Touch the Core: Exploring Task Dependence Among Hybrid Targets for Recommendation

Xing Tang*
FiT, Tencent
Shenzhen, China
xing.tang@hotmail.com

Yang Qiao*
FiT, Tencent
Shenzhen, China
sunnyqiao@tencent.com

Fuyuan Lyu*†
McGill University & MILA
Montreal, Canada
fuyuan.lyu@mail.mcgill.ca

Dugang Liu‡
Guangdong Laboratory of Artificial
Intelligence and Digital Economy (SZ)
Shenzhen, China
dugang.ldg@gmail.com

Xiuqiang He‡
FiT, Tencent
Shenzhen, China
xiuqianghe@tencent.com

ABSTRACT

As user behaviors become complicated on business platforms, online recommendations focus more on how to touch the core conversions, which are highly related to the interests of platforms. These core conversions are usually continuous targets, such as *watch time*, *revenue*, and so on, whose predictions can be enhanced by previous discrete conversion actions. Therefore, multi-task learning (MTL) can be adopted as the paradigm to learn these hybrid targets. However, existing works mainly emphasize investigating the sequential dependence among discrete conversion actions, which neglects the complexity of dependence between discrete conversions and the final continuous conversion. Moreover, simultaneously optimizing hybrid tasks with stronger task dependence will suffer from volatile issues where the core regression task might have a larger influence on other tasks. In this paper, we study the MTL problem with hybrid targets for the first time and propose the model named Hybrid Targets Learning Network (HTLNet) to explore task dependence and enhance optimization. Specifically, we introduce label embedding for each task to explicitly transfer the label information among these tasks, which can effectively explore logical task dependence. We also further design the gradient adjustment regime between the final regression task and other classification tasks to enhance the optimization. Extensive experiments on two offline public datasets and one real-world industrial dataset are conducted to validate the effectiveness of HTLNet. Moreover, online A/B tests on the financial recommender system also show that our model has improved significantly. Our implementation is available here[1].

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Multi-task Learning, Task Dependence, Hybrid Targets, Recommendation

1 INTRODUCTION

Recommender systems (RS) have played a crucial role in online business platforms, which provide personalized candidates (e.g., ads, videos, funds, apps, etc) based on user interests [37]. Usually, the recommendation model in these platforms is required to meet the demand of multiple objectives due to the increasing complexity of user behaviors. For example, the fund recommendation model should consider the likelihood of the user clicking a fund and the likelihood of the user purchasing the fund [15]. However, online platforms have begun to make efforts to touch the core conversion [10, 13, 27]. As shown in Figure 1, video-sharing platforms value most users' watch time, and fund investment platforms care how much the user invests. Along with preceding discrete click or purchase targets, these continuous core targets consist of hybrid targets. Hence, learning tasks with hybrid targets simultaneously in the recommendation model benefits the online platforms.

Previously, multi-task learning (MTL) [1] has been introduced in the recommendation to make predictions for multiple targets, attracting much attention and making a success in lots of application [7, 9, 15, 24, 26]. By jointly training multiple tasks in a single model, the MTL can improve performance and decrease computational cost by knowledge sharing and cross-task transferring [28]. A vital issue for the success of MTL models is modeling relationships between multiple tasks. Unlike implicit relationships that require a well-designed learning paradigm, the explicit task dependence must be explored among these targets [22, 29]. As an example illustrated in Figure 1, recommending a video with long *watch time* is
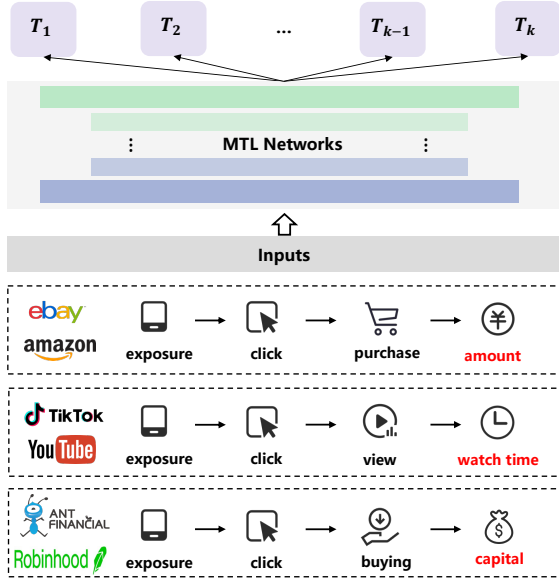
Figure 1: Illustration of hybrid target learning paradigm in various business platforms. The red one denotes core targets.

dependent on the task of whether the user *view* it, which sequentially depends on whether the user *click* it. This prior is greatly helpful for modeling these targets simultaneously in the sequential dependence MTL (SDMTL) paradigm.

However, the differences between the core and preceding targets make the modeling more challenging. On the one hand, the divergence exists in the modeling objectives. The task of the core target aims to model the distribution of continuous value, compared with preceding tasks modeling the click/conversion rate. Nevertheless, the high conversion rates seldom indicate the high value of core targets, which is contrary to the assumption that a high click rate indicates a high conversion rate in sequential dependence multitask learning [15, 25]. Therefore, exploring task dependence among hybrid targets is more complicated than traditional MTL. On the other hand, the tasks of core targets are usually regression tasks, while the preceding tasks are classification tasks. Obviously, the gradients based on regression loss will not be in accordance with the gradients of classification loss from the view of magnitude and direction. This fact will lead to training stability, and thus degrading the performance [19]. Hence, the optimization strategy designed for hybrid target learning is necessary.

There have been plenty of state-of-the-art general MTL models for recommendation. Some mainly investigate how to learn implicit task relationships and extract corresponding representations. Usually, they introduce an ensemble of expert submodules and gating networks to learn task relationships. Although many efforts have been devoted to learning implicit relationships among tasks [9, 14, 18], these approaches are still limited in two aspects. First, the negative information will transfer with the shared structure in an unpredictable manner. No matter whether the two tasks are related, the information will pass between them. The task dependence priors are absent in modeling, which makes the prediction more difficult. As a result, some methods resort to formulating the

learning paradigm as SDMTL [15, 22, 25, 29, 35]. Assuming all the targets are discrete, some specific designed architectures and losses are proposed. However, these works neglect the complexity of the dependence between the classification and regression tasks. As SDMTL assumes $\hat{y}^k - \hat{y}^{k-1} = P(t_k = 0, t_{k-1} = 1)$, implying the difference of prediction scores between adjacent tasks $(k-1, k)$ is the probability of the task k not happening while the task $k-1$ is observed. However, the equation can no longer hold when the prediction is continuous since we cannot conclude that different continuous values lead to the same probability. Besides, some works focus on optimization strategies to enhance performance. Gradients between tasks should be handled to avoid training instability and performance deterioration [4, 7, 19, 33]. These methods have no special preference for hybrid target learning, which should seriously consider the fact that regression tasks may have a dominant influence on the network gradients.

