STEM: Unleashing the Power of Embeddings for Multi-Task Recommendation

Liangcai Su [*1], **Junwei Pan** [*2], **Ximei Wang**[2], **Xi Xiao**[1†], **Shijie Quan**[2], **Xihua Chen**[2], **Jie Jiang**[2],

[1]Shenzhen International Graduate School, Tsinghua University
[2]Tencent Inc.
sulc21@mails.tsinghua.edu.cn, xiaox@sz.tsinghua.edu.cn,
{jonaspan, messiwang, justinquan, tinychen, zeus}@tencent.com

Abstract

Multi-task learning (MTL) has gained significant popularity in recommender systems as it enables simultaneous optimization of multiple objectives. A key challenge in MTL is negative transfer, but existing studies explored negative transfer on all samples, overlooking the inherent complexities within them. We split the samples according to the relative amount of positive feedback among tasks. Surprisingly, negative transfer still occurs in existing MTL methods on samples that receive comparable feedback across tasks. Existing work commonly employs a shared-embedding paradigm, limiting the ability of modeling diverse user preferences on different tasks. In this paper, we introduce a novel Shared and Task-specific EMbeddings (STEM) paradigm that aims to incorporate both shared and task-specific embeddings to effectively capture task-specific user preferences. Under this paradigm, we propose a simple model STEM-Net, which is equipped with an All Forward Task-specific Backward gating network to facilitate the learning of task-specific embeddings and direct knowledge transfer across tasks. Remarkably, STEM-Net demonstrates exceptional performance on comparable samples, achieving positive transfer. Comprehensive evaluation on three public MTL recommendation datasets demonstrates that STEM-Net outperforms state-of-the-art models by a substantial margin. Our code is released at https://github.com/LiangcaiSu/STEM.

Introduction

Recently, multi-task learning has drawn great interest in recommender systems since they are able to optimize multiple objectives (*e.g.*, `Like`, `Share`, `Finish`) simultaneously. The effectiveness of multi-task recommendation (MTR) depends on leveraging knowledge from other tasks to *help* the learning of each task. However, prior works have identified negative transfer (Torrey et al. 2010) and seesaw phenomenon (Tang et al. 2020), whereby multi-task learning models may not always outperform single-task models. To address negative transfer, MMoE (Ma et al. 2018a) and PLE (Tang et al. 2020) incorporate *task-specific* gate networks and experts, respectively. Additionally, techniques such as gradient clipping (Yu et al. 2020; Chen et al. 2018; He et al. 2022) and task-aware optimizer (Yu et al. 2020; Chen et al. 2018; Yang et al. 2023) have also been employed to alleviate gradient conflicts.

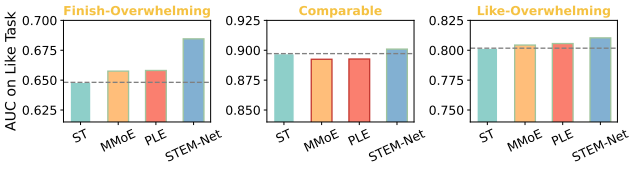


Figure 1: Existing MTL models such as MMoE and PLE suffer from negative transfer on the comparable subset of TikTok test samples, while STEM-Net achieves positive transfer. STEM-Net also outperforms MMoE and PLE on task-overwhelming subsets. ST: Single-Task.

Existing methods tend to treat all samples in a task as a whole, overlooking the inherent intricacies within them. Consequently, it remains unclear *where negative transfer occurs*. To address this concern, we split the test set of TikTok, a public MTL recommendation dataset, into three subsets according to the relative amount of feedback between task `Finish` and `Like`: `Finish`-Overwhelming, Comparable, and `Like`-Overwhelming. We then evaluate the performance of two popular MTL models, MMoE and PLE, and the single-task model on task `Like` over these subsets, since it has much fewer positive samples and therefore suffers from negative transfer. As shown in Figure 1, compared to the single task model, both MMoE and PLE demonstrate a notable performance lift on the `Finish`-Overwhelming subset by incorporating additional knowledge of task `Finish` to resolve the feedback sparsity on `Like`. They also have a slight performance boost on the `Like`-Overwhelming subset where task `Like` itself can already learn well, but stills benefits from the complementary knowledge from `Finish`. However, to our surprise, on the subset that receives comparable positive feedback from both tasks, the performance of existing methods is *inferior* to that of the single-task model.

The performance drop of existing MTL methods on the comparable subset prompts us to reconsider the design of MTL recommendation models. Intuitively, users may possess diverse and sometimes even conflicting preferences over items across various tasks, which become more pro-

[*]These authors contributed equally.
[†]Corresponding author.

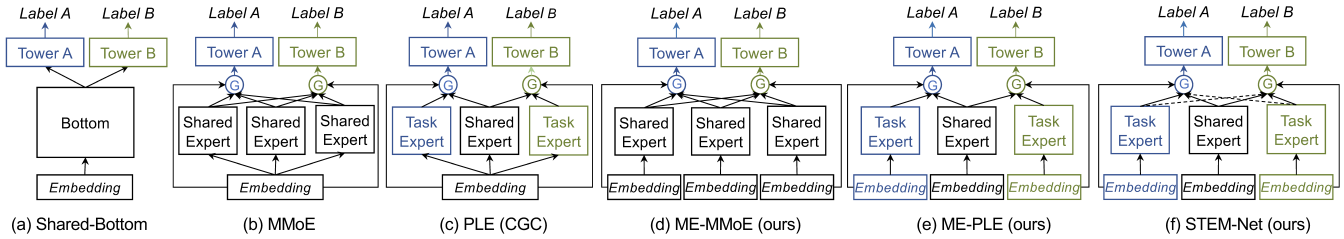


Figure 2: Comparison between representative MTL models and our proposed STEM-Net. Dot lines denote connections with stop-gradient operation.

nounced when there are sufficient signals from multiple tasks, as on the comparable subset here. The preference of users is captured through user and item embeddings in recommendation models. However, existing MTL methods, including MMoE (Ma et al. 2018a) and PLE (Tang et al. 2020), all follow a *shared-embedding paradigm*. That is, they learn a universal embedding for each user and item, shared across tasks. Such a paradigm is only able to capture a single preference of users, hindering the ability to capture the preference divergence across tasks.

