Multitask Ranking System for Immersive Feed and No More Clicks: A Case Study of Short-Form Video Recommendation

Qingyun Liu
Google DeepMind
Mountain View, California, USA
qyl@google.com

Zhe Zhao
Google DeepMind
Mountain View, California, USA
zhezhao@google.com

Liang Liu
Google Inc
Mountain View, California, USA
liangliu@google.com

Zhen Zhang
Google Inc
Mountain View, California, USA
zgzhen@google.com

Junjie Shan
Google Inc
Mountain View, California, USA
junjieshan@google.com

Yuening Li
Google Inc
Mountain View, California, USA
yueningl@google.com

Shuchao Bi
Google Inc
Mountain View, California, USA
shuchaobi@google.com

Lichan Hong
Google DeepMind
Mountain View, California, USA
lichan@google.com

Ed H. Chi
Google DeepMind
Mountain View, California, USA
edchi@google.com

ABSTRACT

In recent years, social media users spend significant amount of time on Short-Form Video (SFV) platforms. Its success in creating an immersive viewership experience is not only from the content, but also due to its unique UI innovation: instead of providing choices for users to click, SFV platforms actively recommend content to users to watch one at a time. In this paper, we highlight unique challenges rooted from such UI changes for SFV recommendation system design. Firstly, there is yet much unexplored for sources of system biases under the new UI, as there are no clicks nor the common click-based position biases. Additionally, when training multiple types of user activities, positive labels for activities like sharing and commenting can be much sparser and more skewed than traditional click-based recommendation systems, as the latter can filter non-click impressions when generating "post-click" activities.

To tackle these challenges, we introduce a unified multi-task ranking framework which puts two novel components all together into an overall system for SFV recommendation. First, we identify that there are position biases of SFVs in the recommendation sequence, namely "watch trail biases", and introduce biases correction using trail-related information. Second, to get the most benefits from multi-task learning, especially co-training tasks with extremely skewed and sparse labels, we adapt a disentangle regularization to mitigate task conflicts, introduce loss upweighting for sparse task co-training and adopt a meta-learning algorithm for efficient weight selection. We demonstrate the effectiveness and efficiency of the framework on one of today's largest SFV platforms. Our framework has been deployed to the production system for more than 6 months.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615489

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Recommender systems**; • **Computing methodologies** → **Multi-task learning**.

KEYWORDS

Short-form Video; Recommender Systems; Multi-task Learning

ACM Reference Format:
Qingyun Liu, Zhe Zhao, Liang Liu, Zhen Zhang, Junjie Shan, Yuening Li, Shuchao Bi, Lichan Hong, and Ed H. Chi. 2023. Multitask Ranking System for Immersive Feed and No More Clicks: A Case Study of Short-Form Video Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583780.3615489

1 INTRODUCTION

Recent years have witnessed the evolution of online recommendation from information filtering to multi-stage deep learning based retrieval and ranking systems [13, 18, 19, 45, 51]. While many of the innovations target at challenges from scalability and efficiency on user generated content [13, 16, 17, 31], we want to spotlight one recent innovation in the UI and viewership experience called "immersive feed". Instead of providing different options for users to choose from (*e.g.* by clicking), Short-Form Video (SFV, videos with less than one minute length) platforms often actively recommend content to users to watch one at a time [47]. It takes extremely low cost for users to interact with the item, *e.g.* loop, swipe, do engagements such as like, share, comment. With such engaging experience for users, SFV platforms have risen to be the latest social media stars, *e.g.* YouTube Shorts, TikTok, Instagram Reels.

The innovative immersive feed posts unique challenges to the recommendation system design. The quality for users' lean-back experience (*i.e.* users receive information in a passive way) relies heavily on the system's ability to capture user interests and provide satisfying recommendations. Specifically, recommendation

systems often have biases as models are trained with user logs generated from the existing systems [18, 47], and such biases are actually training data biases needed to be account for. It is essential to understand sources for system biases in SFVs as there are no click-based position biases [21], a common cause in traditional recommendation systems. Additionally, when capturing different types of user behaviors, positive labels for behaviors like commenting and sharing are more sparse and skewed than traditional click-based recommendation systems. This is because click-based systems can filter non-click impressions when generating "post-click" activities [6, 12, 34]. For example, jointly learning from users' watching and commenting signals can be quite challenging, as comments happen on average less than once per thousand videos watches, which can introduce severe task conflicts in multitask learning.

In this paper, we focus on the ranking stage of recommendation systems which sorts a few hundred candidates generated by the previous retrieval stage. To meet the above mentioned challenges, we introduce an efficient architecture based on the widely adopted multi-task learning (MTL) framework [12, 41, 51] that naturally captures different user behaviors. Specifically, an MTL ranking model have multiple prediction tasks and each task is to predict one type of user behaviors, *e.g.* watches, comments, likes. MTL optimizes multiple tasks simultaneously. Figure 1 shows our MTL framework for SFV ranking where user logs are processed as training data. Input features are consumed by a shared bottom layer, followed by Multi-gate Mixture-of-Experts (MMoE) [18, 33] module. MMoE shares multiple expert sub-models across different tasks and trains a gating network for each task to selectively combine the experts. Outputs from MMoE are consumed by task-specific parameters (*e.g.* MLP) to produce task predictions, which are compared to labels extracted from user behavior signals. We highlight two novel components in this MTL framework:

Biases correction (red components in Figure 1). We identify the existence of watch trail biases in SFV systems, and alleviate them by applying a learnt bias term for each task as a normalizer or regularizer to the main model.

