PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information

Jianxin Chang*
Chenbin Zhang*
changjianxin@kuaishou.com
zhangchenbin@kuaishou.com
Kuaishou, China

Yiqun Hui
huiyiqun@kuaishou.com
Kuaishou, China

Dewei Leng
lengdewei@kuaishou.com
Kuaishou, China

Yanan Niu
niuyanan@kuaishou.com
Kuaishou, China

Yang Song
yangsong@kuaishou.com
Kuaishou, China

Kun Gai
gai.kun@qq.com
Unaffiliated, China

ABSTRACT

With the increase of content pages and display styles in online services such as online-shopping and video-watching websites, industrial-scale recommender systems face challenges in multi-domain and multi-task recommendations. The core of multi-task and multi-domain recommendation is to accurately capture user interests in different domains given different user behaviors. In this paper, we propose a plug-and-play ***P**arameter and **E**mbedding **P**ersonalized **Net**work (**PEPNet**)* for multi-task recommendation in the multi-domain setting. PEPNet takes features with strong biases as input and dynamically scales the bottom-layer embeddings and the top-layer DNN hidden units in the model through a gate mechanism. By mapping personalized priors to scaling weights ranging from 0 to 2, PEPNet introduces both parameter personalization and embedding personalization. Embedding Personalized Network (EPNet) selects and aligns embeddings with different semantics under multiple domains. Parameter Personalized Network (PPNet) influences DNN parameters to balance interdependent targets in multiple tasks. We have made a series of special engineering optimizations combining the Kuaishou training framework and the online deployment environment. We have deployed the model in Kuaishou apps, serving over 300 million daily users. Both online and offline experiments have demonstrated substantial improvements in multiple metrics. In particular, we have seen a more than 1% online increase in three major scenarios.

ACM Reference Format:
Jianxin Chang, Chenbin Zhang, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. 2018. PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research. The order of the authors is determined by coin flip.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Traditional recommendation models focus on the single predictions task (e.g. CTR) in a single domain[6, 21, 37], which is training using examples collected from a **single domain** and serving the prediction of a **single task**. However, in real-world applications, the need for recommendation are fragmented across different scenarios. As the number of pages increases, recommender systems face the critical problem that data fragments are located in different domains. For example, the Taobao application has scenarios such as pre-purchase(Guess What You Like), in-purchase(Choose Again and Again), and post-purchase(Guess What You Like after Purchase), as shown in Figure 1. In addition, multiple buttons are usually designed on each page for users to interact with. To obtain user feedback and provide a better experience, recommender systems need to model the probability of user interaction with these buttons in multiple tasks, that is, the user preference for different targets. For example, the various targets of the Kuaishou app in Figure 1, such as like, follow, forward, and comment, are used to obtain the user's preference degree for short-videos.

Since there are overlapping users and items in different scenarios, the multiple domains have commonalities. And different targets have dependencies between their function and label definition, so there are connections between multiple tasks. Training separate models for each target in each domain is not only unacceptable in terms of cost and iterative efficiency, but also not utilizing the full amount of data and ignoring the commonalities between the data can lead to suboptimal performance. However, mixing all the data directly and training with a unified model ignores the differences between domains and tasks. The inability to align embeddings with different semantics will result in **domain seesaw** [31] due to the different distributions of user behaviors and item candidates in multiple scenarios. Since different targets have distinctive sparsity and influence each other, the inability to balance the targets of multiple tasks can lead to the **task seesaw** [33].

At present, **multi-domain learning** and **multi-task learning** have made great progress in recommender systems. But in real applications, we cannot simply and directly reuse multi-domain or multi-task learning methods in multi-domain and multi-task joint settings, respectively. Multi-task methods focus on fitting target

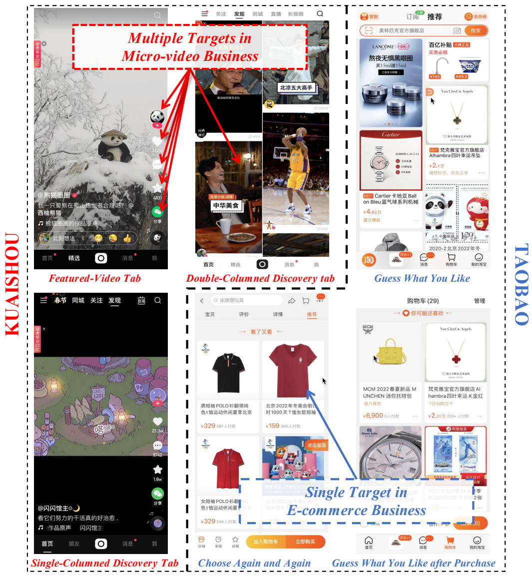


Figure 1: Comparison of short-video scenarios in Kuaishou and e-commerce scenarios in Taobao. Recommendations are made for different domains on different pages. In addition, multiple tasks are carried out for each domain, e.g., *like, follow, foward, and comment* for short videos.

distributions of different tasks, but ignore the semantic differences in the feature space under multi-domain settings. Multi-domain methods focus on aligning the embedding distributions under different domains, but ignore dependencies in the label space under multi-task settings. As shown in Figure 2, compared with separate multi-task learning or multi-domain learning, multi-task learning and multi-domain learning occur simultaneously in real industry and are more complex. On the one hand, the amounts of training data and the distribution of features varies greatly in different domains. On the other hand, different targets in the same domain and the same target in different domains have gap in sparsity. Different from task seesaw phenomenon and domain seesaw phenomenon, we call it **the imperfectly double seesaw phenomenon**. The phenomenon is more severe in industry-scale recommender systems as the number of domains and targets increases. Due to the requirement for high efficiency and low cost in real industries, a plug-and-play network is urgently needed to solve the challenges of multi-domain and multi-task. Personalization modeling is the core of recommender systems. Augmenting personalization of the model helps capture the degree of user preference for items in different situations. Multi-domain and multi-task settings can be viewed as users interacting with items in different situations, so more accurate personalization estimates can alleviate the imperfectly double seesaw problem. But simply using personalized priors as the bottom input, the effect becomes extremely weak after being passed to the top layers. How to infuse personalized priors into the model in the right place and in the right way is critical and worth exploring, especially for multiple domains and tasks.

To address this issue, we propose a **Parameter and Embedding Personalized Network (PEPNet)** for the multi-task and

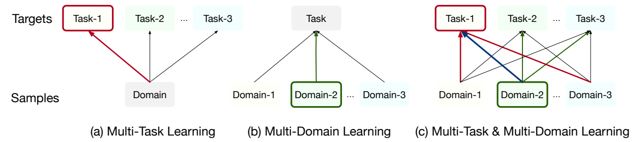


Figure 2: Comparison of multi-task, multi-domain and multi-task multi-domain learning. Best viewed in color.

multi-domain recommendation, which fully exploits the relationship between tasks and eliminates domain bias via augmenting personalization. Compared with existing works in multi-task learning [33, 46] and multi-domain learning [16, 20, 38], PEPNet is an efficient plug-and-play network. PEPNet takes features with personalized priors as input and dynamically scales the bottom-layer embeddings or the top-layer DNN hidden units in the model through the gate mechanism, which are called domain-specific EPNet and task-specific PPNet. Embedding Personalized Network (EPNet) firstly adds domain information to the bottom embedding part to generate domain-specific gates, and then perform element-wise product with the original features to get the personalized embedding of the domain. Parameter Personalized Network (PPNet) concatenates user and item information, etc. with domain-specific personalized embeddings as the input in the top DNN part, but removes the back-propagation of the original embeddings. Personalized scaling weights are then generated for each DNN hidden unit of each task tower through the gating mechanism and applied to the next layer. By mapping personlized priors to scaling weights ranging from 0 to 2, EPNet selects and aligns embeddings with different semantics under multiple domains, and PPNet influences DNN parameters to balance interdependent targets in multiple tasks.

