Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations

Hongyan Tang
Tencent PCG
China
violatang@tencent.com

Junning Liu
Tencent PCG
China
korchinliu@tencent.com

Ming Zhao
Tencent PCG
China
marcozhao@tencent.com

Xudong Gong
Tencent PCG
China
xudonggong@tencent.com

ABSTRACT

Multi-task learning (MTL) has been successfully applied to many recommendation applications. However, MTL models often suffer from performance degeneration with negative transfer due to the complex and competing task correlation in real-world recommender systems. Moreover, through extensive experiments across SOTA MTL models, we have observed an interesting seesaw phenomenon that performance of one task is often improved by hurting the performance of some other tasks. To address these issues, we propose a Progressive Layered Extraction (PLE) model with a novel sharing structure design. PLE separates shared components and task-specific components explicitly and adopts a progressive routing mechanism to extract and separate deeper semantic knowledge gradually, improving efficiency of joint representation learning and information routing across tasks in a general setup. We apply PLE to both complicatedly correlated and normally correlated tasks, ranging from two-task cases to multi-task cases on a real-world Tencent video recommendation dataset with 1 billion samples, and results show that PLE outperforms state-of-the-art MTL models significantly under different task correlations and task-group size. Furthermore, online evaluation of PLE on a large-scale content recommendation platform at Tencent manifests 2.23% increase in view-count and 1.84% increase in watch time compared to SOTA MTL models, which is a significant improvement and demonstrates the effectiveness of PLE. Finally, extensive offline experiments on public benchmark datasets demonstrate that PLE can be applied to a variety of scenarios besides recommendations to eliminate the seesaw phenomenon. PLE now has been deployed to the online video recommender system in Tencent successfully.

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412236

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Retrieval models and ranking**;

KEYWORDS

Multi-task Learning, Recommender System, Seesaw Phenomenon

ACM Reference Format:
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 (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412236

1 INTRODUCTION

Personalized recommendation has played a crucial role in online applications. Recommender systems (RS) need to incorporate various user feedbacks to model user interests and maximize user engagement and satisfaction. However, user satisfaction is normally hard to tackle directly by a learning algorithm due to the high dimensionality of the problem. Meanwhile, user satisfaction and engagement have many major factors that can be learned directly, e.g. the likelihood of clicking, finishing, sharing, favoriting, and commenting etc. Therefore, there has been an increasing trend to apply Multi-Task Learning (MTL) in RS to model the multiple aspects of user satisfaction or engagement simultaneously. And in fact, it has been the mainstream approach in major industry applications[11, 13, 14, 25].

MTL learns multiple tasks simultaneously in one single model and is proven to improve learning efficiency through information sharing between tasks [2]. However, tasks in real-world recommender systems are often loosely correlated or even conflicted, which may lead to performance deterioration known as *negative transfer* [21]. Through extensive experiments in a real-world large-scale video recommender system and public benchmark datasets, we find that existing MTL models often improve some tasks at the sacrifice of the performance of others, when task correlation is complex and sometimes sample dependent, i.e., multiple tasks could not be improved simultaneously compared to the corresponding single-task model, which is called *seesaw phenomenon* in this paper.

Prior works put more efforts to address the negative transfer but neglect the seesaw phenomenon, e.g., cross-stitch network [16] and sluice network [18] propose to learn static linear combinations to

fuse representations of different tasks, which could not capture the sample dependence. MMOE [13] applies gating networks to combine bottom experts based on the input to handle task differences but neglects the differentiation and interaction between experts, which is proved to suffer from the seesaw phenomenon in our industrial practice. Hence, it is critical to design a more powerful and efficient model to handle complicated correlations and eliminate the challenging seesaw phenomenon.

To achieve this goal, we propose a novel MTL model called Progressive Layered Extraction (PLE), which better exploits prior knowledge in the design of shared network to capture complicated task correlations. Compared with roughly shared parameters in MMOE, PLE explicitly separates shared and task-specific experts to alleviate harmful parameter interference between common and task-specific knowledge. Furthermore, PLE introduces multi-level experts and gating networks, and applies progressive separation routing to extract deeper knowledge from lower-layer experts and separate task-specific parameters in higher levels gradually.

To evaluate the performance of PLE, we conduct extensive experiments on real-world industrial recommendation dataset and major available public datasets including census-income [5], synthetic data [13] and Ali-CCP [1]. Experiment results demonstrate that PLE outperforms state-of-the-art MTL models across all datasets, showing consistent improvements on not only task groups with challenging complex correlations, but also task groups with normal correlations in different scenarios. Besides, significant improvement of online metrics on a large-scale video recommender system in Tencent demonstrates the advantage of PLE in real-world recommendation applications.

The main contributions of this paper are summarized as follows:

- Through extensive experiments in the large-scale video recommender system at Tencent and public benchmark datasets, an interesting seesaw phenomenon has been observed that SOTA MTL models often improve some tasks at the sacrifice of the performance of others and do not outperform the corresponding single-task model due to the complicatedly inherent correlations.
- A PLE model with novel shared learning structure is proposed to improve shared learning efficiency then address the seesaw phenomenon and negative transfer further, from the perspective of joint representation learning and information routing. Besides recommendation applications, PLE is flexible to be applied to a variety of scenarios.
- Extensive offline experiments are conducted to evaluate the effectiveness of PLE on industrial and public benchmark datasets. Online A/B test results in one of the world's largest content recommendation platforms at Tencent also demonstrate the significant improvement of PLE over SOTA MTL models in real-world applications, with 2.23% increase in view-count and 1.84% increase in watch time, which generates significant business revenue. PLE has been successfully deployed to the recommender system now and can be potentially applied to many other recommendation applications.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=408

2 RELATED WORK

Efficient multi-task learning models and application of MTL models in recommender systems are two research areas related to our work. In this section, we briefly discuss related works in these two areas.