Task conflicts reduction (blue components). Facing tasks with extremely skewed and sparse distributions in SFVs, we first adapt a disentangle-based regularization [20] to MMoE for overall model generalization. We then overcome degraded sparse task learning with task importance upweighting. Finally, we efficiently select weights for sparse tasks based on a meta-learning algorithm.

We summarize our contributions as follows:

- We identify recommendation system design challenges created from the UI innovation of SFVs, *i.e.* an immersive feed without clicks. By actively recommending one SFV at a time instead of providing many options, it introduces challenges like new system biases and more severe task conflicts compared to traditional click-based recommendation systems.
- We conduct analysis to study biases created by the new UI design, and identify the existence of biases in user watch trails. We model such watch trail biases as a learnt bias term to be applied to the main model.
- When developing a MTL framework that jointly learns different types of user behaviors, we focus on co-training scenarios with both dense and extremely sparse tasks. Specifically, we introduce disentangle regularization to mitigate general task conflicts, apply task upweighting to improve sparse task performance during co-training, and adopt a meta learning based strategy for efficient sparse task weight selection.
- We conduct extensive offline and live experiments to show the effectiveness and efficiency of the framework, on one of today's largest SFV recommendation system. Our framework (as shown in Figure 1) has been deployed to production for more than 6 months.

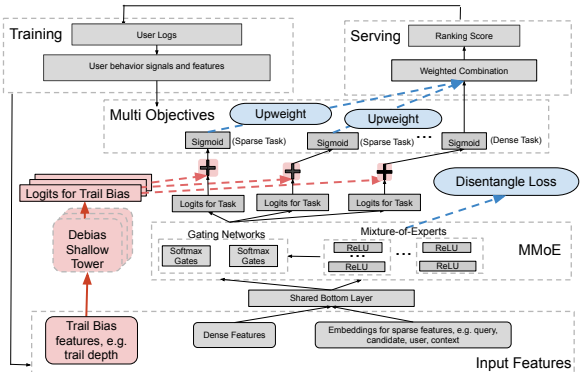

Figure 1: Architecture of the multi-task learning framework for SFV ranking, with novel components on watch trail biases correction (red), and task conflicts reduction (blue).

2 RELATED WORK

2.1 Industrial Recommendation Systems

Many industry-scale systems adopt multi-stage recommendations where a ranking stage follows a candidate retrieval stage, such as YouTube [12], Google Play [45], Facebook [19], Pinterest [13]. Multitask learning (MTL) has been widely adopted in ranking systems [12, 18, 41, 51] as it efficiently optimizes different objectives by training a single model [7], and MMoE is a popular modeling choice [18, 33, 51] that helps mitigate task conflicts.

With the growing attention on SFV recommendations, despite some potential ethical concerns [36], SFVs are reported to provide significant benefits and support in various areas such as socializing [32], emotional support [32], book reading [37], tourism [29], and education [49]. Specifically, [28] designs a recommendation system for relevant SFVs after users clicking like buttons. [5, 6] adopt Reinforcement Learning techniques to optimize goals such as user retention or watch-time. [17] develops a real-time reranking system for server-side recommendation. Here we highlight challenges introduced by SFV systems' unique UI design, *i.e.* providing users one SFV at a time instead of making them choose from many options, resulting in no clicks. We target at optimizing users' overall experience instead of single metric.

Selection biases often exist in industrial-level systems [18, 47, 51] with the wide adoption of implicit feedbacks as their training data. Position biases [1, 21, 43] are most commonly discovered and conventionally click based [21, 34]: items ranked at top are more likely to be clicked and treated as of good quality in training, regardless of their actual qualities. For SFVs, [47] identifies video duration biases and proposes Duration-Deconfounded Quantile-based (D2Q) watch-time prediction, while we examine whether position biases still exist without any clicks. Though propensity-weighted methods are popular to correct for biased distribution [43,

46], for real-world recommendation systems we adapt an efficient strategy from [51] to fast adjust to data distribution changes.

2.2 Task Conflicts and Multitask Weights

As in SFV modeling tasks may have extremely skewed and sparse distributions, we study existing work related to task conflicts [40] and deep learning with class imbalance [22]. To reduce task conflicts that often cause degraded performance, one type of solution is using flexible model architectures, *e.g.* MMoE [33], attention-based [30] design, under-parameterized self-auxiliaries [44]. Others focus on the optimization process, *e.g.* drop conflict gradient [11], look ahead to capture task interactions [14]. We adapt a decorrelation regularization technique [20] in a light-weighted way on top of MMoE to meet efficiency challenges in industrial applications.

