Multi-Objective Ranking for Live-Streaming: Balancing Fresh and Delayed Signals with Segment-Aware Targeting

Xiaoyi Gu
Twitch Interactive
San Francisco, CA, USA
guixiaoy@twitch.tv

Julia Tavares
Twitch Interactive
San Francisco, CA, USA
juliatav@twitch.tv

Eder Santana
Twitch Interactive
San Francisco, CA, USA
edesanta@twitch.tv

Carlos Mendoza-Cardenas
Twitch Interactive
San Francisco, CA, USA
gcarlmen@twitch.tv

Nikita Mishra*
Amazon Prime Video
Sunnyvale, CA, USA
mishniki@amazon.com

Saad Ali
Twitch Interactive
San Francisco, CA, USA
sali@twitch.tv

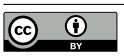
Abstract

One of the most challenging problems entertainment live-streaming services face in recommendation systems is that user behaviors are sparse and delayed, and interaction data exhibits bias for different user segments. Unlike e-commerce applications where user actions follow linear sequences, live-streaming viewers engage in multiple concurrent behaviors of watching, chatting, following, and spending, each occurring with varying delays. We address these challenges through three key contributions: 1) a delayed window approach that extends feedback collection beyond immediate responses, 2) a multi-model architecture that combines fresh and delayed signals, and a segment-aware targeting module that optimizes ranking scores differently across user lifecycle stages, and 3) Multi-gate Mixture-of-Experts (MMoE) integration that jointly models correlated targets while reducing model parameters by 41.9% compared to independent models. Online A/B testing demonstrates significant improvements, including a +0.09% increase in Daily Active Viewers (DAV), generating millions more annual active viewer days, and +0.56% increase in highly engaged viewers' capped Average Revenue Per User (ARPU). Viewer-segment targeting achieved an additional +0.15% DAV improvement for newer and less engaged viewers, while MMoE enhancement added +0.08% overall DAV and +0.27% new follows. The proposed system processes ranking requests with low latency, providing a scalable approach for balancing multiple business objectives across diverse user populations. In addition, we tested the multi-model architecture on the Twitch mobile live feed and achieved a +1.12% increase in positive user-channel interactions (clicks, follows, and likes), demonstrating applicability beyond the primary use case.

CCS Concepts

• **Information systems** → **Personalization**; *Learning to rank*; • **Computing methodologies** → Machine learning algorithms.

*Work performed while at Twitch Interactive.



RecSys '26, Minneapolis, MN, USA

ACM ISBN 979-8-4007-2284-4/26/09
https://doi.org/10.1145/3773078.3831867

Keywords

Multi-Objective Optimization, Recommendation System, Ranking, Live-streaming Services

ACM Reference Format:
Xiaoyi Gu, Julia Tavares, Eder Santana, Carlos Mendoza-Cardenas, Nikita Mishra, and Saad Ali. 2026. Multi-Objective Ranking for Live-Streaming: Balancing Fresh and Delayed Signals with Segment-Aware Targeting. In *20th ACM Conference on Recommender Systems (RecSys '26), September 27–October 02, 2026, Minneapolis, MN, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3773078.3831867

1 Introduction

A live-streaming service like Twitch is a two-sided marketplace where viewers and streamers interact through multiple mechanisms, including watching content, chatting, following channels, and participating in commerce activities. For sustainable business growth, it is important for services to satisfy the needs of all stakeholders in this ecosystem: viewers seek to both discover new content and form bonds with existing communities; streamers aim to grow audiences and monetize their channels. Therefore, the service's personalized recommendation system needs to balance immediate engagement with long-term retention and revenue goals.

Multi-objective optimization (MOO) in recommendation systems has advanced significantly through multi-task learning architectures that enable flexible knowledge sharing across objectives. Notable approaches include Multi-gate Mixture-of-Experts (MMoE) [20] and subsequent architectures [19, 28, 30], which have demonstrated success in large-scale industrial applications [37]. To balance competing objectives, researchers have explored gradient-based methods [3, 36] and Pareto optimization strategies [1, 17, 34].

While these MOO approaches have proven effective for e-commerce and video-on-demand (VOD) applications, live-streaming services present unique challenges that require specialized solutions. In particular, we identify three core challenges that are amplified in live-streaming where viewers engage through multiple concurrent behaviors of watching, chatting, following, and spending.

The first is **target sparsity**: high-value user actions such as follows, subscriptions, and purchases occur at rates orders of magnitude lower than primary engagement signals like clicks and short views, making them difficult to model effectively. In live-streaming, this challenge is especially pronounced — our analysis of millions

of impressions from 1 million viewers over 7 days showed a progression of decreasing action density: short watch rates were approximately 2.5x lower than click-through rates, longer watch rates 3.4x lower, chat rates approximately 20x lower, and follow rates nearly 90x lower, with monetary transactions occurring at even lower frequencies.