The contributions of this work can be summarized as follows:

- We propose a Parameter and Embedding Personalized Network (PEPNet) to predict multiple tasks in multiple domains. PEPNet is an efficient, low-cost deployment and plug-and-play method that can be injected in any network. We evaluate PEPNet and other SOTA methods on the industrial short-video dataset, and extensive experiments demonstrate the effectiveness of our method in mitigating the imperfect double seesaw phenomenon.
- We deploy PEPNet in the recommendation system of Kuaishou which had an average of 1 billion monthly average users (MAU) in 2022. Up to now, the deployment of PEPNet brings a more than 1% increase in watch time and around 2% improvement on multiple interactive targets which validates its superiority. Our method can be generalized to other setups, and researchers can benefit from the lessons learned in our deployment.

2 METHODOLOGY

This section presents the detailed design for alleviating the imperfectly double seesaw problem. We elaborate on problem formulation, network structure of the proposed PEPNet and deployment in Kuaishou, one of the largest short-video platforms in China.

2.1 Problem Formulation

Here we define the notations and problem settings of our study. The model uses sparse/dense inputs such as user historical behavior, user profile features, item features, context features and so on. The predicted target $\hat{y}_i$ is the user u preference on an item p in the i-th

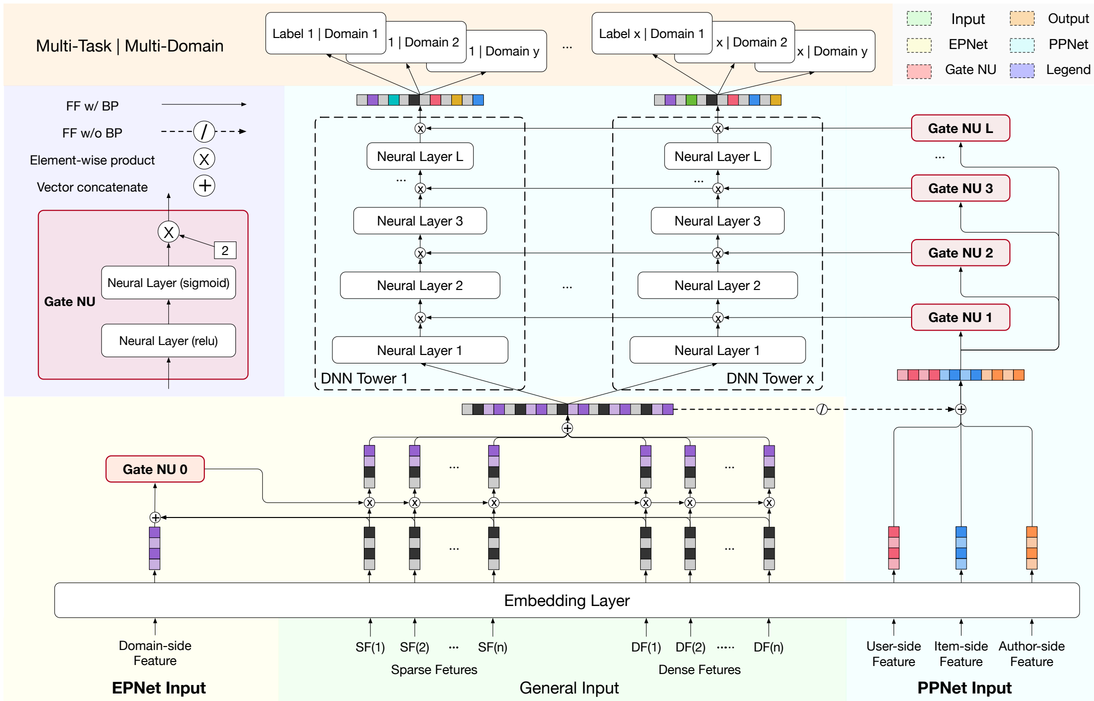


Figure 3: PEPNet consists of Gate NU, EPNet and PPNet. Gate NU is the basic unit that utilizes prior information to generate gated and amplified valid signals. EPNet increases the model's domain awareness at the Embedding layer, and Stacking PPNet on each task DNN tower enhances task personalization. The same set of multi-targets is estimated in multiple domains. PEPNet can be plugged and played in any network. Best viewed in color.

task and domain d, which is calculated via:

$$\hat{y}_i = \mathrm{f}(\{E(u_1), \ldots, E(u_t) \oplus E(p_1), \ldots, E(p_j) \oplus E(c_1), \ldots, E(c_k)\}_d) \tag{1}$$

where $u_1, ..., u_t$ indicate that the user-side features include the user's historical behavior, user profile, and user ID, etc. $p_1, ..., p_j$ indicate the target item features including item ID (iid), author ID (aid), etc. The $c_i, ..., c_k$ indicate the other features which include the context feature and combine feature. The $\{\}_d$ means the examples from the domain d. The $E(*)$ means the sparse/dense features are mapped to the learnable embedding by the embedding layer after the bucketing algorithm, and $\oplus$ means concatenation.

For a real-world application, the item candidate pool and part of the users are shared in multiple scenarios. Due to the different consumption purposes, users will change their behavior tendencies on the same item in different scenarios. To better capture user tendency for multiple behaviors and enhance the connection within multiple scenarios, the recommender needs to make multi-task predictions for multiple domains D , simultaneously. Notes the model input is $\{x, y_i, D\}$, where x is the feature as mentioned above. y_i is the label of each task and $d \in D$ is the domain indicator that indicates which domain this example is collected.

- **Input**: The sparse/dense inputs such as user historical behaviors, user profile features, item features, and other context features.
- **Output**: A recommendation model that estimates users' multiple targets in multiple domains, e.g. like, follow, forward, etc.

2.2 Network Structure

Figure 3 illustrates the network structure of our proposed PEPNet model. The model is made up of the following three parts, which we will elaborate on one by one.

- **Gate Neural Unit.** Gate NU is the basic unit of EPNet and PPNet, which is a gated structure generated based on prior information.
- **Embedding Personalized Network.** EPNet takes domain information as input and uses Gate NU for domain-specific personalization, enhancing the ability of the bottom layer of the model to express features across domains.
- **Parameter Personalized Network.** PPNet uses user information and item information to generate gates and adjust the parameters of each layer in different task towers, balancing the interdependent targets of the top layer of the model.