In this paper, we address the hybrid targets learning problem by proposing a Hybrid Targets Learning Net (HTLNet). Two main challenges are identified for our HTLNet. The first challenge is exploring the dependence among hybrid targets. We tackle this challenge by introducing label embedding and information fusion units in our model. The label embedding unit enables the adjacent tasks to transfer the prediction information following task dependence, and the information fusion unit ensures the information is adaptively transferred. Therefore, HTLNet explores task dependence among hybrid targets through explicit label embedding and implicit task representation. The second challenge is optimizing the complex model. Optimization is extremely hard to handle as HTLNet aims to train regression and classification tasks in a unified model. Therefore, we further develop an optimization strategy to conquer the gradients in a shared layer. Our major contributions are summarized as follows:

- This paper first distinguishes the hybrid targets learning problem in the recommendation, which aims to model the sequential auxiliary discrete targets and core continuous targets in an MTL model concerning the core target.
- We propose a novel model HTLNet that incorporates label and task information to touch the core task. By developing a corresponding optimization strategy, HTLNet can effectively explore the dependence among hybrid targets in a stable way.
- Extensive offline experiments on public and real-world product datasets are conducted. The results demonstrate the superiority of the proposed model. Furthermore, online experiments also verify the stability and effectiveness of HTLNet.

We organize the rest of the paper as follows. In Section 2, we briefly introduce related works. Section 3 formulates the MTL problem with hybrid targets in the recommendation. In section 4, HTLNet is introduced in detail. In Section 5, experimental details are given to verify our HTLNet. Finally, we conclude this paper in Section 6.

2 RELATED WORK

Multi-task Learning has broad applications in various fields [2, 3, 5, 11, 12, 16, 17, 20, 21, 32, 36]. In recommendation, many MTL network architectures are designed based on Multi-gate Mixture-of-Experts (MMoE) [14]. Progressive Layered Extraction (PLE) [18]

separates task-common and task-specific parameters to avoid parameter conflict. AdaTT [9] leverages a residual mechanism and a gating mechanism for task-to-task fusion, which adaptively learns both shared knowledge and specific knowledge. This research line fails to utilize the dependence among targets, which is far from satisfactory in modeling sequential dependence multi-task learning (SDMTL). A typical application of SDMTL is estimating post-click conversion rate (CVR) [15, 25, 35, 38]. These works formulate the problem as $pCTCVR = pCTR \times pCVR$ and employ an auxiliary click-through rate (CTR) task to enhance the performance. Besides, DCMT [38] and ESCM^2 introduce imputation and inverse propensity weighting tasks to predict unbiased CVR estimation. Despite two dependent tasks, some methods expand the problem into multiple sequential dependence tasks [22, 26, 29]. AITM [29] proposes a novel information transfer mechanism between different conversion steps to explore the dependence among these targets. TAFE [22] investigates the differences between SDMTL and MTL problems and proposes a feature learning module to tackle SDMTL. However, these works assume the targets are discrete conversion steps, neglecting the complexity of hybrid targets learning. Finally, the training stability of multitask ranking models has attracted attention recently [19]. Besides, optimization strategy has been a mainstream way to enhance the performance [4, 7, 33]. These works are all based on the general MTL network, which does not consider the network architecture of hybrid targets learning.

3 PRELIMINARY

This section will elaborate on the MTL problem with hybrid targets in the recommendation. Generally, the recommendation model will rank items according to many objectives including one or more conversion targets and the core target.

First, consider the prediction of the core target in recommendation over an input space $\mathbf{X}$, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ represents input features. Given a large dataset of data points $\{\mathbf{x}_i, y_i\}$ where $\mathbf{x}_i \in \mathbf{X}$ denotes the feature vector and $y_i \in \mathbb{R}$ is the corresponding core continuous target. Then, we assume it takes the user T conversion steps to complete the core targets. In each step t, $y_i^t \in \{0, 1\}$ denotes whether the user completes the conversion and satisfies the constraint in sequential dependence that $y_i^1 \geqslant y_i^2 \geqslant \cdots \geqslant y_i^T$. Therefore, $\{\mathbf{y}, \mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^T\}$ consists of hybrid targets for all the data points. Meanwhile, we introduce these auxiliary sequential tasks to improve the core target's prediction accuracy and provide predictions of these conversions. The MTL problem with hybrid targets is then formulated as follows:

$$
\begin{aligned}
\hat{y}_i &= f(\mathbf{x}_i | \Theta, y_i^1, y_i^2, \cdots, y_i^T), \\
\hat{y}_i^T &= f(\mathbf{x}_i | \Theta, y_i^1, y_i^2, \cdots, y_i^{T-1}), \\
&\cdots \\
\hat{y}_i^1 &= f(\mathbf{x}_i | \Theta),
\end{aligned}
\tag{1}
$$

where Θ denotes the parameters of MTL model f, $\hat{y}_i$ is the prediction of core target and $\hat{y}_i^t, t \in \{1, \cdots, T\}$ are predictions of preceding targets. Note that the first equation denotes the regression task aiming to improve the core target prediction, and the remaining equations are classification tasks, composing a typical sequential dependence MTL (SDMTL) [22] respectively. One critical issue for

learning this problem is exploring the dependence among hybrid targets with the MTL model according to Equation 1.

The parameters Θ of the MTL model usually consist of three components, i.e. $\{\theta_s, \theta_{core}, \{\theta_t\}_{t=1}^T\}$, where θ_s is the shared parameters, θ_{core} is the parameters for core targets prediction, and θ_t denotes task specific parameters. The Θ is learned by jointly minimizing the core target task loss and other task losses:

$$
\mathcal{L}(f(\mathbf{X}; \Theta), \{\mathbf{y}, \{\mathbf{y}^t\}_{t=1}^T\}) = \mathcal{L}_{core}(f(\mathbf{X}; \theta_s, \theta_{core}), \mathbf{y}) + \sum_{t=1}^T \mathcal{L}_t(f(\mathbf{X}; \theta_s, \theta_t), \mathbf{y}^t),
$$

where $\mathcal{L}_{core}$ can be either mean squared error(MSE) [34], and $\mathcal{L}_t$ is the Binary Cross-Entropy (BCE) loss due to the hybrid targets. With one optimizing step, the gradient for the MTL model is computed as follows:

$$
G = \nabla_\Theta \mathcal{L}_{core} + \sum_{t=1}^T \nabla_\Theta \mathcal{L}_t.
\tag{2}
$$

As shown in Equation 2, one optimization step relies on the gradient, which is composed of two parts: the gradient for the core targets task and the sum of gradients for other tasks. The gradient component with a larger magnitude can influence the gradient more. Another critical issue for learning the MTL problem with hybrid targets is stabilizing the optimization and enhancing performance [19].

4 METHOD

To learn the hybrid targets effectively with the MTL model, we propose a novel network architecture, Hybrid Targets Learning Network (HTLNet). The overall framework of HTLNet is illustrated in Figure 2, which mainly consists of three components: (1) task towers for hybrid targets, (2) the label embedding unit (LEU) encoding the labels of auxiliary sequential tasks, (3) the information fusion unit (IFU) utilizing the information of all preceding tasks. Note that the latter two components make our model explore task dependence effectively. Furthermore, a gradient adaption method based on the HTLNet is also introduced, which enables the optimization to be stable and enhances the model performance.

