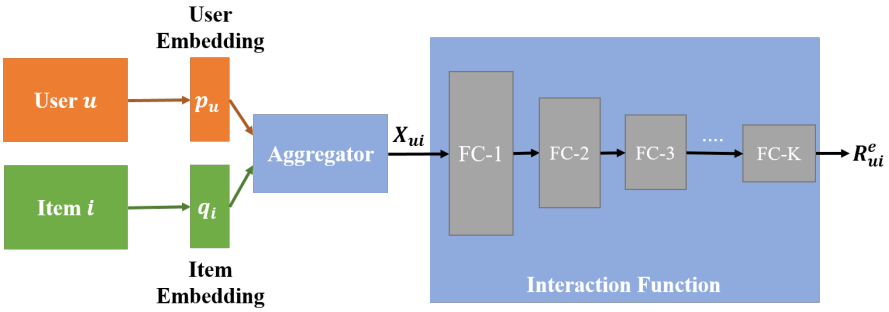



Fig. 1: Base network architecture for top-N item recommendation task.

important considering fact that more data regulations are being imposed on user data [29].

Network-based transfer learning has been studied in contexts of computer vision and natural language processing research. In [32], it is found that the first layers of Convolutional Neural Network (CNN) are highly transferable. The following works lead to the developments of various transfer techniques for image classification task. In [21], the output layer of a pre-trained CNN is replaced by an adaptation layer, while the remaining layers are transferred to the target domain. In [9], only the convolutional layers are transferred, while all the fully-connected layers of CNN networks are fine-tuned with learning rate determined by Bayesian Optimization. A recent evaluation [14] found that there is a strong correlation between ImageNet accuracy and transfer accuracy among popular image classification networks. To improve the performance of factoid question answering (QA) tasks on small datasets, the model parameters trained on a large dataset are used to initialize the target model's weights, with a modified loss function to avoid catastrophic forgetting [31]. In [12], an universal language modeling fine tuning (ULMFiT) is presented, featuring discriminative fine-tuning, slanted trianglar learning rates, and gradual unfreezing. In [11], an adapter-based parameter efficient transfer learning for NLP is proposed.

## 3 Network-based Transfer Learning

In this section, the top-N item recommendation task is defined and a neural network-based approach is introduced. Then, we describe how to transfer the a pre-trained network from a source domain to a target domain.

### 3.1 Top-N item recommendation task

Along with rating prediction [3], top-N item recommendation is one of most important tasks in recommender systems. Suppose that we need to recommend $N$ items to individual users of a particular domain (e.g., online book stores,

e-commercial websites). Let $\mathcal{U}$ and $\mathcal{I}$ respectively denotes the sets of users and items. We define the variables $\{R_{ui}\}$ to represent user-item interactions as follows.

$$R_{ui} = \begin{cases} 1 & \text{if user } u \text{ has interacted with item } i \\ 0 & \text{otherwise} \end{cases}$$

Here, an interaction can be a purchase or rating of the item, a click on the item's advertisement, or a visit to the item's website. In this paper, we assume that only implicit feedback is available. Thus, user-item interactions are represented by binary values. The set of items that a user $u$ has interacted with in the past is denoted by $\mathcal{I}_u$, i.e., $\mathcal{I}_u = \{i | R_{ui} = 1\}$. The top-N item recommendation problem can be formulated as follows.

*For a user $u \in \mathcal{U}$, determine $N$ items $\{i_1, i_2, .., i_N\} \in \mathcal{I} \setminus \mathcal{I}_u$ that have the highest likelihoods that the user $u$ will interact with.*

Existing methods for top-N item recommendation task can be classified into two main groups, namely *content-based*, and *collaborative filtering*. Content-based methods simply calculate the similarity between candidate items and the items the user has interacted with, then select top-N items with highest similarity scores. On the other hand, collaborative filtering predicts the interaction score by using preference from many users. In this paper, we focus on model-based CF to predict the value $R_{ui}$ for every item $i \in \mathcal{I} \setminus \mathcal{I}_u$. The model is built on top of a neural network and will be described in the next section.

### 3.2 Neural Network Model

In this paper, we follow the NeuMF framework proposed in [10] to build the base network as follows. Each user/item is characterized by a latent vector or embedding. The user-item interactions are modeled by an interaction function. Similar to [10], the interaction function is a Multi-layer Perceptron network, which is learned during training.

