Direct Routing Gradient (DRGrad): A Personalized Information Surgery for Multi-Task Learning (MTL) Recommendations

Yuguang Liu [1], **Yiyun Miao** [2], **Luyao Xia** [3]
[1] Whisper Bond Technologies Inc.
[2] Independent Researcher
[3] Tongji University
log_whistle@163.com, myothone@gmail.com, luyao.x@tongji.edu.cn

Abstract

Multi-task learning (MTL) has emerged as a successful strategy in industrial-scale recommender systems, offering significant advantages such as capturing diverse users' interests and accurately detecting different behaviors like "click" or "dwell time". However, negative transfer and the seesaw phenomenon pose challenges to MTL models due to the complex and often contradictory task correlations in real-world recommendations. To address the problem while making better use of personalized information, we propose a personalized Direct Routing Gradient framework (DRGrad), which consists of three key components: router, updater and personalized gate network. DRGrad judges the stakes between tasks in the training process, which can leverage all valid gradients for the respective task to reduce conflicts. We evaluate the efficiency of DRGrad on complex MTL using a real-world recommendation dataset with 15 billion samples. The results show that DRGrad's superior performance over competing state-of-the-art MTL models, especially in terms of AUC (Area Under the Curve) metrics, indicating that it effectively manages task conflicts in multi-task learning environments without increasing model complexity, while also addressing the deficiencies in noise processing. Moreover, experiments on the public Census-income dataset and Synthetic dataset, have demonstrated the capability of DRGrad in judging and routing the stakes between tasks with varying degrees of correlation and personalization.

Introduction

Multi-task learning (Caruana 1997), which leverages information sharing and knowledge transfer between multiple tasks, is widely applied in recommendation systems (Lim et al. 2022). In real-world recommendation scenarios, different tasks have varying levels of importance. Some tasks, such as "click" or "dwell time," significantly impact online performance and serve as the primary training tasks, despite being challenging to train. The remaining tasks, named "engagement" or "business" heads, can provide finer-grained information and easier to converge, such as " like behavior" reflects the direction and degree of user preference for items. Although these different tasks each have their own emphasis, they are not isolated; instead, they exhibit potential interrelations. Hence, MTL inevitably has the problems of seesaw and negative transfer (Lakkapragada et al. 2023), which means the improvement of a certain task may accompany with others degradation, or some may be affected by the noise of other tasks.

Many existing studies ignore the importance relationship between tasks in business scenarios, such as adaptive weights method (Navon et al. 2022; Yang et al. 2023) and gradient surgery approaches (Yu et al. 2020; Liu et al. 2021). Well hand-crafted MTL Network, like AC-MMOE (Li and Xu 2023), solves the seesaw and negative transfer well. Nevertheless, higher model computation will lead to performance issues, especially for the online recommendation (Fabbri et al. 2022; Ben-Porat et al. 2022; Wang et al. 2023) with stricter response time. IGBv2 (Dai, Fei, and Lu 2023) introduces reinforcement learning to balance the tasks weights dynamically, which is alse computation bound and difficult in convergence. Existing approaches have improved the "seesaw" and "negative transfer" issues, but they also introduce new problems, such as higher model computation, deficiency of noise processing, or ignore the importance among tasks in business.

To address these problems, we utilize the gradient relationship between tasks to concrete stakes between tasks, defined in Fig 1(a), and split the specific task into two parts to reduce its noise impact on the overall loss (shown in Fig 1(b), named Split-MMoE). Motivated by self-supervised router network (Li et al. 2023), we propose the supervised and end-to-end training router and updater network to strengthen cooperation and reduce conflict. Furthermore, we introduce a personalized gate network, similar to PPNet (Chang et al. 2023), to mitigate gradient conflicts among users.

In summary, we propose the personalized Direct Routing Gradient (DRGrad) method, which addresses the "seesaw" and "negative transfer" problems without compromising performance or causing information distortion. The main contributions of this work are as follows:

- To solve "seesaw" and "negative transfer" problems, we propose a router network, which judges the stakes adaptively according to gradient direction between tasks. The router autonomously identifies the optimal gradients from auxiliary tasks and seamlessly integrates them into the current task.
- To address the performance and information distortion

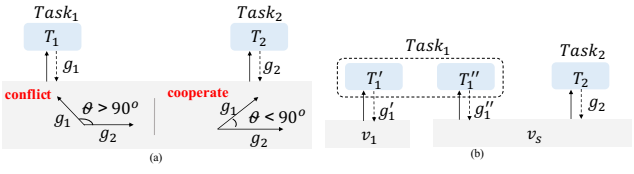

Figure 1: (a) defines θ as the angle between gradients. When $\theta > 90°$, gradients will update in opposite directions, resulting in conflicts. When $\theta < 90°$, different gradients will cooperate with each other. (b) seperates $task_1$ into two parts, one uses a dedicated layer and the other shares layer with $task_2$.

issues, we adopt a well hand-crafted network structure to divide the task into two parts, assisting the router and updater networks to realize dynamic adjustment. The distinct structures can mitigate the influence of noise on tasks. The updater dynamically aggregates these structures dynamically, guided by the output of the router.

- To introduce more personalization information, we propose the personalized gate network. The core method employs a PPNet-like structure (Chang et al. 2023), using personalized features such as user IDs. Applying this network structure to the underlying share layer of DRGrad can provide personalized gradient information for the router network, which can solve "seesaw" and "negative transfer" problems at a finer granularity.