Motivated by the limitations of the shared-embedding paradigm, this paper introduces a Shared and Task-specific EMbeddings (STEM) paradigm. STEM aims to *incorporate both shared and task-specific embeddings* to learn common and task-specific user preference. Under this paradigm, we design a simple model, namely STEM-Net, which begins by introducing the shared and task-specific embedding tables. STEM-Net then constructs a set of experts, where each expert is either shared across tasks, utilizing only shared embeddings, or task-specific, utilizing embeddings specific to a particular task. Furthermore, STEM-Net employs an All Forward Task-specific Backward gating network for each task tower, which a) *receive forward from shared and all task-specific experts* and b) employs a *stop-gradient operation on the experts of the other tasks* to learn task-specific embeddings by preventing a task's gradients from updating embeddings of the other tasks.

We conduct extensive evaluation of STEM-Net on three subsets of the TikTok dataset. STEM-Net consistently outperforms both MMoE and PLE on the `Finish` and `Like`-Overwhelming subsets. Moreover, STEM-Net exhibits remarkable performance in the comparable subset, surpassing the Single-Task `Like` model. Furthermore, we assess STEM-Net on three public MTL recommendation datasets. In all cases, STEM-Net surpasses state-of-the-art models by a substantial margin, further validating its effectiveness. We outline our contributions as follows:

- We delve into the intricacies of samples to investigate the negative transfer in MTL recommendation models. Surprisingly, our investigation reveals the presence of negative transfer on samples that receive comparable feedback from both tasks.
- We propose a novel paradigm called STEM (Shared and Task-specific EMbeddings) for multi-task recommendation. Under this paradigm, we design a simple model called STEM-Net equipped with an All Forward Task-specific Backward gating network to facilitate direct knowledge transfer across tasks while learning task-specific embeddings.
- We conduct comprehensive experiments and ablation studies on three MTL recommendation datasets and provide compelling evidence of STEM-Net's effectiveness. In online A/B testing, STEM-Net achieves significant improvement in GMV (Gross Mechanise Value) over the base model (MMoE), and it has been successfully deployed in Tencent's online advertising platform.

Related Work

Evolution of the MTL Model Architectures

In this section, we provide a comprehensive review of multi-task recommendation models that are based on the widely adopted *Embedding-Expert-Gate-Tower* architecture (Caruana 1997; Ma et al. 2018a; Tang et al. 2020; Yang et al. 2022; Hazimeh et al. 2021; Qin et al. 2020). This architecture comprises four key modules: *Embeddings*, which employ dense vectors to represent sparse features; *Experts*, which extract knowledge from the input features; *Gates*, which aggregate the outputs of the experts using attentive weights; and *Towers*, which are task-specific classifiers. Figure 2 illustrates representative models at various levels of task specificity. Notably, recent works have witnessed a gradual shift in the placement of task-specific modules, from top modules such as towers (Caruana 1997; Misra et al. 2016) or gates (Ma et al. 2018a; Pan et al. 2019) toward bottom modules such as experts (Tang et al. 2020).

Tower-level task-specific models, where each task has an independent tower and is updated only by its own loss, as shown in Figure 2(a). All remaining parameters are shared across tasks. Shared-Bottom (Caruana 1997) is a representative of these models, also known as hard parameter sharing. OMoE (Ma et al. 2018a), Cross-stitch (Misra et al. 2016) are variants of Shared-Bottom with shared experts (*i.e.* bottom).

Gate-level task-specific models, which employ a gating network to each task to weight the outputs of experts. MMoE is a representive of these models, as shown in Figure 2(b). Dselect-k (Hazimeh et al. 2021), MoSE (Qin et al. 2020) and MT-FwFM (Pan et al. 2019) also fit into this category. All parameters in the modules below the gates, including experts and embeddings, are shared across tasks.

Expert-level task-specific models, where each task has its own experts on the basis of the Tower/Gate-level task-

specific model. PLE (Tang et al. 2020) is a representative model, which utilizes both task-specific and shared experts. Numerous subsequent works follow this design including PFE (Xin et al. 2022), MLPR (Wu et al. 2022) and TAML (Liu et al. 2023). Moreover, some methods based on sparse routings, such as SNR (Ma et al. 2019) and CSRec (Bai et al. 2022), are also Expert-level task-specific models since some parameters of their experts are task-specific. In these models, even though parameters of some experts are task-specific, the embeddings are still shared across tasks.

In contrast to the above studies, we focus on **Embedding-level task-specific models**. We will discuss this in detail in the method section.

Delve into Negative Transfer in MTL Recommenders

In MTL recommendation, negative transfer refers to the phenomenon where knowledge or information learned from one task adversely affects the performance of another task. Existing research primarily focuses on investigating negative transfer across all samples as a whole, often overlooking the inherent intricacies within samples. To illustrate this point, let us consider an MTL setting involving two tasks, denoted as task `A` and `B`. Samples with limited positive feedback from `A` may intuitively benefit from MTL by receiving additional feedback from `B`. Besides, samples with abundant positive feedback from `A` can also get performance lift by incorporating complementary feedback from `B`. However, when a comparable amount of feedback exists for both tasks `A` and `B`, negative transfer may occur due to the possible contradictory user preferences over items between two tasks.