Target sparsity is compounded by **delayed feedback**, where user actions may occur hours or days after initial exposure rather than immediately. This creates a fundamental tension: labeling non-responses as negatives too early injects noise, while waiting too long makes training data stale. In live-streaming, a viewer might watch a recommended channel multiple times over several weeks before deciding to follow, and may only subscribe after establishing a stronger bond through repeated viewership.

The third challenge is **user segment bias**: highly engaged users dominate training data, biasing models away from newer or less active users who represent critical growth potential. This is compounded by differing optimization needs across user lifecycle stages: less engaged users benefit from recommendations focused on immediate engagement to encourage repeated visits, while highly engaged users benefit more from deeper engagement and monetization actions.

To address these challenges, we present a multi-objective optimization framework for a large-scale live-streaming service. We use Twitch's recommendation system as our case study, extending a single engagement-focused ranking algorithm [6] to a multi-objective framework that jointly optimizes engagement, retention, and monetization. We validate our approach through comprehensive offline and online A/B experiments, and further demonstrate generalization to Twitch's feed ranking model.

Our main contributions are as follows:

- We show that separating fresh and delayed signals is a strong driver of improvement for live-streaming recommendation, and operationalize this through a multi-model architecture combining Fresh Signal Models (FSM) for immediate engagement with Delayed Signal Models (DSM) that aggregate sparse actions over a 14-day delayed window. We introduce a viewer segment targeting (VST) module applying different optimization across user lifecycle stages at inference time.
- We demonstrate that jointly modeling deeper engagement and delayed actions in MMoE while keeping shallow engagement as an independent FSM improves performance while reducing model parameters by 41.9%.
- We validate through staged online A/B on Twitch's recommendation system serving millions of daily viewers. The multi-model with delayed window improved DAV[1] by +0.09% and highly engaged viewers' capped ARPU[2] by +0.56%. The VST module further boosted less engaged viewers' DAV by +0.15%. The MMoE enhancement added +0.08% overall DAV and +0.27% new follows[3]. We achieve these improvements while maintaining system latency under 110ms p99 at scale.

[1]DAV: Daily Active Viewers
[2]ARPU: Average Revenue Per User
[3]DAV and ARPU are top-line challenging product metrics; improvements of this magnitude represent significant business impact at Twitch's scale.

2 Related Work

2.1 Multi-Objective Optimization

Multi-objective optimization (MOO) in recommender systems has advanced significantly through multi-task learning frameworks that enable joint optimization of multiple, often conflicting objectives [27]. Early approaches employed shared-bottom architectures [23] where all tasks share a common representation. Multi-gate Mixture-of-Experts (MMoE) [20] advanced this paradigm by introducing task-specific gating mechanisms over shared expert networks, enabling flexible knowledge sharing across tasks. This architecture has demonstrated success in large-scale systems such as YouTube recommendations [37]. Subsequent work has explored more sophisticated architectures including SNR [19] that modularized the shared hidden layers, PLE [28] that separates the task-common and task-specific parameters explicitly, MSSM [10] that learns features selectively, and HoME [30] with a hierarchy meta experts structure. For sequential user behaviors, ESMM [21] models dependencies between objectives in e-commerce conversion funnels, where actions follow a defined order. AITM [32] adaptively learns sequential dependence among multi-step conversions in display advertising.

Researchers have also explored gradient-based methods [3, 36], Pareto optimization [1, 17, 22], policy learning [14], efficient adaptation [26], and objective ensembling and alignment [4, 29, 33] to balance competing or correlated objectives.

While these MOO approaches have proven effective for industrial applications such as e-commerce and VOD applications, they do not adequately address the unique challenges of live-streaming environments, particularly multiple concurrent delayed feedback and viewer segment bias, which we tackle in this work.

2.2 Delayed Feedback and Target Sparsity

In live-streaming recommendation systems, unobserved interactions are not necessarily negative [7]. The delayed feedback challenge is compounded by target sparsity, particularly for high-value events such as follows, subscriptions, and purchases that occur at extremely low frequencies, which is also observed in advertising conversion scenarios [1, 11, 18]. Research in display advertising has addressed delayed feedback through probabilistic delay models [5], nonparametric approaches [35], loss function corrections [15], multi-window methods [13], and auxiliary correction models [31].

However, most existing methods primarily address sequential conversion scenarios where a single target action (like purchase) follows an initial interaction (like click). They do not address the unique challenges of live-streaming services, where multiple viewer actions (watching, chatting, following) can occur independently after content consumption and require joint optimization. Meanwhile, sparse actions like subscriptions and purchases have insufficient positive labels without extended observation windows of days, making them challenging to capture within short windows [16].