Related Works

Well Hand-Crafted DNN. MMoE (Multi-gate Mixture-of-Experts) (Ma et al. 2018) implements a gate network for each task to alleviate the conflict. Nevertheless, there is no interaction between experts, which may bring noise and result in the absence of capturing complex information between tasks. Branched MTL (Vandenhende et al. 2019) utilizes employed tasks' affinities to build branches automatically. SNR (Ma et al. 2019) uses coding variables to control the connection between sub-networks and performs multi-level stacking of the networks, but the dynamic generation is still computation bound. PLE (Tang et al. 2020) improves the efficiency of shared learning and further solves the seesaw from the perspective of joint representation learning, while it's difficult to decouple the complicated relationships. AC-MMoE (Li and Xu 2023) applies attention and convolution to MMoE to relieve the computation bound. Nevertheless, problem of conflict between tasks still exists in the layers shared by all tasks, and it's difficult for high complexity model to convergent.

Multi-Task Weight. The weight or gradient perspective can effectively solve the aforementioned complexity problems. Nash-MTL (Navon et al. 2022) regards gradient combination as a bargaining game, and propose the Nash Bargaining Solution as a principled approach to multi-task learning. IGB (Dai, Fei, and Lu 2023) assigns task weights for improvable gap loss balancing and introduce reinforcement learning to MTL. AdaTask (Yang et al. 2023) proposes a Task-wise Adaptive learning rate approach and separate the accumulative gradients of each task so that no task would dominate the overall accumulative gradients. It improves the "seesaw" and "negative transfer" problems with lower model complexity, but may result in the loss of interactive information. The Pareto optimal solution (Lin et al. 2019) can generate sets of parameters to improve the effectiveness of all indicators, while complex calculations are difficult to implement in industrial scenarios. PCGrad (Yu et al. 2020) defines conflict by the cosine value between gradients direction and rotates the gradients of "conflict tasks" into the vertical direction. Nevertheless, it may convergent to the Pareto set rather than the optimal point. To solve this, CAGrad (Liu et al. 2021) seeks the gradient update direction by maximizing the task with the least loss reduction but ignores the overall impact of high-noise tasks. The aforementioned researches tackle the "seesaw" and "negative transfer" problems from the perspective of gradient relationships, providing a less computationally intensive implementation approach, which inspired our work. However, the minimum loss may come from the "engagement" or "business" heads, and continuously optimizing the loss of these tasks may lead to neglecting primary tasks such as "click," disregarding the differences in importance between various tasks in the business, and the impact of the high-noise tasks. Our research work focuses on addressing these issues.

The proposed DRGrad incorporates gradient operations into the model architecture, isolating primary tasks from others, which utilizes router network to maintain the original gradient information. It can route cooperative information to primary tasks without interference from the "engagement" and "business" secondary tasks.

Proposed Method

The gradient direction of different samples between MTL tasks dynamically changes during training. To leverage cooperative gradients and mitigate conflicts, we propose an end-to-end framework DRGrad, which dynamically judge the stakes between tasks.

DRGrad comprises of three core components namely Router, Updater and Personalized gate network. The router and updater networks better rectify the gradients to optimize the task's performance, and the personalized gate network is introduced to achieve personalized gradient related to users and fine-grained update of parameters. In training step t, we define the relevant quantities:

- $g_1{}'(t)$, $g_1{}''(t)$, $g_2(t)$: the gradient of $T_1{}'$, $T_1{}''$ and T_2, where $T_1{}'$, $T_1{}''$ T_2 are DNN.
- $\mu_1'(t)$, $\mu_1''(t)$: the aggregation coefficient from Updater.

As illustrated in Fig. 2, $task_1$ is the primary training task, and $task_2$ represent "engagement" and "business" heads, which can be expanded to much more tasks. To mitigate conflicts and reduce the impact of noise on the primary task, we partition task1 into two components and introduce the Router network. The Router network routes relevant gradients to the dedicated layer v_1 of $task_1$ and differentiates between conflicting and cooperative gradients from the shared