2.1 Multi-Task Learning Models

Hard parameter sharing [2] shown in Fig. 1a) is the most basic and commonly used MTL structure but may suffer from negative transfer due to task conflicts as parameters are straightforwardly shared between tasks. To deal with task conflicts, cross-stitch network [16] in Fig. 1f) and sluice network [18] in Fig. 1g) both propose to learn weights of linear combinations to fuse representations from different tasks selectively. However, representations are combined with the same static weights for all samples in these models and the seesaw phenomenon is not addressed. In this work, the proposed PLE (Progressive Layered Extraction) model applies progressive routing mechanism with gate structures to fuse knowledge based on the input, which achieves adaptive combinations for different inputs.

There have been some studies applying the gate structure and attention network for information fusion. MOE [8] first proposes to share some experts at the bottom and combine experts through a gating network. MMOE [13] extends MOE to utilize different gates for each task to obtain different fusing weights in MTL. Similarly, MRAN [24] applies multi-head self-attention to learn different representation subspaces at different feature sets. The expert and attention module are shared among all tasks and there is no task-specific concept in MOE, MMOE (shown in Fig. 1) and MRAN. In contrast, our proposed CGC (Customized Gate Control) and PLE model separate task-common and task-specific parameters explicitly to avoid parameter conflicts resulted from complex task correlations. Even though there exists theoretical possibility for MMOE to converge to our network design, the prior knowledge on network design is important and MMOE can hardly discover the convergence path in practice. Liu et al. [10] apply task-specific attention networks to fuse shared features selectively but different tasks still share the same representation before fusion in attention network. None of the previous works has explicitly addressed the issues of joint optimization of representation learning and routing, especially in an inseparable joint fashion, while this work makes the first effort to propose a novel progressive separation fashion on the general framework of joint learning and routing.

There have also been some works utilizing AutoML approaches to find a good network structure. SNR framework [12] controls connections between sub-networks by binary random variables and applies NAS [26] to search for the optimal structure. Similarly, Gumbel-matrix routing framework [15] learns routing of MTL models formulated as a binary matrix with Gumbel-Softmax trick. Modeling routing process as MDP, Rosenbaum et al. [17] applies MARL [19] to train the routing network. The network structures in these works are designed with certain simplified assumptions and are not general enough. The routing network in [17] selects no more than one function block for each task in each depth, which reduces the expressivity of the model. Gumbel-matrix routing network [15] imposes the constraint on the representation learning as each task's input needs to merge to one representation at each layer. Moreover,

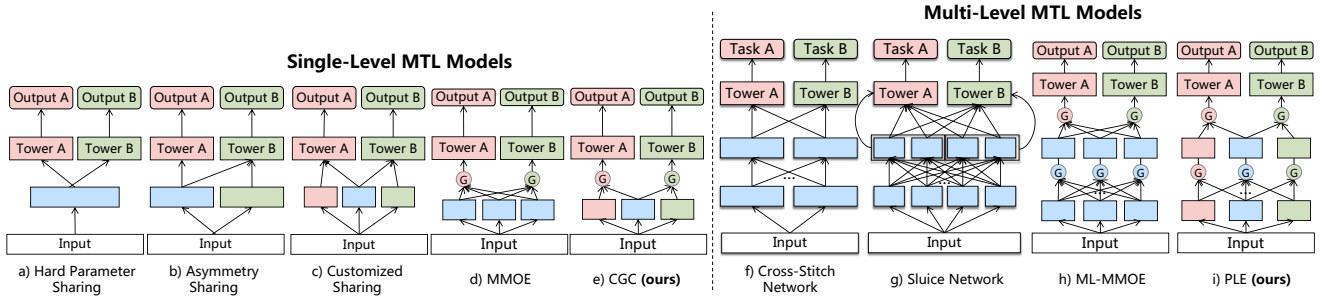

Figure 1: Network Routing of MTL Models. Blue rectangles and circles represent shared layers and gating networks respectively, pink and green rectangles represent task-specific layers, pink and green circles denote task-specific gating networks for different tasks.

the fusing weights in these frameworks are not adjustable for different inputs, and the expensive searching cost is another challenge for these approaches to find the optimal structure.

2.2 Multi-Task Learning in Recommender Systems

To better exploit various user behaviors, multi-task learning has been widely applied to recommender systems and achieved substantial improvement. Some studies integrate traditional recommendation algorithms such as collaborative filtering and matrix factorization with MTL. Lu et al. [11] and Wang et al. [23] impose regularization on latent representations learned for the recommendation task and explanation task to optimize them jointly. Wang et al. [22] combine collaborative filtering with MTL to learn user-item similarity more efficiently. Compared to PLE in this paper, these factorization based models exhibit lower expressivity and could not fully exploit commonalities between tasks.

As the most basic MTL structure, hard parameter sharing has been applied to many DNN based recommender systems. The ESSM [14] introduces two auxiliary tasks of CTR (Click-Through Rate) and CTCVR and shares embedding parameters between CTR and CVR (Conversion Rate) to improve the performance of CVR prediction. Hadash et al. [7] propose a multi-task framework to learn parameters of the ranking task and rating task simultaneously. The task of text recommendation in [1] is improved through sharing representations at the bottom. However, hard parameter sharing often suffers from negative transfer and seesaw phenomenon under loose or complex task correlations. In contrast, our proposed model introduces a novel sharing mechanism to achieve more efficient information sharing in general.

Besides hard parameter sharing, there have been some recommender systems applying MTL models with more efficient shared learning mechanism. To better exploit correlations between tasks, Chen et al. [3] utilize hierarchical multi-pointer co-attention [20] to improve the performance of the recommendation task and explanation task. However, tower networks of each task share the same representation in the model, which may still suffer from task conflicts. Applying MMOE [13] to combine shared experts through different gating networks for each task, the YouTube video recommender system in [25] can better capture task differences and optimize multiple objectives efficiently. Compared with MMOE which treats all experts equally without differentiation, PLE in this paper explicitly separates task-common and task-specific experts and adopts a novel progressive separation routing to achieve significant improvement over MMOE in real-world video recommender systems.