2.3 Viewer Segment Bias

While multi-objective optimization has advanced recommendation systems, existing work rarely addresses how to handle training data imbalance across user lifecycle stages. Users at different engagement levels contribute disproportionately to training data, and

meanwhile require different optimization strategies within a unified multi-objective framework.

Bias in recommendation systems has been extensively studied [7, 25], but much research focuses on item-level or algorithmic biases [2, 24] rather than segment imbalances. Fu et al. [12] identified unfairness toward inactive users and proposed reranking approaches. We address segment bias within multi-objective optimization through segment-aware targeting that applies pre-calibrated weights per viewer lifecycle stage.

3 Preliminaries

3.1 Two-Stage Recommendation System

Twitch's recommendation system follows a two-stage Deep Neural Networks architecture [8]. The first stage employs a two-tower model to retrieve tens of channels from a catalog of millions of creators [6]. The real-time ranking stage uses a deep neural network model to predict the minutes play probability given a viewer v_i and retrieved channels $C_k, k = 1, ..., n$. The system ranks the probabilities and shows the ordered channels to the viewers. In this work, we focus on the ranking stage.

3.2 Multi-Objective Optimization For Twitch

While the base ranking model focused solely on engagement through predicting minutes play, Twitch's long-term business goals require optimizing multiple objectives simultaneously, including engagement, retention, and monetization. The real-time ranking model should optimize these high-level objectives by encouraging specific viewer actions, which serve as model training targets[4], and meanwhile stays adaptive to viewer segments. This section introduces Twitch's viewer actions and their correlation with the objectives, and defines viewer segments.

Viewer Actions and Objectives. Viewers on Twitch engage through five key actions:

- Short-form Minutes Play (SMP): Playing a channel for τ_s minutes. SMP represents immediate viewer interest after clicking into a channel.
- Long-form Minutes Play (LMP): Playing a channel for τ_l minutes, indicating deeper content engagement beyond initial curiosity.[5]
- Chatting: in a channel's community.
- Following: a channel for easier access and notifications.
- Spending: on subscriptions and gifts to support a channel.

These actions serve different business objectives: *SMP*, *LMP*, *chat*, and *follow* primarily drive new viewer engagement and retention. Sustained engagement can indirectly lead to monetization through purchases and subscriptions. Direct *spend* actions increase product revenue and indicate deeper viewer-channel bonds. This natural progression of viewer actions provides appropriate targets for optimizing engagement, retention, and revenue simultaneously.

Viewer Segments. Newer and established viewers behave differently. To better understand viewer behavior, Twitch segments viewers into two main categories based on engagement levels:

Early (E): New (account age <= M days) or less frequent (in the last M days, chat or visit days <= N days) viewers.[6]

Dedicated (D): Frequently engaging viewers and enthusiasts (account age > M days, and in the last M days, visit days > N).

4 Methods

We present our approach through progressive enhancements to address sparse delayed feedback and viewer segment bias. We begin with problem formulation (Section 4.1), then introduce the delayed window (Section 4.2), Multi-Model (Section 4.3), Viewer Segment Targeting (Section 4.4), and MMoE Enhancement (Section 4.5) with design rationale. Each stage was validated through comprehensive offline and online experiments.

4.1 Problem Formulation

Given a set of viewers $\mathcal{V}$ and channels C, let $\mathbf{x}_{v_i,c_j}$ denote the feature vector that combines viewer and channel signals for viewer $v_i \in \mathcal{V}$ and channel $c_j \in C$. To optimize multiple business objectives simultaneously, we construct a ranking function that combines predictions for all viewer actions $a \in \mathcal{A} = \{SMP, LMP, chat, follow, spend\}$:

$$F^{Ranking}(\mathbf{x}_{v_i,c_j}) = \sum_{a \in \mathcal{A}} w_a \cdot p_a(\mathbf{x}_{v_i,c_j}) \tag{1}$$

where $p_a(\mathbf{x}_{v_i,c_j})$ represents the predicted probability of viewer action a, and w_a are the weights that balance different business objectives. While this formulation is straightforward, its implementation faces key challenges previously discussed: sparse and delayed feedback signals, and the need for segment-specific optimization. Additionally, scalability is critical – Twitch serves millions of users daily, requiring efficient real-time inference. In the following sections, we present our solutions to address these challenges while maintaining real-time performance requirements.