Figure 1 shows the architecture of the base network used in this paper. As aforementioned, each user $u \in \mathcal{U}$ is characterized by an embedding vector $p_u \in \mathbb{R}^{d_u}$, where $d_u$ is the user embedding size. Similarly, each item $i \in \mathcal{I}$ is mapped to an item embedding vector $q_i \in \mathbb{R}^{d_i}$ where $d_i$ is the length of the item embedding vectors. In this paper, we assume that the user and item embeddings have the same size, i.e., $d_u = d_i$. Given an interaction between user $u$ and item $i$, the corresponding user and item embeddings are aggregated by the *aggregator*, forming $X_{ui}$, which is the input of the *interaction function*. In this paper, the aggregator simply concatenates the user and item embedding vectors as follows.

$$X_{ui} = [p_u, q_i] \tag{1}$$

The interaction function consists of $K$ fully connected layers FC-$k$ ($1 \leq k \leq K$). Let $s_k$ denote the size of layer FC-$k$. The output $y_k \in \mathbb{R}^{s_k}$ of layer FC-$k$ ($1 \leq k \leq K$) is given by,

$$y_k = \begin{cases} f_k(X_{ui} * W_k + b_k) & \text{if } k = 1 \\ f_k(y_{k-1} * W_k + b_k) & \text{if } k > 1 \end{cases} \tag{2}$$

where $f_k$, $W_k \in \mathbb{R}^{s_{k-1} \times s_k}$, and $b_k \in \mathbb{R}^{s_k}$ respectively denotes the activation function, weight, and bias of layer FC-$k$. The outermost FC layer (i.e., FC-$K$) is also referred to as *output layer*. The base network parameter set $\theta$ includes the user and item embeddings and the layers' weights and biases.

$$\theta = \{\{p_u\}_{u \in \mathcal{U}}, \{q_i\}_{i \in \mathcal{I}}, \{W_k, b_k\}_{1 \leq k \leq K}\} \tag{3}$$

The parameter set $\theta$ is learned so as to minimize a loss $L$, which is a function of the predicted interaction and the actual ones.

$$\min_{\theta} \frac{1}{|\mathcal{U}| \times |\mathcal{I}|} \sum_{(u,i)} L(R_{ui}, R^e_{ui}) \tag{4}$$

Where $R^e_{ui}$ is the predicted interaction interaction of user $u$ and item $i$. In this paper, since the interaction values are binary, we adopt the binary cross-entropy loss function.

### 3.3 Network-based Transfer Learning Mechanism

In this paper, we are interested in the transferability of deep neural networks learned on a source domain to improve performance on a target domain. As aforementioned, since we assume that data sharing is not available, instance-based transfer is not applicable since it requires transferring of data instances from the source domain to the target domain. Feature-based transfer (e.g., [19]) requires prior knowledge of shared users/items, which is unknown in this case, and so cannot be applied. Thus, a network-based transfer approach [23] is used. Given a target domain $D_T$ and a learning task $T$, the goal here is to improve performance on $D_T$ by transferring knowledge of the pre-trained network learned on a source domain $D_S$.

The key assumption of network-based transfer approach is that the neural networks of the source and target domains should share some parameters. Let $\theta_S$ and $\theta_T$ respectively denotes the parameter set of the source and target networks. Then, the parameter sets can be decomposed into two sub-sets, one contains shared parameters (i.e., $\theta_0$) and another contains domain-specific parameters (i.e., $v_S$ and $v_T$) as follows.

$$\theta_S = \theta_0 \cup v_S \tag{5}$$

$$\theta_T = \theta_0 \cup v_T \tag{6}$$

The common parameters $\theta_0$ are learned on the source domain and then transferred to the target domain. During training at the target domain, the common (transferred) parameters are frozen, whereas domain-specific parameters ($v_T$) are learned.

Since user/item linkages are not allowed in our problem setting, the user and item embedding vectors are non-transferable, and so they are in domain-specific parameter set $v_T$. Transferable parameters consists of the weights and

Table 1: Transfer settings of fully-connected layers of the base network.

| **Setting** | **Layers to transfer** |
|---|---|
| Config-1 | FC-1, FC-2, FC-3, FC-4 |
| Config-2 | FC-1, FC-2, FC-3 |
| Config-3 | FC-1, FC-2 |
| Config-4 | FC-1 |
| Config-5 | FC-2, FC-3, FC-4 |
| Config-6 | FC-3, FC-4 |
| Config-7 | FC-4 |

biases of individual fully-connected (FC) layers of the interaction function. In this paper, we perform transfer in layer basis, in which all parameters of a given layer are transferred as a whole. More fine-grain transfer options are reserved for our future work. We consider different transfer configurations as will described in the next section.