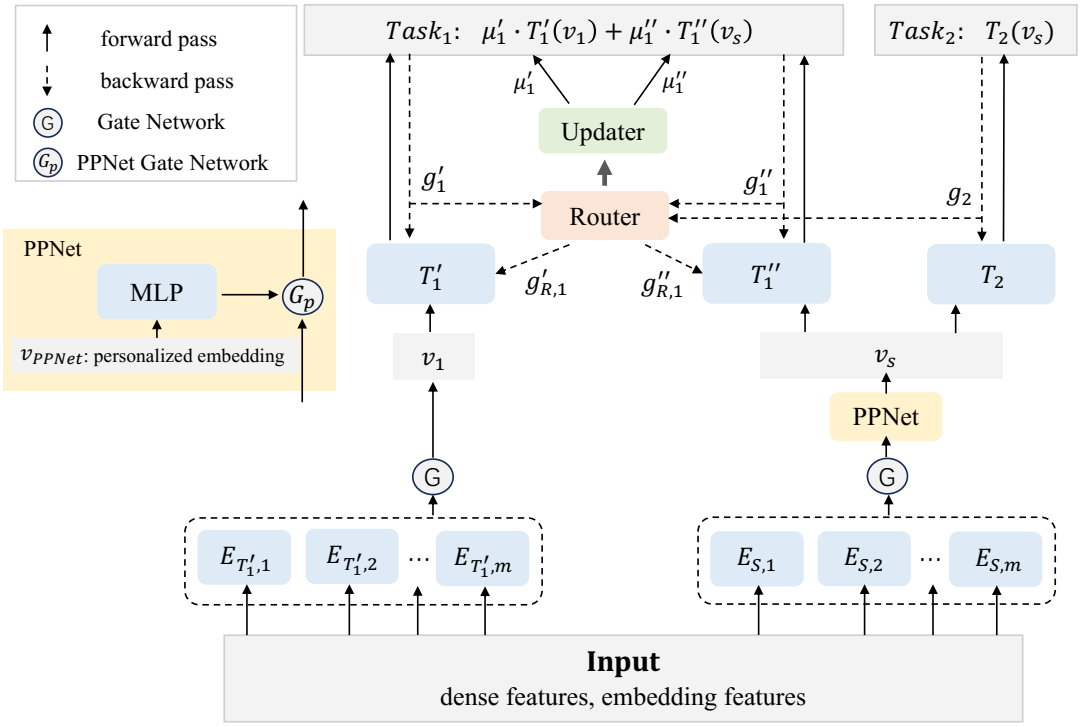

Figure 2: DRGrad model structure. The DNN tower $T_1^{'}$ takes the dedicated tensor v_1 as its input, and $T_1^{''}$ shares the same input tensor, named v_s, with T_2. $Task_1$ is aggregated by the output of $T_1^{'}$ and $T_1^{''}$, refer as $T_1^{'}(v_1)$ and $T_1^{''}(v_s)$. The Tensor v_{PPNet} is the input of PPNet, containing the personalized embedding of users. G is the Gate Network, using softmax function and G_p is Gate Network for PPNet, using sigmoid function.

Algorithm 1: Training Algorithm with DRGrad

Initialize: $\mu_1^{'}, \mu_1^{''} = 0.5, \gamma > 0$

1: **for** $t = 0$ **to** max_train_step **do**
2: Compute $Loss(t) = \Sigma_{i=1}^{n} \alpha_{task_i}$ (t) * $Loss_{task_i}(t)$
3: Compute $g_1^{'}(t)$, $g_1^{''}(t)$, $g_2(t)$
4: Compute $g_{R,1}^{'}(t)$, $g_{R,1}^{''}(t)$ by router network (Eq. 1 and Eq. 2)
5: Update $\omega_{T_1^{'}}(t)$, $\omega_{T_1^{''}}(t)$ and $\omega_{T_2}(t)$ through Eq. 4 and Eq. 5
6: Update all parameters $\omega(t)$ using $\nabla_{\omega_{(t)}} Loss(t)$
7: Compute output of updater network $\mu_1^{'}$, $\mu_1^{''}$, through Eq. 6
8: **end for**

layer v_s. The updater network collaborates with the router network to dynamically aggregate the two components of $task_1$. The personalized Gate Network employs PPNet to incorporate personalized information into the shared layer, addressing the "seesaw" and "negative transfer" problems at a finer granularity. The code is in appendix A.5

Router Network

The router network takes effect during back-propagation, which routes the gradient of other tasks to the primary task's DNN network by accessing the relationship among $g_1^{'}$, $g_1^{''}$ and g_2. When the $task_1$ is separated by Split-MMoE in Fig. 1(b), the router network will route the coupling information to the corresponding task, while preventing interference between $task_1$ and $task_2$. This approach ultimately improves the accuracy of all tasks.

Fig. 3(a) illustrates the router network, which accesses the influence relationship through the cosine similarity of the gradient. The router network defines similarity ξ_1 and ξ_2 in Eq. 1, and further calculates the adaptive weights λ_1 and λ_2, where $\| \cdot \|_2$ represents the L2 normalization of x, and γ denotes the hyperparameter.

$$\begin{aligned} \xi_1 &= \frac{g_1^{'} * g_2}{\|g_1^{'}\|_2 * \|g_2\|_2}, \lambda_1 = [clip(\frac{\| g_1^{'} \|_2}{\| g_2 \|_2}, 0, 1)]^{\gamma} \\ \xi_2 &= \frac{g_1^{'} * g_1^{''}}{\|g_1^{'}\|_2 * \|g_1^{''}\|_2}, \lambda_2 = [clip(\frac{\| g_1^{'} \|_2}{\| g_1^{''} \|_2}, 0, 1)]^{\gamma} \end{aligned} \tag{1}$$

Router's outputs $g_{R,1}^{'}$ and $g_{R,1}^{''}$, defined in Eq. 2, can provide additional gradient information from $g_1^{''}$ and g_2 for $task_1$ based on the direction relationship ξ_1 and ξ_2. $\xi_{i=1,2}$ in the router network determines the value of indicative function $\mathbf{1}_{\{\mathbf{cond}\}}$ in Eq. 3.

$$\begin{aligned} g_{R,1}^{'} &= (1 - \mathbf{1}_{\{\xi_1 < \mathbf{0}\}} * \xi_1) * \lambda_1 * g_1^{''} + \mathbf{1}_{\{\xi_2 \geq \mathbf{0}\}} * \lambda_2 * g_2, \\ g_{R,1}^{''} &= -\mathbf{1}_{\{\xi_1 * \xi_2 < \mathbf{0}\}} * \xi_1 * \xi_2 * g_1^{''} \end{aligned} \tag{2}$$