4.1 Framework of HTLNet

We first introduce the details of HTLNet. The core idea is to explore the task dependence among hybrid targets by introducing label embedding and transferring both implicit and explicit information among tasks. As shown in Figure 2, all the tasks are shared with an embedding layer. With the input $\mathbf{x}_i$, the features usually include the user, item, and context features in the recommendation, which are denoted as a vector with m fields:

$$
\mathbf{x}_i = \{x_{i_1}, x_{i_2}, \cdots, x_{i_m}\}.
\tag{3}
$$

Following the input, an embedding matrix $\mathbf{E} \in R^{M \times d}$, where M is the number of features, and d is the embedding dimension, is used in the shared embedding layer to get the corresponding embedding vector for $j - th$ field:

$$
e_{i_j} = x_{i_j} \mathbf{E}.
\tag{4}
$$

Then, the vectors are all stacked together as shared embedding vector, $\mathbf{e_i} = [e_{i_1}, e_{i_2}, \cdots, e_{i_m}]$.
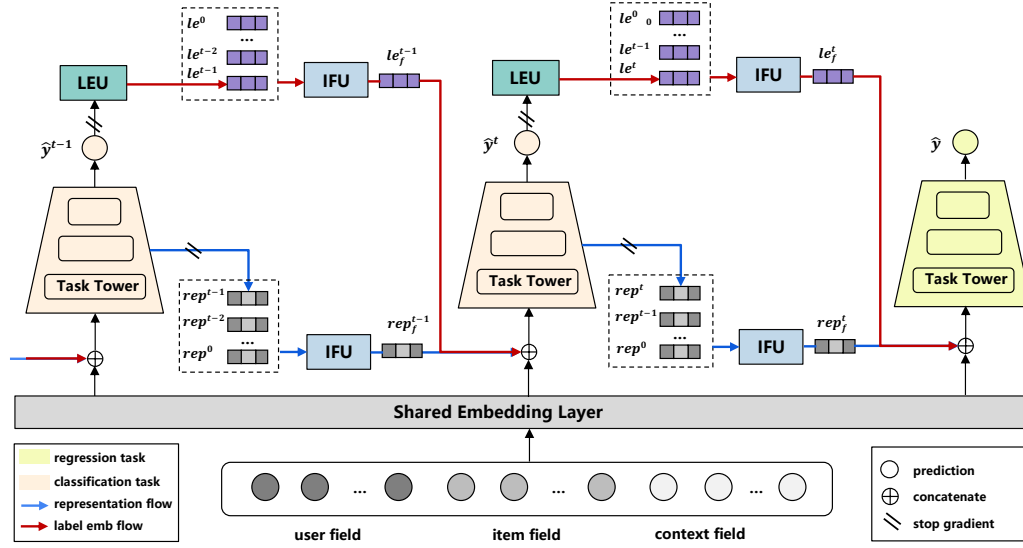
Figure 2: The overall framework of HTLNet. LEU represents the Latent Embedding Unit, which encodes label information, and the Information Fusion Unit, which adaptive infuses all preceding tasks.

Notice that the core continuous target introduces the difficulty of sharing information between hybrid targets. To explore the task dependency relationship, it is not enough to depend solely on shared embedding. Moreover, the prediction of the core target will benefit from the information from the preceding tasks. Therefore, we incorporated explicit and implicit information transferred from preceding tasks as input to the task tower. Specifically, for two adjacent tasks $t-1$ and t, the input for task t is computed as:

$$h_{input,i}^t = concat(rep_f^{t-1}, le_f^{t-1}, \mathbf{e_i}), \quad (5)$$

where rep_f^{t-1} and le_f^{t-1} are the outputs from the IFUs of task $t-1$ respectively. Notice that the le_f^{t-1} is the explicitly label information, and the rep_f^{t-1} is the implicit information from the task tower, which can be the output from a particular layer of the task tower. One way to encode label information is to use it directly as a constraint in the final loss [22, 29], which is inappropriate in hybrid target learning and will make the prediction of the core target suffer from the deficiency of preceding label information. We introduce the LEU here to address the issue, which will be illustrated in detail afterward.

For the classification tasks, the task tower gives the prediction probability of instance i:

$$\hat{y}_i^t = \sigma(MLP(h_{input,i}^t)), \quad (6)$$

where the MLP is multi-layer perception, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. The loss for these tasks is usually binary cross entropy:

$$\mathcal{L}_t = -\frac{1}{n} \sum_{i=1}^{n} [y_i^t \log(\hat{y}_i^t) + (1-y_i^t)\log(1-\hat{y}_i^t)]. \quad (7)$$

(a) Label Embedding Unit   (b) Information Fusion Unit
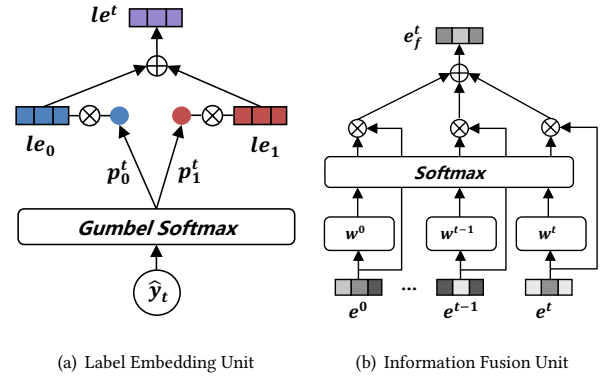
Figure 3: Illustration of Label Embedding Unit and Information Fusion Unit. $\otimes$ denotes element-wise multiplication. Note that $p_0^t + p_1^t = 1$.

For the core continuous target, the task tower gives the numerical prediction of instance i:

$$\hat{y}_i = MLP(h_{input,i}), \quad (8)$$

and the MSE loss is adopted from training:

$$\mathcal{L}_{core} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \quad (9)$$

4.1.1 Label Embedding Unit. The LEU aims to encode the label information explicitly, as shown in Figure 3(a). Unlike the previous study [3], which solely focused on implicit information among task towers, encoding label information can let a task utilize the labels of all its preceding tasks as formulated in Equation 1. However, the discrete 0/1 label information can hardly be delivered directly to the core continuous target due to the hybrid targets.

Therefore, an embedding table consisting of two rows is introduced for each classification task in our setting, which is $E^t \in R^{2 \times L_d}$ for label 0/1. The embedding table enables a one-to-one mapping from a label to a L_d-dimensional trainable vector. Each row contains the information corresponding to the label 0 or 1, denoted le_0 and le_1, respectively. An alternative to mapping the embedding is using the golden labels of tasks, which will introduce the train-test discrepancy. Specifically, the golden labels are only available in the training stage, which is missing in the test set. Moreover, directly using prediction labels for tasks leads to unstable training due to cascading errors between tasks.

In our LEU, we sample a label from the predicted probability distribution instead of predefined labels. If the sampling label is a misleading prediction of the final core task, the back-propagated gradients from the core target will penalize the embedding. The only issue is the sampling operation is not differentiable, which is incompatible with our framework. To address this issue, we further introduce the Gumbel-softmax re-parameter operation [8] here to approximate the sampling by:

$$p^t = \frac{exp(\frac{log(\hat{y}^t)}{\tau})}{exp(\frac{log(\hat{y}^t)}{\tau}) + exp(\frac{log(1-\hat{y}^t)}{\tau})}, \tag{10}$$

where τ is the decayed temperature parameter to control the smoothness of the Gumbel-softmax, when τ approximates to zero, the output will be a two-dimensional one-hot vector. The Gumbel-softmax allows HTLNet to calculate the soft weighted embedding value instead of hard selecting particular embedding, thus significantly reducing cascading errors. The output of LEU is then formalized as the weighted sum over the label embedding table:

$$le^t = [p^t, 1 - p^t]E^t. \tag{11}$$

Unlike directly utilizing the discrete label, this label embedding mechanism can transfer label information from discrete targets to the core continuous target.