## 4 Evaluation

### 4.1 Experiment Setup

**Base Neural Network Parameters** The user and item embedding sizes are both set to 32. The interaction function consists of $K = 4$ fully connected layers with the sizes of 64, 32, 16, and 8. It should be noted that the size of the first hidden layer of the interaction function network is equal to the sum of the user and item embedding sizes. We compare performance in terms of Hit Ratio (HR) with a baseline in which the base network are trained from scratch using only data in the target domain. For both the transfer options and baseline, Adam optimizer is used. The learning rate is set to 0.001. The batch size is 256. The number of epoch is 100. For each method/option, we run the experiment ten times and report the average values.

**Transfer Configurations** To investigate the transfer learning performance, we consider seven transfer configurations of the base neural network as shown in Table 1. The configurations differs based on which fully-connected layers are being transferred. It should be noted that the user/item embeddings are not transferable.

**Evaluation Protocol** To evaluate the proposed method, we follow the leave-one-out evaluation protocol [10]. Specifically, for a user, a test item is randomly chosen among the items that the user have interacted with. In addition, 99 negative items, which have not been interacted by the user, are randomly selected. The predicted scores for the test and negative items are calculated. Then, the test item is ranked against the negative ones based on the predicted scores. The

performance metric of hit ratio (HR) is computed as follows. Let $h_u$ denote the hit position (rank) of the test item of user $u$ against the negative items. HR@N is defined as:

$$\text{HR@N} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \max(0, 1 - \lfloor (h_u/(N+1)) \rfloor) \tag{7}$$

Here, $\lfloor . \rfloor$ is the floor function. The HR has the range in [0, 1] where a higher value indicates better performance. In this paper, we use $HR$@10 as the performance metric.

**Datasets** In our evaluation, eighteen real-world datasets from Amazon Review database [20] are used. The original datasets are preprocessed by removing users and items with less than 20 interactions. Statistics of all datasets are shown in Table 2. We train the base network from the scratch by randomly initializing weights and evaluate the performance on each dataset (i.e., baseline). As can be seen in Table 2, the three datasets of Book, Movie, and Kindle have the highest recommendation performance. Thus, those datasets are chosen as the source domains. The remaining fifteen datasets are taken as target domains.

Table 2: Statistics and baseline (non-transfer) performance of eighteen datasets used in our experiment. The three datasets of Book, Movie, Kindle are taken as source domains.

| ID | Dataset | #users | #items | #ratings | sparsity (%) | baseline ($HR$@10) |
|---|---|---|---|---|---|---|
| 1 | Book | 46276 | 148785 | 2453521 | 99.96 | **0.66** |
| 2 | Movie | 8396 | 25839 | 449685 | 99.79 | **0.64** |
| 3 | Kindle | 13742 | 21883 | 566622 | 99.81 | **0.61** |
| 4 | Sport | 2826 | 14016 | 109229 | 99.72 | 0.25 |
| 5 | Clothing | 11975 | 69009 | 743040 | 99.91 | 0.18 |
| 6 | CD | 5019 | 12847 | 193170 | 99.70 | 0.49 |
| 7 | Pet | 2153 | 7591 | 95753 | 99.41 | 0.26 |
| 8 | DigitalMusic | 230 | 2116 | 7146 | 98.53 | 0.20 |
| 9 | Home | 2002 | 9962 | 73672 | 99.63 | 0.13 |
| 10 | Toy | 2181 | 9577 | 66321 | 99.68 | 0.34 |
| 11 | Videogame | 514 | 1996 | 16496 | 98.39 | 0.31 |
| 12 | Art | 531 | 3492 | 17804 | 99.04 | 0.27 |
| 13 | Automotive | 2196 | 13435 | 87418 | 99.70 | 0.15 |
| 14 | Cellphone | 146 | 1726 | 3813 | 98.49 | 0.21 |
| 15 | Food | 1451 | 6134 | 59899 | 99.33 | 0.21 |
| 16 | Instrument | 173 | 1271 | 7185 | 96.73 | 0.17 |
| 17 | Office | 543 | 2768 | 19309 | 98.72 | 0.29 |
| 18 | Garden | 213 | 1877 | 5957 | 98.51 | 0.20 |

Table 3: Performance gain (%) of different transfer configurations compared to the baseline (non-transfer) of individual target domains when Book is the source domain. A positive (negative) value means positive (negative) transfer. The last column shows the best configuration and the corresponding $HR$@10.