2.2.1 ***Gate Neural Unit(Gate NU).*** Inspired by the LHUC algorithm [32], where the key idea is to learn a specific hidden unit contribution for each speaker, PEPNet introduces a gating mechanism called Gate Neural Unit that personalizes network parameters for different users. The Gate Neural Unit, short for Gate NU, consists of two neural network layers. Denote the inputs of Gate NU $x^{(0)}$, the weight $\mathbf{W}^{(0)}$ and the bias $\boldsymbol{b}^{(0)}$ for the first network layer. Relu is choosed as the activation function for the first layer for the function. The first layer is formulated as follows,

$$\boldsymbol{x}_1 = Relu\left(\boldsymbol{x}^{(0)}\mathbf{W}^{(0)} + \boldsymbol{b}^{(0)}\right), \tag{2}$$

Then, Gate NU uses the Sigmoid function to generate gate, which limits the output to $[0, 1]$. γ is a hyperparameter which set as 2.

$\mathbf{W}^{(1)}$ and $\boldsymbol{b}^{(1)}$ are the weight and bias of the second layer. The second layer is formulated as follows,

$$\boldsymbol{x}_2 = \gamma * Sigmoid\left(\boldsymbol{x}^{(1)}\mathbf{W}^{(1)} + \boldsymbol{b}^{(1)}\right), \boldsymbol{x}_2 \in [0, \gamma] \tag{3}$$

From Equation 2 and 3, Gate NU uses the prior information $x^{(0)}$ to generate the gating vector and uses the hyperparameter γ to further amplify the effective signal. Next, we elaborate on how to use this gating mechanism in combination with EPNet and PPNet.

2.2.2 *Embedding Personalized Network(EPNet).*

The EPNet model shares the same embedding layer for the sake of computational and memory costs, where

$$E(*) = E(SF) \oplus E(DF). \tag{4}$$

SF are the sparse features and DF are the dense features. $E(*)$ is commonly known as the share-bottom structure, which has several drawbacks in practice, focusing on the commonalities but ignoring the difference between multiple domains.

For shared EPNet, we use the domain-side features $E(df) \in \mathbb{R}^k$ as input, such as domain ID and statistical feature. For the specific data example of the i domain, we denote the rest feature as $E(*) \in \mathbb{R}^d$, where d is the input dimension. $\mho_{ep}$ is the Gate NU for the embedding layer, the EPNet output $\delta_{domain} \in \mathbb{R}^d$ is given by

$$\delta_{domain} = \mho_{ep}(E(df)). \tag{5}$$

We use an external Gate NU network to transform the embedding and align distributions across multiple domains without altering the original embedding layers. The transformed embedding is

$$\mathbf{O}_{ep} = \delta_{domain} \otimes E(*), \tag{6}$$

where $\mathbf{O}_{ep} \in \mathbb{R}^d$, and $\otimes$ denotes the element-wise multiplication.

2.2.3 *Parameter Personalized Network(PPNet).*

To augment information regarding the task-specific personalization, we use user/item/author-side feature($uf/if/af$) as PPNet's inputs, such as user ID, item ID, author ID, and the side information features, e.g., user age/gender, item tag/topic/popularity, etc. Specifically, the detailed PPNet structure is as follows:

$$\begin{aligned} \mathbf{O}_{prior} &= E(uf) \oplus E(if) \oplus E(af), \\ \delta_{task} &= \mho_{pp}(\mathbf{O}_{prior} \oplus (\oslash(\mathbf{O}_{ep}))). \end{aligned} \tag{7}$$

where $E(uf) \in \mathbb{R}^u$, $E(if) \in \mathbb{R}^i$, $E(af) \in \mathbb{R}^a$.

PPNet concatenates the EPNet's output and the features $\mathbf{O}_{prior}$ with strong personalized priors, which gives the model more perception of the prior information. The prior information about personalization can be obtained to a certain extent from user ID, item ID and author ID, where the author refers to the producer of short-videos in Kuaishou. In order not to affect the embedding that has been updated in EPNet, we perform the operation of stopping the gradient $\oslash$ on the output of EPNet. The $\mho_{pp}$ denotes the Gate NU for parameter personalization in the DNN layers. In the traditional model of previous work, all hidden units are treated equally and fed to the next layer. We use element-wise multiplication to select and amplify valid signals as follows:

$$\mathbf{O}_{pp} = \delta_{task} \otimes \mathbf{H}, \tag{8}$$

where $\mathbf{H}$ is the hidden unit in each DNN layer of task towers. Parameter sharing in multi-task learning greatly reduces the size of DNN parameters, but some information is lost between multiple shared targets, resulting in unbalanced performance. For example, the tasks of predicting Follow and Like share the DNN parameters, but Follow task has fewer positive samples. After the gradients of the two are accumulated, some signals of Follow will be covered by Like. So for each task, we insert the PPNet $\mathbf{O_{pp}}^{l}$ as above in the l-th layer in each DNN task tower to strengthen the prior information of task personalization as follows:

$$\begin{aligned} \mathbf{O}_{pp}^{(l)} &= \delta_{task}^{(l)} \otimes \mathbf{H}^{(l)}, \\ \mathbf{O}_{pp}^{(l+1)} &= f(\mathbf{O}_{pp}^{(l)}\mathbf{W}^{(l)} + \boldsymbol{b}^{(l)}), l \in \{1, ..., n\}, \end{aligned} \tag{9}$$

where n is the number of DNN layers of each task tower and f is the activation function. For the first $n-1$ layers, the activation function f uses Relu. f in the last layer is Sigmoid without amplification coefficients γ, which is different from Gate NU. After obtaining prediction scores for multiple targets on multiple domains in the last layer, the binary cross-entropy is employed for optimization.

2.3 Engineering Optimization Strategy

To deploy PEPNet in Kuaishou's large-scale recommendation scenarios, we make the following engineering optimization strategies:

- **Feature score elimination strategy**: Because mapping one embedding vector per ID will quickly fill up the server's memory resources. To ensure that the system can perform for a long time, we design a special parameter server to achieve a conflict-free and memory-efficient Global Shared Embedding Table (GSET). GSET uses the feature score elimination strategy to control the memory footprint to always be below a preset threshold. However, traditional cache elimination strategies such as LFU and LRU only consider the frequency information of entities, and are mainly used to maximize the cache hit ratio.
- **DNN/Embedding layer Updating**: Since the system adopts online learning, it will accumulate data for a while for training. We refer to the smallest unit of training data as a pass, and each pass updates the online inference model. Due to a large number of users, authors and items, this will lead to the rapid expansion of the features of user ID, item ID and author ID. Some ID features of the platform may expire or become cold, so storing all the ID features is not efficient. It will blindly increase the redundancy of the system, bringing additional storage and computing overhead. We add two strategies for feature eviction. One is to set a specific number of a feature, and the excess will be expelled. The other is to set the expiration time of the ID features, keep the ID features that have been updated frequently, and delete those which does not reach the required number of updates. Similarly, we will check the corresponding embedding when the model is updated and only update the changed embedding.
- **Training strategy**: Due to the business characteristics of short-video scenarios in Kuaishou, the ID features change rapidly. In practice, we find the updating of embedding is more frequent than DNN model parameters. To better capture changes in the bottom-layer embeddings and stably update the top-layer DNN parameters in the case of online learning, we update the embedding part and the DNN parameter part separately and adopts different update strategy. In the bottom-layer embedding, we use the AdaGrad optimizer and the learning rate is set to 0.05. While

Table 1: Statistics of the dataset used in experiments. See context for details.