4.2 Delayed Window Framework

To address target sparsity and delayed feedback, we expand the observation window for positive signal collection. Aggregating across multiple days makes the sparse targets denser and captures delayed feedback. For sparse actions *chat*, *follow*, and *spend*, we formalize this approach as follows. For a recommendation session at time t where viewer v_i is exposed to channel c_j, we define the delayed window DW as:

$$\begin{aligned} DW_{v_i,c_j}(t, \Delta t) = \{(a, t') : a \in \mathcal{A}, t \le t' \le t + \Delta t, \\ \text{viewer } v_i \text{ performs action } a \text{ on channel } c_j \text{ at time } t'\} \end{aligned} \tag{2}$$

where t' represents any timestamp within the delayed window interval. For each action, this transforms sparse immediate observations into aggregated denser delayed engagement signals through:

$$y^{delayed}_{v_i,c_j} = \mathbb{I}[|DW_{v_i,c_j}(t, \Delta t)| > 0] \tag{3}$$

where $\mathbb{I}[\cdot]$ is the indicator function that equals 1 if the corresponding target action occurs within the delayed window, and 0 otherwise.

[4] Throughout this paper, we use *objective* to refer to business goals, *target* for viewer actions used in training, and *task* for the corresponding prediction problems.

[5] $\tau_l \gg \tau_s$, where τ_l and τ_s are watch-time thresholds; exact values are proprietary.

[6] M and N are service-specific thresholds; exact values are proprietary.

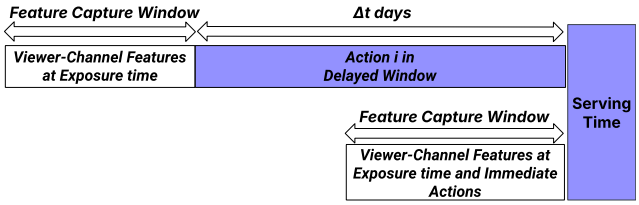


Figure 1: Timeline illustration of Delayed Window approach. Top: DSM training data uses exposure-time features with delayed window targets collected over Δt days. Bottom: FSM training data uses exposure-time features and immediate actions, ensuring fresh information at serving.

4.3 Multi-Model Architecture

The Δt delayed window effectively alleviates viewer action sparsity. However, relying solely on delayed signals presents a trade-off: while delayed feedback provides comprehensive action information, it may miss recent changes in viewer preferences that are captured by immediate signals at inference time. As illustrated in Figure 1, using only delayed targets means that features would be Δt days old by serving time, potentially missing recent viewer behavior patterns. To balance comprehensive historical feedback with fresh viewer preferences, we propose a multi-model (MM) architecture.

For less sparse viewer actions such as SMP, we use fresh signals without delayed windows and train Fresh Signal Models (FSM) to capture immediate engagement. The delayed window approach is only applied to sparse actions $a \in \{chat, follow, spend\}$, which are learned by Delayed Signal Models (DSM). Each FSM learns from dense and immediate actions, while each DSM is trained on delayed targets. All models are trained in parallel. During inference, both types of models output predictions $p_a(\mathbf{x}_{v_i,c_j})$ for their corresponding actions. This architecture provides the foundation for further enhancements in viewer segment targeting and model efficiency.

4.4 Viewer Segment Targeting (VST)

With DSMs and FSMs providing action-specific predictions, we address viewer segment bias through segment-aware weighting of model outputs. Since D viewers contribute roughly 4x more training samples than E viewers in our data, model predictions are inherently skewed toward D viewer behavior. Using the same ensemble weights across all viewer segments would therefore propagate this bias into the final rankings. Moreover, business objectives vary across viewer lifecycle stages: engagement and retention are crucial for E viewers, while D viewers can be optimized for monetization, as monetization indicates deeper viewer-channel bonds and revenue. To address these varying optimization needs while maintaining operational simplicity, we apply lightweight segment-conditioned objective scalarization at inference time rather than training separate segment-specific ranking models.

Segment-Aware Ensemble. For each viewer v_i, we determine their segment $s \in \mathcal{S} = \{E, D\}$ based on account age and engagement level described in Section 3.2. We then apply the segment-conditioned weights to the final ranking score:

$$F^{VST}(\mathbf{x}_{v_i,c_j}) = \sum_{a \in \mathcal{A}} w_{a,s} \cdot p_a(\mathbf{x}_{v_i,c_j}) \tag{4}$$

where $w_{a,s}$ represent weights for action $a \in \mathcal{A}$ and viewer segment $s \in \mathcal{S}$. The models are trained on all the viewers, while during inference, we serve the predictions differently. This post-training weighting approach preserves deployment simplicity and online iteration stability while enabling segment-specific optimization.