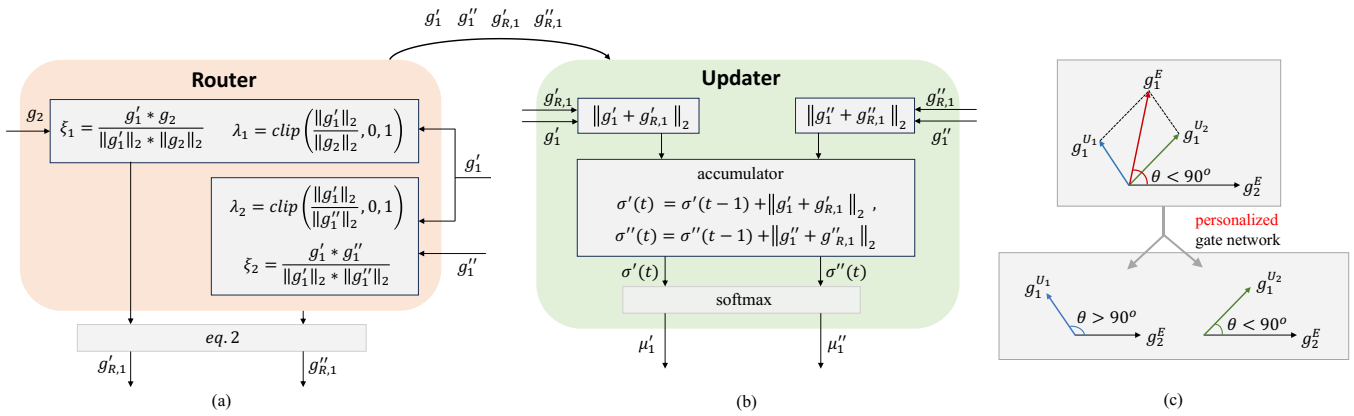

Figure 3: (a) is Router network. The gradients $g_1^{'}$, $g_1^{''}$ and g_2 are the inputs of Router Network, which come from $task_1$ and $task_2$. The processed gradients $g_{R,1}^{'}$ and $g_{R,1}^{''}$ are the outputs, used to update the parameters of $T_1^{'}$, $T_1^{''}$. (b) is Updater network. Gradient $g_1^{'}$, $g_1^{''}$, $g_{R,1}^{'}$ and $g_{R,1}^{''}$ are the inputs of Updater Network, and the outputs $\mu_1^{'}$, $\mu_1^{''}$ are used to aggregate $task_1$ dynamically. (c) is Personalized Gradients, g_1^E represents the gradient expectation of all users, $g_1^{U_1}$ represents user U_1.

$$\mathbf{1}_{\{\mathbf{cond}\}} = 1 \quad if \ \mathbf{cond} \ else \ 0 \tag{3}$$

As shown in Fig. 2, in training step t, the origin gradient $g_1^{'}(t)$ and the output $g_{R,1}^{'}(t)$ of router network are used to update the parameter $\omega_{T_1^{'}}(t)$ of $T_1^{'}$ DNN network in Eq. 4, where opt is the optimizer and η denotes learning rate.

$$\omega_{T_1^{'}}(t) \leftarrow \omega_{T_1^{'}}(t-1) - \eta * opt(g_1^{'}(t) + g_{R,1}^{'}(t)) \tag{4}$$

For $T_1^{''}$ DNN network, we use its gradient $g_1^{''}(t)$ and router's output $g_{R,1}^{''}(t)$ to update parameter $\omega_{T_1^{''}}(t)$. For T_2 DNN network, we use only its gradient $g_2(t)$ to update its parameter.

$$\begin{aligned} &\omega_{T_1^{''}}(t) \leftarrow \omega_{T_1^{''}}(t-1) - \eta * opt(g_1^{''}(t) + g_{R,1}^{''}(t)), \\ &\omega_{T_2}(t) \leftarrow \omega_{T_2}(t-1) - \eta * opt(g_2(t)) \end{aligned} \tag{5}$$

The router network is the superset of PCGrad (Yu et al. 2020), while its convergence can alse be proven. Router network will not directly rotate the gradient and damage the information, but serve as additional gradient information to promote tasks learning.

ξ_1	ξ_2	Gradient for v_1	Gradient for v_s
≥ 0	≥ 0	$g_1^{'} + \beta_1 * g_1^{''} + \beta_2 * g_2$	$g_2 + g_1^{''}$
≥ 0	< 0	$g_1^{'} + \beta_1 * g_1^{''}$	$g_2 + (1 - \xi_1 * \xi_2) * g_1^{''}$
< 0	≥ 0	$g_1^{'} + \beta_1 * (1 - \xi_1) * g_1^{''} + \beta_2 * g_2$	$g_2 + (1 - \xi_1 * \xi_2) * g_1^{''}$
< 0	< 0	$g_1^{'} + \beta_1 * (1 - \xi_1) * g_1^{''}$	$g_2 + g_1^{''}$

Table 1: Analysis of router network. According to the direction between gradients, the output will be discussed in four cases.

Table 1 presents the gradients of the upstream layers operated by the router network, where β_i is the coefficient constant. For v_s, $task_1^{''}$ shares the same vector v_s with $task_2$, which facilitates more effective information sharing between the two tasks. The routed gradients $g_{R,1}^{''}$ to $task_1^{''}$ can avoid "seesaw" between two tasks in this layer. For this layer alone, the router network is equivalent to PCGrad algorithm, however, the key difference is that DRGrad will route the gradient to the downstream DNN parameters of tasks. For v_1, the router network can assess information that contributes to $task_1{'}$ from both $task_1{''}$ and $task_2$. In addition, $task_1{'}$ and $task_1{''}$ share the same label, and the rotated $g_1^{''}$ can provide additional task fusion information.