Dataset	Users	Items	Instances	Sparsities of each target(%)
Domain A	76k	9,474k	48,037k	$< 5/< 1/< 0.3/< 0.3/\approx 20/\approx 50$
Domain B	110k	5,205k	68,348k	$< 3/< 0.3/< 0.3/< 0.1/\approx 60/\approx 50$
Domain C	88k	5,588k	78,197k	$< 2/< 0.5/< 0.5/< 0.1/\approx 60/\approx 50$

the DNN parameters are updated by the Adam optimizer with the learning rate 5.0e-06.

3 OFFLINE EXPERIMENTS

In this section, we conduct offline experiments on real-world industrial datasets to evaluate our proposed method, with the purpose of answering the following three questions.

- **RQ1:** How does the proposed method perform compared with state-of-the-art recommenders? What about the performance in multi-task and multi-domain scenarios?
- **RQ2:** Can PPNet and EPNet in the proposed method address the double seesaw problems in multi-task and multi-domain recommendation, respectively?
- **RQ3:** What is the effect of different components and implementations in the proposed method?

3.1 Experimental Settings

3.1.1 Datasets and Metrics. Existing public datasets are not suitable for experiments in the context of the imperfectly double seesaw phenomenon, that is, these datasets do not satisfy the setting of multiple tasks in multiple domains. To evaluate PEPNet in real-world situation for multi-domain and multi-task recommendation, we need a large scale dataset with rich domains and tasks. Unfortunately, existing public datasets do not satisfy the setting of multiple tasks in multiple domains suffering the imperfectly double seesaw problem in the real business. We thus collect an industrial dataset from Kuaishou, one of the largest short-video platforms in China.

We extract a subset of the logs from Sept. 11st to Sept. 22nd, 2022, a total of 12 days. We consider three domains in our work that is the **Double-Columned Discovery Tab**, the **Featured-Video Tab**, and the **Single-Columned Slide Tab**, annotated as Domain A, B and C in our experiments. Six types of user interactions are predicted as binary targets, namely **Like**, **Follow**, **Foward**, **Hate**, **Click** and **EffView**. EffView, short for *effective view*, is defined as 1 if the watch time reaching 50% or more of the distribution formed by all samples, and 0 otherwise.

We use data of the first 10 days as the training set, the 11th day for validation, and the last day for the test. We further filter out users that have less than 10 interactions and items interacted by less than 10 users. Basic statistics of the datasets are summarized in Table 1. We leave the details of the datasets analyzed in Appendix A.1. And we evaluate the models with two widely-adopted accuracy metrics including AUC and GAUC [45].

3.1.2 Baselines and Implementations. To demonstrate the effectiveness of our PEPNet model, we compare it with several state-of-the-art methods. The baselines fall into three categories: general recommenders that only deal with a single task on a single domain, multi-task recommenders that ignore the impact of multiple domains on different tasks, and multi-task and multi-domain recommenders that consider comprehensive. For general recommenders, we include **DeepFM** [11], **DCN** [36], **xDeepFM** [21], **DCNv2** [37]. With respect to multi-task recommenders, we compare with **DCNv2-MT**(a varient of DCNv2), **SharedBottom**, **MMoE** [24], **PLE** [33]. We also propose some variants **PLE-MD**, **SharedTop**, **SpecificTop**, **SpecificAll** for comparison to fill the gap of multi-task and multi-domain recommendation.

General Recommenders: We train each label in each domain separately to report the multi-task and multi-domain results of the general recommender.

- **DeepFM** [11] is a widely used general recommender, which replaced the wide part of WDL [6] with Factorization Machine.
- **DCN** [36] replaces FM of DeepFM with Cross Network to model the linear cross-feature.
- **xDeepFM** [21] further introduces vector-wise idea into the Cross part of DCN to learn feature crosses efficiently.
- **DCNv2** [37] is the state-of-the-art model which uses a mixture of low-rank DCN to achieve a healthier trade-off between model performance and latency.

Multi-task Recommenders: We train multiple targets in each domain separately to report the multi-task and multi-domain results of the multi-task recommender.

- **DCNv2-MT** simply extends the state-of-the-art general recommender DCNv2 to multi-task scenarios, which shares the main model between different tasks and use different dense layers to generate score after obtaining logits.
- **SharedBottom** is the most common multi-task model that shares the parameters of the bottom layers and uses specific task tower to generate corresponding scores.
- **MMoE** [24] shares several expert submodels and a gating network across all tasks to implicitly model relationships between multiple task with different label spaces.
- **PLE** [33] is the state-of-the-art method which sets up independent experts for each task and considers the interaction between experts based on retaining the shared experts in MMoE.

Multi-task and Multi-domain Recommenders: There is little work dedicated to solving multi-task and multi-domain recommendations at the same time, and many cross-domain requirements in real applications are met through multi-task learning.

- **PLE-MD** simply extends the state-of-the-art multi-task recommender PLE [33] to multi-domain scenarios, which shares the input embedding layer across different domains, and add domain ID as domain indicator feature.
- **SharedTop**: To fill the gap of multi-task and multi-domain recommendation work, we first extend SharedBottom to multi-domain recommendation, sharing the bottom embedding layer instead of the bottom dnn layer. Further, we share the top task tower across different domains based on this model.
- **SpecificTop**: Different from SharedTop, this method adopts different task towers for the same task on different domains which is called SpecificTop, while the bottom embedding layer is still shared between domains.
- **SpecificAll**: Different from SpecificTOP, this method not only distinguishes different top task towers on different domains, but also adopts specific bottom embedding layers.

Table 2: Performance comparison of different methods in terms of all six task metrics on three domains. The best and second-best results are highlighted in boldface and underlined respectively. $*$ indicates that the performance difference against the second-best result is statistically significant at 0.05 level. The experimental results are averaged over multiple times.

Method	Domain A | Double-Columned Discovery Tab (AUC)						Domain A | Double-Columned Discovery Tab (GAUC)					
	Like	Follow	Foward	Hate	Click	EffView	Like	Follow	Foward	Hate	Click	EffView
DeepFM	0.8606	0.8025	0.7539	0.7092	**0.6998**	0.6908	0.6294	0.6401	0.6077	0.5490	0.5895	0.5815
DCN	0.8687	0.8017	0.7599	0.7178	0.6958	0.7038	0.6379	0.6533	0.6082	0.5378	**0.5961**	0.5893
xDeepFM	0.8706	0.8074	0.7828	0.7279	0.6961	0.7045	0.6459	0.6525	0.6126	0.5319	0.5973	0.5901
DCNv2	0.8725	0.8102	0.7615	0.7176	0.6973	0.7046	0.6441	0.6545	0.6161	0.5360	0.5963	0.5909
DCNv2-MT	0.8708	0.7949	0.7541	0.6489	0.6931	0.7007	0.6508	0.6468	0.6037	0.5187	0.5942	0.5907
SharedBottom	0.8685	0.7585	0.7587	0.7172	0.6922	0.7000	0.6301	0.6112	0.5782	0.4801	0.5933	0.5824
MMoE	0.8664	0.7676	0.7615	0.7306	0.6928	0.7010	0.6295	0.6155	0.5764	0.4998	0.5903	0.5806
PLE	0.8736	0.7991	0.7773	0.7674	0.6931	0.7006	0.6337	0.6420	0.5854	0.5338	0.5918	0.5812
PLE-MD	0.8708	0.8001	0.7612	0.731	0.6912	0.7041	**0.6585**	0.5985	0.5398	0.5455	0.5903	0.5881
SharedTop	0.8709	0.7973	0.7682	0.7601	0.6925	0.7035	0.6454	0.6506	0.6214	0.5502	0.5936	0.5872
SpecificTop	0.8700	0.7906	0.7624	0.7012	0.6928	0.7042	0.6435	0.6578	0.6131	0.4780	0.5939	0.5870
SpecificAll	0.8673	0.7705	0.7618	0.7122	0.6926	0.7010	0.5924	0.6269	0.6076	0.5119	0.5621	0.5819
PEPNet	**0.8797***	**0.8258***	**0.7911***	**0.7887***	0.6957	**0.7080***	0.6549	**0.6704***	**0.6397***	**0.5517**	0.5950	**0.5938***