To verify this hypothesis, we can split the testing samples into three subsets according to the relative amount of (expected) positive feedback between task `A` and `B`: `A`-Overwhelming subset $\mathcal{D}_{\text{A-O}}$ and `B`-Overwhelming subset $\mathcal{D}_{\text{B-O}}$, each consisting of samples with *overwhelming positive feedback* from task `A` or `B`, respectively, and Comparable subset $\mathcal{D}_{\text{Comp}}$ which consists of samples with *comparable positive feedback* from both tasks. In particular, we measure the amount of positive feedback for a given sample regarding each task by the *expected* positive feedback from each *single task model*, and then split the samples into subsets according to the gap between task-wise expected positive feedback.

We conducted an empirical analysis on the TikTok dataset, which consists of two tasks: `Like` and `Finish`. We aim to investigate negative transfer on the `Like` task, as it exhibits a significantly lower number of positive samples, rendering it susceptible to be dominated. Following the above mentioned procedure, we first discretize the expected feedback of each task into 10 buckets with equal sample frequency. The relative feedback can then be quantified by the difference in bucket indices: $b(f_A(x)) - b(f_B(x))$. We define `Finish`-Overwhelming subset as the set of samples with a bucket index difference in the range of $[-9, -4]$, the Comparable subset with a range of $(-4, 6]$ and `Like`-Overwhelming with a range of $(6, 9]$. We evaluate the performance of two existing MTL models, MMoE and PLE, and the `Like` single task model, on the `Like` task. As depicted in Figure 1, both MMoE and PLE demonstrate improved performance when there is overwhelming feedback from either task. However, both models exhibit inferior performance compared to the single task model on subset $\mathcal{D}_{\text{Comp}}$. We attribute the performance drop of MMoE and PLE to that there may be contradictory user preference over items between `Like` and `Finish` on the comparable subset. Shared embedding methods, such as MMoE and PLE, is incapable to capture such contradictory preference. We'll investigate contradictory user preference in the Performance Evaluation Section. In the following, we'll present our STEM paradigm and STEM-Net under this paradigm.

Method

The STEM Paradigm

In recommendation, each sample consists of M active features, i.e., $x = \{x_1, x_2, \cdots, x_M\}$, where M denotes the number of fields and x_i represents the active feature of the i-th field. All existing methods follow a shared-embedding paradigm, that is, they have only one shared embedding table, and learn one shared embedding for each feature.

Under STEM paradigm, in addition to the shared embedding table $E^S \in \mathcal{R}^{N \times K}$ in the shared-embedding paradigm, we also employ T task-specific embedding tables $\{E^1, \ldots, E^t, \ldots, E^T\}$, one for each task t. Here N denotes the total number of features across all fields, and K denotes the embedding dimension. For a given feature x_i, we get the task-specific embeddings $\{\boldsymbol{v}_i^t\}$ as well as the shared embedding $\boldsymbol{v}_i^S$ as follows :

$$\begin{aligned} \boldsymbol{v}_i^t &= \texttt{Lookup}(x_i, E^t), \\ \boldsymbol{v}_i^S &= \texttt{Lookup}(x_i, E^S). \end{aligned} \tag{1}$$

We concatenate the task-specific and shared embeddings across all active features as follows:

$$\begin{aligned} \boldsymbol{h}_0^t &= [\boldsymbol{v}_1^t, \ldots, \boldsymbol{v}_i^t, \ldots, \boldsymbol{v}_M^t], \\ \boldsymbol{h}_0^S &= [\boldsymbol{v}_1^S, \ldots, \boldsymbol{v}_i^S, \ldots, \boldsymbol{v}_M^S], \end{aligned} \tag{2}$$

where the concatenated task-specific embedding $\boldsymbol{h}_0^t$ and shared-embedding $\boldsymbol{h}_0^S$ are the input to the experts, as discussed below.

The STEM-Net Model

Under STEM paradigm, our proposed STEM-Net further employ Shared & Task-Specific Experts and an All Forward Task-specific Backward gating network to facilitate the learning of task-specific embeddings and knowledge transfer across tasks.

Shared & Task-Specific Experts Following PLE (Tang et al. 2020), we employ shared and task-specific experts. However, each shared or task-specific expert group is equipped with an independent embedding table, so as to prevent any parameters interference. In particular, the shared experts only takes $\boldsymbol{h}_0^S$ as the input, which consists of embeddings from the E^S. Experts for task t only takes $\boldsymbol{h}_0^t$ as the input, consisting of embeddings from E^t. For brevity,