Regarding convergence, the router network performs an incremental operation on the existing gradient. With the clip limitation $0 \leq \lambda_j \leq 1, 0 \leq E[\mathbf{1}_{\{\mathbf{X}\}}(\xi_j) * \xi_j] \leq E(\xi_j) \leq 1$, and $0 \leq E[\mathbf{1}_{\{\mathbf{X}\}}(\xi_j)] \leq 1$, according to Eq. 8, DRGrad maintains the original gradient direction, and the scale values remain bounded. Consequently, the training process is guaranteed to converge.

$$\begin{aligned} g &= g_1^{'} + g_1^{''} + g_2 + g_{R,1}^{'} + g_{R,1}^{''} \\ &= g_1^{'} + (1 + \mathbf{1}_{\{\mathbf{X}\}} * \lambda_2) * g_2 + (2 - \mathbf{1}_{\{\mathbf{X}\}} * \lambda_1 - \mathbf{1}_{\{\mathbf{X}\}} * \xi) * g_1^{''} \\ |g| &\leq |g_1^{'}| + |2 * g_2| + |2 * g_1^{''}| \end{aligned} \tag{6}$$

Updater Network

The updater network is designed to cooperate with the router network to achieve dynamic weight update for task aggregation during each training step t. Specifically, to prevent mutual influence between tasks, we divide $task_1$ into two components, which are placed in the dedicated layer and the shared layer respectively. As depicted in Fig. 3(b), to dynamically obtain the weights of two components, we employ an updater network, which generates dynamic weights, $\mu_1^{'}$ and $\mu_1^{''}$. These weights change based on the inputs and outputs of the router network and are used to update the msagnitude between $T_1^{'}(v_1)$ and $T_1^{''}(v_s)$.

The updater network updates itself during back-

propagation and takes effect during forward-propagation. In training step t, accumulated variables $\sigma^{'}(t)$, $\sigma^{''}(t)$ are updated according to the input and output of the router network, and $\mu_1^{'}(t)$, $\mu_1^{''}(t)$ are obtained by applying the softmax function to $\sigma^{'}(t)$ and $\sigma^{''}(t)$, respectively.

$$\begin{aligned}
&\sigma^{'}(t) = \sigma^{'}(t-1) + \|g_1^{'} + g_{R,1}^{'}\|_2 \\
&\sigma^{''}(t) = \sigma^{''}(t-1) + \|g_1^{''} + g_{R,1}^{''}\|_2 \\
&\mu_1^{'}(t) = \frac{e^{\sigma^{'}(t)}}{e^{\sigma^{'}(t)} + e^{\sigma^{''}(t)}}, \quad \mu_1^{''}(t) = \frac{e^{\sigma^{''}(t)}}{e^{\sigma^{''}(t)} + e^{\sigma^{''}(t)}}
\end{aligned} \tag{7}$$

The final output of $task_1$ is the weighted sum of $T_1^{'}(v_1)$ and $T_1^{''}(v_s)$ in Eq. 7, where $\mu_1^{'}$, $\mu_1^{''}$ are variables updated by the output of the updater network automatically through Eq. 6 above during the training process.

$$T_1 = \mu_1^{'} * T_1^{'}(v_1) + \mu_1^{''} * T_1^{''}(v_s) \tag{8}$$

In summary, by utilizing the input and output of the router network as the input of the updater network and accumulating the changes, the weights of each component of $task_1$ can be aggregated dynamically.

Personalized Gate Network

The two tasks share the same vector v_s, which contains information from all tasks. However, the personalized information in v_s is limited. The Personalized Gate Network introduces personalized information to the shared layer, aiming to solve the "seesaw" and "negative transfer" problems at a finer granularity. Gradients mostly represent the expected value of all users, rather than the personalized gradient for a specific user. Therefore, the implement of personalized gate network, a PPNet-like structure, can provide finer-grained personalized information for v_s. Combined with the router, personalized gate network can achieve personalized gradients. PPNet's input v_{PPNet} is consist of personalized features, such as userId, itemId and authorId. The output of personalized gate network is $v_s = 2 * v_s \otimes sigmoid(v_{PPNet} * \omega_{PPNet})$.

Multiplying the output of PPNet to the v_s can enrich personalized information in network. As shown in Fig. 3(c), g_1^E and g_2^E represent the expected value of gradients. The angle between two gradients is denoted by θ, which represents the relationship between the gradients of all users. But for individual users, the relationship between the gradients of each task may differ from the overall. So PPNet can provide personalized gradients of each user, like $g_1^{U_1}$ and $g_1^{U_2}$, which may have different angles compared to the original gradients. For each user, $g_1^{U_1}$ and $g_1^{U_2}$ are more representative of the relationship between different behaviors and items. When incorporating PPNet to v_s, it will rotate the gradients g_2 and $g_1^{''}$ towards more personalized directions. This enables the router network to obtain finer-grained personalized gradients and provide more accurate routing output.

Experiment

Baseline Models. The backbone is MMoE (Multi-gate Mixture-of-Experts) (Ma et al. 2018) with shared bottom structure, and we choose the following MTL models with different shared network architectures for comparison: SNR (Ma et al. 2019), PLE (Tang et al. 2020), AC-MMOE (Li and Xu 2023), PCGrad (Yu et al. 2020), CAGrad (Liu et al. 2021), AdaTask (Yang et al. 2023), Nash-MTL (Navon et al. 2022), and IGBv2 (Dai, Fei, and Lu 2023) algorithms, which are the same amount of parameters with DRGrad to verify the effectiveness. Experiment setup is in appendix A.1.