4.5 Multi-Gate Mixture-of-Experts Enhancement

Building upon our multi-model architecture with viewer segment targeting, we further optimize the system's efficiency while maintaining its effectiveness. While the parallel models (Section 4.3) effectively capture distinct signals for each target, maintaining separate models for each action increases system complexity and training costs. To address this, we consolidate multiple independent DSMs into a single multi-task learning model while preserving the benefits of segment-aware targeting. We experiment with Multi-gate Mixture-of-Experts (MMoE) [20], Shared-Bottom, and CGC [28] (the core extraction module of PLE) as candidate MTL backbones in Section 5; here we describe the architecture using MMoE as the representative example.

As illustrated in Figure 2, our final FSM-MMoE-VST architecture combines a Fresh Signal Model for shallow engagement (SMP) with MMoE for joint modeling of deeper engagement, retention, and monetization targets (LMP, chat, follow, spend), integrated with the VST module at inference. Our task grouping is motivated by distinct characteristics of viewer actions:

- SMP (shallow engagement): Serves as a critical early indicator requiring immediate prediction, maintained as an independent FSM.
- LMP (deep engagement): Represents sustained viewer interest beyond initial exploration. While LMP uses immediate labels rather than delayed windows, its role as a deeper engagement signal aligns naturally with delayed action targets, making it suitable for joint modeling.
- Delayed actions (chat, follow, spend): Represent community engagement, retention, and monetization objectives, captured using 14-day delayed windows to address sparsity.

Based on these characteristics, our final architecture combines an independent FSM for SMP with an MMoE model ($K = 4$ expert networks) that jointly handles $\{LMP, chat, follow, spend\}$.[7] MMoE's task-specific gating mechanisms allow different targets to selectively leverage shared expert representations while maintaining specialization. The segment-aware targeting is preserved through segment-specific weighting at inference.

Formally, our final ranking function combines predictions from both FSM and MMoE:

$$F^{FSM-MMOE-VST}(\mathbf{x}_{v_i,c_j}, s) = w_{smp,s} \cdot p^{FSM}_{smp}(\mathbf{x}_{v_i,c_j}) + \sum_a w_{a,s} \cdot p^{MMoE}_a(\mathbf{x}_{v_i,c_j}) \tag{5}$$

[7] Number of experts selected via hyperparameter optimization (HPO).

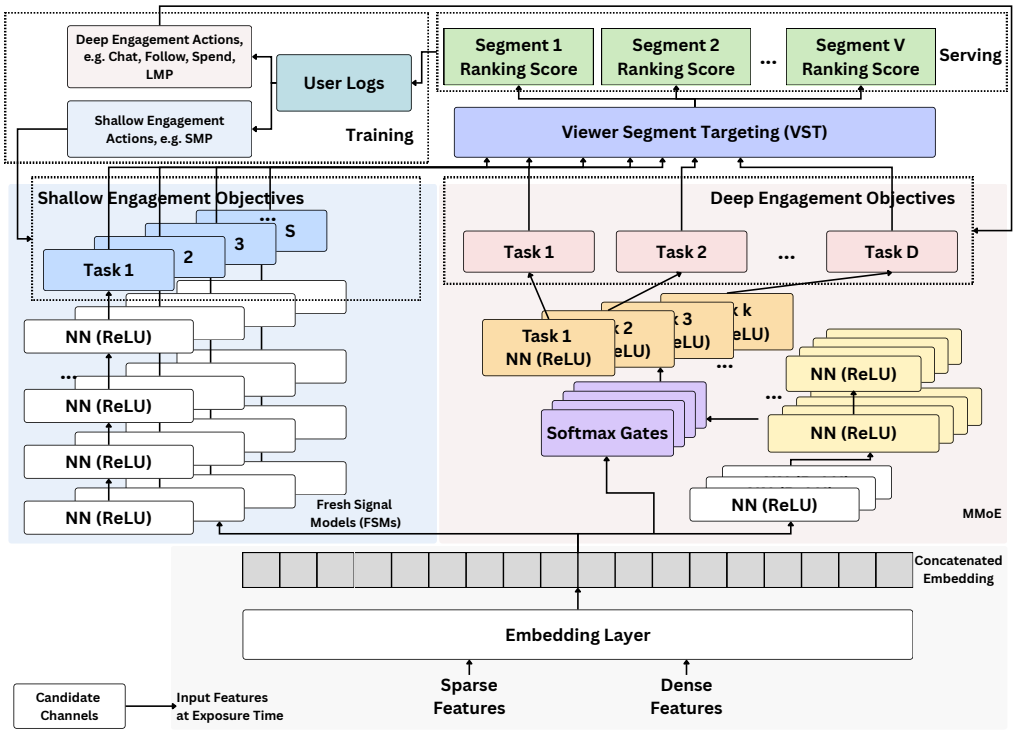


Figure 2: The FSM-MMoE-VST architecture. FSM handles shallow engagement (SMP); MMoE jointly models deep engagement tasks including immediate (LMP) and delayed targets (chat, follow, spend) with 14-day window aggregation. VST applies segment-conditioned weights at inference for Early (E) and Dedicated (D) viewers.

where $a \in \{LMP, chat, follow, spend\}$, $s \in \mathcal{S} = \{E, D\}$ is the viewer segment, and $w_{smp,s}$ and $w_{a,s}$ are segment-specific weights for FSM and MMoE outputs respectively.