3 SEESAW PHENOMENON IN MTL FOR RECOMMENDATION

Negative transfer is a common phenomenon in MTL especially for loosely correlated tasks [21]. For complex task correlation and especially sample dependent correlation patterns, we also observe the seesaw phenomenon where improving shared learning efficiency and achieving significant improvement over the corresponding single-task model across all tasks is difficult for current MTL models. In this section, we introduce and investigate the seesaw phenomenon thoroughly based on a large-scale video recommender system in Tencent.

3.1 An MTL Ranking System for Video Recommendation

In this subsection, we briefly introduce the MTL ranking system serving Tencent News, which is one of the world's largest content platforms and recommends news and videos to users based on the diverse user feedbacks. As shown in Fig. 2, there are multiple objectives to model different user behaviors such as click, share, and comment in the MTL ranking system. In the offline training process, we train the MTL ranking model based on user actions extracted from user logs. After each online request, the ranking model outputs predictions for each task, then the weighted-multiplication based ranking module combines these predicted scores to a final score through a combination function shown in Equation 1, and recommends top-ranked videos to the user finally.

$$score = {p_{VTR}}^{w_{VTR}} \times {p_{VCR}}^{w_{VCR}} \times {p_{SHR}}^{w_{SHR}} \times \cdots \times {p_{CMR}}^{w_{CMR}} \times f(video\_len), \quad (1)$$

where each w determines the relative importance of each predicted score, $f(video\_len)$ is a non-linear transform function such as sigmoid or log function in video duration. $w_{VTR}, w_{VCR}, w_{SHR}, w_{CMR}$ are hyper-paramters optimized through online experimental search to maximize online metrics.

Out of all tasks, VCR (View Completion Ratio) and VTR (View-Through Rate) are two important objectives modeling key online metrics of view-count and watch time respectively. Specifically,

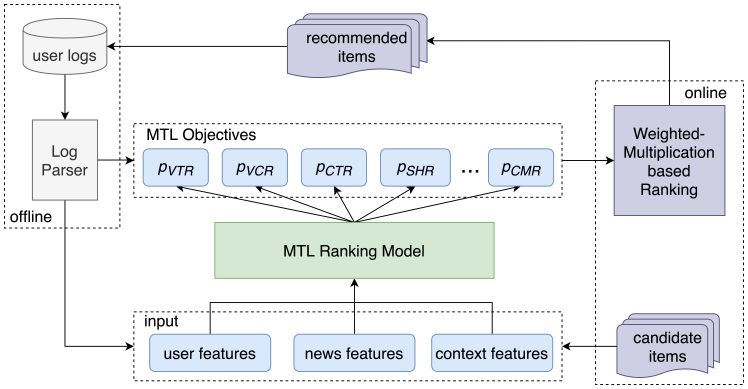

Figure 2: An MTL Ranking System for Video Recommendation

VCR prediction is a regression task trained with MSE loss to predict the completion ratio of each view. VTR prediction is a binary classification task trained with cross-entropy loss to predict the probability of a valid view, which is defined as a play action that exceeds a certain threshold of watch time. The correlation pattern between VCR and VTR is complex. First, the label of VTR is a coupled factor of play action and VCR, as only a play action with watch time exceeding the threshold will be treated as a valid view. Second, the distribution of play action is further complicated as samples from auto-play scenarios in WIFI exhibit higher average probability of play, while other samples from explicit click scenarios without auto-play exhibit lower probability of play. Due to the complex and strong sample dependent correlation pattern, a seesaw phenomenon is observed when modeling VCR and VTR jointly.

3.2 Seesaw Phenomenon in MTL

To better understand the seesaw phenomenon, we perform experimental analysis with the single-task model and SOTA MTL models on the complicatedly correlated task-group of VCR and VTR in our ranking system. Besides hard parameter sharing, cross-stitch [16], sluice network [18] and MMOE [13], we also evaluate two innovatively proposed structures called asymmetric sharing and customized sharing:

- **Asymmetric Sharing** is a novel sharing mechanism to capture asymmetric relations between tasks. According to Fig. 1b), bottom layers are shared asymmetrically between tasks, and representation of which task to be shared depends on relations between tasks. Common fusion operations such as concatenation, sum-pooling, and average-pooling can be applied to combine outputs of bottom layers of different tasks.
- **Customized Sharing** shown in Fig. 1c) separates shared and task-specific parameters explicitly to avoid inherent conflicts and negative transfer. Compared with the single-task model, customized sharing adds a shared bottom layer to extract sharing information and feeds the concatenation of the shared bottom layer and task-specific layer to the tower layer of the corresponding task.

Fig. 3 illustrates experiment results, where bubbles closer to upper-right indicate better performance with higher AUC and lower MSE. It is worth noting that 0.1% increase of AUC or MSE contributes significant improvement to online metrics in our system, which is also mentioned in [4, 6, 14]. One can see that hard parameter sharing and cross-stitch network suffer from significant negative transfer and perform worst in VTR. Through innovative sharing mechanism to capture asymmetric relations, asymmetric sharing achieves significant improvement in VTR but exhibits significant degeneration in VCR, similar to sluice network. Owing to explicit separation of shared layers and task-specific layers, customized sharing improves VCR over the single-task model while still slightly suffers in VTR. MMOE improves over the single-task model on both tasks but the improvement of VCR is only +0.0001 on the borderline. Although these models exhibit different learning efficiency with these two challenging tasks, we clearly observe the seesaw phenomenon that the improvement of one task often leads to performance degeneration of the other task, as no one baseline MTL model lies in 2*nd* quadrant completely. Experiments with SOTA models on public benchmark datasets also exhibit clear seesaw phenomenon. Details would be provided in Section 5.2.