| Target domain | config-1 | config-2 | config-3 | config-4 | config-5 | config-6 | config-7 | best config ($HR$@10) |
|---|---|---|---|---|---|---|---|---|
| Sport | -11.40 | -32.20 | -30.68 | -13.15 | 2.64 | **3.17** | 1.44 | config-6 (0.255) |
| Clothing | **19.90** | -14.81 | -19.21 | -9.99 | 4.59 | 4.26 | 0.61 | config-1 (0.213) |
| CD | -1.78 | -26.51 | -28.66 | -11.96 | -2.15 | **0.14** | -1.03 | config-6 (0.489) |
| Pet | -4.41 | -26.28 | -24.14 | -9.26 | **0.02** | -0.65 | -0.30 | config-5 (0.255) |
| DigitalMusic | -10.74 | -11.92 | -11.49 | -6.17 | **-0.43** | -6.06 | -9.53 | config-5 (0.200) |
| Home | 1.80 | -11.67 | -12.68 | -4.10 | **10.89** | 8.59 | 3.81 | config-5 (0.143) |
| Toy | -8.31 | -25.06 | -23.68 | -9.18 | -0.88 | -1.47 | **0.38** | config-7 (0.342) |
| Videogame | -0.43 | -28.77 | -27.38 | -11.52 | 0.97 | 1.07 | **1.58** | config-7 (0.312) |
| Art | 2.80 | -20.51 | -26.08 | -4.78 | 6.47 | **9.05** | 8.48 | config-6 (0.291) |
| Automotive | -4.59 | -17.01 | -19.35 | -5.25 | 1.85 | **2.74** | 0.24 | config-6 (0.158) |
| Cellphone | -19.85 | -10.03 | -21.30 | -17.75 | -12.58 | **-7.75** | -9.36 | config-6 (0.196) |
| Food | 0.35 | -26.52 | -22.33 | -9.76 | **4.51** | 0.30 | -1.15 | config-5 (0.218) |
| Instrument | -13.61 | -7.38 | -10.14 | **0.66** | -3.24 | -4.63 | -7.38 | config-4 (0.168) |
| Office | **-0.52** | -13.22 | -16.68 | -5.52 | -2.98 | -0.80 | -2.47 | config-1 (0.268) |
| Garden | -8.29 | -14.23 | -16.00 | -6.65 | -4.51 | -7.52 | **-0.80** | config-7 (0.201) |

Table 4: Performance gain (%) of different transfer configurations compared to the baseline (non-transfer) of individual target domains when Movie is the source domain.

| Target domain | config-1 | config-2 | config-3 | config-4 | config-5 | config-6 | config-7 | best config ($HR$@10) |
|---|---|---|---|---|---|---|---|---|
| Sport | 0.58 | -17.34 | -14.62 | -12.76 | **3.10** | 2.35 | 0.71 | config-5 (0.254) |
| Clothing | **15.42** | -15.49 | -9.38 | -12.76 | 7.90 | 3.29 | -1.35 | config-1 (0.205) |
| CD | -0.32 | -16.68 | -12.33 | -10.37 | -1.25 | **1.65** | -0.15 | config-6 (0.497) |
| Pet | **1.90** | -12.69 | -10.25 | -10.65 | -0.02 | -0.72 | -1.80 | config-1 (0.260) |
| DigitalMusic | **-0.86** | -3.46 | -3.81 | -4.73 | -2.92 | -8.66 | -8.66 | config-1 (0.199) |
| Home | **17.58** | -3.11 | -2.68 | -1.79 | 14.24 | 4.90 | -0.04 | config-1 (0.151) |
| Toy | **0.89** | -14.11 | -8.66 | -9.88 | -1.41 | -2.18 | -0.74 | config-1 (0.344) |
| Videogame | **8.18** | -9.63 | -9.19 | -10.58 | 1.07 | 3.61 | 2.28 | config-1 (0.332) |
| Art | 5.95 | -9.08 | -8.66 | -8.30 | 4.39 | 5.24 | **6.22** | config-7 (0.284) |
| Automotive | **8.24** | -14.03 | -8.84 | -7.96 | 4.82 | 2.98 | 0.57 | config-1 (0.166) |
| Cellphone | **6.72** | -3.55 | 4.52 | -0.06 | -9.38 | -5.48 | -5.49 | config-1 (0.227) |
| Food | **10.70** | -10.62 | -12.57 | -4.30 | 7.91 | 3.10 | -0.96 | config-1 (0.231) |
| Instrument | **7.41** | -1.24 | 1.15 | -1.45 | 2.53 | -7.38 | -8.77 | config-1 (0.180) |
| Office | **5.15** | -4.77 | -6.75 | -1.83 | -0.17 | -1.25 | -1.76 | config-1 (0.303) |
| Garden | 4.74 | -1.81 | -4.28 | -7.05 | -3.43 | **4.75** | 0.81 | config-6 (0.213) |