For imbalance learning [22], one main remediation is on data-level to process data distributions and decrease the imbalance level, *e.g.* re-sampling [3, 42], transfer learning [26]. Another is on algorithm-level to modify the learning or decision process and focus more on the minority class, *e.g.* cost-sensitive learning that adjusts the class-specific weight [25, 48], applying new loss functions [27, 39] like Focal Loss [27] to focus more on hard examples. The novel challenge we face in SFV systems is the extremely imbalance distributions under MTL set-up with large-scale data.

Another related area is multitask weighting as MTL usually combines tasks by a weighted sum of losses. Weights can be adjusted adaptively based on predefined heuristics like gradient descent, *e.g.* Gradnorm [10], Uncertainty Reweighting [24]. Besides, Pareto Optimization based methods cast MTL as multi-objective optimization [38]. Recently, meta learning [15, 50] is applied to task weighting, *e.g.* [35] optimizes the generalization performance estimated by the test losses. Differently, we target at extremely sparse task scenarios, and need algorithms to be efficient for large-scale data, *e.g.* learn weights additively without retraining the whole model.

3 SYSTEM OVERVIEW

We follow a similar ranking problem definition and multi-task learning (MTL) framework as [12, 18, 51], where a ranking system sorts a few hundred candidate videos by their utility to users. To estimate a candidate utility, we apply MTL to model multiple objectives, or "tasks", to predict different user behaviors simultaneously, *e.g.* watches, comments, likes. Specifically, tasks can be classification problems, *e.g.* whether a user comments on a SFV, or regression problems, *e.g.* how long a user watches a SFV. Then these predicted user behaviors will be merged by a manually tuned or learned combination function to user utility towards the candidate.

We generate training data from user logs as shown in Figure 1. Models are sequentially trained to consume data from earlier days to recent days chronologically, while data samples from the same day are randomly shuffled. Input features can be related to the candidate (*e.g.* video information and statistics), the query (*e.g.* user's watch history), and the context (*e.g.* devices, user demographics). Outputs are the predictions on different tasks such as watches, comments, likes. Those predictions are then combined in weighted multiplication as the final ranking score for each candidate video in serving, where such weights can be manually tuned or learnt. Our system adopts point-wise losses [4] due to efficiency concern at serving time.

4 MODELING TRAIL BIASES

We identify the existence of watch trail biases and propose a simple yet effective bias correction method to mitigate the issue.

4.1 Identify Watch Trail Biases

To explore biases introduced by SFV's innovative UI design, *i.e.* immersive feed, we focus on position biases, which are most common for implicit feedback training [21, 47, 51]. While position biases are conventionally click based [21, 34], there are no clicks in SFVs as we only present one item at a time to the user sequentially. Instead, a new concept of "watch trail" is introduced: consecutive video watches connected by the user's swiping ups and downs, from the very first watch (*i.e.* start of the trail) till the user quits the platform.

To evaluate trail biases, we conduct analysis of user behaviors on trail positions with randomly promoted SFV recommendations. Figure 2 shows relative movement of three types of user behaviors (*e.g.* looping/swiping) for a certain trail position to the first trail position. A value close to 1 means user behavior doesn't change by trail positions, <1 means users interacting more at the beginning of the trail, and vice versa. We see that relative metrics keep deviating from 1 along the trail, which proves the existence of trail biases.

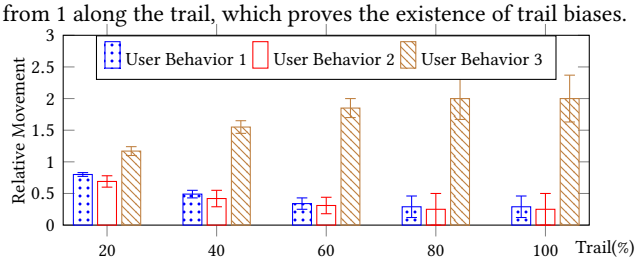

Figure 2: Relative movement of different user behaviors at trail position x% to the 1st trail position for randomly promoted SFVs, which keeps deviating from 1 along the trail.

4.2 Modeling Trail Biases

In the context of a real-world recommendation system, the approach for modeling biases should be efficient for large-scale data and effective in fast adaptation to changing user data distributions. Inspired by how [51] models click biases, we model trail biases directly from training data and apply the bias terms as regularizers to the main model.

The high level idea is to have the model learn a calibration factor that offsets biases based on features related to trail biases. This allows the model to focus on a fair comparison of candidate qualities. As shown in the red components in Figure 1, given a task, the bias term is learnt through a shallow tower and then added to the task logits to serve final predictions. The inputs are trail bias related features which can be non-serving, *i.e.*, features treated as missing at serving time. We apply a 10% drop-out for all those features during training to alleviate training-serving skewness.

5 MITIGATING SFV TASK CONFLICTS

To capture various user behaviors in SFV, we co-train tens of tasks. Unique challenges include more severe task conflicts with extremely

skewed user behavior signals, and degraded sparse task quality when co-training with dense tasks. To tackle these challenges, we first introduce a regularization loss to disentangle the representations of each expert in our MMoE structure. Then we empirically find that upweighting the losses of sparse tasks can improve sparse task performance without hurting (too much) on dense tasks. At last, we adopt a simple meta learning algorithm to learn the optimal weight of the sparse tasks.