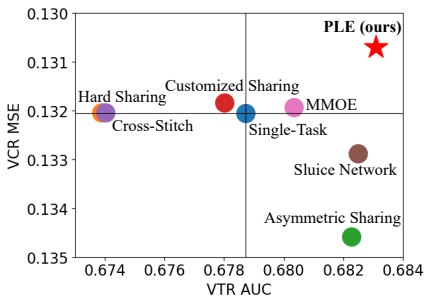

Figure 3: Seesaw Phenomenon under Complex Task Correlation

As aforementioned, the correlation pattern between VCR and VTR is complicated and sample dependent. Specifically, there are some partially ordered relations between VCR and VTR, and different samples exhibit different correlations. Thus, cross-stitch and sluice network that combine shared representations with same static weights for all samples could not capture the sample dependence and suffer from the seesaw phenomenon. Applying gates to obtain fusing weights based on the input, MMOE handles task difference and sample difference to some extent, which outperforms other baseline MTL models. Nevertheless, experts are shared among

all tasks in MMOE without differentiation, which could not capture the complicated task correlations and may bring harmful noise to some tasks. Moreover, MMOE neglects the interactions between different experts, which limits the performance of joint optimization further. In addition to VCR and VTR, there are many complicatedly correlated tasks in industrial recommendation applications as human behaviors are often subtle and complex, e.g., CTR prediction and CVR prediction in online advertising and e-commerce platform [14]. Therefore, a powerful network that considers differentiation and interactions between experts is critical to eliminate the challenging seesaw phenomenon resulted from complex task correlation.

In this paper, we propose a Progressive Layered Extraction (PLE) model to address the seesaw phenomenon and negative transfer. The key idea of PLE is as follows. First, it explicitly separates shared and task-specific experts to avoid harmful parameter interference. Second, multi-level experts and gating networks are introduced to fuse more abstract representations. Finally, it adopts a novel progressive separation routing to model interactions between experts and achieve more efficient knowledge transferring between complicatedly correlated tasks. As shown in Fig. 3, PLE achieves significant improvement over MMOE in both tasks. Details of structure design and experiments would be described in Section 4 and Section 5 respectively.

4 PROGRESSIVE LAYERED EXTRACTION

To address the seesaw phenomenon and negative transfer, we propose a Progressive Layered Extraction (PLE) model with a novel sharing structure design in this section. First, a Customized Gate Control (CGC) model that explicitly separates shared and task-specific experts is proposed. Second, CGC is extended to a generalized PLE model with multi-level gating networks and progressive separation routing for more efficient information sharing and joint learning. Finally, the loss function is optimized to better handle the practical challenges of joint training for MTL models.

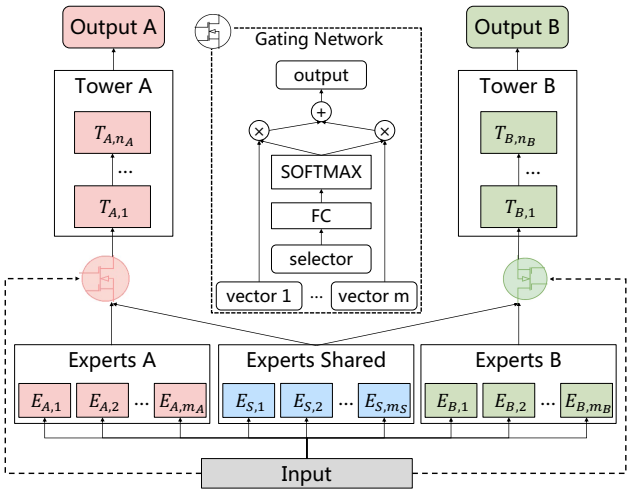

Figure 4: Customized Gate Control (CGC) Model

4.1 Customized Gate Control

Motivated by customized sharing which achieves similar performance with the single-task model through explicitly separating shared and task-specific layers, we first introduce a Customized Gate Control (CGC) model. As shown in Fig. 4, there are some expert modules at the bottom and some task-specific tower networks at the top. Each expert module is composed of multiple sub-networks called experts and the number of experts in each module is a hyper-parameter to tune. Similarly, a tower network is also a multi-layer network with width and depth as hyper-parameters. Specifically, the shared experts in CGC are responsible for learning shared patterns, while patterns for specific tasks are extracted by task-specific experts. Each tower network absorbs knowledge from both shared experts and its own task-specific experts, which means that the parameters of shared experts are affected by all tasks while parameters of task-specific experts are only affected by the corresponding specific task.

In CGC, shared experts and task-specific experts are combined through a gating network for selective fusion. As depicted in Fig. 4, the structure of the gating network is based on a single-layer feed-forward network with SoftMax as the activation function, input as the selector to calculate the weighted sum of the selected vectors, i.e., the outputs of experts. More precisely, the output of task k's gating network is formulated as:

$$g^k(x) = w^k(x)S^k(x), \tag{2}$$

where x is the input representation, and $w^k(x)$ is a weighting function to calculate the weight vector of task k through linear transformation and a SoftMax layer:

$$w^k(x) = \text{Softmax}(W_g^k x), \tag{3}$$

where $W_g^k \in R^{(m_k+m_s)\times d}$ is a parameter matrix, m_s and m_k are the number of shared experts and task k's specific experts respectively, d is the dimension of input representation. $S^k(x)$ is a selected matrix composed of all selected vectors including shared experts and task k's specific experts:

$$S^k(x) = [E^T_{(k,1)}, E^T_{(k,2)}, \ldots, E^T_{(k,m_k)}, E^T_{(s,1)}, E^T_{(s,2)}, \ldots, E^T_{(s,m_s)}]^T \tag{4}$$

Finally, the prediction of task k is:

$$y^k(x) = t^k(g^k(x)), \tag{5}$$

where t^k denotes the tower network of task k.

Compared with MMOE, CGC removes connections between a task's tower network and task-specific experts of other tasks, enabling different types of experts to concentrate on learning different knowledge efficiently without interference. Combined with the benefit of gating networks to fuse representations dynamically based on the input, CGC achieves more flexible balance between tasks and better deals with task conflicts and sample-dependent correlations.