5 Experiments

5.1 Experimental Setup

5.1.1 Delayed Window Analysis. To determine the optimal delayed window size for sparse targets (chat, follow, spend), we analyzed positive sample ratios across windows of {7, 14, 21, 28, 35} days using historical impressions from 6 million viewers in the training dataset. Weekly intervals were chosen to balance captured signals with data aggregation efficiency because Twitch users present different behaviors in weekdays and weekends. We observed substantial density improvements with the delayed window approach: chat targets increase by around 9 times in positive labels with just a 7-day window and by 12 times at 14 days. However, extending windows indefinitely introduces diminishing returns and raises concerns about data aggregation computation and signal staleness. In our final implementation, we apply the delayed window to the sparse targets *chat*, *follow*, and *spend*, while *SMP* and *LMP* use immediate labels.

5.1.2 Model Architecture and Training. We used 7 days of historical impression data from 6 million viewers for training. For the Fresh Signal Model (FSM), *SMP* is trained on all impressions with immediate targets. For the MMoE component, *LMP* uses immediate labels, while delayed targets (*chat*, *follow*, *spend*) use a 14-day delayed window. Our evaluation dataset comprises 1-day historical data from 1 million viewers, with a 35-day forward window for ground truth labels to ensure comprehensive coverage of delayed actions[8].

In the multi-model architecture, the FSM uses four fully-connected layers (size=1000) while DSMs use four layers (size=300), all with ReLU activation. For the MMoE-enhanced architecture, we maintain the same FSM and replace the independent DSMs with an MMoE model consisting of three shared bottom layers (size=1000), four expert networks (two layers: 512, 256), task-specific gating networks (four layers: 128 each), and one-layer task towers (size=64). We also evaluate Shared-Bottom and CGC [28] as alternative MTL backbones, in both single unified (all five targets) and FSM-based (paired with an independent SMP model) configurations. We exclude sequential conversion models such as ESMM [21] and AITM [32] as they assume sequential action funnels (e.g., click → purchase), whereas live-streaming viewers' important actions occur independently without a fixed order. All models use binary cross entropy loss and process the same feature set, including viewer demographics, channel characteristics, and viewer-channel interactions. We use the Adam optimizer with learning rates in [0.001, 0.005, 0.01] and batch sizes in [1024, 2048], selected through hyperparameter optimization. We use NDCG on the validation set for model selection.

[8] Spending actions, such as subscriptions, can be monthly.

5.1.3 Viewer Segment Targeting Strategy. We hypothesize that E and D viewers have different optimization needs. E viewers, being newer and less familiar with streamers and their communities, benefit most from recommendations focused on immediate engagement (*SMP*). In contrast, D viewers can be optimized for deeper engagement (*chat*, *follow*, *LMP*) as well as monetization (*spend*).

To counteract training data bias where E viewers contribute fewer samples, we considered applying higher loss weights to E viewers' samples during training. However, this approach did not improve performance and introduced additional training complexity in our context. Instead, we apply differentiated weighting strategies at inference time. For E viewers, we prioritize *SMP* with minimal weights on other actions. For D viewers, we balance multiple objectives. We selected candidate weights in offline tests and finalized the production configuration through iterative A/B testing on business metrics and guardrails. Specific weight values are proprietary but follow the prioritization strategy described above. This approach allows us to tailor recommendations to viewer lifecycle stages – in the early stage, we encourage them to watch and connect with the streamers they are interested in; later on, we encourage them to build deeper bonds by interacting with streamers. Importantly, this segment-aware behavior is achieved purely through differentiated weighting at inference, avoiding the complexity of maintaining separate segment-specific models.

5.2 Offline Results

5.2.1 Delayed Window Selection. Using a 2025 June-July dataset, we analyzed the impact of different delayed window sizes on the multi-model performance. Figure 3 shows NDCG@6 results for both engagement (measured by LMP - Long Minutes Play) and monetization (measured by Spend) across different window lengths.[9] The analysis reveals distinct patterns across viewer segments:

For E viewers, the 1-day window capturing immediate feedback performed worst, while extending to 7-14-21 days significantly improved both LMP and Spend NDCG@6. This suggests that aggregating delayed feedback helps better understand newer viewers' behavior patterns.