Method	Domain B | Featured-Video Tab (AUC)						Domain B | Featured-Video Tab (GAUC)					
	Like	Follow	Foward	Hate	Click	EffView	Like	Follow	Foward	Hate	Click	EffView
DeepFM	0.8901	0.8616	0.7738	0.8017	0.7156	0.7044	0.6247	0.6388	0.6020	0.5573	0.6106	0.6018
DCN	0.8949	0.8618	0.7783	0.8083	0.7152	0.7072	0.6342	0.6493	0.5992	0.5603	0.6105	0.6065
xDeepFM	0.9027	0.8670	0.7796	0.8071	0.7191	0.7075	0.6378	0.6563	0.6006	0.5647	0.6109	0.6127
DCNv2	0.9040	0.8601	0.7767	0.8111	0.7190	0.7072	0.6408	0.6525	0.6059	0.5769	0.6149	0.6130
DCNv2-MT	0.9008	0.8523	0.7687	0.7886	0.7185	0.7074	0.6365	0.6465	0.6011	0.5716	0.6148	0.6143
SharedBottom	0.8876	0.8629	0.7746	0.8399	0.7154	0.7033	0.6267	0.6415	0.606	0.5598	0.6098	0.6092
MMoE	0.8889	0.8611	0.7760	0.8325	0.7155	0.7037	0.6294	0.6499	0.6061	0.5841	0.6127	0.6126
PLE	0.8905	0.8677	0.7625	0.8326	0.7157	0.7033	0.6304	0.6472	0.5939	0.5822	0.6106	0.6095
PLE-MD	0.8606	0.7949	0.6184	0.7724	0.5288	0.5946	0.5712	0.6111	0.5251	0.5330	0.5666	0.5596
SharedTop	0.9002	0.8647	0.7705	0.8302	0.7185	0.7070	0.6239	0.6505	0.6001	0.5835	0.6125	0.6101
SpecificTop	0.8139	0.7534	0.6834	0.6525	0.3859	0.4016	0.5633	0.6033	0.5767	0.5224	0.4995	0.4996
SpecificAll	0.8790	0.8565	0.7746	0.8300	0.7161	0.7044	0.6266	0.6411	0.6047	0.5640	0.6115	0.6119
PEPNet	**0.9042**	**0.8837***	**0.7974***	**0.8587***	**0.7203***	**0.7092**	**0.6431**	**0.6705***	**0.6257***	**0.6207***	**0.6189***	**0.6208***

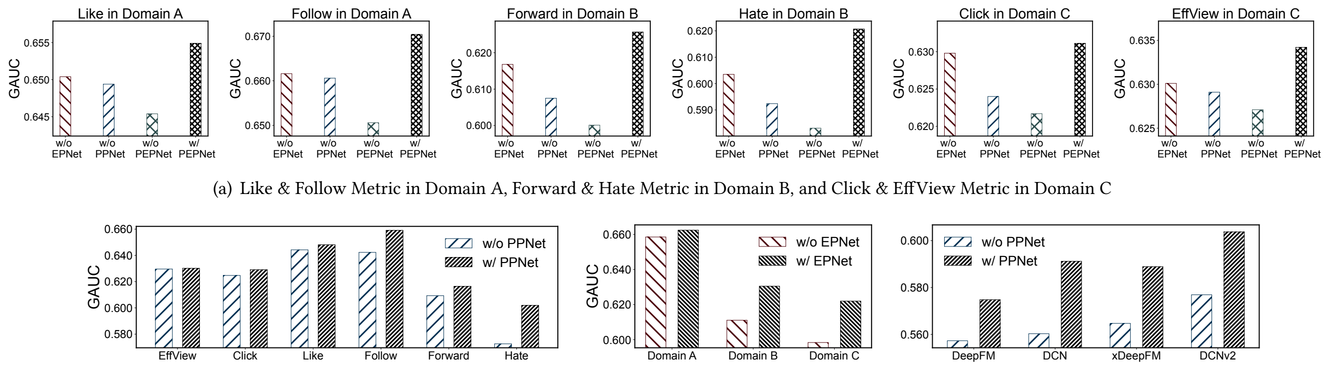
Method	Domain C | Single-Columned Slide Tab (AUC)						Domain C | Single-Columned Slide Tab (GAUC)					
	Like	Follow	Foward	Hate	Click	EffView	Like	Follow	Foward	Hate	Click	EffView
DeepFM	0.8945	0.8571	0.7783	0.8406	0.7154	0.7107	0.6350	0.6379	0.6024	0.5763	0.6350	0.6202
DCN	0.8962	0.8598	0.7801	0.8431	0.7142	0.7136	0.6402	0.6451	0.6082	0.5805	0.6209	0.6231
xDeepFM	0.9013	0.8633	0.7796	0.8514	0.7192	0.7178	0.6431	0.6465	0.6055	0.5738	0.6227	0.6272
DCNv2	0.9025	0.8603	0.7806	0.8521	0.7261	0.7181	0.6455	0.6505	0.6192	0.5827	0.6240	0.6292
DCNv2-MT	0.9022	0.8583	0.7710	0.8430	0.7273	0.7182	0.6442	0.6423	0.6093	0.5726	0.6247	0.6296
SharedBottom	0.9017	0.8574	0.7677	0.8346	0.7242	0.7152	0.6359	0.6436	0.6194	0.5834	0.6222	0.6240
MMoE	0.9014	0.8565	0.7667	0.8432	0.7245	0.7143	0.6334	0.6370	0.6131	0.5663	0.6214	0.6232
PLE	0.9018	0.8651	0.7723	0.8507	0.7246	0.7155	0.6345	0.6467	0.6142	0.6053	0.6233	0.6257
PLE-MD	0.7237	0.7621	0.5203	0.7146	0.4437	0.4491	0.5432	0.5984	0.4770	0.4740	0.5470	0.5005
SharedTop	0.9019	0.8605	0.7641	0.8458	0.7249	0.7180	0.6337	0.6424	0.6169	0.5863	0.6217	0.6271
SpecificTop	0.2056	0.6330	0.5199	0.6426	0.4833	0.454	0.4214	0.5018	0.4778	0.5156	0.4919	0.4821
SpecificAll	0.9011	0.8582	0.7683	0.8510	0.7244	0.7148	0.6333	0.6375	0.6174	0.5810	0.6208	0.6237
PEPNet	**0.9063***	**0.8843***	**0.7927***	**0.8589***	**0.7296**	**0.7203**	**0.6501***	**0.6720***	**0.6373***	**0.6212***	**0.6311***	**0.6342***

3.1.3 Hyper-parameter Settings. In offline experiments, we implement all the models based on TensorFlow[1]. We use Adam [18] for optimization with the initial learning rate as 0.001. The batch size is set as 1024 and the embedding size is fixed to 40 for all models. Xavier initialization [9] is used here to initialize the parameters. All methods use a two-layer feedforward neural network with hidden sizes of [100, 64] for interaction estimation. We apply careful grid-search to find the best hyper-parameters. The number of experts in MMoE, PLE, SharedBottom and its variants is searched in [4, 6, 8]. All regularization coefficients are searched in $[1e^{-7}, 1e^{-5}, 1e^{-3}]$.