4.2 Progressive Layered Extraction

CGC separates task-specific and shared components explicitly. However, learning needs to shape out deeper and deeper semantic representations gradually in deep MTL, while normally it is not crystally clear whether the intermediate representations should be treated

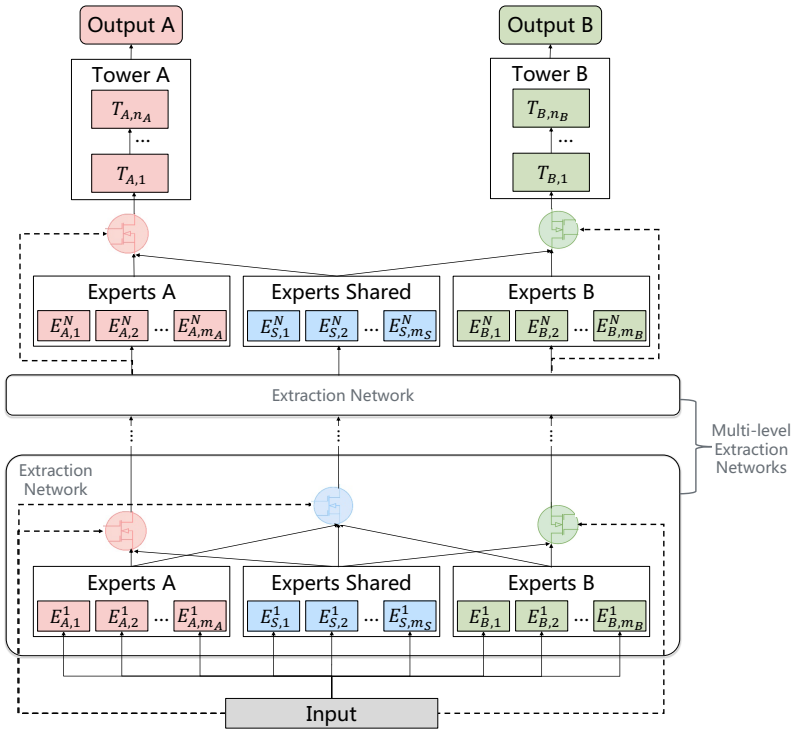

Figure 5: Progressive Layered Extraction (PLE) Model

as shared or task-specific. To address this issue, we generalize CGC with Progressive Layered Extraction (PLE). As depicted in Fig. 5, there are multi-level extraction networks in PLE to extract higher-level shared information. Besides gates for task-specific experts, the extraction network also employs a gating network for shared experts to combine knowledge from all experts in this layer. Thus parameters of different tasks in PLE are not fully separated in the early layer as CGC but are separated progressively in upper layers. The gating networks in higher-level extraction network take the fusion results of gates in lower-level extraction network as the selector instead of the raw input, as it may provide better information for selecting abstract knowledge extracted in higher-level experts.

The calculation of weighting function, selected matrix, and gating network in PLE are the same as that in CGC. Specifically, the formulation of the gating network of task k in the jth extraction network of PLE is:

$$g^{k,j}(x) = w^{k,j}(g^{k,j-1}(x))S^{k,j}(x), \tag{6}$$

where $w^{k,j}$ is the weighting function of task k with $g^{k,j-1}$ as input, and $S^{k,j}$ is the selected matrix of task k in the jth extraction network. It is worth noting that the selected matrix of the shared module in PLE is slightly different from task-specific modules, as it consists of all shared experts and task-specific experts in this layer.

After calculating all gating networks and experts, we can obtain the prediction of task k in PLE finally:

$$y^k(x) = t^k(g^{k,N}(x)) \tag{7}$$

With multi-level experts and gating networks, PLE extracts and combines deeper semantic representations for each task to improve generalization. As shown in Fig. 1, the routing strategy is full connection for MMOE and early separation for CGC. Differently, PLE adopts a progressive separation routing to absorb information from all lower-layer experts, extract higher-level shared knowledge, and separate task-specific parameters gradually. The process of progressive separation is similar to the extraction process from a compound for desired product in chemistry. During the process of knowledge extraction and transformation in PLE, lower-level representations are jointly extracted/aggregated and routed at the higher-level shared experts, obtaining the shared knowledge and distributing to specific tower layers progressively, so as to achieve more efficient and flexible joint representation learning and sharing. Although the full connection routing of MMOE seems like a general design of CGC and PLE, practical studies in Section 5.3 show that MMOE is not able to converge to the structure of CGC or PLE, in spite of the existence of possibility.

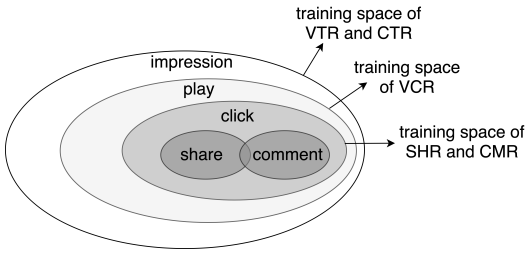

Figure 6: Training Space of Different tasks

4.3 Joint Loss Optimization for MTL

Having designed the efficient network structure, we now focus on training task-specific and shared layers jointly in an end-to-end manner. In multi-task learning, a common formulation of joint loss

is the weighted sum of the losses for each individual task:

$$L(\theta_1, \ldots, \theta_K, \theta_s) = \sum_{k=1}^{K} \omega_k L_k(\theta_k, \theta_s), \tag{8}$$

where θ_s denotes shared parameters, K is the number of tasks, L_k, ω_k and θ_k are loss function, loss weight, and task-specific parameters of task k respectively.