Table 5: Performance gain (%) of different transfer configurations compared to the baseline (non-transfer) of individual target domains when Kindle is the source domain.

| Target domain | config-1 | config-2 | config-3 | config-4 | config-5 | config-6 | config-7 | best config ($HR$@10) |
|---|---|---|---|---|---|---|---|---|
| Sport | -6.04 | -16.91 | -22.22 | -10.57 | **3.89** | 2.25 | 1.35 | config-5 (0.256) |
| Clothing | **6.22** | -18.22 | -16.46 | -10.86 | 2.22 | 2.09 | 0.50 | config-1 (0.189) |
| CD | -5.47 | -15.68 | -19.36 | -9.99 | **1.56** | 0.96 | 0.15 | config-5 (0.496) |
| Pet | -3.88 | -14.75 | -15.53 | -10.37 | **2.73** | 0.05 | -0.96 | config-5 (0.262) |
| DigitalMusic | -11.27 | -7.58 | -7.80 | -7.58 | **-1.07** | -8.88 | -5.62 | config-5 (0.198) |
| Home | 2.30 | -3.97 | -4.59 | -2.72 | **15.29** | 3.70 | 1.05 | config-5 (0.148) |
| Toy | -9.69 | -13.63 | -16.36 | -8.00 | -1.40 | **-0.55** | -0.73 | config-6 (0.339) |
| Videogame | 0.56 | -7.08 | -17.78 | -9.38 | **6.15** | 5.20 | 1.07 | config-5 (0.326) |
| Art | 5.54 | -6.89 | -15.36 | -2.22 | **11.49** | 7.14 | 6.08 | config-5 (0.298) |
| Automotive | -0.64 | -8.28 | -10.85 | -4.91 | **7.49** | 0.42 | 0.79 | config-5 (0.165) |
| Cellphone | -6.11 | -2.01 | -8.87 | -12.09 | -6.45 | **1.29** | -4.84 | config-6 (0.215) |
| Food | 2.43 | -16.48 | -13.35 | -6.42 | **8.67** | 3.53 | 0.40 | config-5 (0.227) |
| Instrument | 1.07 | -4.11 | -6.26 | -3.66 | **1.53** | -2.20 | -3.58 | config-5 (0.170) |
| Office | **3.76** | -4.48 | -9.84 | -5.21 | 1.11 | -1.57 | -0.43 | config-1 (0.299) |
| Garden | -10.70 | -12.43 | -7.52 | -8.38 | -6.94 | -0.35 | **0.35** | config-7 (0.204) |

### 4.2 Evaluation Results

In the first part of our experiment, we aim to answer the first and second questions regarding the transferability of the base network, namely *Q1: Does transfer learning lead to better recommendation on the target domain?* and *Q2: How to transfer a neural network for the best transfer performance?*. Table 3, Table 4, Table 5 show the gains of seven transfer configurations compared to the baseline (non-transfer method) of individual target domains when the source domain is Book, Movie, and Kindle, respectively. A positive (negative) value indicates positive (negative) transfer. The last column of each table shows the configuration with the highest gain and the corresponding $HR$@10.

It can be seen that, for all three source domains, transferring the neural network can improve the performance of most target domains. Among the fifthteen target domains, fourteen domains are benefited from transferring from at lest one source domain. In particular, the number of target domains with positive transfer are 11, 14, and 13 when the source domain is Book, Movie, and Kindle, respectively. Transferring can improve the Hit Ratio on the target domain by up to 20% from the Book domain, up to 17% from the Movie domain, and up to 15% from the Kindle domain. There are 10 target domains in which positive transfer occurs with all three source domains, namely Automotive, Home, Food, Art, Clothing, CD, Pet, Sport, Video, and Instrument. For the domains when the negative transfer occurs, the Hit Ratio is reduced by 1-8%(Book), 1-4%(Movie), and 1-5%(Kindle) compared to the baseline method. For the DigitalMusic domain, transfer learning always causes performance degradation compared to the baseline for both three source domains.