Evaluation and ablation Studies. We use the AUC of each task to measure the model's performance and reflect the noise processing capability. In particular, there is a correspondence between the AUC indicator and online effects. For example, the "click" task corresponds to the online CTR effect, in industrial scenarios, even a small improvement in click AUC (e.g. 0.0010) can lead to a significant increase in online CTR (e.g. 0.8%). Besides, we consider the training time and latency in online serving to reflect the model complexity. To further investigate the effectiveness of key components proposed in the DRGrad model, we design a series of ablation studies. Three variants are considered to simplify DRGrad by: 1) using Split-MMoE network only to validate its effectiveness, 2) using Split-MMoE in collaborate with the router, as shown in Fig. 1(b), to examine the effectiveness of split structure. 3) removing the personalized gate network.

Effectiveness Verification

We verify the effectiveness of the proposed DRGrad using a real-world dataset from a recommender, which consists of 15 billion daily samples collected from a real-world application.

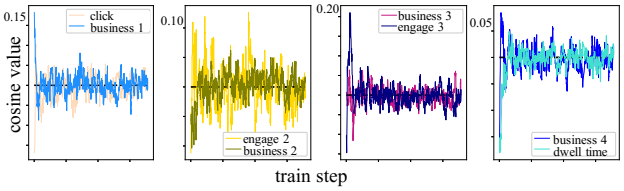

Figure 4: Grad's direction with respect to click in Fig. 1(b). The gradient direction between tasks fluctuates violently between positive and negative.

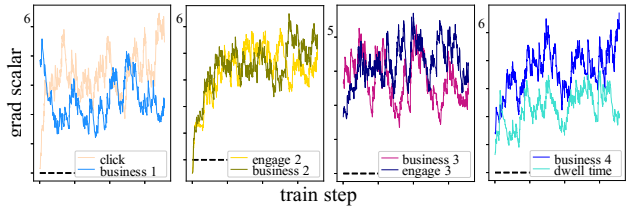

Figure 5: Grad's scalar with respect to click in Fig. 1(b). The scale of the gradient between tasks is large and the convergence trend is not obvious.

Stakes in Baseline. The changes of cosine similarity between tasks in baseline model are shown in Fig. 4. During

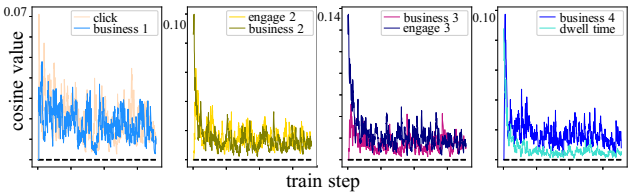

Figure 6: Grad's direction to click in DRGrad model. The direction between the gradients becomes same direction and is easier to converge.

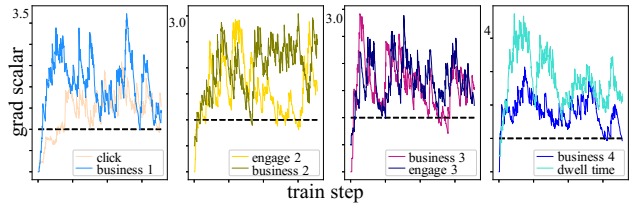

Figure 7: Grad's scalar to click in DRGrad model. The ratio between gradients becomes smaller and converges faster.

the training process, the cosine values between gradients of "click" and other tasks fluctuate significantly between positive and negative values. Although the cosine values exhibit a convergence trend, the trend for the "engagement2 task" is not evident. Therefore, each task constantly alternates between conflict and cooperation with the "click" task, so as to other tasks. Fig. 5 depicts the variation in gradient scale between tasks in the baseline model. The gradients of each task exhibit large scales and fluctuations, which can affect the gradient updates of the shared layers. In addition, the convergence trend is not apparent during the training process.

Effectiveness for DRGrad. Fig. 6 illustrates the gradient relationships between the primary task and auxiliary tasks in the DRGrad model. With the incorporation of router network, the auxiliary task positively affects the updating of the current primary task. Moreover, compared to the baseline results in Fig. 4, the convergence trend of the cosine values for each task is more pronounced towards zero, which shows that the direction of each gradient relative to the primary task gradually changes to the vertical direction. Thereby, the DRGrad model enhances the cooperation and has certain regularity for the conflict, further simplifying the complex relationships in the shared tower. In DRGrad model, the gradient scales between the primary task and others are shown in Fig. 7. Compared with the baseline results in Fig. 5, DRGrad model has apparent normative effect on the scale of the gradient, which can more effectively prevent task from being affected by other tasks with larger gradient.

Artificially Synthesized Dataset Results

The real-world dataset cannot completely decouple the cooperation and conflict between tasks. To verify the model's effectiveness in reducing conflict and enhancing cooperate between tasks, we designed a synthetic dataset in appendix A.4 with labels indicating absolute conflict or cooperation. The synthesized dataset consists of 110,000 samples, with 100,000 used for training and the remaining 10,000 for testing. The dataset contains 32 features, 6 of which are sparse. θ is the artificially direction between the task $task_1$ and secondary task $task_2$ while x denotes the input features used to generate labels. The functions $rand(a,b)$, $randint(a,b)$ represent random numbers and random integers between a and b, respectively. $N(a,b)$ represents a random value from a normal distribution with mean a and variance b. $label_1$ and $label_2$ are the labels of the primary task $task_1$ and secondary task $task_2$, respectively. To introduce conflict between the two tasks, we set $-1 < cos(\theta) < 0$. For cooperation, we set $0 < cos(\theta) < 1$.