For D viewers, we observe an interesting trade-off: shorter windows (1-7 days) excel in LMP NDCG@6 but underperform in Spend NDCG@6 compared to longer windows. This pattern indicates that while longer aggregation periods help capture sparse, delayed actions like spending, excessive window lengths introduce noise by attributing actions to impressions they are no longer associated with, degrading engagement signal quality.

Based on these results, we identify 14 days as the optimal window length, achieving significant spend improvements while maintaining strong engagement performance. Using a 21-day window leads to similar performance as 14 days, with slightly better Spend and slightly worse LMP NDCG@6. However, considering the data aggregation computation and staleness, we still choose 14 days. This analysis demonstrates that appropriate delayed feedback aggregation can enhance multi-objective learning effectiveness, particularly for capturing sparse actions without compromising immediate engagement signals.

5.2.2 Model Performance. With a 14-day delayed window, we experimented with our proposed architectures. The baseline model is a single-objective point-wise deep neural network modeling SMP trained on Twitch dataset. According to Table 1, our architectural progression shows clear improvements.

First, MM with delayed window improved overall Spend NDCG@6 by 2.07% (0.0726 to 0.0741), primarily driven by D viewers (0.0780 to 0.0797), while incurring a slight LMP NDCG@6 decrease (-0.45% overall). Adding VST successfully recovered the LMP performance, particularly for E viewers, while maintaining the Spend improvements. This validates our strategy of prioritizing engagement for growth-potential E viewers while preserving monetization for established D viewers.

Comparing single unified MTL models (rows 4-6) against the DNN baseline reveals a consistent pattern: jointly modeling all five targets in a single architecture severely degrades LMP performance (-4.2% to -4.6%) regardless of the MTL backbone (MMoE, Shared-Bottom, or CGC), demonstrating that shallow engagement (SMP) conflicts with deeper engagement and delayed targets when modeled together.

Among FSM-MTL based architectures (rows 7-9), all three MTL backbones achieve comparable offline performance, suggesting that the fresh/delayed signal separation is the primary driver of improvement rather than the specific multi-task architecture. We evaluated both Shared-Bottom and MMoE in online A/B testing, where MMoE demonstrated stronger performance (Section 5.3); CGC, which showed comparable offline results, remains a candidate for future online evaluation.

All three FSM-MTL based architectures improve over the DNN baseline, with FSM + MMoE + VST improving LMP NDCG@6 by 0.12% (0.2459 to 0.2462) and Spend NDCG@6 by 1.79% (0.0726 to 0.0739), with gains across both viewer segments (E LMP: +0.24%, D LMP: +0.12%). The comparable performance across MTL backbones reinforces our key finding: the architectural separation of fresh and delayed signals is the dominant factor in our context.

5.3 Online Experiments

Based on the promising offline results, we conducted three 14-day A/B experiments to evaluate our approach stage by stage.[10]

Experiment 1: Multi-Model with Delayed Targets. The baseline uses a single-objective deep neural network optimizing SMP with a monetization heuristic feature, Monetization Attach Ratio (MAR), which was the previous A/B test winner for revenue optimization. Our treatment implements the multi-model architecture: an FSM for immediate engagement (SMP) combined with independent DSMs for delayed targets (chat, follow, spend) using a 14-day delayed window. The experiment impacted millions of visitors, with results in Table 2 (Exp. 1). We capped ARPU at a fixed daily threshold to reduce variance from high spenders.

The results demonstrate that our multi-model with delayed targets significantly improves metrics across segments: overall DAV (+0.09%, $p < 0.01$) and LMP (+0.16%, $p < 0.01$), translating to millions of annual active viewer days. D viewers show both monetization (+0.56% ARPU, $p < 0.05$) and engagement gains, while E

[9] Twitch's left navigation displays 6 recommended channels by default

[10] Follow metrics are omitted from Exp. 1-2 for brevity.