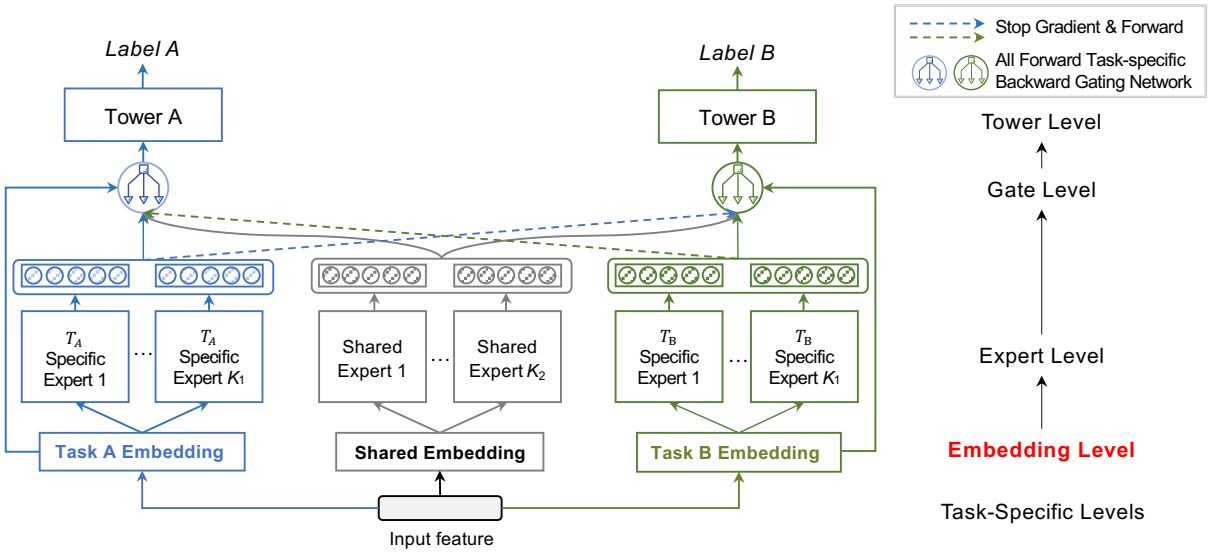


Figure 3: Overview of STEM-Net.

we assume that each task-specific expert group and shared expert group contains K_1 and K_2 experts respectively. Each expert is a multi-layer perceptrons (MLPs) over the input:

$$\begin{aligned} \boldsymbol{h}_i^t &= \texttt{MLP}_i^t(\boldsymbol{h}_0^t), \forall i = 1, \ldots, K_1, \\ \boldsymbol{h}_j^S &= \texttt{MLP}_j^S(\boldsymbol{h}_0^S), \forall j = 1, \ldots, K_2, \end{aligned} \tag{3}$$

where $\boldsymbol{h}_i^t$ represents output of the i-th expert for task t and $\boldsymbol{h}_j^S$ represents the output of the j-th shared expert.

All Forward Task-specific Backward Gating Network The gating mechanism aims to integrate outputs from experts for each task's tower. In STEM-Net, we'd like to receive the outputs from all experts, including both shared and task-specific ones, to maximize knowledge transfer, while at the same time learn task-specific embeddings, so as to capture task-specific preference. We achieve this by designing an All Forward Task-specific Backward gating network, that connects the tower of each task to all experts, with a stop gradient operation on the experts of the other tasks. Formally, the output of the gating network for task t is formalized as:

$$\begin{aligned} \boldsymbol{o}^t = &\sum_i^{K_1} \boldsymbol{g}_i^{t \to t} \boldsymbol{h}_i^t + \sum_i^{K_2} \boldsymbol{g}_i^{S \to t} \boldsymbol{h}_i^S \\ &+ \sum_{t' \in \mathcal{T}, t' \neq t} \sum_i^{K_1} \boldsymbol{g}_i^{t' \to t} \texttt{SG}(\boldsymbol{h}_i^{t'}), \end{aligned} \tag{4}$$

where $\texttt{SG}(\cdot)$ is the stop gradient operator, and $\boldsymbol{g}^{t \to t} \in \mathcal{R}^{K_1}$ denotes the attentive weight on connections between task t's corresponding tower and experts, $\boldsymbol{g}^{t' \to t} \in \mathcal{R}^{K_1}$ denotes the weight between the expert of task t' and tower of task t, $\boldsymbol{g}^{S \to t} \in \mathcal{R}^{K_2}$ denotes the weight between shared experts and tower of task t, respectively. These weights are computed with a softmax over the concatenated embeddings:

$$[\boldsymbol{g}^{t \to t}, \{\boldsymbol{g}^{t' \to t}\}, \boldsymbol{g}^{S \to t}] = \texttt{Softmax}(W_g^t(\boldsymbol{h}_0^t + \boldsymbol{h}_0^S)), \tag{5}$$

where $W_g^t \in \mathcal{R}^{d \times (K_1 \times T + K_2)}$.

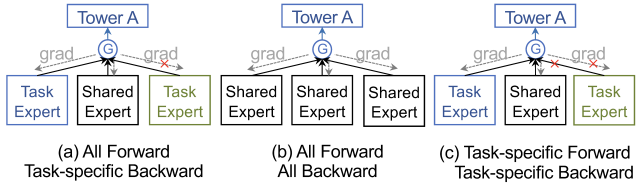


Figure 4: Comparison of gating networks of MMoE (All Forward All Backward), PLE (Task-specific Forward Task-specific Backward) and our STEM-Net (All Forward Task-specific Backward).

Towers and Loss Function Finally, we assign each task an independent tower and obtain the final output as:

$$\hat{y}_t = \sigma(\texttt{MLP}^t(\boldsymbol{o}^t)), \tag{6}$$

where σ is the sigmoid function. We choose binary cross-entropy loss function as the objective function for each task, and the final loss function is formulated as follows:

$$\mathcal{L} = -\sum_t^T y_t \log(\hat{y}_t) + (1 - y_t) \log(1 - \hat{y}_t), \tag{7}$$

where y_t is the ground truth of task t.

Comparison of Gating Networks MMoE adopts an All Forward All Backward gating network, that each task's tower receives output from all experts, and propagate the gradients to all experts, too. All Backward makes MMoE unable to learn task-specific embeddings, even if we assign independent embedding tables for each expert, denoted as Multi-Embedding MMoE (ME-MMoE for short).

PLE adopts a Task-specific Forward Task-specific Backward gating network, where each task's tower receives outputs from shared experts and its own experts, but not from experts of the other tasks. Similarly, each task's tower only

Dataset	#User	#Items	#Samples	#Fields	#Tasks	Positive Ratio (%)
TikTok	560K	1800K	223.4M/24.8M/27.6M	8	2	28.31/1.60
QK-Video	970K	760K	95.9M/12.0M/12.5M	16	2	24.01/2.03
KuaiRand1K	1K	189K	10.9M/0.39M/0.42M	32	8	37.76/1.54/0.10/0.26/0.08/0.10/26.17/1.78

Table 1: Statistics of processed datasets.