4.1.2 Information Fusion Unit. The IFU is proposed to fuse all the information from the preceding tasks. Combined with LEU, there are two kinds of information to fuse, i.e., label embedding and task representation. The IFU adopts a similar attention mechanism in [29, 30] as illustrated in Figure 3(b).

Specifically, the preceding task vectors for task t can be denoted as $[\mathbf{e}^0, \mathbf{e}^1, \cdots, \mathbf{e}^t]$, where $\mathbf{e}$ is either le from LEU or rep from task tower representation. The attention is designed to allocate the weights of these transferred information adaptively.

$$\mathbf{e}_f^t = \sum_{u \in [\mathbf{e}^0, \mathbf{e}^1, \cdots, \mathbf{e}^t]} w^t h_1(\mathbf{u}) \tag{12}$$

where w^t is formulated as:

$$w^t = \frac{exp(\hat{w}_{\mathbf{u}})}{\Sigma_u exp(\hat{w}_{\mathbf{u}})}, \quad \hat{w}_{\mathbf{u}} = \frac{< h_2(\mathbf{u}), h_3(\mathbf{u}) >}{\sqrt{k}} \tag{13}$$

Where $< \boldsymbol{.}, \boldsymbol{.} >$ denotes the dot product, k is the hidden dimension, and $h_1(\boldsymbol{.}), h_2(\boldsymbol{.}), h_3(\boldsymbol{.})$ are learnable kernels for transforming input information into a new output space. Finally, we get label information le_f^t and task representation rep_f^t from IFU to explore the task dependence effectively.

4.2 Optimization Strategy for HTLNet

Despite exploring the task dependence, the designed network poses some optimization challenges. First, the loss of core continuous targets differs from other tasks, which means the gradients from the core target tower are very far from other tasks, leading to performance deterioration. Second, all the label embeddings and task representations will be transferred to the core target tower, allowing the core target loss to mainly influence the optimization of other tasks with the gradient backpropagation to label prediction and task representation. Hence, we are motivated to propose an optimization strategy for our HTLNet.

Two ways for a task to influence other tasks are shared embedding and transferred information. However, as discussed above, the transferred information will disturb the preceding task's label prediction and task representation by LEU and IFU. Hence, we cut off the influence of transferred information, leaving the shared embedding as the only way to influence each other:

$$LEU(stop\_gradient(\hat{y}^t)), \quad IFU(stop\_gradient(rep^t)). \tag{14}$$

Then, we target dealing with the gradient of shared embedding from the view of direction and magnitude. Denoted $G_{core} = \nabla_\Theta \mathcal{L}_{core}$ and $G_t = \nabla_\Theta \mathcal{L}_t$ in Equation 2, we consider the pairs of task gradients $\{G_{core}, G_t\}_{t=1}^T$ to cope with the gradient conflict of shared embedding between discrete tasks and our core continuous task. For every classification task t and core task, we first eliminate the gradient direction conflict with the core task being the target gradient for shared embedding [33]:

$$G_t = G_t - \alpha \frac{G_t \bullet G_{core}}{\|G_{core}\|^2} G_{core}$$
$$s.t. \quad G_t \bullet G_{core} < 0, \tag{15}$$

where α is a hyperparameter to control the amount of removing the conflict. As a variant of method in [33], when the gradient of preceding task t conflicts with the core task, meaning cosine similarity is negative, the Equation 15 projects the G_t onto the normal plane of G_{core}. After iterations, all the G_ts will not conflict with G_{core}, which means all preceding tasks are optimized towards core tasks.

Besides gradient conflict, another issue is the magnitude discrepancy between G_{core} and G_t. Notice that $\mathcal{L}_{core}$ is MSE loss, the magnitude of which is usually more significant than the Logloss for $\mathcal{L}_t$. Inspired by [7], we adaptive adjust the gradient magnitude of task t toward the core task in a direct way:

$$G_t = \gamma \cdot \frac{\|G_{core}\|}{\|G_t\|} \cdot G_t + (1 - \gamma) \cdot G_t, \tag{16}$$

where γ is a hyperparameter tuned to balance the gradient magnitude between task t and the core task, $\frac{\|G_{core}\|}{\|G_t\|}$ is the calculated weight to adjust the gradients. We introduce a hyperparameter C to clip the weight to avoid the explosion of gradients in a batch. When $\frac{\|G_{core}\|}{\|G_t\|}$ is too large, it will clip up to C, while it is too small, it will clip down to $\frac{1}{C}$. Combining both direction and magnitude, we summarize the gradient process for shared embedding in the Algorithm 1.

Algorithm 1: Gradient for shared parameters

Input: Tasks shared model parameters θ_s, core target task
loss $\mathcal{L}_{core}(\theta_s)$, preceding sequential task losses
$\mathcal{L}_t(\theta)$, projection relax factor α, balance relax factor
γ, balance clip threshold C

Result: Gradient of shared model parameters G_s

1 $G_{core}(\theta_s) \leftarrow \nabla_{\theta_s} \mathcal{L}_{core}(\theta_s)$
2 **for** $t \leftarrow 1$ to T **do**
3    $G_t(\theta_s) \leftarrow \nabla_{\theta_s} \mathcal{L}_t(\theta_s)$
4   **if** $G_t \cdot G_{core} < 0$ **then**
5       $G_t = G_t - \alpha \frac{G_t \cdot G_{core}}{\|G_{core}\|^2} G_{core}$
6   **end**
7    $weight = \frac{\|G_{core}\|}{\|G_t\|}$;
8   **if** $weight < 1/C$ **then**
9       $weight \leftarrow 1/C$
10   **end**
11   **if** $weight > C$ **then**
12       $weight \leftarrow C$
13   **end**
14    $G_t = \gamma \cdot weight \cdot G_t + (1 - \gamma) \cdot G_t$
15 **end**
16 $G_s = G_{core} + \sum_{t=1}^{T} G_t$
17 **ouput** G_s

5 EXPERIMENTS

In this section, we first outline the experimental setup for our model. Two public and one private industry datasets are used to evaluate the performance. We design experiments to answer the following research questions:

- **RQ1**: Does HTLNet achieve superior performance on MTL for hybrid targets compared with mainstream MTL models?
- **RQ2**: How do the proposed network structure and optimization strategy affect the HTLNet? Can we only rely on one component to achieve the same performance?
- **RQ3**: How well do the proposed LEU and IFU affect the performance of HTLNet?
- **RQ4**: Does the optimization strategy for HTLNet perform better than the previous methods?
- **RQ5**: Can the HTLNet also achieve better performance in the online setting?

5.1 Experimental Setup

5.1.1 Datasets. The datasets in previous MTL works only consist of tasks with a single type of label, which can not be directly used for evaluation in hybrid target learning. Hence, we define related tasks with hybrid targets from two public datasets, including video and revenue scenarios, and collect one product dataset from a real-world fund recommendation scenario.