5.1 Disentangle for Multitask Learning

Compared with training single tasks, multitask learning (MTL) [18, 41, 51] becomes the choice for many industry systems as it improves learning efficiency by sharing parameters among tasks. MTL is also expected to improve generalization of tasks with inductive transfer. However, MTL in SFVs faces additional challenges compared to traditional click-based recommendations. First, as SFVs are built for users' lean-back experience, user behavior signals can be extremely skewed, which might hurt generalization of some tasks. Figure 3 shows the distribution for an example regression task, whose skewness is intensified by looping (*i.e.* one SFV is automatically repeated unless users swipe or engage). Second, with an increasing number of tasks, the risks of task conflicts rise. It also increases potential spurious correlation between features and tasks according to a recent research [20], where a causal feature for task A might be spurious for task B.

To improve overall MTL generalization, one intuition is to make the shared latent representation space more structured. We adapt the Multi-Task Causal Representation Learning (MT-CRL) [20] by applying a disentangle regularization alone over the shared MMoE structure. It aims to make latent expert representations more independent of each other, *i.e.* more disentangled. Specifically, for an MMoE structure where Z_i denotes the latent representation for expert i, we first calculate the inter-latent representations' Pearson correlation ρ between i and other experts, say expert j:

$$\rho(Z_i, Z_j) = \frac{Cov(Z_i, Z_j)}{\sqrt{Cov(Z_i, Z_i) \cdot Cov(Z_j, Z_j))}}$$

where $Cov(Z_i, Z_j) = [Z_i - \overline{Z}_i]^T [Z_j - \overline{Z}_j]$. We then regularize such correlation by minimizing its Frobenius norm and treat the regularization as an additional loss L_{reg} to be added to the final loss calculation L:

$$L = L_{model} + \lambda \cdot L_{reg} = L_{model} + \lambda \cdot \sum_{i<j} ||\rho(Z_i, Z_j)||_F^2$$

L_{model} denotes the groundtruth loss (*i.e.* loss between predictions and labels). We tune a multiplier λ for L_{reg} to decide its importance.

5.2 Upweighting Sparse Task Losses

Another challenge for SFV ranking comes from training dense and extremely sparse tasks together. Specifically, consider a binary classification task with labels $\{0, 1\}$. *e.g.* to model how likely a user likes/dislikes a SFV, given a (user, item) pair, assign 1 if the user liked/disliked this item and 0 vice versa. Label mean is the count of 1 over the count of all examples. Dense tasks usually have balanced label mean ($\sim$ 0.5). Sparse tasks have much smaller label mean, *i.e.*, much fewer positives compared to negatives. Specifically, label mean for sparse tasks in SFVs usually $< 1e^{-2}$, and can be extremely small to the scale of $1e^{-4}$ or $1e^{-5}$. Compared to traditional click-based recommendations, negative labels can be filtered by click impressions and such "post-click" tasks are usually denser. Co-training them is inherently challenging as gradient updates from sparse tasks can be swamped by dense tasks on the shared layers [6].

As mentioned in Sec 2.2, there are many imbalance learning (IL) algorithms and it is costly to explore all on real-world datasets. To choose candidate algorithms that work for extremely sparse scenario, we construct a synthetic dataset for empirical study. Inspired by [23, 33, 42], we generate two tasks in a MTL set-up with one dense and one sparse task (label mean ranging from $1e^{-6}$ to $1e^{-2}$). We follow Sec 2.2 to cover popular algorithms from each IL category: *e.g.* sampling [3], cost-sensitive learning [25], applying new losses. We generate 1M examples and multiple runs for each algorithm, and show the top algorithm for each category in boosting sparse task performance in Figure 4. We omit results for dense task as the impact are negligible, and discuss their real world impact in Sec 6.3.2. We find that: (1) without applying IL strategies (black triangle), sparse task suffers from co-training and the more sparse, the more suffering; (2) upweighting sparse task losses (red dot) generates best performance.

Formally, given an existing task set $T_{exist} = \{t_1, t_2, ...\}$ in a MTL set-up, T_{sparse} (T_{dense}) contain all sparse (dense) tasks in T_{exist}. With sparse task upweighting, the groundtruth loss L_{model} is:

$$L_{model} = L_{dense} + L_{sparse} = \sum_{t_i \in T_{dense}} L_i + \sum_{t_i \in T_{sparse}} w_i L_i$$

L_i is loss for task t_i, and $w_i > 1$ is t_i's loss weight if $t_i \in T_{sparse}$.

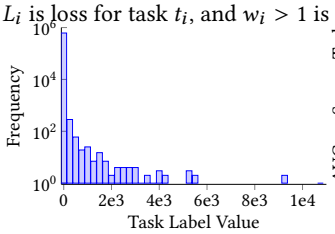

Figure 3: Extremely skewed distribution for an example regression task.

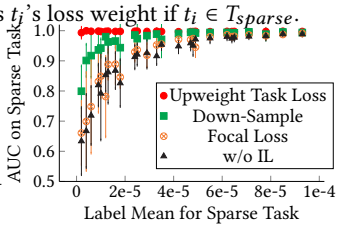

Figure 4: Compare imbalance learning (IL) algorithms on a synthetic dataset.