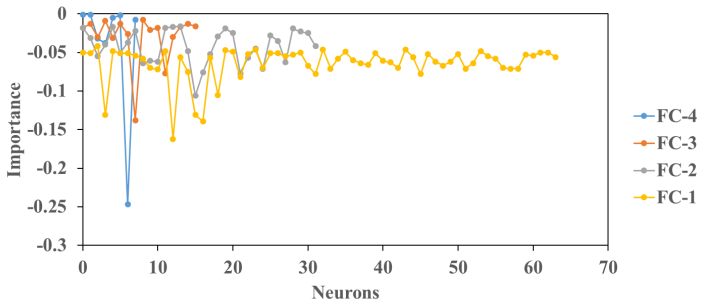


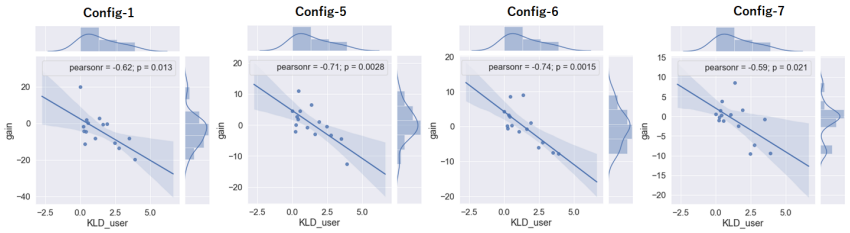

Fig. 2: Importance of individual layers of the source domain model (Book).

It can also be noted that the best transfer configuration varies across target domains and source domains. When Book is the source domain, the best transfer configuration under the Clothing and Instrument domains are Config-1 and Config-4, respectively. For the four domains of Sport, CD, Art, and Automotive, Config-6 yields the highest gains. Especially, Config-2 and Config-3 are in no case the best. Config-4 leads to negative transfer with all target domains except for Instrument. Generally, the performance of those three configurations are 10-30% lower than that of the baseline.

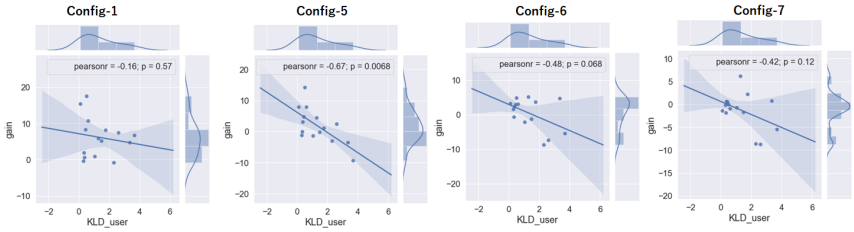
When transferring from the Movie domain (i.e., Table 4), Config-1 is the best configuration for ten target domains. The Config-5, Config-6, and Config-7 configurations are the best configuration for only one target domain domain. Again, it can be seen that the Config-2, Config-3 and Config-4 configurations results in negative transfer for all target domains. As can be seen in Table 5, Config-5 are the best configuration for most target domains when Kindle is the source domain. For the two domains of Clothing and Office, Config-1 achieve the highest gains. Again, it can be seen that the Config-2, Config-3, and Config-4 configurations cause negative transfer in all target domains.

To understand the importance of individual layers, we follow the method proposed in [17] to calculate the importance of individual fully connected (FC) layers. Specifically, to evaluate the importance of a neuron, the log-likelihoods of the correct label with and without the presence of the neuron are compared, and the importance is calculated. Fig. 2 shows the importance values of individual neurons of different FC layers. It can be seen that the layers close to the outputs are generally more importance than those close to the inputs. This result may be a hint to explain why the three configurations of Config-2, Config-3, and Config-4, where the FC-4 is not transferred, are worsen than the other configurations. This issue will be studied further in our future work.

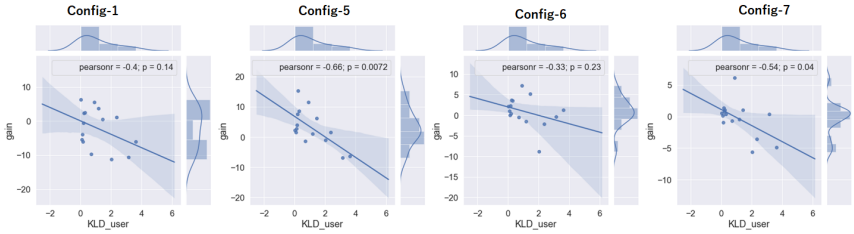
From the above results, we can have the following remarks regarding the transferability of neural networks for recommendation systems.




(a) Source: Book




(b) Source: Movie




(c) Source: Kindle

Fig. 3: Relationship between KLD values and the gains of the Config-1, Config-5, Config-6, and Config-7 configurations for different source domain a) Book, b) Movie, and c) Kindle.

- Transferring the pre-trained network from three source domains of Book, Movie, and Kindle can improve the recommendation performance on most of the target domains.
- For a given source domain, different target domains require different transfer configurations. Especially, Config-1 is preferable when Movie is the source domain, whereas Config-5 achieves highest gains for the highest number of target domains when Kindle is the source domain.
- Config-2, Config-3, and Config-4 always lead to negative transfer. This indicates that transferring of the source model contain the layers close to the output such as in case of Config-1, Config-5, Config-6, and Config-7.