- KuaiRand [6]: This is a sequential recommendation dataset collected from the video-sharing mobile app. Despite providing randomly exposed items, it also contains various user behaviors, such as clicking, sharing, and liking the app. In our experiments,

we use the KuaiRand-pure and choose three hybrid targets defined in the dataset: click, long view, and watch time, setting watch time as the core target.
- Kaggle-Revenue [23]: This Kaggle Acquire Valued Shoppers Challenge competition dataset contains a complete basket-level shopping history for customers and companies. The data process follows [23], and we define three hybrid targets: user repurchase in one year, user repurchase in one month, and the amount of user repurchase in one month, which is the core target.
- Product: The product dataset is collected from the recommendation logs from an online fund recommendation platform in the past three months, which records users' click and purchase fund behaviors. We define three hybrid targets: click, purchase, and the amount of purchase.

The statistics of these three datasets are shown in Table 1. The datasets are divided chronologically into training, validation, and test sets, keeping the ratio 8 : 1 : 1.

Table 1: Statistics of three datasets.

		KuaiRand			
name	sample	watch time		click	long view
metrics	size	mean	std	ratio	ratio
value	1186059	6.94	18.79	17.62%	8.50%

		Kaggle-Revenue			
name	sample	repurchase 1M amt		repurchase 1Y	repurchase 1M
metrics	size	mean	std	ratio	ratio
value	5041007	4.52	165.69	85.68%	41.05%

		Product			
name	sample	purchase amt		click	purchase
metrics	size	mean	std	ratio	ratio
value	19160294	1627.05	8009.42	8.65%	4.82%

5.1.2 Baselines. We choose baseline models from four aspects. First, the **DNN** is a three-layer MLP structure for a single task. Second, **Shared-Bottom** [1], **MMoE** [14], **PLE** [18], and **AdaTT** [9] are SOTA models in MTL for recommendation. Third, **ESMM** [15] and **AITM** [29] are MTL models that take the sequential dependence into consideration. Finally, **MetaBalance** [7] is the recent optimization method for MTL.

5.1.3 Evaluation Metrics. Because our experiments have hybrid targets, we introduce the evaluation metrics for the classification and regression tasks, respectively. Two widely used metrics, AUC and LogLoss, are adopted for all the classification tasks. Regarding regression tasks, we adopt NRMSE (Normalized Root-Mean-Square Error) and NMAE (Normalized Mean Absolute Error) to evaluate watch time and purchase amount following [31, 34]. Besides these two metrics for regression, Gini and Spearman [23] are also adopted to measure purchase amounts because this prediction aims to rank the users according to purchase amounts. Note that AUC, Gini, and Spearman all measure the rank results; a more significant value indicates a better performance. On the contrary, Logloss, NRMSE, and NMAE suggest a smaller value will lead to a better result.

5.2 Overall Performance (RQ1)

The compared experimental results on two public datasets and one product dataset are separately presented in Table 2 due to the

Table 2: Results on two public and one product datasets.

Dataset	Task	Metrics	DNN	SB	MMoE	PLE	AdaTT	ESMM	AITM	MB	HTLNet
KuaiRand	click	AUC↑	0.7512	0.7399	0.7379	0.7409	0.7396	0.7303	0.7391	<u>0.7541</u>	**0.7574***
		Logloss↓	0.4074	0.4115	0.4127	0.4110	0.4120	0.4259	0.4118	<u>0.4065</u>	**0.4035***
	long view	AUC↑	0.7698	0.7685	0.7645	0.7694	0.7683	0.7540	0.7672	<u>0.7768</u>	**0.7787***
		Logloss↓	0.2536	0.2543	0.2566	0.2542	0.2550	0.2832	0.2552	<u>0.2527</u>	**0.2509**
	watch time	NRMSE↓	0.8979	0.8968	0.8970	<u>0.8963</u>	0.8975	0.8981	0.8978	0.8967	**0.8937***
		NMAE↓	0.9553	0.9620	0.9621	0.9648	<u>0.9508</u>	0.9605	0.9533	0.9509	**0.9093***
Kaggle-Revenue	repurchase 1Y	AUC↑	0.6892	0.6568	0.6696	0.6732	0.6720	0.6361	0.6728	<u>0.6896</u>	**0.6960***
		Logloss↓	0.3830	0.3931	0.3903	0.3885	0.3888	0.6871	0.3871	<u>0.3825</u>	**0.3790***
	repurchase 1M	AUC↑	0.6255	0.6134	0.6161	0.6178	0.6187	0.6046	0.6151	<u>0.6267</u>	**0.6296***
		Logloss↓	0.6542	0.6606	0.6593	0.6598	0.6580	0.7755	0.6579	<u>0.6539</u>	**0.6520**
	repurchase 1M amount	NRMSE↓	0.9985	0.9984	0.9987	0.9988	0.9987	0.9989	<u>0.9982</u>	0.9985	**0.9980**
		NMAE↓	1.0902	1.1064	<u>1.0623</u>	1.0708	1.0969	1.1981	1.1274	1.1265	**1.0585***
		Spearman↑	0.2339	0.2340	0.2357	0.2332	<u>0.2419</u>	0.2379	0.2204	0.2376	**0.2475***
		Gini↑	0.5093	0.5090	0.5093	0.5136	<u>0.5185</u>	0.5131	0.5029	0.5100	**0.5233***
Product	click	AUC↑	0.7914	0.7968	0.7749	0.7936	0.7958	0.7069	<u>0.8036</u>	0.7965	**0.8049***
		Logloss↓	0.1508	0.1498	0.1544	0.1500	0.1497	0.2324	<u>0.1479</u>	0.1492	**0.1472**
	convert	AUC↑	0.8929	0.8991	0.8770	0.8983	0.9000	0.8378	0.9003	<u>0.9008</u>	**0.9023**
		Logloss↓	0.1105	0.1062	0.1119	0.1054	0.1040	0.1982	<u>0.1034</u>	0.1035	**0.1030**
	purchase amount	NRMSE↓	0.8475	0.8441	<u>0.8351</u>	0.8507	0.8401	0.8594	0.8377	0.8365	**0.8346**
		NMAE↓	0.8471	0.8499	0.8846	0.9276	0.9234	<u>0.8469</u>	0.8807	0.9094	**0.8383***
		Spearman↑	0.1759	0.1779	0.2060	<u>0.2088</u>	0.2014	0.1892	0.1956	0.1954	**0.2244***
		Gini↑	0.7982	0.8085	0.8254	0.8274	<u>0.8279</u>	0.8128	0.8212	0.8219	**0.8362***

Here the best and second best results are marked in **bold** and <u>underline</u> fonts, respectively. Each experiment is repeated 10 times for statistical confidence, and * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between our method and the best baseline.

space. Overall, our MTLNet achieves the best metrics in all target prediction tasks for all the datasets, which verifies the effectiveness of HTLNet. We summarize our insights as follows.

Firstly, compared with single-task DNN models, most MTL models perform better on all tasks in the corresponding dataset, demonstrating that MTL can predict all hybrid tasks well. This justifies the reason we introduce MTL to learning hybrid targets. Among all the models, our HTLNet can improve performance significantly due to the carefully designed network architecture and corresponding optimization strategy.