5.3 Meta Learning for Weight Selection

Finding proper weights for sparse tasks is non-trivial due to trade-off between sparse tasks vs. dense tasks. Existing research [24, 38] that tries to learn weights of task losses in a MTL setup might not be directly applicable to extremely imbalance task co-training. To avoid grid-search which is not realistic on large-scale recommendations, we adopt a simple meta learning strategy to select proper weights for new sparse tasks using limited data points [15, 35].

With T_{exist} and a new sparse task t_{new}, we train a meta learner G to select loss weight w_{new} for t_{new} from a given weight set W. We define the combined metrics $M(w)$ for co-training T_{exist} and t_{new} given a specific $w \in W$ used for upweighting t_{new} in training:

$$M(w) = \sum_{t_i \in \{t_{new}\} \cup T_{exist}} \alpha_i \cdot M_{t_i}(w)$$

$M_{t_i}(w)$ is the metric for task t_i, *e.g.* AUC [2], given w as loss weight for t_{new} in co-training. Note that using different w can affect the metrics significantly for not only t_{new} but also T_{exist}, due to task conflicts. α_i is the importance for t_i to emphasize on different tasks.

For simplicity, we use 1.0 for our experiments. G predicts the combined metrics as $\widetilde{M}$, with feature vector f and parameters θ_G:

$$\widetilde{M}(w) = G(f(t_{new}), f(T_{exist}), w; \theta_G)$$

G's features include per-task gradients from each layer of the co-training model, task label means and loss weights, and their crosses. We use a simple MLP with ReLU layers of size [256, 1] for G, and θ_G are learnt through back propagation of minimizing losses between M and $\widetilde{M}$. To generate training examples for G, which is a pair of new sparse task and existing tasks, we enumerate each sparse task from T_{exist} with different upweights, and the rest in T_{exist}. After G is trained, we choose w_{new} that optimizes G's predictions:

$$w_{new} = \arg\max_{w \in W} G$$

For each t_{new} we repeat 10 runs and select final w_{new} by majority vote for better robustness.

6 EXPERIMENTS

We describe how we evaluate the ranking framework through both offline and live experiments on one of today's largest SFV platforms.

6.1 Experiment Setup and Evaluation Metrics

Our model is built upon a real-world SFV platform with tens of billions of user interactions on a corpus of millions of items. It trains tens of tasks simultaneously, which label mean can be as sparse as $1e^{-4}$ or $1e^{-5}$. We use TFRS[1] to build models and Tensor Processing Units (TPUs) for training. Similar to the set-ups in other industrial recommendation systems [8, 12, 45], we search for optimal hyper-parameters (*e.g.* learning rate, batch size) by live experiments. The proposed multi-task learning (MTL) framework in Figure 1 has been deployed to the production system for more than 6 months.

We conducted extensive offline and live experiments to evaluate the MTL framework, especially for the novel components on trail biases correction, and task conflicts reduction. For offline experiments, we measure model quality by AUC for classification tasks and RMSE for regression tasks. For live experiments, we carry out A/B tests where a portion of real user traffic is diverted to both control and treatment models for over 2 weeks. To measure user performance, we focus on two types of live metrics:

- **Overall Enjoyment** for users on the platform [8, 9].
- **Task-specific metrics** for certain type of user behaviors, *e.g.*, number of user likes/dislikes.

For all tables in this paper, we use **bold** numbers to denote the best result for a given metric, and * for numbers significant with 95% Confidence Interval (CI).

6.2 Modeling Trail Biases

To evaluate the impact of modeling trail biases, we compare:

Control: the MTL-based model in Figure 1 without trail biases correction (*i.e.* without red components).

Treatment: the Control model adding trail biases correction technique described in Sec 4.2. The debias tower can be added to a single task, or multiple tasks with shared embeddings for trail biases related features.

[1]Tensorflow Recommenders: https://www.tensorflow.org/recommenders

Table 1 demonstrates live results for Treatment models against the Control (offline metrics omitted for brevity). As debiasing for all tasks outperforms other combination of multiple tasks, we only show results for all tasks. We see that modeling trail biases boosts model performance even when it is applied to a single task, while applied to all tasks generates the highest gain.

Applied Task	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	All Tasks
Overall Enjoyment	0.39%*	0.42%*	0.81%*	0.16%	0.84%*	0.30%	**1.96%***

Table 1: Live metrics (the higher the better) on applying debias tower to different tasks, vs. the model w/o debiasing.[2]

6.3 Mitigating SFV Task Conflicts

We evaluate each technique to improve MTL generalization individually: disentangle regularization, upweighting sparse tasks in co-training, and meta-learning for efficient weight selection.

6.3.1 Disentangle for Multi-task Learning. We compare between:

Control: the MTL-based model in Figure 1 without task conflicts reduction (*i.e.* without blue components)

Treatment: the Control model only adding the disentangle regularization on MMoE, as described in Sec 5.1.