In the second part of our experiment, we investigate how different factors affect the transfer performance, i.e, Question Q3. It is well-known that transfer

learning is more effective if the source and target domain are related [32]. Thus, we first examine the impact of the *relatedness* between a source domain and a target domain on the transfer performance. In this paper, we use the similarity between data distributions of the source and target domains to measure the relatedness. For that purpose, we first calculate the histogram of the number of purchases per user $H_u$ of individual domains. Then, we use the KL-Divergence (KLD) to measure the relatedness $R(D^S, D^T)$ between a source domain $D^S$ and a target domain $D^T$ as follows.

$$R(D^S, D^T) = KLD(H_u(D^S), H_u(D^T)) \tag{8}$$

Figure 3 shows the relationship between KLD values and the gains of the Config-1, Config-5, Config-6, and Config-7 configurations for three source domains. The line in each figure show the linear regression fit of the data with an 95% confident interval. The Pearson correlation coefficients (PCC) and p-values are also shown. Because the Config-2, Config-3, Config-4 result in negative transfer for most of the cases, they are excluded in this part. As can be seen in Fig. 3a, when the Book is the source domain, the transfer gain correlates to the KLD values, in which higher KLD value tends to lead to lower transfer learning performance. Especially, this trend is clearly shown in cases of Config-5 and Config-6 where $|PCC| > 0.7$. In case of Movie as the source domain (i.e., Fig. 3b), only the gain of Config-5 shows correlations with the KLD values, whereas the correlations between the three configurations of Config-1, Config-6, and Config-7 are not statistically significant, i.e., $P\text{-}value > 0.05$. As for the Kindle domain(i.e., Fig. 3c), the correlation between transfer gain and KLD can be observed for Config-5 and Config-7, but not for Config-1 and Config-6.

Table 6: Pearson correlation coefficients(P-values) between the transfer performance of transfer configurations and a) the target domain sizes and b) target domain sparsity.

(a) Target domain dataset size

| Config | Book | Movie | Kindle |
|---|---|---|---|
| Config-1 | 0.74 (0.002) | 0.39 (0.151) | 0.35 (0.198) |
| Config-5 | 0.28 (0.318) | 0.35 (0.205) | 0.03 (0.917) |
| Config-6 | 0.32 (0.249) | 0.26 (0.357) | 0.13 (0.640) |
| Config-7 | 0.17 (0.533) | 0.07 (0.805) | 0.14 (0.606) |

(b) Target domain sparsity

| Config | Book | Movie | Kindle |
|---|---|---|---|
| Config-1 | 0.48 (0.069) | 0.07 (0.801) | 0.03 (0.913) |
| Config-5 | 0.47 (0.074) | 0.35 (0.198) | 0.32 (0.252) |
| Config-6 | 0.55 (0.032) | 0.56 (0.030) | 0.31 (0.260) |
| Config-7 | 0.54 (0.037) | 0.53 (0.040) | 0.45 (0.089) |

Next, we investigate how characteristics of the target domain affect the transfer performance. Specifically, we consider two key characteristics of the target domain, namely dataset size and sparsity. Table 6 show the correlation coefficients (P-value) between transfer performance and a) target domain dataset size and b) target domain sparsity. It can be seen in Table 6a that the correlation between the transfer performance with the target domain's dataset size

is low ($|PCC| < 0.4$) and statistically insignificant ($P\text{-}value > 0.05$), except for Config-1 configuration with Book as the source domain. As shown in Table 6b, the target domain's sparsity has higher correlation to the transfer performance than the dataset size for Config-5, Config-6, and Config-7. However, the PCC values are generally low ($|PCC| < 0.6$). Especially, there is almost no correlation between the transfer performance of Config-1 and the sparsity when Movie and Kindle are source domains.

## 5 Conclusions

In this paper, we investigate the transferability of deep neural networks for top-N item recommendation task in recommender systems. Specifically, we adopt MLP as the base network, and investigate seven transfer configurations using eighteen real-world datasets. Experimental results shows that transferring layers of the interaction network enhance performance on most of the target domains by up to 20% in terms of Hit Ratio. Especially, in contrast to neural networks for computer vision and NLP tasks, the layers close to the output are more transferable than those close to the input. We also found that the best transfer configuration highly depends on the source and target domains. Hence, different from other tasks such as image classification, the transfer configuration should be carefully chosen to achieve good performance in embedding-based recommendation. Further investigation reveals that the relatedness between the source and target domain measured in terms of KL-Divergence affects the transfer performance, whereas the sizes and sparsity of target domains have little impacts on the transfer performance. In future work, we will focus on developing transfer techniques to dynamically decide the optimal transfer configuration.