Model	Task A	Task B	Task C	Task D	Task E	Task F	Task G	Task H	Avg. AUC	MTL Gain
Single-Task	0.7534	0.9293	0.8294	0.8943	0.8572	0.8821	0.7650	0.8358	0.8433	-
Shared-Bottom	0.7535	0.9261	0.8162	0.8881	0.8228	0.7820	0.7642	0.8340	0.8234	-0.0199
OMoE	0.7549	0.9273	0.8404	0.8923	0.8352	0.8750	0.7655	0.8349	0.8407	-0.0026
MMoE	0.7541	0.9278	0.8268	0.8901	0.8591	0.8908	0.7647	0.8360	0.8437	+0.0003
PLE	0.7537	0.9290	0.8362	0.8885	0.8449	0.8940	0.7643	0.8374	0.8435	+0.0002
ME-MMoE	**0.7555**	0.9288	0.8310	0.8912	0.8500	0.8668	**0.7658**	**0.8385**	0.8410	-0.0024
ME-PLE	0.7536	**0.9294**	0.8353	**0.8970**	0.8521	0.8871	0.7637	0.8381	0.8445	+0.0012
STEM-Net	0.7523	0.9282	**0.8420**	0.8910	**0.8635**	**0.9070**	0.7637	0.8359	**0.8480**	**+0.0047**

Table 2: Overall performance on KuaiRand1K.

Model	Finish		Like		Avg.
	Logloss↓	AUC ↑	Logloss↓	AUC↑	AUC ↑
Single-Task	0.5111	0.7505	0.0558	0.9058	0.8281
Shared-Bottom	0.5112	0.7504	0.0560	0.9022	0.8263
OMoE	**0.5103**	**0.7516**	0.0559	0.9029	0.8273
MMoE	0.5105	0.7511	0.0560	0.9018	0.8265
PLE	0.5105	0.7511	0.0560	0.9016	0.8264
ESMM	0.5111	0.7503	0.0564	0.9012	0.8258
AITM	0.5109	0.7506	0.0560	0.9026	0.8266
ME-MMoE	0.5114	0.7502	0.0557	0.9045	0.8274
ME-PLE	0.5120	0.7492	0.0560	0.9058	0.8275
STEM-Net	0.5104	0.7513	**0.0553**	**0.9095**	**0.8304**

Table 3: Overall performance on TikTok.

Model	Click		Like		Avg.
	Logloss↓	AUC ↑	Logloss↓	AUC↑	AUC ↑
Single-Task	0.2826	0.9234	0.0378	0.9400	0.9317
Shared-Bottom	0.2857	0.9235	0.0380	0.9389	0.9312
OMoE	0.2826	0.9238	**0.0373**	0.9394	0.9316
MMoE	0.2813	0.9238	0.0379	0.9401	0.9319
PLE	0.2832	0.9238	0.0375	0.9399	0.9318
ESMM	0.2847	0.9208	0.0378	0.9368	0.9288
AITM	0.2836	0.9237	0.0386	0.9398	0.9318
ME-MMoE	0.2815	**0.9239**	0.0375	0.9407	0.9323
ME-PLE	0.2818	0.9238	0.0374	0.9410	0.9324
STEM-Net	0.2816	0.9237	0.0381	**0.9426**	**0.9331**

Table 4: Overall performance on QK-Video.

backward propagate gradients to the shared experts and its own experts, but not to the other tasks' experts. Task-specific Forward makes PLE can't fully transfer the knowledge from other tasks *directly* besides the shared expert, even if we assign independent embeddings for each expert (group) in PLE, denoted as ME-PLE. The comparison of our gating network with those of MMoE and PLE is shown in Figure 4.

Performance Evaluation

Experimental Setup

Public Datasets. We choose three public datasets, namely TikTok, QK-Video (Yuan et al. 2022), and KuaiRand1K (Yuan et al. 2022) for performance evaluation. We replace features from all datasets that appeared less than 10 times in the training set by a default value. The statistics of the processed dataset is presented in Table 1.

Baselines. To establish a performance benchmark for comparison, we implement a **Single-Task** model, as well as popular MTL methods including **Shared-Bottom** (Caruana 1997), **OMoE** (Ma et al. 2018a), **MMoE** (Ma et al. 2018a) and **PLE** (Tang et al. 2020). The Single-Task model adopts MLPs (Multi-Layer Perceptrons) that are trained only for a single task. In implementation, it is essentially equivalent to a Shared-Bottom model with only one tower. For the Tiktok and QK-Video datasets, we introduce two additional baselines: **ESMM** (Ma et al. 2018b) and **AITM** (Xi et al. 2021). Furthermore, we provide the Multi-Embedding versions of MMoE and PLE as strong baselines to study the effect of simply including additional embeddings, namely **ME-MMoE** (Fig. 2(d)) and **ME-PLE** (Fig. 2(e)). For ME-MMoE, we allocate independent embeddings for each expert, while for ME-PLE, we assign separate embeddings for task-specific and shared experts. Note that ME-PLE also follows the STEM paradigm, in the sense that it also learns shared and task-specific embeddings. ME-MMoE and ME-PLE use the same gating network as MMoE and PLE, i.e., All Forward All Backward and Task-specific Forward Task-specific Backward, respectively.