Table 2 shows live results for Treatment models against the Control, with tuning the multiplier λ for regularization loss. Besides Overall Enjoyment, we also add Regression Task Metric (for the task mentioned in Figure 3) and Classification Task Metric (*e.g.* number of user likes/dislikes) for comparison in a finer granularity. We see disentangle regularization boosts overall performance as well as task-specific metrics. λ is not overly sensitive to Overall Enjoyment, as long as it is kept within a reasonable range like $0.05 \sim 1$.

Loss Multiplier λ	0.001	0.01	0.05	0.5	1.0	10.0
Overall Enjoyment	0.15%*	0.18%*	**0.33%***	0.30%*	0.29%*	0.19%*
Regression Task Metric	0.47%*	0.35%*	**0.67%***	0.21%*	0.49%*	0.29%*
Classification Task Metric	0.99%*	0.77%*	1.09%*	**1.32%***	0.39%*	0.79%*

Table 2: Live metrics (the higher the better) on adding disentangle loss to the MMoE structure, vs. the model w/o disentangle loss.

6.3.2 Upweighting Sparse Task Losses. For both Control and Treatment models below, we adopt the same MMoE structure as the Treatment models in Table 2, which has disentangled loss added:

Control (Separate Training): train sparse tasks separately upon input features as shown in Figure 5(a). Each sparse task has its own task tower (with stop gradient to prevent negative transferring to embedding layers), and only dense tasks share a bottom layer and the MMoE structure.

Treatment (Co-Training): co-train sparse and dense tasks with the shared bottom layer and the MMoE structure as shown in Figure 5(b). Besides the treatment that applies no imbalance learning (IL) on sparse tasks, we explore promising IL strategies from Sec 5.2, including (1) **Sampling**: down-sample/up-sample to each sparse task separately. We experimented with random sampling, sampling based on task metrics, different sampling ratios, and achieved best results from negative down-sampling [12, 31] with random samples till each task becomes a balanced class. (2) **Upweighting**: upweight the loss for each sparse task in L_{model} as described in Sec 5.2. We

[2]Notations for all tables in this paper: bold numbers are the best results for a metric, and numbers with * are significant with 95% Confidence Interval in live experiments.

Weight Selection Strategy	T_{new} label mean 1e-3					T_{new} label mean 1e-4				
	AUC(t_{new})	$\overline{AUC}$(T$_{exist}$)	Overall Enjoyment	t_{new} Metric	T_{exist} Metric	AUC(t_{new})	$\overline{AUC}$(T$_{exist}$)	Overall Enjoyment	t_{new} Metric	T_{exist} Metric
Handpick (100% data)	0.795	**0.792**	-	-	-	0.764	**0.795**	-	-	-
US	0.772	0.791	-0.16%*	-3.80%*	-0.49%*	0.744	0.794	-0.16%*	-1.60%*	-0.49%*
UR	0.790	0.790	-0.50%*	1.93%*	-1.49%*	0.754	0.793	-0.50%*	-0.10%*	-1.49%*
Ours (5% data)	0.772	0.791	-0.16%	-3.80%	-0.49%	0.744	0.794	-0.16%	-1.60%	-0.49%
Ours (10% data)	**0.806**	0.791	-0.08%*	**2.87%***	-0.20%*	**0.787**	0.794	-0.23%	**3.91%**	-0.52%
Ours (20% data)	0.795	**0.792**	**0.00%***	0.00%*	**0.00%***	0.764	**0.795**	**0.00%***	0.00%*	**0.00%***

Table 3: Offline and live metrics (the higher the better) on different weight selection strategies to add sparse task t_{new} to the existing co-training set-up with task set T_{exist}. $\overline{AUC}$ is the averaged AUC for a task set. Live metrics are compared against Handpick (see Sec 6.3.3 for details). Handpick and Ours are noted with the amount of user log data used in weight selection.

grid searched a unified upweight value for all sparse tasks due to limited training budgets, which worked well in practice.

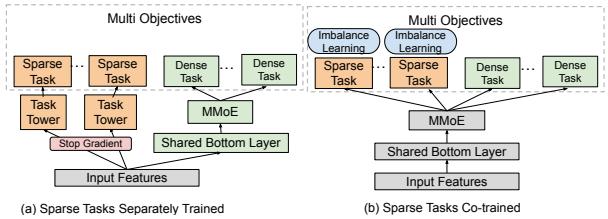

Figure 5: Different architectures for sparse task training in MTL. (a) trains sparse tasks separately from dense tasks (Control), with only input features shared. (b) co-trains sparse tasks with dense tasks (Treatment), with most layers shared and imbalance learning applied to sparse tasks only.

Table 4 presents live results for Treatment models against the Control. Besides Overall Enjoyment, we present task-specific metrics for sparse tasks, *e.g.* number of user likes/dislikes, and metrics for dense tasks, *e.g.* rate of users completing most of the videos. We have the following observations: (1) Compared to separately training sparse tasks, co-training sparse and dense tasks without applying IL techniques greatly degrades model quality. This is consistent with other observations [6]; (2) Co-training with sparse task upweighting achieves overall better performance than separately training sparse tasks; (3) Increasing upweight value tends to improve sparse task metrics while hurting dense task metrics. A good trade-off happens around upweight 50. We also find that by parameter sharing of co-training, model parameters reduce by 60%.