However, there exist several issues, making joint optimization of MTL models challenging in practice. In this paper, we optimize the joint loss function to address two critical ones encountered in real-world recommender systems. The first problem is the heterogeneous sample space due to sequential user actions. For instance, users can only share or comment on an item after clicking it, which leads to different sample space of different tasks shown in Fig. 6. To train these tasks jointly, we consider the union of sample space of all tasks as the whole training set, and ignore samples out of its own sample space when calculating the loss of each individual task:

$$L_k(\theta_k, \theta_s) = \frac{1}{\sum_i \delta_k^i} \sum_i \delta_k^i loss_k(\hat{y}_k^i(\theta_k, \theta_s), y_k^i) \tag{9}$$

where $loss_k$ is task k's loss of sample i calculated based on prediction $\hat{y}_k^i$ and ground truth y_k^i, $\delta_k^i \in \{0, 1\}$ indicates whether sample i lies in the sample space of task k.

The second problem is that the performance of an MTL model is sensitive to the choice of loss weight in the training process [9], as it determines the relative importance of each task on the joint loss. In practice, it is observed that each task may have different importance at different training phases. Therefore, we consider the loss weight for each task as a dynamic weight instead of a static one. At first, we set an initial loss weight $\omega_{k,0}$ for task k, then update its loss weight after each step based on the updating ratio γ_k:

$$\omega_k^{(t)} = \omega_{k,0} \times \gamma_k^t, \tag{10}$$

where t denotes the training epoch, $\omega_{k,0}$ and γ_k are hyper-parameters of the model.

5 EXPERIMENTS

In this section, extensive offline and online experiments are performed on both the large-scale recommender system in Tencent and public benchmark datasets to evaluate the effectiveness of proposed models. We also analyze the expert utilization in all gate-based MTL models to better understand the working mechanism of gating networks and verify the structure value of CGC and PLE further.

5.1 Evaluation on the Video Recommender System in Tencent

In this subsection, we conduct offline and online experiments for task groups with complex and normal correlations as well as multiple tasks in the video recommender system at Tencent to evaluate the performance of proposed models.

5.1.1 Dataset. We collect an industrial dataset through sampling user logs from the video recommender system serving Tencent News during 8 consecutive days. There are 46.926 million users, 2.682 million videos and 0.995 billion samples in the dataset. As mentioned before, VCR, CTR, VTR, SHR (Share Rate), and CMR (Comment Rate) are tasks modeling user preferences in the dataset.

5.1.2 Baseline Models. In the experiment, we compare CGC and PLE with single-task model, asymmetric sharing, customized sharing, and the SOTA MTL models including cross-stitch network, sluice network, and MMOE. As multi-level experts are shared in PLE, we extend MMOE to ML-MMOE (multi-layer MMOE) shown in Fig. 1h) by adding multi-level experts for fair comparison. In ML-MMOE, higher-level experts combine representations from lower-level experts through gating networks and all gating networks share the same selector.

5.1.3 Experiment Setup. In the experiment, VCR prediction is a regression task trained and evaluated with MSE loss, tasks modeling other actions are all binary classification tasks trained with cross-entropy loss and evaluated with AUC. Samples in the first 7 days are used for training and the rest samples are test set. We adopt a three-layer MLP network with RELU activation and hidden layer size of [256, 128, 64] for each task in both MTL models and the single-task model. For MTL models, we implement the expert as a single-layer network and tune the following model-specific hyper-parameters: number of shared layers, cross-stitch units in hard parameter sharing and cross-stitch network, number of experts in all gate-based models. For fair comparison, we implement all multi-level MTL models as two-level models to keep the same depth of networks.

Table 1: Performance on VTR/VCR Task Group

Models	AUC VTR	MSE VCR	MTL Gain VTR	MTL Gain VCR
Single-Task	0.6787	0.1321	-	-
Hard Parameter Sharing	0.6740	0.1320	-0.0047	+1.8E-5
Asymmetric Sharing	0.6823	0.1346	+0.0036	-0.0025
Cross-Stitch	0.6740	0.1320	-0.0047	+1.6E-5
Sluice Network	0.6825	0.1329	+0.0038	-0.0008
Customized Sharing	0.6780	0.1318	-0.0007	+0.0002
MMOE	0.6803	0.1319	+0.0016	+0.0001
ML-MMOE	0.6815	0.1329	+0.0028	-0.0009
CGC	**0.6832**	0.1320	**+0.0045**	+3.5E-5
PLE	**0.6831**	**0.1307**	**+0.0044**	**+0.0013**

Besides common evaluation metrics such as AUC and MSE, we define a metric of **MTL gain** to quantitively evaluate the benefit of multi-task learning over the single-task model for a certain task. As shown in Equation 11, for a given task group and an MTL model q, MTL gain of q on task A is defined as the task A's performance improvement of MTL model q over the single-task model with the same network structures and training samples.

$$\text{MTL gain} = \begin{cases} M_{MTL} - M_{single}, & M \text{ is a positive metric} \\ M_{single} - M_{MTL}, & M \text{ is a negative metric} \end{cases} \tag{11}$$

5.1.4 Evaluation on Tasks with Complex Correlation. To better capture the major online engagement metrics, e.g., view count and watch time, we first conduct experiments on task-group of

VCR/VTR. Table 1 illustrates the experiment results and we mark best scores in bold and performance degeneration (negative MTL gain) in gray. It is shown that CGC and PLE significantly outperform all baseline models in VTR. Due to the complex correlation between VTR and VCR, we can clearly observe the seesaw phenomenon with the zigzag gray distribution that some models improve VCR but hurt VTR while some improve VTR but hurt VCR. Specifically, MMOE improves both tasks over single-task but the improvement is not significant, while ML-MMOE improves VTR but hurts VCR. Compared to MMOE and ML-MMOE, CGC improves VTR much more and improves VCR slightly as well. Finally, PLE converges with similar pace and achieves significant improvement over the above models with the best VCR MSE and one of the best VTR AUCs.