3.2 Overall Performance (RQ1)

Table 2 illustrates the results on the datasets from three domains. From the results, we have the following observations:

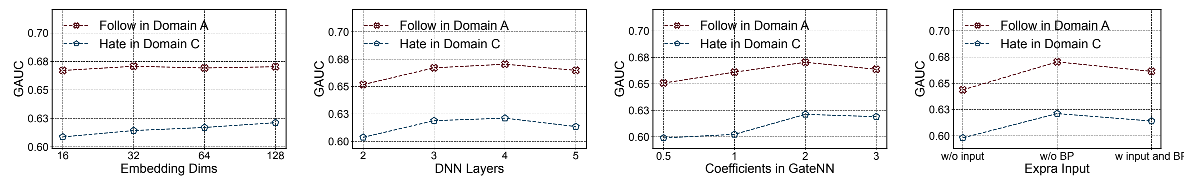
- **Our proposed method consistently achieves the best performance.** We can observe that our model PEPNet significantly outperforms all baselines in terms of all six task metrics on three domains. Specifically, our model improves GAUC on average by around 0.02 on Domain A, 0.01 on Domain B and 0.02 on Domain C with p-value < 0.05. For the average performance of each task



(a) Like & Follow Metric in Domain A, Forward & Hate Metric in Domain B, and Click & EffView Metric in Domain C

(b) All task metrics in Domain C (c) Follow Metric in all domains (d) Hate Metric in Domain C

Figure 4: Effectiveness of the sub-modules PPNet and EPNet in the proposed PEPNet model.



(a) Embedding dims in EPNet (b) DNN layers in PPNet (c) Coefficients in Gate NU (d) Extra Input and BP

Figure 5: Performance of PEPNet model with different settings and implementations.

on three domains, Like is increased by 0.01, Follow is increased by 0.03, Foward is increased by 0.02, Hate is increased by 0.03, Click is increased by 0.005, and EffView is increased by 0.01. The improvement is more obvious on the more sparse domain and task, which verifies that our method can balance multi-task and multi-domain recommendation problems more effectively. It significantly reduces the difficulty of modeling sparse domains and sparse tasks in a cross-domain and cross-task manner.

- **General recommenders cannot balance the task seesaw.** The general recommender performs well on dense tasks (Click) in dense domains (Domain B), but performs poorly on sparse tasks (Forward) in sparse domains (Domain A). Simply extending the general recommender (DCNv2) to multi-task (DCNv2-ML) results in some tasks (Like) getting better and some tasks (Hate) getting worse. This shows that the centralized general model has seesaw problems when faced with multi-task estimation, resulting in unbalanced performance across tasks. Compared with this, SharedBottom with a shared parameter layer and specific task towers obtains balanced performance improvements in all metrics on some domains (Domain C). It demonstrates that specially designed multi-task recommenders can alleviate the task seesaw phenomenon. And the more complex the design of the shared parts and specific parts of the model (MMoE and PLE), the more obvious the performance improvement. But they still suffer from poor performance on sparse domains (Domain A).
- **Multi-task recommenders cannot balance the domain seesaw.** Even the most powerful multi-task recommenders (PLE), when extended to multi-domain (PLE-MD), still appears that some domains (Domain A) get better and some domains (Domain C) get worse, i.e. the domain seesaw phenomenon. The reason is that the top-level label space and the bottom-level embedding space have inconsistencies. The limitation of multi-task models that model separate domains is that they cannot consider cross-domain and cross-task information simultaneously. The multi-task and multi-domain variants SharedTop built on SharedBottom, an early effort in multi-task learning, can alleviate the double seesaw phenomenon to a certain extent. With the task tower being specific to domains, SpecificTop only brings better results in some domains (Domain A) while increasing the number of parameters several times. While SpecificAll further divides the embedding space, which ignores the shared knowledge between domains and deteriorates the recommendation effect. Our method plugs gated networks based on shared bottom embedding layers and shared top task towers to capture the user's personalized bias across domains and tasks, achieving the best performance with a small number of parameters.

3.3 Ablation Study (RQ2)

In real applications, products of different businesses usually have isolated data but shared targets. There is no effective work for fusing data and distinguishing targets of different businesses. Firstly, we try to filter out a certain domain data and task target directly, and the online effect is negative, which is equivalent to losing a lot of user behavior information. Secondly, adding domain distinguishing features directly on top of the network has a negative effect in the real environment, indicating that the DNN parameter has a poor ability to perceive domain features. Thirdly, dealing with task features at Embedding layer is also negative, which shows embedding is not suitable for learning task information. In conclusion, all these online attempts validate the motivation of PEPNet's design.

To further verify the effectiveness of the sub-modules in the proposed PEPNet model, we compare the offline performance of the model without PPNet module, without EPNet module, without both modules, and the full model, as shown in Figure 4 (a). Furthermore, we study the generalization ability of PEPNet as a plug-and-play module on other settings than multi-task and multi-domain recommendation problems. Specifically, we compare the effect of adding EPNet module to a single-task and multi-domain model in Figure 4 (c), the effect of adding PPNet to a multi-task and single-domain model in Figure 4 (d), and the effect of adding PPNet to a single-task and single-domain model in Figure 4 (e).

The results in Figure 4 (a), (b) and (c) show the effectiveness of capturing cross-domain and cross-task information via EPNet and PPNet. Embedding personalization of EPNet and parameter personalization of PPNet can bring further performance improvement, respectively. In Figure 4 (d), adding pure parameter personalization to a single-task and single-domain model can also bring benefits for general recommendation tasks, which also illustrates the importance of modeling personalization deviation in recommendation.

3.4 Hyper-parameter Study (RQ3)

To study the influence of different settings and implementations in the proposed model, we conduct hyperparameter experiments. First, we compare the performance of EPNet with different embedding sizes for each input feature in Figure 5 (a), and the effect of the number of DNN layers coupled with PPNET in Figure 5 (b). Second, since we propose to add coefficients on the Sigmoid in Gate NU to amplify or reduce the difference between different dimensions, we evaluate the recommendation performance under different coefficients in Figure 5 (c). Finally, we study the role of general input in EPNET and PPNET, and compare the effect on model performance of removing input, adding input, and adding input but removing BackPropagation(BP) in Figure 5 (d).