Hyper-Parameter Settings. We implement all methods based on Pytorch and use Adam (Kingma and Ba 2015) as the optimizer. We set the learning rate as $\{1e^{-3}, 5e^{-4}, 1e^{-4}\}$, the batch size as 4096, and the l_2 regularization factor of embedding as $1e^{-6}$. We set the dimension of the embedding to 16, and each expert/bottom is an MLP with hidden units of $[512, 512, 512]$. The towers and the gate networks of all methods are MLPs with hidden units of $[128, 64]$. The number of task-specific and shared experts is chosen from $\{1, 2, 4, 8\}$. Grid search is used to find opti-

Variants	Tiktok		KuaiRand1K	
	AUC	#Param	AUC	#Param
STEM-Net-$\varnothing$	0.8261	1.00x	0.8382	1.00x
STEM-Net-(user id, item id)	0.8302	1.95x	0.8448	1.36x
STEM-Net-(user side)	0.8301	1.23x	0.8437	1.00x
STEM-Net-(item side)	0.8260	2.62x	0.8193	2.05x
STEM-Net-(all features)	0.8304	2.85x	0.8480	2.06x

Table 5: Performance of STEM-Net variants where only selected feature fields deploy task-specific.

mal hyper-parameters for all methods.

Overall Performance

The comparison between STEM-Net and the baselines on the three datasets is presented in Tables 3, 4 and 2.We have the following observations. First, STEM-Net achieves the best average AUC across all datasets. Specifically, STEM-Net exhibits an average AUC improvement of approximately $4e^{-3}$, $2e^{-3}$, and $4e^{-3}$ over the state-of-the-art model (PLE) on three datasets, respectively.

Second, STEM-Net achieves positive transfer over the single task model on the `Like` task on both TikTok and QK-Video dataset for the first time. These tasks have much less positive feedback than the other tasks, making them fragile to negative transfer. STEM-Net tackles negative transfer by learning task-specific embedding and hence capturing task-specific preference on this task.

Third, to make a fair comparison regarding number of parameters, we compare STEM-Net with another two multi-embedding baselines: ME-MMoE and ME-PLE. STEM-Net and ME-PLE both have $T + 1$ embedding tables, while ME-MMoE has the same number of embedding tables as experts (typically greater than T). STEM-Net still beats these two methods on three datasets. Furthermore, ME-PLE, which also follows STEM paradigm, outperforms ME-MMoE, especially on `Like` on TikTok and QK-Video datasets, validating the effectiveness of task-specific embeddings in capturing task-specific user preference.

Which Feature Fields Should Be Task-Specific?

We investigate which feature fields should be task-specific to learn diverse user preferences across tasks. To this end, we designed STEM-Net variants that only learn task-specific embeddings for selected fields F, denoted as STEM-Net-(F). Results on the TikTok dataset are presented in Table 5.

First, if no features are task-specific in STEM-Net, denotes as STEM-Net-$\varnothing$, its performance is comparable with that of MMoE and PLE. This proves that task-specific embeddings or the STEM paradigm (facilitated by All Forward Task-specific Backward gating) is the key for STEM-Net's performance lift, rather than the model architecture itself.

Second, our main purpose is to capture user's preference on items in STEM-Net, so we wonder if we can still achieve decent performance lift when only making embeddings of User ID and Item ID task-specific. We evaluate the performance of the corresponding model STEM-Net-(user id, item id), observing that it also achieves competitive performance with STEM-Net. This verify our hypothesis that STEM-Net's performance lift is mainly attributed to its ability to capture diverse user preference over items.

Further, we are curious whether user side (e.g., user id and device id in Tiktok) or item side (e.g., item id, author id, item city, channel, music id and video_duration in Tiktok) features are more critical to have task-specific embeddings. We design two new variants, STEM-Net-(user side) and STEM-Net-(item side), and observe the former one perform better, indicating that user side features are more effective than item in capturing diverse user preference among tasks.

Input	SG	`Finish`		`Like`		Avg.
		AUC	Logloss	AUC	Logloss	AUC
$\boldsymbol{h}_0^t$	✓	0.7509	0.5107	0.9077	0.0555	0.8293
	✗	0.7511	0.5106	0.9023	0.0560	0.8267
	△	+0.0001	-0.0001	-0.0054	+0.0005	-0.0026
$\boldsymbol{h}_0^S$	✓	0.7510	0.5106	0.9089	0.0553	0.8300
	✗	0.7511	0.5105	0.9037	0.0558	0.8274
	△	+0.0001	0.0000	0.0052	+0.0005	-0.0026
$\boldsymbol{h}_0^t + \boldsymbol{h}_0^S$	✓	0.7513	0.5104	0.9095	0.0553	0.8304
	✗	0.7510	0.5106	0.9008	0.0562	0.8259
	△	-0.0002	+0.0003	-0.0080	+0.0009	-0.0045

Table 6: The effect of input and stop gradient to gating network.