Secondly, HTLNet is statistically significant on most tasks compared to the other baselines. This necessitates considering both network architecture design and optimization strategy to improve the performance of hybrid target learning. Specifically, HTLNet performs consistently well on core target tasks and preceding sequential tasks compared with ESMM, AITM, and MetaBalance, which indicates that our HTLNet architecture can explore the dependence between hybrid targets. As to MMoE, AdaTT, and PLE, they achieve better performance on core target tasks compared to others. However, because of the optimization difficulty for hybrid targets, they perform worse on the preceding classification tasks.

Thirdly, among MTL models except for HTLNet, MetaBalance performs better in auxiliary tasks, demonstrating that optimization strategy is a critical factor in hybrid target learning. On the other hand, the general MTL model performs slightly better than the SDMTL model on core target regression tasks, i.e., AITM and ESMM. We conjecture that these SDMTL models are mainly designed to explore the dependence among classification targets, neglecting the peculiarity of regression tasks. On the contrary, our HTLNet first carefully designs the network architecture to explore

the dependence among hybrid targets and provide a corresponding optimization strategy, further improving performance.

Moreover, an alternative loss to Kaggle-Revenue and product datasets is lognormal loss [23], which has been widely adopted in Lifetime Value Prediction. To handle this special target, we compare ZILN and HTLNet with ZILN architecture on these two datasets as shown in Table 3. Compared with MSE loss in Table 2, the lognormal loss improves the performance of all the tasks, which indicates that the lognormal loss is a better inductive bias for revenue prediction. Notice that our HTLNet still performs better with Ziln, indicating the superiority and generalization of our HTLNet on hybrid target learning.

5.3 Ablation Study (RQ2)

We conduct ablation studies on the KuaiRand dataset to analyze the effectiveness of the main components.

As discussed in the previous section, our HTLNet contributes to hybrid target learning in network architecture and optimization strategy. Hence, we introduce two variants of HTLNet: (1) **HTLNet w/o Architecture** replaces the designed architecture with a shared-bottom network, which means there are no special modules for exploring the task dependencies. (2) **HTLNet w/o Optimization** removes the gradient process described in Algorithm 1, which only relies on the designed architecture.

We present the results in Table 4 with the evaluation metrics on all tasks. It can be concluded that HTLNet outperforms all variants consistently. Specifically, **HTLNet w/o Optimization** is the worst variant in this setting. This phenomenon further justifies our contribution to propose an optimization strategy based on the designed architecture. This is necessary in hybrid target learning,

Table 3: Results of HTLNet with ZILN Loss on Kaggle-Revenue and Product datasets

Dataset	Task	Metrics	ZILN	HTLNet-ZILN
Kaggle-Revenue	repurchase 1Y	AUC↑	-	**0.7022***
		Logloss↓	-	**0.3780***
	repurchase 1M	AUC↑	0.6320	**0.6348***
		Logloss↓	0.6507	**0.6499**
	repurchase 1M amount	NRMSE↓	0.8832	**0.8540***
		NMAE↓	1.1463	**1.1447**
		Spearman↑	0.2525	**0.2568***
		Gini↑	0.5304	**0.5341***
Product	click	AUC↑	-	**0.8091***
		Logloss↓	-	**0.1466***
	convert	AUC↑	0.9041	**0.9082***
		Logloss↓	0.1024	**0.1021**
	purchase amount	NRMSE↓	0.8884	**0.8693**
		NMAE↓	0.8794	**0.8570**
		Spearman↑	0.2444	**0.2547***
		Gini↑	0.8691	**0.8749***

Table 4: Ablation Study of HTLNet on KuaiRand dataset

Dataset	KuaiRand					
Task	click		long view		watch time	
Metrics	AUC↑	Logloss↓	AUC↑	Logloss↓	NRMSE↓	NMAE↓
HTLNet	**0.7574**	**0.4035***	**0.7787***	**0.2509**	**0.8937**	**0.9093***
w/o Architecture	0.7470	0.4082	0.7746	0.2527	0.8955	0.9630
w/o Optimization	0.7420	0.4104	0.7713	0.2538	0.8970	0.9480

Here * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between

HTLNet and the best-performed baseline.

where the objects vary with each other significantly. Moreover, the **HTLNet w/o Architecture** is slightly better than **HTLNet w/o Optimization** while still worse than HTLNet. This indicates that our LEU and IFU effectively transfer information from task labels and representations.

5.4 Analysis of Network Architecture (RQ3)

To better understand the effects of architecture in our HTLNet, we further design experiments to investigate the network architecture on the KuaiRand dataset.

Table 5: Network Architecture Analysis on KuaiRand dataset

Dataset	KuaiRand					
Task	click		long view		watch time	
Metrics	AUC↑	Logloss↓	AUC↑	Logloss↓	NRMSE↓	NMAE↓
HTLNet	**0.7574**	**0.4035**	**0.7787**	**0.2509**	**0.8937**	**0.9093***
w/o representation	0.7526	0.4058	0.7773	0.2512	0.8951	0.9248
w/o label embedding	0.7565	0.4044	0.7785	0.2511	0.8950	0.9198

Here * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between

HTLNet and the best-performed baseline.

Notice that two kinds of information are transferred to explore the dependence among hybrid targets. We thus investigate how these two kinds of information affect the performance. We introduce two variants of the HTLNet network architecture. (1) **HTLNet w/o representation** removes the IFU for task representations fusion. (2) **HTLNet w/o label embedding** removes the LEU for each task, equivalent to removing the label information transferred among tasks. The results are summarized in Table 5. Overall, removing any information from the original HTLNet will degrade the performance. **HTLNet w/o representation** performs slightly better than

HTLNet w/o label embedding, which indicates the information is limited from preceding tasks with only label embedding.
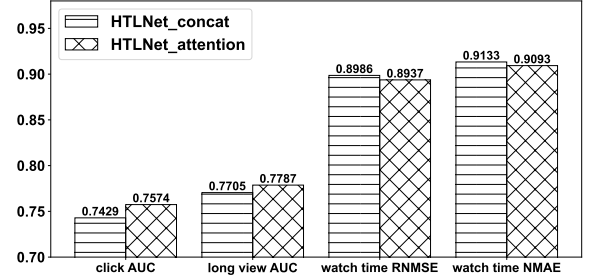
Figure 4: Comparison of the different operations in IFU.

5.5 Analysis of Optimization Strategy (RQ4)

We also investigate the effects of information fusion in HTLNet. As the information fusion operation in IFU is attention, we replace it with a simple concatenation operation. The results are given in Figure 4. The performance of classification tasks suffers a drop with the concatenation operation, as directly fusing information from a preceding task cannot avoid the negative transfer. Over the click AUC, HTLNet also performs better with the attention operation, indicating that transferring inappropriate information will influence preceding tasks with shared embedding. As to the core target, the attention mechanism can adaptively learn what and how much information to transfer from preceding tasks.