From the results, we can observe that the performance of EPNet is robust under embeddings of different dimensions, and even small dimension with only 16 still keeps an excellent performance. As the number of DNN layers increases, the performance of PPNet becomes better, but after a certain number of layers, a too deep neural network will lead to overfitting. The coefficient of Sigmoid in Gate NU performs best when the value is 2, because its output range is (0,2) centered at 1, which can better balance the scaling effect. Adding general input and removing BackPropagation(BP) in PPNet is better than other settings, which shows that this manner can make better use of input information and model user personalization without affecting the backbone network.

4 ONLINE A/B TESTING

To further verify the effectiveness of the PEPNet, we conduct online feed recommendation A/B testing. By 2022, PEPNet is deployed and servers more than 10 business domains on Kuaishou and Kuaishou Express App. We split online A/B test traffic by device-IDs evenly for the tested models. We compute the overall improvements of three representative domains, that is Domain A (Double-Columned Discovery Tab), Domain B (Featured-Video Tab), Domain C (Single-Columned Slide Tab) . Unlike CTR and GMV in e-commerce scenarios, short-video scenarios pay attention to the following metrics: Like, Follow, Forward, Watch Time and App Usage. Note that Watch Time measures the average using time of each user and App Usage means average app usage time per day. From Table 3, we can see that all metrics improves significantly compared with the previous SOTA method deployed in our online service.

Table 3: Online gains in three representative domain. Note that in the Kuaishou short-video recommendation scenario, 0.1% increase in App Usage is a significant improvement that requires much effort to achieve.

	Like	Follow	Forward	Watch time	App usage
Domain A	+1.08%	+1.43%	+1.31%	+1.25%	+0.312%
Domain B	+1.36%	+1.81%	+1.55%	+1.93%	+1.132%
Domain C	+2.11%	+2.23%	+1.43%	+2.12%	+1.623%

5 RELATE WORK

Our work is based on traditional CTR prediction and extended to multi-domains and multi-tasks, simultaneously serving multiple targets in multiple businesses. In this section, we discuss related works in CTR prediction, multi-domain learning and multi-task learning areas, and gating mechanisms in the recommendation. We leave the details of related work in Appendix A.2.

CTR prediction[44, 45] mainly focuses on estimating a single target in a single domain. With the continuous improvement and development of the business, joint training from different domain data sources needs to be considered. In addition, in order to continuously improve the user experience, we also have other targets such as likes and follows that need to be optimized at the same time. Under multi-task and multi-domain settings, existing work [24, 30, 33, 42] has made efforts to solve the problem of "multi-task seesaw". However, the challenges encountered by multi-task learning are more complicated in the multi-domain scenario. The task label space of different input data sources is not consistent, so all domains sharing one multi-task model will cause the "multi-domain seesaw" problem[31]. In real industrial scenarios, the cost of training different multi-task models in different domains is unacceptable, and it is impossible to explore correlations between different domains to improve performance. In this paper, we call this **the imperfectly double seesaw problem**. Different from the above research, our approach focuses on achieving the balance of performance and efficiency. On the other hand, the previous work[13, 14, 23] also tried to use the gating mechanism to control both feature-level and instance-level information can be passed to the downstream layers. Compared with these efforts, We propose the basic innovative and plug-and-play Gate NU to achieve the effects of domain personalization and task personalization.

6 CONCLUSION AND FUTURE WORK

In this paper, we studied the imperfectly double seesaw problem, where some business domains have much less data than others and suffer from sparse labels in multi-task learning. Then, we proposed a Parameter and Embedding Personalized Network (PEPNet) which learned the heterogeneous relationships between multiple domains and multiple tasks. In Kuaishou's unique recommendation scene, domain personalization and task personalization were fully considered, which greatly improved the user's consumption experience. And for the characteristics of short-video recommendation, we made engineering strategies to optimize during training and online inferring. We have **deployed** the model at Kuaishou apps. All online and offline experiments from multiple domains achieve significant improvements in both app usage and engagement.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*.

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79, 1 (2010), 151–175.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19 (2007), 137.

[5] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[7] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research* 26 (2006), 101–126.

[8] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European conference on computer vision*. Springer, 597–613.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. Melbourne, Australia., 2782–2788.

[12] Jiang Guo, Darsh J Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. *arXiv preprint arXiv:1809.02256* (2018).

[13] Tongwen Huang, Qingyun She, Zhiqiang Wang, and Junlin Zhang. 2020. GateNet: Gating-Enhanced Deep Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2007.03519* (2020).

[14] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

[15] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[16] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. 2019. Contrastive adaptation network for unsupervised domain adaptation. In *In CVPR*.

[17] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *CIKM*.

[20] Pan Li and Alexander Tuzhilin. 2020. Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 331–339.

[21] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *arXiv preprint arXiv:1803.05170* (2018).

[22] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1871–1880.

[23] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 825–833.

[24] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[25] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[26] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-task Learning. 3994–4003.

[27] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[28] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *Proceeding of The 29th ACM International Conference on Information and Knowledge Management*. 2685–2692.

[29] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *Proceedings of the16th International Conference on Data Mining*. IEEE, 1149–1154.

[30] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142* 2 (2017).

[31] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[32] Pawel Swietojanski, Jinyu Li, and Steve Renals. 2016. Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, 8 (2016), 1450–1463.

[33] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*.

[34] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.

[35] Jialei Wang, Steven CH Hoi, Peilin Zhao, and Zhi-Yong Liu. 2013. Online multi-task collaborative filtering for on-the-fly recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*. 237–244.

[36] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[37] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*. 1785–1797.

[38] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3745–3755.

[39] Zhibo Xiao, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Hao Wang. 2020. Deep Multi-Interest Network for Click-through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

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

[41] Sicheng Zhao, Bo Li, Pengfei Xu, and Kurt Keutzer. 2020. Multi-source domain adaptation in the deep learning era: A systematic survey. *arXiv preprint arXiv:2002.12169* (2020).

[42] Sicheng Zhao, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. 2019. Multi-source domain adaptation for semantic segmentation. *arXiv preprint arXiv:1910.12181* (2019).

[43] Sicheng Zhao, Chuang Lin, Pengfei Xu, Sendong Zhao, Yuchen Guo, Ravi Krishna, Guiguang Ding, and Kurt Keutzer. 2019. Cycleemotiongan: Emotional semantic consistency preserved cyclegan for adapting image emotions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2620–2627.

[44] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. Honolulu, Hawaii, USA, 5941–5948.

[45] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.

[46] Yongchun Zhu, Fuzhen Zhuang, and Deqing Wang. 2019. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5989–5996.

Table 4: The overlap rate after scaling by a certain ratio.

Dataset	Domain A & B	Domain B & C	Domain A & C
User Overlap	73.94%	13.94%	10.49%
Item Overlap	16.94%	28.64%	61.61%

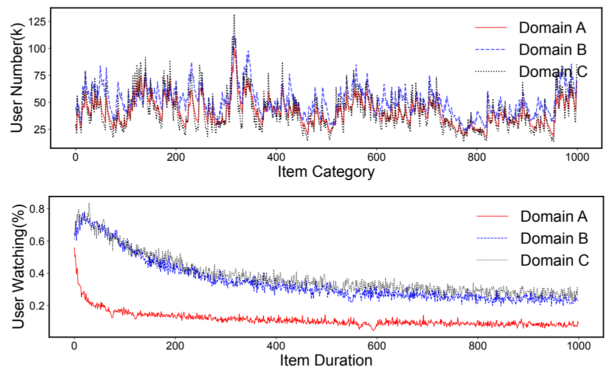


Figure 6: User preference for different video types and viewing progress for different video durations on three domains.