Effectiveness of the Proposed Gating Network

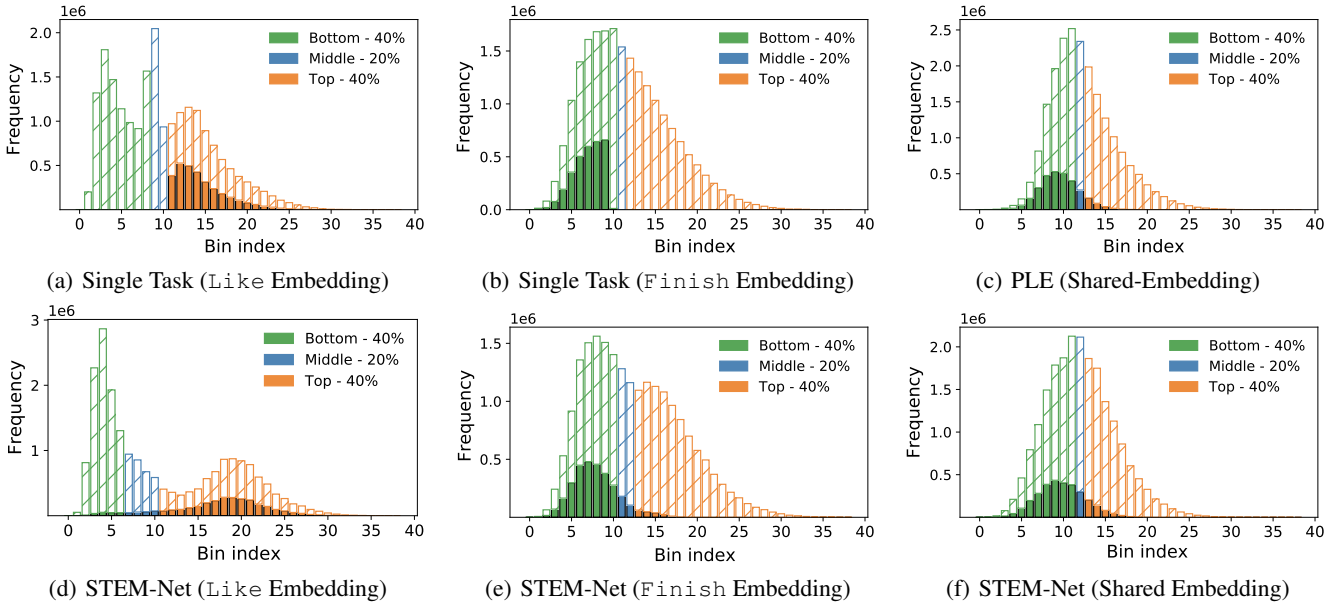
In STEM-Net, our proposed All Forward Task-specific Backward gating network is critical to learn task-specific embeddings via stop gradients (SG) operation. Furthermore, the input to the gating network is also worth discussing (Fei et al. 2021; Chang et al. 2023). Thus, we present three variants by replacing the input of Eq. 5, i.e., $\boldsymbol{h}_0^t + \boldsymbol{h}_0^S$, by $\boldsymbol{h}_0^t$ or $\boldsymbol{h}_0^S$, and conduct ablation study of SG on each variant. Experimental results are shown in Table 6. Here are two observation:

(a) The stop gradient operation proves to be critical for performance lift. As shown in Table 6, across all input settings, introducing SG (highlighted in gray) leads to an AUC improvement of approximately $5e^{-3}$ to $8e^{-3}$ for the `Like` task. Without SG, the embeddings are essentially shared between tasks, making the model similar to ME-MMoE.

(b) The best performance is achieved when the input consists of both task-specific and shared concatenated embeddings, resulting in an improvement of $1e^{-3}$ in average AUC. We argue this is because that task-specific embeddings are exclusively updated by their corresponding tasks, which restricts their ability to perceive the information of the other tasks. Consequently, integrating shared embeddings as a constituent of the input aids the gating network in perceiving common information across tasks.

Contradictory User Preference Analysis

We hypothesize that on the comparable subset where there is enough feedback from both tasks, there may be contradictory user preference over items across tasks. For example, a



(a) Single Task (`Like` Embedding) (b) Single Task (`Finish` Embedding) (c) PLE (Shared-Embedding)

(d) STEM-Net (`Like` Embedding) (e) STEM-Net (`Finish` Embedding) (f) STEM-Net (Shared Embedding)

Figure 5: The distance distribution of the contradictory user item pair set S (with solid color) as well as the whole user item pair set (with slash lines) regarding: the single task `Like` (a) and `Finish` embedding (b), the PLE embedding (c), and the `Like` (d) and `Finish`-specific (e) embedding and shared embedding (f) in STEM-Net.

user u may be inclined to an item i regarding task `A`, but be declined to it or neutral regarding task `B`. For these user item pairs, shared embedding methods fail to learn such contradictory preference, since they have only a single shared embedding table, and therefore can learn *only a single* preference. In contrast, STEM-Net should be able to capture such contradictory preference by task-specific embeddings.

We conduct the follow analysis to validate the above hypothesis. We select a set of contradictory user item pairs S whose Euclidean distance[1] are among the top-40% regarding single task `Like` embedding (Fig. 5(a)) and among the bottom-40% regarding single task `Finish` embedding (Fig. 5(b)). These user item pairs correspond to 9.63% of all samples. We plot the distance distribution of these pairs regarding the shared embedding from PLE in Fig. 5(c) and observe that PLE learn small distances for most of them, which is similar to the distribution of single task `Finish`, while contradictory to that of single task `Like`.

In STEM-Net, the distance distribution of `Like` (Fig. 5(d)) and `Finish`-specific (Fig. 5(e)) embedding table are consistent to single task correspondence, showing case that STEM-Net is able to learn the contradictory user item preference. Note that similar to the PLE emebdding, the distance distribution of shared embedding table (Fig. 5(c)) in STEM-Net is also similar to the distribution of single task `Finish` while contradictory to that of single task `Like`.

Online A/B Test

Online Deployment Since 2022, STEM-Net has been developed on Tencent's display advertising platform over various scenarios. These advertising scenarios consists of several tasks including `Follow`, `Activation`, `Fulfill Sheet`, and `Pay`.

[1]We choose Euclidean distance as the similarity metric because the experts adopt MLPs.

Scenario	Follow	Activation	Fulfill Sheet	Pay	Avg.
Scenario 1	+0.29%	+0.33%	+0.29%	+0.35%	+0.32%
Scenario 2	+0.13%	+0.22%	+0.33%	+0.27%	+0.24%
Scenario 3	+0.47%	+0.39%	+0.28%	+0.78%	+0.48%

Table 7: AUC Lift of Online A/B Test.

Performance The production model follows an MMoE architecture, where each expert is NFwFM, a variant of NFM (He and Chua 2017) which replaces the vanilla FM (Rendle 2010) by FwFM (Pan et al. 2018). We equip the production model with shared and task-specific embeddings. The overall improvements of all tasks are shown in Table **??**, indicating a significant improvement of STEM-Net over the production MMoE model. In particular, STEM-Net brings 0.32%, 0.24%, and 0.48% average AUC lifts for three representative scenarios, leading to 4.2%, 3.9%, and 7.1% GMV lift in our online A/B test.