Table 6: Optimization Strategy Analysis on KuaiRand dataset

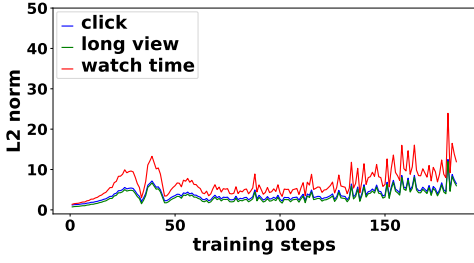| Dataset	KuaiRand					
Task	click		long view		watch time	
Metrics	AUC↑	Logloss↓	AUC↑	Logloss↓	NRMSE↓	NMAE↓
HTLNet	**0.7574**	**0.4035**	**0.7787***	**0.2509**	**0.8937**	**0.9093***
w/o stop_gradient	0.7292	0.4214	0.7549	0.2684	0.8974	0.9344
w/o gradient conflict	0.7516	0.4067	0.7743	0.2523	0.8946	0.9179
w/o gradient magnitude	0.7424	0.4109	0.7710	0.2547	0.8987	0.9321
HTLNet-Gradient Surgery	0.7432	0.4104	0.7718	0.2546	0.8992	0.9444
HTLNet-MetaBalance	0.7517	0.4070	0.7773	0.2519	0.8948	0.9136

Here * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between

HTLNet and the best-performed baseline.

Another critical component of the HTLNet is the optimization strategy. We analyze optimization strategy using several experiments on the KuaiRand dataset.

The first one compares HTLNet with relevant variants. (1) **HTLNet w/o stop_gradient**, **HTLNet w/o gradient conflict** and **HTLNet w/o gradient magnitude** represents the HTLNet without any stop gradient operations in Equation.14, HTLNet without gradient direction conflict in Equation.15 and HTLNet without gradient magnitude in Equation.16 respectively. (2) **HTLNet-Gradient Surgery** employs the Gradient Surgery [33] method on the HTLNet architecture. (3) **HTLNet-MetaBalance** employs the MetaBalance method on the HTLNet architecture. The results are illustrated in Table 6. From the results, we conclude that the operation of the stop gradient affects the performance significantly among all the gradient operations. We believe that the regression targets interfering with other tasks directly make the optimization volatile. Gradient Surgery and MetaBalance are the optimization methods for general MTL models. The results show that they perform worse than our
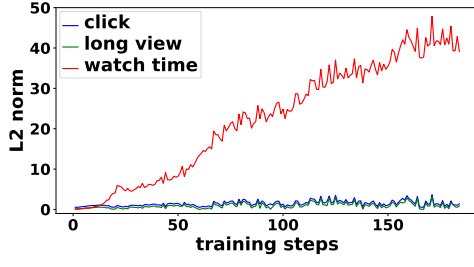
(a) HTLNet without Gradient Process

(b) HTLNet with Gradient Process

Figure 5: The illustrative case of the effect of optimization on the KuaiRand dataset. The legend indicates the different tasks. The Y-axis is the average shared embedding gradient magnitude over all mini-batch iterations in one training step.

optimization strategy, which suggests that our optimization method is suitable and customized for hybrid target learning.

In addition, we give an illustrative case to analyze the effect of the gradient process on shared embedding. As shown in Figure 5, the magnitudes of gradient w.r.t shared embedding layer of the HTLNet without or with gradient process presents a totally different tendency. The upper one (without the gradient process) shows that the gradient from the *watch time* target is diverging from the *click* and *long view* targets. Meanwhile, the magnitude of *watch time* gradient becomes increasingly more significant than the other two preceding targets. The one with a gradient process can easily control the magnitude of all three tasks, thus making them converge simultaneously. This observation indicates that the convergence of the gradient poses a severe issue in our proposed model and further explains why our model outperforms baselines.

5.6 Online Experiments (RQ5)

This section investigates whether HTLNet performs better in the online recommendation scenario. We conducted an A/B test in the online fund recommendation scenario to measure the benefits of HTLNet compared with the online baseline [14]. Unlike other recommendation scenarios, the fund recommendation scenario focuses on not only the click/conversion behaviors but also the users' purchase amount on the platform. We allocate 10% serving traffic for 14 days. Both models are trained in a single cluster, where each node contains a 96-core Intel(R) Platinum 8255C CPU, 256GB RAM, and 8 NVIDIA TESLA A100 GPU cards. Three online metrics, i.e., click ratio (CTR), purchase ratio (CVR), and purchase amount (core target), are measured in the online performance.
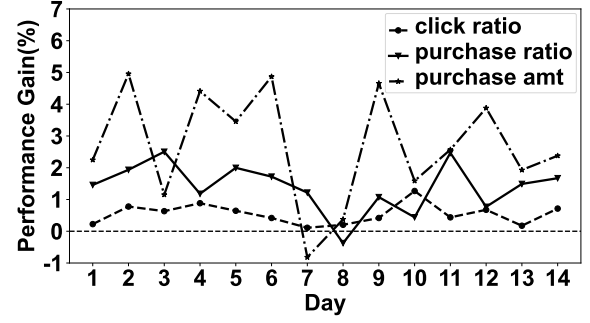

Figure 6: Online relative performance gains of three online metrics in 14 consecutive days.

Figure 6 shows the relative promotion of three corresponding objectives in 14 consecutive days. On most days, HTLNet achieves significant performance improvements, especially in terms of purchase amount. The accumulated gains of CTR, CVR, and purchase amount are +0.54%, +1.4%, and +2.69%. These significant improvements in the online scenario prove the effectiveness of HTLNet. However, the purchase amount is considered more fluctuant due to the difficulties in the regression target, which further emphasizes our study.

6 CONCLUSION

In this paper, a novel model, HTLNet, is devoted to learning hybrid targets for the first time. To effectively explore the dependence among the hybrid targets, HTLNet introduces a label embedding unit to map the label into the dense vector containing the information of preceding labels. Then an information fusion unit incorporates preceding information adaptively, which helps the task prediction. Moreover, an optimization strategy is also proposed to solve the gradient issue raised by our network architecture. Compared to several multi-task approaches, the performance gains of HTLNet on both public and product datasets demonstrate its effectiveness in exploring dependence. Besides, we also conduct an online A/B test to further verify our HTLNet performs well in a large-scale fund recommendation scenario.

Limitations. Despite HTLNet demonstrating superior effectiveness over other baselines, it requires more training times than other baselines. Additionally, HTLNet posits the hypothesis that the objective of the final core target is to be continuous while those of all the preceding tasks are to be discrete. While this is the most common scenario in the real world, we are interested in generalizing HTLNet into other complex cases.

ACKNOWLEDGMENTS

REFERENCES

[1] Rich Caruana. 1997. Multitask Learning. *Machine Learning* 28, 1 (1997), 41–75.
[2] Heyan Chai, Zhe Yin, Ye Ding, Li Liu, Binxing Fang, and Qing Liao. 2023. A Model-Agnostic Approach to Mitigate Gradient Interference for Multi-Task Learning. *IEEE Trans. Cybern.* 53, 12 (2023), 7810–7823.

[3] Wenqing Chen, Jidong Tian, Liqiang Xiao, Hao He, and Yaohui Jin. 2020. Exploring Logically Dependent Multi-task Learning with Causal Inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2213–2225.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm, Sweden, 793–802. http://proceedings.mlr.press/v80/chen18a.html

[5] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural Multi-task Recommendation from Multi-behavior Data. In *35th IEEE International Conference on Data Engineering, ICDE*. IEEE, Macao, China, 1554–1557.