A APPENDIX

A.1 Dataset Analysis

We provide more detailed dataset visualization and analysis to help understand the imperfectly double seesaw problem alleviated by PEPNet. Different from previous work, the same item candidate pool, as well as author pool and a huge part of users, are shared in Kuaishou's multiple scenarios. There are also the most overlap items in Kuaishou's scenarios, as shown in Table 4. To better model our business scenarios, we perform analysis on the datasets as follows. To illustrate the inconsistency of distributions across domains with the same targets, we make visualizations as follows. Figure 6 shows how many users interact with each category of video under different domains, and the viewing progress of different items' duration after bucketing. Figure 7 shows the relation between target behaviors and age segments on different domains. In summary, although domains share the same video pool and include lots of overlapped users, it can be seen that video exposure and user behavior are different in different domains. This indicates that users have different consumption intentions in different domains and experience a differentiated recommendation ecosystem.

A.2 Related Work

A.2.1 ***Click-Through Rate Prediction.*** Click-Through Rate(CTR) Prediction is the most important growth engine for E-commerce and streaming Internet companies, which can improve user experience and increase company revenue at the same time. The traditional shallow CTR models, e.g. Logistic Regression (LR), Factorization Machine (FM) and Gradient Boosting Decision Tree (GBDT), with their strong interpretability and lightweight training deployment requirements, were widely used in the early days.

Due to the powerful ability of deep learning to capture high-order feature cross, and user potential interests. To capture the cross-over features from different fields and get rid of complex

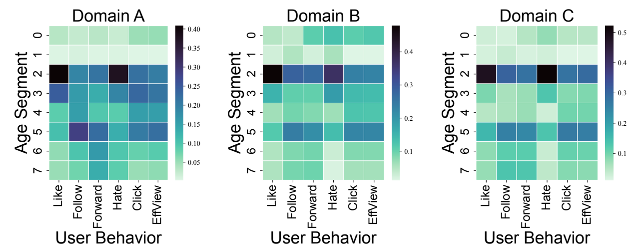


Figure 7: Visualization of the relation between users' age segment and their behavior preference on three domains.

manual feature engineering, FNN[40] uses FM to pre-train the embedding layer and then inputs the processed dense features into DNN. PNN [29] transfers the vector inner/outer product from pre-training directly to the neural network. WDL [6] jointly trains a wide linear model and a deep neural network to combine the memory and generalization advantages. DeepFM [11] replaces the wide part of WDL with FM, thus no longer relying on manual feature engineering. DCN[36, 37] replaces FM of DeepFM with Cross Network, and xDeepFM[21] further introduces the idea of vector-wise into the Cross part of DCN.

Different from the above work, some methods focus on learning the expression of potential interest from the user's historical behavior. DIN [45] uses the target-attention mechanism to activate users' different interests for different target items. DIEN [44] combines the attention mechanism with GRU to model the dynamic evolution of user interest. SASRec [17] and BST [5] use self-attention which is a parallel structure to replace GRU to efficiently model user behavior sequences. MIND [19] and DMIN [39] believe that a single vector may not be enough to capture the user's complex multi-interest patterns. MIMN [27] proposes a memory-based architecture to extract user interest from behavior histories that are thousands of in length. SIM [28] uses two cascaded search units to extract user interests, which can further model lifetime historical behavior data.

A.2.2 ***Multi-domian Learning.*** Multi-domain learning is an extension of domain adaptation and belongs to transductive transfer learning. Transfer learning can use source domains with sufficient labeled data to complete targeted tasks for target domains with little or no labeled data. When the data distributions in the source and target domains are different, but the two tasks are the same, this special kind of transfer learning is named Domain Adaptation (DA) [7]. Models trained directly on the source domain generally perform poorly on the target domain due to not satisfying the Independent and Identically Distributed (IID) assumption, a phenomenon known as Negative Transfer[3, 4]. The basic idea of domain adaptation is to align the data of different distributions of source and target domains into a unified space to obtain domain-invariant features.

Existing DA methods mainly employ a joint architecture of source and target domains[41]. It aligns the different domains through four methods: Discrepancy-based methods [16], Adversarial generative methods [10, 43], Adversarial discriminative methods [34], and Reconstruction based methods [8].

Different from general domain adaptation problems, multi-source domain adaptation involves multiple source domains with different distributions, and multi-target domain adaptation aims to transfer to multiple target domains [46]. The key to solving such problems is alignment strategies on multiple domains, which are mainly divided into two categories [41]. Latent space transformation methods optimize the discrepancy loss or adversarial loss to align the latent spaces of different domains [12]. Intermediate domain generation methods generate an intermediate adaptive domain for each source domain that is indistinguishable from the target domain [42].

Different from the previously mentioned work, multi-domain learning in recommendation scenarios increasingly weakens the concept of source and target domains, emphasizing on improving the recommendation effect of multiple domains at the same time.

A.2.3 ***Multi-task Learning***. Multi-task learning aims to learn multiple related tasks at the same time, and facilitate the learning of each specific task by mining shared information. Early linear models [2] used shared sparse representations to learn across multiple tasks. In the deep learning's period, the hard parameter sharing method may cause negative transfer due to task differences. To achieve better performance, some studies deal with optimization with soft parameter sharing method. Misra et al. [26] and Ruder et al. [30] respectively propose cross-stitch network and sluice network to learn linear combinations of task-specific hidden layers. Other methods use gating mechanisms and attention mechanisms for information fusion. The MOE proposed by Jacobs et al. [15] uses the gate structure to combine several experts shared at the bottom. Liu et al. [22] proposed that MTAN consists of a shared network and several task-specific attention modules.

In recommendation systems, early models based on collaborative filtering and matrix factorization [35] express lower expressivity and ignore the correlation between tasks. Due to irreplaceable advantages such as simplicity and efficiency, the hard parameter sharing at the bottom is also widely used in recommendation systems. [24] propose MMoE to share all experts in different tasks and use different gates for each task to extend MOE. ESSM [25] is based on a soft parameter sharing structure and simultaneously optimizes two related tasks with sequential modes to alleviate the sparsity of the prediction target. Based on retaining the shared experts in MMoE, PLE [33] sets up independent experts for each task and considers the interaction between different experts.

A.2.4 ***Gating Mechanisms in Recommendation***. Gating mechanism is widely used in recommendation systems [13] which uses the gates to automatically adjust parameters between modeling shared information and modeling task-specific information. Most recently, Huang *et al.* [13] propose to directly enhance the model ability by their method Gate-Net and propose a novel structure such that feature embedding gate layer and the hidden layer gate to dynamically increase the weights of important features and decrease the weights of uninformative features via the Gating mechanism. The method of Ma *et al.* [23] is similar to ours, in which both feature-level and instance-level gating modules adaptively control what item latent features and which relevant item can be passed to the downstream layers. Besides, it still thoroughly implicitly model the user's personalization from the user's behavior sequence. However, this method doesn't consider cross-domain issues. The method proposed by Huang *et al.* [14] requires training a SENET Layer for each feature, which is costly for the real-world recommendation model with high-dimensional and sparse features.