- ith sparse feature: $x = e^{rand(0,1)*randint(1,i+2)} + rand(0,1)*randint(1,i+2)^{\frac{i}{2}+1}$
- $label_1$: $10*(\frac{4*x^2}{\|x^2\|_2}+5*e^{\frac{x}{\|x\|_2}}+6*sin(x)+N(0.01,0.002)$
- $task_2$'s label $label_2$: $cos(\theta)*label_1+N(0.01,0.002)$

Table 2 demonstrates that DRGrad achieves improvements on both tasks, with a more significant improvement on the primary task $label_1$. When the two tasks are in the same direction, DRGrad slightly increases the AUC of $label_2$ and significantly increases the AUC of $label_1$. When the two tasks are in opposite directions, DRGrad yields more substantial improvements in the AUC of both tasks. These results indicate that DRGrad can effectively alleviate conflicts while enhancing cooperation between tasks.

Real-World Dataset Results

To evaluate the effectiveness of the proposed method on real-world large-scale datasets, we chose the UCI Census-Income Dataset and a Real-World Recommendation Dataset. This allows for more reliable and easily interpretable results in actual business scenarios.

UCI Census-Income Dataset. The UCI census-income dataset is based on 1994 census data and consists of 299,285 demographic records of American adults with 40 fe atures. The tasks aim to predict whether the income exceeds \$50K and whether this person's marital status is never married. We provide the data processing method in appendix A.3. As shown in Table 3, the split structure has brought improvement in AUC, but DRGrad can improve more significantly. Since there are no personalized features like userid in the Census-income dataset, DRGrad w/o PPNet achieves state-of-the-art AUC on both tasks with absolute improvement gains of 0.0028 and 0.0004, respectively.

Real-World Recommendation Dataset. The recommendation dataset consists of 15 billion daily samples from a real-world application. There are two main tasks "click" and "dwell time", and several auxiliary tasks like "business" and "engagement heads". As shown in Table 3, DRGrad model achieves SOTA offline AUC for two main tasks "click" and "dwell time" with improvements of 0.25% and 0.12%. These improvements have a significant impact on online dwell time and Click-Through Rate (CTR). It is worth

	Cooperate, $E(cos(g_1, g_2)) \geq 0$			Conflict, $E(cos(g_1, g_2)) < 0$		
	MMoE	Split-MMoE(Fig. 1(b))	DRGrad	MMoE	Split-MMoE(Fig. 1(b))	DRGrad
$label_1$ AUC	0.9521	0.9568	<u>0.9710</u>	0.8735	0.8807	<u>0.9212</u>
$label_2$ AUC	0.9473	0.9544	<u>0.9596</u>	0.8712	0.8828	<u>0.9140</u>

Table 2: Comparison of effects on synthesized dataset. Best results are underscored. Regardless the cooperative or conflict relationship between tasks, DRGrad performs better.

	15 Billion Samples Industry Data						UCI Census-Income Data	
Method	Click AUC	Click Gain	Dwell Time AUC	Dwell Time Gain	Train Time	Latency	Task1 AUC	Task2 AUC
MMoE	0.7624	-	0.7477	-	**389min**	113ms	0.9387	0.9927
Split-MMoE(Fig. 1(b))	0.7626	0.0002	0.7481	0.0004	**394min**	114ms	0.9393	0.9928
SNR	0.7636	0.0012	0.7480	0.0003	437min	129ms	0.9519	0.9943
PLE	0.7635	0.0011	0.7480	0.0003	413min	117ms	0.9522	0.9945
AC-MMoE	0.7637	0.0013	0.7483	0.0006	453min	122ms	0.9523	0.9945
PCGrad	0.7634	0.0010	0.7479	0.0002	**391min**	113ms	0.9506	0.9931
CAGrad	0.7629	0.0005	0.7485	0.0008	402min	113ms	0.9521	0.9929
Nash-MTL	0.7635	0.0011	0.7482	0.0005	396min	114ms	0.9534	0.9946
Adatask	0.7640	0.0016	0.7483	0.0006	390min	113ms	0.9532	0.9947
IGBv2	0.7643	0.0019	0.7482	0.0005	426min	126ms	0.9529	0.9948
DRGrad (ours)	<u>0.7651</u>*	0.0027	<u>0.7493</u>*	0.0016	**395min**	113ms	<u>0.9550</u>*	<u>0.9949</u>*

Table 3: Test AUCs on real-world dataset with the best results underscored. A small improvement in click AUC (e.g. 0.0010) can lead to a significant increase in online CTR (e.g. 0.8%) while DRGrad obtains 0.25% and 0.12% absolute AUC gain for click and dwell time. * indicates the statistical significance for $p \leq 0.01$ compared with the best baseline over paired t-test.

	CTR	Dwell Time
DRGrad model	<u>1.79%</u>*	<u>0.5712%</u>*

Table 4: Online relative gains compared to MMoE). DRGrad obtains 1.79% CTR gain and 0.5712% dwell time gain for the online APP compared with MMoE.