[6] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22)*. ACM, Atlanta, GA, USA, 3953–3957.

[7] Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. 2022. MetaBalance: Improving Multi-task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks. In *WWW '22: The ACM Web Conference 2022*. ACM, New York, NY, USA, 2205–2215.

[8] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France.

[9] Danwei Li, Zhengyu Zhang, Siyang Yuan, Mingze Gao, Weilin Zhang, Chaofei Yang, Xi Liu, and Jiyan Yang. 2023. AdaTT: Adaptive Task-to-Task Fusion Network for Multitask Learning in Recommendations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD*. ACM, New York, NY, USA, 4370–4379.

[10] Yu Li, Yi Zhang, Lu Gan, Gengwei Hong, Zimu Zhou, and Qiang Li. 2021. RevMan: Revenue-aware Multi-task Online Insurance Recommendation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, Virtual Event, 303–310.

[11] Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, and Shu Liu. 2023. A Scale-Invariant Task Balancing Approach for Multi-Task Learning. *CoRR* abs/2308.12029 (2023). https://doi.org/10.48550/ARXIV.2308.12029

[12] Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, Shu Liu, and James Kwok. 2023. Dual-balancing for multi-task learning. (2023).

[13] Xiao Lin, Xiaokai Chen, Linfeng Song, Jingwei Liu, Biao Li, and Peng Jiang. 2023. Tree Based Progressive Regression Model for Watch-Time Prediction in Short-Video Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA,). Association for Computing Machinery, New York, NY, USA, 4497–4506.

[14] 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, KDD 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, London, UK, 1930–1939.

[15] 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, SIGIR*. ACM, New York, NY, USA, 1137–1140.

[16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-task Learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*. IEEE Computer Society, Las Vegas, NV, USA, 3994–4003. https://doi.org/10.1109/CVPR.2016.433

[17] Yi Ouyang, Bin Guo, Xing Tang, Xiuqiang He, Jian Xiong, and Zhiwen Yu. 2021. Mobile App Cross-Domain Recommendation with Multi-Graph Neural Network. *ACM Trans. Knowl. Discov. Data* 15, 4 (2021), 55:1–55:21. https://doi.org/10.1145/3442201

[18] 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 *RecSys 2020: Fourteenth ACM Conference on Recommender Systems*, Rodrygo L. T. Santos, Leandro Balby Marinho, Elizabeth M. Daly, Li Chen, Kim Falk, Noam Koenigstein, and Edleno Silva de Moura (Eds.). ACM, Virtual Event, Brazil, 269–278.

[19] Jiaxi Tang, Yoel Drori, Daryl Chang, Maheswaran Sathiamoorthy, Justin Gilmer, Li Wei, Xinyang Yi, Lichan Hong, and Ed H. Chi. 2023. Improving Training Stability for Multitask Ranking Models in Recommender Systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 4882–4893. https://doi.org/10.1145/3580305.3599846

[20] Xing Tang, Qiguang Miao, Yi-Ning Quan, Jie Tang, and Kai Deng. 2015. Predicting individual retweet behavior by user similarity: A multi-task learning approach. *Knowledge Based System* 89 (2015), 681–688.

[21] Xing Tang, Yang Qiao, Yuwen Fu, Fuyuan Lyu, Dugang Liu, and Xiuqiang He. 2023. OptMSM: Optimizing Multi-Scenario Modeling for Click-Through Rate Prediction. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*. Springer Nature Switzerland, Cham, 567–584.

[22] Xuewen Tao, Mingming Ha, Qiongxu Ma, Hongwei Cheng, Wenfang Lin, Xiaobo Guo, Linxun Chen, and Bing Han. 2023. Task Aware Feature Extraction Framework for Sequential Dependence Multi-Task Learning. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys*. ACM, New York, NY, USA, 151–160.

[23] Xiaojing Wang, Tianqi Liu, and Jingang Miao. 2019. A deep probabilistic model for customer lifetime value prediction. *arXiv preprint arXiv:1912.07753* (2019).

[24] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. Multi-Task Deep Recommender Systems: A Survey. *CoRR* abs/2302.03525 (2023). arXiv:2302.03525

[25] Hong Wen, Jing Zhang, Yuan Wang, Fuyu Lv, Wentian Bao, Quan Lin, and Keping Yang. 2020. Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR*. ACM, New York, NY, USA, 2377–2386.

[26] Yunpeng Weng, Xing Tang, Liang Chen, and Xiuqiang He. 2023. Curriculum Modeling the Dependence among Targets with Multi-task Learning for Financial Marketing. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. ACM, New York, NY, USA, 1914–1918.

[27] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning Clicks into Purchases: Revenue Optimization for Product Search in E-Commerce. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 365–374.

[28] Sen Wu, Hongyang R. Zhang, and Christopher Ré. 2020. Understanding and Improving Information Transfer in Multi-Task Learning. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, Addis Ababa, Ethiopia.

[29] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the Sequential Dependence among Audience Multi-step Conversions with Multi-task Learning in Targeted Display Advertising. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 3745–3755.

[30] Dongbo Xi, Fuzhen Zhuang, Bowen Song, Yongchun Zhu, Shuai Chen, Dan Hong, Tao Chen, Xi Gu, and Qing He. 2020. Neural Hierarchical Factorization Machines for User's Event Sequence Analysis. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1893–1896. https://doi.org/10.1145/3397271.3401307

[31] Mingzhe Xing, Shuqing Bian, Wayne Xin Zhao, Zhen Xiao, Xinji Luo, Cunxiang Yin, Jing Cai, and Yancheng He. 2021. Learning Reliable User Representations from Volatile and Sparse Data to Accurately Predict Customer Lifetime Value. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 3806–3816.

[32] Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. 2023. AdaTask: A Task-Aware Adaptive Learning Rate Approach to Multi-Task Learning. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, Washington, DC, USA, 10745–10753.

[33] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., virtual, 5824–5836.

[34] Ruohan Zhan, Changhua Pei, Qiang Su, Jianfeng Wen, Xueliang Wang, Guanyu Mu, Dong Zheng, Peng Jiang, and Kun Gai. 2022. Deconfounding Duration Bias in Watch-Time Prediction for Video Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 4472–4481. https://doi.org/10.1145/3534678.3539092

[35] Dandan Zhang, Haotian Wu, Guanqi Zeng, Yao Yang, Weijiang Qiu, Yujie Chen, and Haoyuan Hu. 2022. CTnoCVR: A Novelty Auxiliary Task Making the Lower-CTR-Higher-CVR Upper. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2272–2276. https://doi.org/10.1145/3477495.3531843

[36] Mingzhu Zhang, Ruiping Yin, Zhen Yang, Yipeng Wang, and Kan Li. 2023. Advances and Challenges of Multi-task Learning Method in Recommender System: A Survey. *CoRR* abs/2305.13843 (2023). https://doi.org/10.48550/ARXIV.2305.13843

[37] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Computing Survey* 52, 1 (2019), 5:1–5:38.

[38] F. Zhu, M. Zhong, X. Yang, L. Li, L. Yu, T. Zhang, J. Zhou, C. Chen, F. Wu, G. Liu, and Y. Wang. 2023. DCMT: A Direct Entire-Space Causal Multi-Task Framework for Post-Click Conversion Estimation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 3113–3125.