Table 2: Performance on CTR/VCR Task Group

Models	AUC CTR	MSE VCR	MTL Gain CTR	MTL Gain VCR
Single-Task	0.7379	0.1179	-	-
Cross-Stitch	0.7220	0.1158	-0.0158	+0.0021
Sluice Network	0.7382	0.1157	+0.0004	+0.0021
MMOE	0.7382	0.1175	+0.0003	+0.0004
ML-MMOE	0.7378	0.1169	-0.0001	+0.0010
CGC	0.7398	0.1155	+0.0020	+0.0023
PLE	**0.7406**	**0.1150**	**+0.0027**	**+0.0029**

5.1.5 Evaluation on Tasks with Normal Correlation. Despite the good performance of CGC and PLE on tasks with really complicated correlations, we further verify their generality on a general task-group of CTR/VCR with normal correlation patterns. As CTR and VCR aim to model different user actions, the correlation between them is simpler. As shown in Table 2, the fact that all models except cross-stitch exhibit positive MTL gain in both tasks shows that the correlation pattern between CTR and VCR is not so complex and does not suffer from the seesaw phenomenon. In this scenario, CGC and PLE still significantly outperform all SOTA models on both tasks with outstanding MTL gain, which verifies that the benefit of CGC and PLE is general, achieving better shared learning efficiency and consistently providing incremental performance improvement across a wide range of task correlation situations, not only for tasks with complex correlations that is hard to cooperate, but also for regularly correlated tasks.

Table 3: Improvement over Single-task Model on Online A/B Test

Models	Total View Count	Total Watch Time
Hard Parameter Sharing	-1.65%	-1.79%
Sluice Network	+0.75%	+1.29%
MMOE	+1.94%	+1.73%
ML-MMOE	+1.96%	+1.10%
CGC	+3.92%	+2.75%
PLE	**+4.17%**	**+3.57%**

5.1.6 Online A/B Testing. Careful online A/B test with task-group of VTR and VCR was conducted in the video recommender system for 4 weeks. We implement all MTL models in our C++ based deep learning framework, randomly distribute users into several buckets, and deploy each model to one of the bucket. The final ranking score is obtained through the combination function of multiple predicted scores described in Section 3. Table 3 shows the improvement of MTL models over the single-task model on online metrics of total view count per user and total watch time per user, the ultimate goal of the system. It is shown that CGC and PLE achieve significant increase in online metrics over all baseline models. Moreover, PLE outperforms CGC significantly on all online metrics, which shows that small improvements of AUC or MSE in MTL yield significant improvements in online metrics. PLE has been deployed to the platform in Tencent since then.

Table 4: MTL gain of CGC and PLE on Multiple Tasks

Task Group	Models	MTL Gain VTR	MTL Gain VCR	MTL Gain SHR	MTL Gain CMR
VTR+VCR +SHR	CGC	**+0.0131**	**+0.0019**	-0.0001	-
	PLE	**+0.0132**	**+0.0036**	**+0.0013**	-
VTR+VCR +CMR	CGC	**+0.0180**	**+0.0012**	-	+0.0000
	PLE	**+0.0197**	**+0.0033**	-	+0.0001
VTR+VCR +SHR+CMR	CGC	**+0.0097**	**+0.0016**	+0.0008	**+0.0012**
	PLE	**+0.0128**	**+0.0017**	**+0.0058**	**+0.0080**

5.1.7 Evaluation with Multiple Tasks. Finally, we explore the scalability of CGC and PLE in more challenging scenarios with multiple tasks. In addition to VTR and VCR, We introduce SHR (Share Rate) and CMR (Comment Rate) to model user feedback actions. It is flexible to extend CGC and PLE to multiple-task cases, only to add a task-specific expert module, gating network, and tower network for each task. As shown in Table 4, CGC and PLE achieve significant improvement over the single-task model nearly on all tasks of all task groups. This shows that CGC and PLE still demonstrate the benefits of promoting task cooperation, preventing negative transfer and seesaw phenomenon for general situations with more than two tasks. PLE outperforms CGC significantly in all cases. Thus, PLE exhibits stronger benefit on improving shared learning efficiency across different sizes of task-groups.

5.2 Evaluation on Public Datasets

In this subsection, we conduct experiments on public benchmark datasets to evaluate the effectiveness of PLE in scenarios besides recommendation further.

5.2.1 Datasets.

- **Synthetic Data** is generated following the data synthesizing process based on [13] to control task correlations. As hyper-parameters for data synthesis are not provided in [13], we randomly sample α_i and β_i following the standard normal distribution, and set c=1, m=10, d=512 for reproducibility. 1.4 million samples with two continuous labels are generated for each correlation.

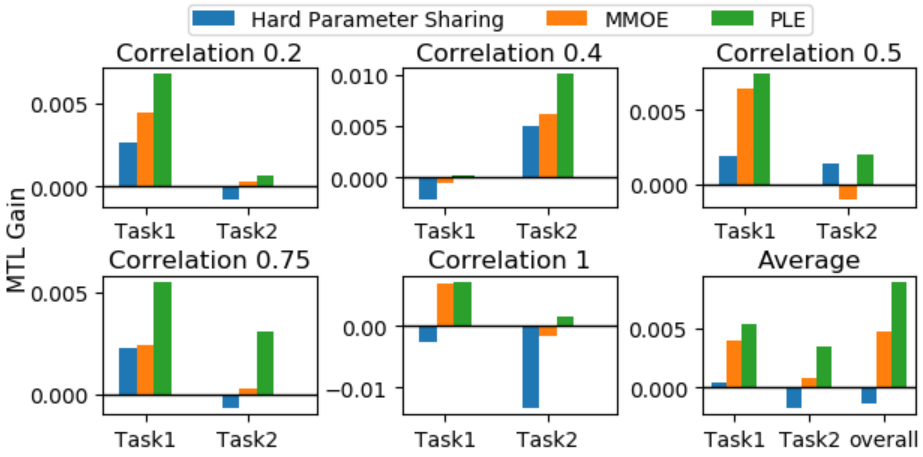

Figure 7: MTL gain on Synthetic Data

- **Census-income Dataset** [5] contains 299,285 samples and 40 features extracted from the 1994 census database. For fair comparison with baseline models, we consider the same task-group as [13]. In detail, task 1 aims to predict whether the income exceeds 50K, task 2 aims to predict whether this person's marital status is never married.
- **Ali-CCP Dataset** [1] is a public dataset containing 84 million samples extracted from Taobao's Recommender System. CTR and CVR (Conversion Rate) are two tasks modeling actions of click and purchase in the dataset.