Imbalance Learning (IL)	Overall Enjoyment	Task 1 Metric (Sparse)	Task 2 Metric (Sparse)	Task 3 Metric (Dense)
No IL Applied	-0.44%*	-7.14%*	-9.92%*	**1.31%***
Sampling	-0.01%	-7.46%*	-3.24%*	0.08%*
Upweighting 10	0.05%	-2.96%*	-1.01%*	0.53%*
Upweighting 20	0.12%*	-2.72%*	-0.09%	0.38%*
Upweighting 50	**0.29%***	0.78%*	3.07%*	-0.11%*
Upweighting 100	0.24%*	2.26%*	3.40%*	-0.21%*
Upweighting 1000	-0.63%*	**6.61%***	**6.96%***	-1.13%*

Table 4: Live metrics (the higher the better) on applying different IL strategies on the sparse task co-trained set-up (Figure 5(b)), vs. the sparse task separately trained (Figure 5(a)).

6.3.3 Meta Learning for Weight Selection. Despite considerable model improvement, there are limitations for the upweighting technique in Sec 6.3.2. First, it applies a unified weight that cannot be customized for different sparse tasks. Second, weight selection by grid searching from live experiments is expensive and takes long to iterate. To investigate efficient weight selection, we follow Sec 5.3 to formally define the problem as: given a co-train set-up as Figure 5(b), an existing task set T_{exist}, a new sparse task t_{new} to be added, select a weight $w_{new} \in [1, 1000]$ that optimizes the overall model performance. We compare the following strategies:

Uniform Scaling (US) [38]: t_{new} is always assigned a uniform weight 1.0 as all other tasks.

Uncertainty Reweighting (UR) [24]: a popular approach to learn multitask weights simultaneously by using homoscedastic task uncertainty (*i.e.* task-dependent uncertainty). w_{new} is learnt instead of chosen from W.

Handpick (Control): grid search w_{new} by live metrics, *i.e.* Overall Enjoyment and task-specific metrics.

Meta Learner (Ours): train G on T_{exist} and choose w_{new} that optimizes G's prediction of combined metrics $\widetilde{M}$ (Sec 5.3).

For each of the above mentioned method, we train a separate model with its selected w_{new} on 100% user logs for offline/live experiment comparison. For w_{new} selection, while no additional training for US and UR, Handpick needs models trained for each $w \in W$ on 100% user logs. Though Ours method also needs multiple models to collect training data for G, it requires only a portion of user logs (as offline metrics are more robust to the usage of user logs) with several w values. We report the usage of user logs in w_{new} selection for Ours and Handpick for efficiency comparison.

We present results on two sparse tasks with different label means in Table 3, and omit similar results for other tasks. We compare AUC for offline metrics, and report live metrics compared against Handpick, as it is supposed to have the best live performance by definition. Live metrics include Overall Enjoyment, and task-specific metrics. t_{new} Metric evaluates t_{new} and varies for the specific task, *e.g.* number of user likes/dislikes. T_{exist} Metric evaluates existing tasks and can be consistent across different t_{new} for fair comparison, *e.g.* metrics from dense tasks like the rate of users completing most of the SFVs. We have consistent observations: (1) Ours can achieve comparable live performance with Handpick; (2) Compared to Handpick, Ours determines weights more efficiently, *e.g.* using 20% data and without enumerating on every $w \in W$. Without relying on all the training data nor numerous live experiments for weight selection, it saves iteration time from days to weeks.

7 CONCLUSION

We focus on unique challenges in SFV recommendation with the absence of clicks and provide solutions for optimizing user engagements. Specifically, we identified the existence of watch trail biases, and severe task conflicts when co-training extremely sparse and skewed tasks. We introduced two novel components to a MMoE based MTL framework to mitigate trail biases and task conflicts in a light-weighted yet effective way. We evaluated our framework on a real-world large dataset and launched to production.

REFERENCES

[1] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 474–482.

[2] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* 30, 7 (1997), 1145–1159.

[3] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks* 106 (2018), 249–259.

[4] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*. 89–96.

[5] Qingpeng Cai, Shuchang Liu, Xueliang Wang, Tianyou Zuo, Wentao Xie, Bin Yang, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Reinforcing User Retention in a Billion Scale Short Video Recommender System. *arXiv preprint arXiv:2302.01724* (2023).

[6] Qingpeng Cai, Zhenghai Xue, Chi Zhang, Wanqi Xue, Shuchang Liu, Ruohan Zhan, Xueliang Wang, Tianyou Zuo, Wentao Xie, Dong Zheng, et al. 2023. Two-Stage Constrained Actor-Critic for Short Video Recommendation. *arXiv preprint arXiv:2302.01680* (2023).

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

[8] Bo Chang, Alexandros Karatzoglou, Yuyan Wang, Can Xu, Ed H Chi, and Minmin Chen. 2023. Latent User Intent Modeling for Sequential Recommenders. In *Proceedings of the web conference 2023 Industrial Track*.