## Acknowledgment

This research was partially supported by JST CREST Grant Number J181401085, Japan.

## References

1. Bansal, T., Belanger, D., McCallum, A.: Ask the gru: Multi-task learning for deep text recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 107–114. Boston, Massachusetts, USA (Sep 2016)
2. Bell, R., Koren, Y., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(08), 30–37 (Aug 2009)
3. Bennett, J., Lanning, S., et al.: The netflix prize. In: Proceedings of KDD cup and workshop. vol. 2007, p. 35. New York, NY, USA. (2007)
4. Chae, D.K., Kang, J.S., Kim, S.W., Lee, J.T.: Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 137–146. Torino, Italy (Oct 2018)

5. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning. pp. 193–200. Corvalis, Oregon, USA (Jun 2007)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
7. Gao, S., Luo, H., Chen, D., Li, S., Gallinari, P., Guo, J.: Cross-domain recommendation via cluster-level latent factor model. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 161–176. Prague, Czech (Sep 2013)
8. Gomez-Uribe, C.A., Hunt, N.: The netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. **6**(4), 13:1–13:19 (Dec 2015)
9. Han, D., Liu, Q., Fan, W.: A new image classification method using cnn transfer learning and web data augmentation. Expert Systems with Applications **95**, 43 – 56 (2018)
10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. pp. 173–182. Perth, Australia (2017)
11. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp (2019)
12. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification (2018)
13. Hu, G., Zhang, Y., Yang, Q.: Conet: Collaborative cross networks for cross-domain recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 667–676. Torino, Italy (Oct 2018)
14. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2661–2671. CA, US
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
16. Li, B., Yang, Q., Xue, X.: Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In: Twenty-First International Joint Conference on Artificial Intelligence. pp. 2052–2057. CA, USA (Jul 2009)
17. Li, J., Monroe, W., Jurafsky, D.: Understanding neural networks through representation erasure (2016)
18. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 97–105. ICML'15, JMLR.org (2015)
19. Man, T., Shen, H., Jin, X., Cheng, X.: Cross-domain recommendation: An embedding and mapping approach. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 2464–2470. Melbourne, Australia (Aug 2017)
20. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 188–197. Hong Kong, China (Nov 2019)

21. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2014)
22. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (Oct 2010)
23. Pan, W., Xiang, E.W., Liu, N.N., Yang, Q.: Transfer learning in collaborative filtering for sparsity reduction. In: Twenty-fourth AAAI conference on artificial intelligence. pp. 230–235. Georgia, USA (Jul 2010)
24. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)
25. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web. pp. 111–112. Florence, Italy (May 2015)
26. Seo, S., Huang, J., Yang, H., Liu, Y.: Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. pp. 297–305. Como, Italy (Aug 2017)
27. Shi, Q., Du, B., Zhang, L.: Domain adaptation for remote sensing image classification: A low-rank reconstruction and instance weighting label propagation inspired algorithm. IEEE Transactions on Geoscience and Remote Sensing **53**(10), 5677–5689 (2015)
28. Smith, B., Linden, G.: Two decades of recommender systems at amazon.com. IEEE Internet Computing **21**(3), 12–18 (May 2017)
29. Voigt, P., Von dem Bussche, A.: The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing (2017)
30. Wang, Z., Du, B., Guo, Y.: Domain adaptation with neural embedding matching. IEEE Transactions on Neural Networks and Learning Systems pp. 1–11 (2019). https://doi.org/10.1109/TNNLS.2019.2935608
31. Wiese, G., Weissenborn, D., Neves, M.: Neural domain adaptation for biomedical question answering. In: Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). pp. 281–289. Vancouver, Canada (Aug 2017)
32. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in neural information processing systems. pp. 3320–3328 (2014)
33. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Comput. Surv. **52**(1), 5:1–5:38 (Feb 2019)
34. Zhu, F., Chen, C., Wang, Y., Liu, G., Zheng, X.: Dtcdr: A framework for dual-target cross-domain recommendation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (Dec 2019)
35. Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., Wu, J.: A deep framework for cross-domain and cross-system recommendations. In: IJCAI International Joint Conference on Artificial Intelligence. pp. 3711–3717. Stockholm, Sweden (Jul 2018)
36. Zhu, Z., Wang, J., Caverlee, J.: Improving top-k recommendation via jointcollaborative autoencoders. In: The World Wide Web Conference. WWW '19 (May 2019)