5.2.2 Experiment Setup. The setup for census-income dataset is the same as [13]. For synthetic data and Ali-CCP dataset, we adopt a three-layer MLP network with RELU activation and hidden layer size of [256, 128, 64] for each task in both MTL models and single-task model. Hyper-parameters are tuned similarly to the experiments in Section 5.1.

5.2.3 Experiment Results. Experiment results on synthetic data shown in Fig. 7 demonstrate that hard parameter sharing and MMOE sometimes suffer from seesaw phenomenon and lost balance between two tasks. On the contrary, PLE consistently performs best for both tasks across different correlations and achieves 87.2% increase in MTL gain over MMOE on average. As results on Ali-CCP and census-income dataset shown in Table 5, PLE eliminates the seesaw phenomenon and outperforms the single-task model and MMOE consistently on both tasks.

Combined with previous experiments on the industrial dataset and online A/B test, PLE exhibits stable general benefit on improving MTL efficiency and performance for different task correlation patterns and different applications.

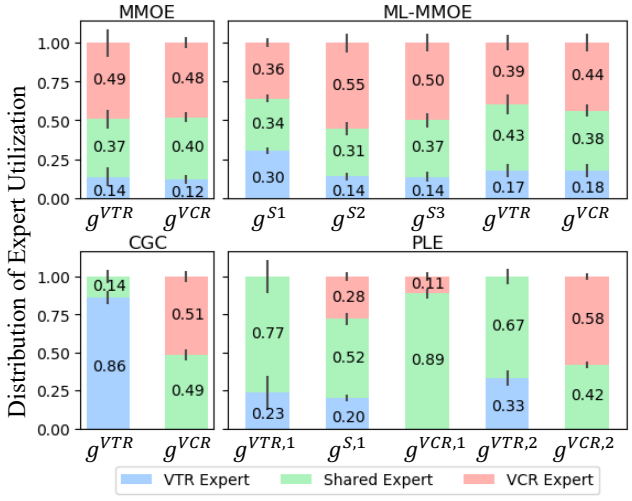

Figure 8: Expert Utilization in Gate-Based Models

5.3 Expert Utilization Analysis

To disclose how the experts are aggregated by different gates, we investigate expert utilization of all gate-based models in VTR/VCR task group of the industrial dataset. For simplicity and fair comparison, we consider each expert as a single-layer network, keep only one expert in each expert module of CGC and PLE, while keep three experts in each layer of MMOE and ML-MMOE. Fig. 8 shows the weight distribution of experts utilized by each gate in all testing data, where the height of bars and vertical short lines indicate mean and standard deviation of weights respectively. It is shown that the VTR and VCR combine experts with significantly different weights in CGC while much similar weights in MMOE, which indicates that the well-designed structure of CGC helps achieve better differentiation between different experts. Furthermore, there is no zero-weight for all experts in MMOE and ML-MMOE, which further shows that it is hard for MMOE and ML-MMOE to converge to the structure of CGC and PLE without prior knowledge in practice, despite the existence of theoretical possibility. Compared with CGC, shared experts in PLE have larger influence on the input of tower networks especially for the VTR task. The fact that PLE performs better than CGC shows the value of shared higher-level deeper representations. In other words, it is demanded that certain deeper semantic representations are shared between tasks thus a progressive separation routing provides a better joint routing and learning scheme.

Table 5: Experiment Results on Census-income and Ali-CCP Dataset

Models	**Census-income** Task1		**Census-income** Task2		**Ali-CCP** CTR		**Ali-CCP** CVR	
	AUC	MTL Gain	AUC	MTL Gain	AUC	MTL Gain	AUC	MTL Gain
Single-Task	0.9445	-	0.9923	-	0.6088	-	0.6040	-
MMOE	0.9393	+0.0048	0.9928	+0.0005	0.6094	+0.0006	0.5738	-0.0302
PLE	**0.9522**	**+0.0078**	**0.9945**	**+0.0022**	**0.6112**	**+0.0024**	**0.6097**	**+0.0057**

6 CONCLUSION

In this paper, we propose a novel MTL model called Progressive Layered Extraction (PLE), which separates task-sharing and task-specific parameters explicitly and introduces an innovative progressive routing manner to avoid the negative transfer and seesaw phenomenon, and achieve more efficient information sharing and joint representation learning. Offline and online experiment results on the industrial dataset and public benchmark datasets show significant and consistent improvements of PLE over SOTA MTL models. Exploring the hierarchical task-group correlations will be the focus of future work.

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 107–114.

[2] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[3] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-attentive multi-task learning for explainable recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2137–2143.

[4] 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*. 7–10.

[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[7] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 451–454.

[8] 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.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

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

[11] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 4–12.

[12] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 216–223.

[13] 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.

[14] 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.

[15] Krzysztof Maziarz, Efi Kokiopoulou, Andrea Gesmundo, Luciano Sbaiz, Gabor Bartok, and Jesse Berent. 2019. Gumbel-Matrix Routing for Flexible Multi-task Learning. *arXiv preprint arXiv:1910.04915* (2019).

[16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3994–4003.

[17] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239* (2017).

[18] Sebastian Ruder12, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *stat* 1050 (2017), 23.

[19] Yoav Shoham, Rob Powers, and Trond Grenager. 2003. Multi-agent reinforcement learning: a critical survey. *Web manuscript* (2003).

[20] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2309–2318.

[21] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 242–264.

[22] 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.

[23] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 165–174.

[24] Jiejie Zhao, Bowen Du, Leilei Sun, Fuzhen Zhuang, Weifeng Lv, and Hui Xiong. 2019. Multiple Relational Attention Network for Multi-task Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1123–1131.

[25] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.

[26] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).