	Click AUC	Dwell Time AUC
MMoE	0.7624	0.7477
Split-MMoE	0.7626	0.7481
Split-MMoE+router network	0.7641	0.7492
DRGrad w/o PPNet	0.7645	0.7491
DRGrad	<u>0.7651</u>	<u>0.7493</u>

Table 5: Results of ablation comparison. Three modules, router, updater, and PPNet structures, are intricately interconnected, resulting in enhanced performance outcomes.

mentioning that in the industry, an offline AUC gain of 0.1% is considered a substantial improvement and can lead to considerable online gains. Compared with well-handed structure, the perspective of gradient or weight will not increase the complexity of the model itself, resulting in almost no change in online latency. DRGrad also shares this advantage. Since the gradient calculation is introduced in the training process, it will often affect the training time by 6 minutes (vs 389 minutes). Compared with the same effect model, the training time is neutral. We conduct an online experiment which obtains the gain of 0.5712% for APP online global dwell time and 1.79% CTR gain for the application's online performance, as shown in Table 4.

Ablation comparison in Table 5 reveals that three key components, router, updater and PPNet network, significantly improve AUC besides split structure. Fig. 8(a) presents the overall loss of the baseline and DRGrad models, demonstrating that the DRGrad model is more conducive to model convergence. The AUCs of each task is shown in Fig. 8(b). The auxiliary tasks have a positive effect, while the primary task has been dramatically improved. Fig. 8(c) shows that the fine-grained routed information can alleviate the complex convergence problem, leading to lower loss for the primary task.

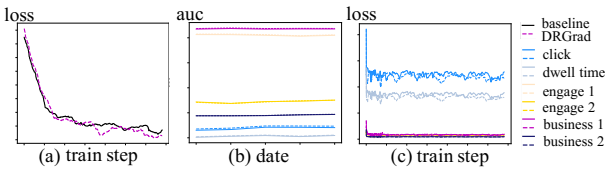

Figure 8: Comparison of loss and AUC (solid line represents baseline, dotted represents DRGrad). DRGrad's loss decreases by an average percent of 3.1 after 300,000 steps.

Conclusion

In this paper, we propose Direct Routing Gradient (DRGrad), a novel gradient routing method that effectively mitigates gradient conflicts and enhances the accuracy of Multi-Task Learning (MTL) models. DRGrad incorporates a split model structure and a personalized gate network adapting to the router network, providing regularization and personalization for the intricate information encapsulated within the shared tower. This method leads to better performance on 11 out of 14 tasks in the real-world recommendation system with billions of daily active users and gets better performance on the public Census-income and synthetic dataset compared to MMoE, SNR, PLE, AC-MMOE, PCGrad, CAGrad, AdaTask, Nash-MTL, and IGBv2 algorithms.

References

Ben-Porat, O.; Cohen, L.; Leqi, L.; Lipton, Z. C.; and Mansour, Y. 2022. Modeling attrition in recommender systems

with departing bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6072–6079.

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

Chang, J.; Zhang, C.; Hui, Y.; Leng, D.; Niu, Y.; and Song, Y. 2023. PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information. *arXiv preprint arXiv:2302.01115*.

Dai, Y.; Fei, N.; and Lu, Z. 2023. Improvable Gap Balancing for Multi-Task Learning. In Evans, R. J.; and Shpitser, I., eds., *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, 496–506. PMLR.

Fabbri, F.; Croci, M. L.; Bonchi, F.; and Castillo, C. 2022. Exposure inequality in people recommender systems: the long-term effects. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 16, 194–204.

Lakkapragada, A.; Sleiman, E.; Surabhi, S.; and Wall, D. P. 2023. Mitigating Negative Transfer in Multi-Task Learning with Exponential Moving Average Loss Weighting Strategies (Student Abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 16246–16247.

Li, J.; Li, J.; Li, J.; Zheng, H.; Liu, Y.; Lu, M.; Wu, L.; and Hu, H. 2023. ADL: Adaptive Distribution Learning Framework for Multi-Scenario CTR Prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, 1786–1790. New York, NY, USA: Association for Computing Machinery. ISBN 9781450394086.

Li, K.; and Xu, J. 2023. AC-MMOE: A Multi-gate Mixture-of-experts Model Based on Attention and Convolution. *Procedia Computer Science*, 222: 187–196. International Neural Network Society Workshop on Deep Learning Innovations and Applications (INNS DLIA 2023).

Lim, N.; Hooi, B.; Ng, S.-K.; Goh, Y. L.; Weng, R.; and Tan, R. 2022. Hierarchical multi-task graph recurrent network for next poi recommendation. In *Proceedings of the 45th international ACM SIGIR conference on Research and development in Information Retrieval*, 1133–1143.

Lin, X.; Chen, H.; Pei, C.; Sun, F.; Xiao, X.; Sun, H.; Zhang, Y.; Ou, W.; and Jiang, P. 2019. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender systems*, 20–28.

Liu, B.; Liu, X.; Jin, X.; Stone, P.; and Liu, Q. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34: 18878–18890.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 216–223.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1930–1939.

Navon, A.; Shamsian, A.; Achituve, I.; Maron, H.; Kawaguchi, K.; Chechik, G.; and Fetaya, E. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*.

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 *Proceedings of the 14th ACM Conference on Recommender Systems*, 269–278.

Vandenhende, S.; Georgoulis, S.; De Brabandere, B.; and Van Gool, L. 2019. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*.

Wang, Y.; Zhang, Y.; Valkanas, A.; Tang, R.; Ma, C.; Hao, J.; and Coates, M. 2023. Structure Aware Incremental Learning with Personalized Imitation Weights for Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 4711–4719.

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. *Advances in Neural Information Processing Systems*, 33: 5824–5836.