Conclusion

In this paper, we propose a novel Share and Task-specific EMbedding paradigm to tackle the negative transfer in MTL recommendation. We design a simple model under such paradigm, namely STEM-Net, which demonstrates compelling performance on comparable samples, achieving positive transfer. In three public datasets and industrial online A/B test, we validate that our proposed STEM-Net achieves significant performance lift over state-of-the-art MTL recommendation models.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61972219), the Overseas Research Cooperation Fund of Tsinghua Shenzhen International Graduate School (HW2021013). The authors would like to thank Zhutian Lin for his help with the contradictory user preference analysis.

References

Bai, T.; Xiao, Y.; Wu, B.; Yang, G.; Yu, H.; and Nie, J. 2022. A Contrastive Sharing Model for Multi-Task Recommendation. In Laforest, F.; Troncy, R.; Simperl, E.; Agarwal, D.; Gionis, A.; Herman, I.; and Médini, L., eds., *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, 3239–3247. ACM.

Caruana, R. 1997. Multitask learning. *Machine learning*, 28(1): 41–75.

Chang, J.; Zhang, C.; Hui, Y.; Leng, D.; Niu, Y.; Song, Y.; and Gai, K. 2023. PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, 3795–3804. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701030.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 794–803. PMLR.

Fei, H.; Zhang, J.; Zhou, X.; Zhao, J.; Qi, X.; and Li, P. 2021. GemNN: Gating-enhanced multi-task neural networks with feature interaction learning for CTR prediction. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 2166–2171.

Hazimeh, H.; Zhao, Z.; Chowdhery, A.; Sathiamoorthy, M.; Chen, Y.; Mazumder, R.; Hong, L.; and Chi, E. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34: 29335–29347.

He, X.; and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 355–364.

He, Y.; Feng, X.; Cheng, C.; Ji, G.; Guo, Y.; and Caverlee, J. 2022. MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks. *WWW*, 2205–2215.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Liu, X.; Jia, Q.; Wu, C.; Li, J.; Quanyu, D.; Bo, L.; Zhang, R.; and Tang, R. 2023. Task Adaptive Multi-learner Network for Joint CTR and CVR Estimation. In *Companion Proceedings of the ACM Web Conference 2023*, 490–494.

Ma, J.; Zhao, Z.; Chen, J.; Li, A.; Hong, L.; and Chi, E. H. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-Task Learning. In *AAAI*, 216–223. AAAI Press.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018a. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *KDD*, 1930–1939. ACM.

Ma, X.; Zhao, L.; Huang, G.; Wang, Z.; Hu, Z.; Zhu, X.; and Gai, K. 2018b. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *SIGIR*, 1137–1140. ACM.

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-Stitch Networks for Multi-task Learning. In *CVPR*, 3994–4003. IEEE Computer Society.

Pan, J.; Mao, Y.; Ruiz, A. L.; Sun, Y.; and Flores, A. 2019. Predicting different types of conversions with multi-task learning in online advertising. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2689–2697.

Pan, J.; Xu, J.; Ruiz, A. L.; Zhao, W.; Pan, S.; Sun, Y.; and Lu, Q. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *WWW*, 1349–1357.

Qin, Z.; Cheng, Y.; Zhao, Z.; Chen, Z.; Metzler, D.; and Qin, J. 2020. Multitask mixture of sequential experts for user activity streams. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3083–3091.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining (ICDM)*, 995–1000. IEEE.

Tang, H.; Liu, J.; Zhao, M.; and Gong, X. 2020. Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations. In *RecSys*, 269–278. ACM.

Torrey, L.; Shavlik, J. W.; Walker, T.; and Maclin, R. 2010. Transfer Learning via Advice Taking. In Koronacki, J.; Ras, Z. W.; Wierzchon, S. T.; and Kacprzyk, J., eds., *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*, 147–170. Springer.

Wu, X.; Magnani, A.; Chaidaroon, S.; Puthenputhussery, A.; Liao, C.; and Fang, Y. 2022. A Multi-task Learning Framework for Product Ranking with BERT. In Laforest, F.; Troncy, R.; Simperl, E.; Agarwal, D.; Gionis, A.; Herman, I.; and Médini, L., eds., *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, 493–501. ACM.

Xi, D.; Chen, Z.; Yan, P.; Zhang, Y.; Zhu, Y.; Zhuang, F.; and Chen, Y. 2021. Modeling the Sequential Dependence among Audience Multi-step Conversions with Multi-task Learning in Targeted Display Advertising. *KDD*.

Xin, S.; Jiao, Y.; Long, C.; Wang, Y.; Wang, X.; Yang, S.; Liu, J.; and Zhang, J. 2022. Prototype Feature Extraction for Multi-task Learning. In *Proceedings of the ACM Web Conference 2022*, 2472–2481.

Yang, C.; Pan, J.; Gao, X.; Jiang, T.; Liu, D.; and Chen, G. 2022. Cross-Task Knowledge Distillation in Multi-Task Recommendation. *AAAI*.

Yang, E.; Pan, J.; Wang, X.; Yu, H.; Shen, L.; Chen, X.; Xiao, L.; Jiang, J.; and Guo, G. 2023. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10745–10753.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient Surgery for Multi-Task Learning. *NeurIPS*, 33: 5824–5836.

Yuan, G.; Yuan, F.; Li, Y.; Kong, B.; Li, S.; Chen, L.; Yang, M.; Yu, C.; Hu, B.; Li, Z.; Xu, Y.; and Qie, X. 2022. Tenrec: A Large-scale Multipurpose Benchmark Dataset for Recommender Systems. *CoRR*, abs/2210.10629.