[9] Bo Chang, Can Xu, Matthieu Lê, Jingchen Feng, Ya Le, Sriraj Badam, Ed Chi, and Minmin Chen. 2022. Recency Dropout for Recurrent Recommender Systems. *arXiv preprint arXiv:2201.11016* (2022).

[10] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*. PMLR, 794–803.

[11] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems* 33, 2039–2050.

[12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[13] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 27th International Conference on World Wide Web*. 1775–1784.

[14] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. 2021. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems* 34, 27503–27516.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* (2017).

[16] Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. 2021. Deep Retrieval: Learning A Retrievable Structure for Large-Scale Recommendations. *Proceedings of the 30th ACM International Conference on Information Knowledge Management (CIKM)*, 524–533.

[17] Xudong Gong, Qinlin Feng, Yuan Zhang, Jiangling Qin, Weijie Ding, Biao Li, Peng Jiang, and Kun Gai. 2022. Real-time Short Video Recommendation on Mobile Devices. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3103–3112.

[18] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep multifaceted transformers for multi-objective ranking in large-scale e-commerce recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2493–2500.

[19] Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim Hazelwood, Mark Hempstead, Bill Jia, et al. 2020. The architectural implications of facebook's dnn-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 488–501.

[20] Ziniu Hu, Zhe Zhao, Xinyang Yi, Tiansheng Yao, Lichan Hong, Yizhou Sun, and Ed H Chi. 2022. Improving Multi-Task Generalization via Regularizing Spurious Correlation. *The Conference on Neural Information Processing Systems* (2022).

[21] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)* 25, 2 (2007), 7.

[22] Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6, 1 (2019), 1–54.

[23] Zhuoliang Kang, Kristen Grauman, and Fei Sha. 2011. Learning with whom to share in multi-task feature learning. In *ICML*.

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

[25] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems* 29, 8 (2017), 3573–3587.

[26] Hansang Lee, Minseok Park, and Junmo Kim. 2016. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE international conference on image processing (ICIP)*. IEEE, 3713–3717.

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[28] Zihan Lin, Hui Wang, Jingshu Mao, Wayne Xin Zhao, Cheng Wang, Peng Jiang, and Ji-Rong Wen. 2022. Feature-aware Diversified Re-ranking with Disentangled Representations for Relevant Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3327–3335.

[29] Jing Liu, Yujie Wang, and Liyan Chang. 2023. How do short videos influence users' tourism intention? A study of key factors. *Frontiers in Psychology* 13 (2023), 1036570.

[30] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1871–1880.

[31] Zhuoran Liu, Leqi Zou, Xuan Zou, Caihua Wang, Biao Zhang, Da Tang, Bolin Zhu, Yijie Zhu, Peng Wu, Ke Wang, et al. 2022. Monolith: Real Time Recommendation System With Collisionless Embedding Table. *arXiv preprint arXiv:2209.07663* (2022).

[32] Xing Lu and Zhicong Lu. 2019. Fifteen seconds of fame: A qualitative study of Douyin, a short video sharing mobile application in China. In *Social Computing and Social Media. Design, Human Behavior and Analytics: 11th International Conference, SCSM 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26-31, 2019, Proceedings, Part I 21*. Springer, 233–244.

[33] 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*. ACM, 1930–1939.

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

[35] Yuren Mao, Zekai Wang, Weiwei Liu, Xuemin Lin, and Pengtao Xie. 2022. MetaWeighting: Learning to Weight Tasks in Multi-Task Learning. In *Findings of the Association for Computational Linguistics: ACL 2022*. 3436–3448.

[36] Kevser Zeynep Meral. 2021. Social media short video-sharing TikTok application and ethics: data privacy and addiction issues. In *Multidisciplinary approaches to ethics in the digital era*. IGI Global, 147–165.

[37] Margaret K Merga. 2021. How can Booktok on TikTok inform readers' advisory services for young people? *Library & Information Science Research* 43, 2 (2021), 101091.

[38] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems* 31 (2018).

[39] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 761–769.

[40] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning?. In *International Conference on Machine Learning*. PMLR, 9120–9132.

[41] 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 *Proceedings of the 14th ACM Conference on Recommender Systems*. 269–278.

[42] HaiYing Wang, Aonan Zhang, and Chong Wang. 2021. Nonuniform Negative Sampling and Log Odds Correction with Rare Events Data. *Advances in Neural Information Processing Systems* 34 (2021), 19847–19859.

[43] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 115–124.

[44] Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, Li Wei, and Ed H Chi. 2022. Can Small Heads Help? Understanding and Improving Multi-Task Generalization. In *Proceedings of the ACM Web Conference*. 3009–3019.

[45] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020*. 441–447.

[46] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[47] 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*. 4472–4481.

[48] Chong Zhang, Kay Chen Tan, and Ruoxu Ren. 2016. Training cost-sensitive deep belief networks on imbalance data problems. In *2016 international joint conference on neural networks (IJCNN)*. IEEE, 4362–4367.

[49] Tongxi Zhang. 2020. A Brief Study on Short Video Platform and Education. In *2nd International Conference on Literature, Art and Human Development (ICLAHD 2020)*. Atlantis Press, 543–547.

[50] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. In *Proceedings of the web conference 2021*. 2220–2231.

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