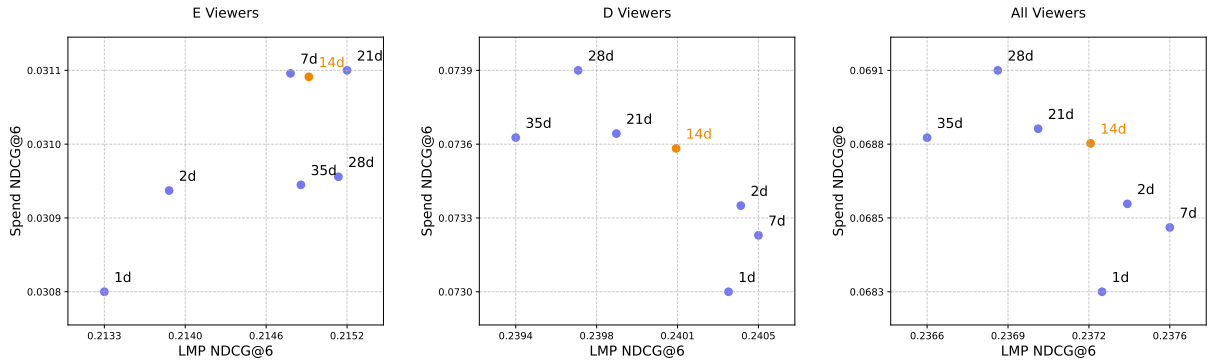


Figure 3: Comparison of LMP and Spend NDCG@6 across delayed window sizes (1-35 days), using consistent multi-model architecture and features. 14-day window highlighted.

Table 1: Offline NDCG@6 comparing different architectures. All models use consistent feature sets and 14-day delayed window where applicable. E and D represent early and dedicated viewers respectively. Bold indicates best performance per column.

	LMP			Spend		
Model	**E**	**D**	**All**	**E**	**D**	**All**
YouTube DNN [8]	0.2090	0.2504	0.2459	0.0282	0.0780	0.0726
MM + Delayed	0.2071	0.2495	0.2448	0.0283	**0.0797**	**0.0741**
MM + Delayed + VST	0.2090	0.2498	0.2453	0.0283	0.0795	0.0740
Single Shared Bottom	0.1967	0.2402	0.2355	0.0275	0.0794	0.0738
Single MMoE	0.1961	0.2395	0.2347	0.0276	**0.0797**	**0.0741**
Single CGC [28]	0.1964	0.2403	0.2355	0.0277	0.0795	0.0739
FSM + Shared Bottom + VST	0.2090	**0.2509**	**0.2463**	0.0283	0.0796	0.0740
FSM + MMoE + VST	**0.2095**	0.2507	0.2462	**0.0284**	0.0794	0.0739
FSM + CGC + VST	**0.2095**	0.2507	0.2462	0.0282	0.0794	0.0738

Table 2: Online A/B experiment results across three stages. All metrics are CUPED [9]. $^{*}p < 0.05$, $^{**}p < 0.01$.

Exp.	Segment	ARPU	DAV	LMP	Follow
1	E	−0.72%	+0.10%**	+0.11%	–
1	D	+0.56%*	+0.08%**	+0.17%**	–
1	All	+0.34%	+0.09%**	+0.16%**	–
2	E	−0.66%	+0.15%*	+0.25%**	–
2	D	+0.33%	−0.01%	−0.09%	–
2	All	+0.23%	+0.03%	+0.07%	–
3	E	+1.45%	+0.09%	+0.12%	+0.16%
3	D	−0.35%	+0.06%	+0.05%	+0.44%**
3	All	−0.01%	+0.08%*	+0.10%*	+0.27%**

viewers improve in engagement (+0.10% DAV, $p < 0.01$) but directionally decline in monetization (-0.72% ARPU). Since D viewers contribute the majority of commerce revenue, the +0.56% capped ARPU improvement represents significant business impact. Notably, this ARPU improvement is achieved over a baseline whose monetization heuristic (MAR) was the previous A/B test winner, making the gains attributable to the multi-model architecture rather than removal of a weak heuristic.

Experiment 2: Viewer Segment Targeting. Using Experiment 1's winner as baseline, we tested segment-aware targeting (Equation 4) where E and D viewers receive segment-conditioned weights.

Our segment-aware approach successfully improves E viewer engagement (+0.15% DAV, +0.25% LMP) while maintaining D performance in the A/B (Table 2, Exp. 2) with the best-performing weights. While overall metrics show modest gains (+0.03% DAV, +0.07% LMP) due to D viewers' slight changes, the improved E engagement represents valuable long-term growth potential.

Experiment 3: MMoE Enhancement. We then used the viewer segment targeting variant as the baseline, and enhanced our architecture by replacing multiple independent models with an FSM + MMoE + VST combination that jointly handles multiple targets.

The MMoE enhancement in Table 2 (Exp. 3) shows significant improvements across multiple metrics: overall total follows (+0.27%, $p < 0.01$), overall DAV (+0.08%, $p < 0.05$), and overall LMP (+0.10%, $p < 0.05$). The D DAV improvement (+0.06%, $p = 0.11$), while not meeting the significance threshold, demonstrates a positive trend. We also evaluated FSM + Shared-Bottom + VST in the same

Table 3: Ablation study results comparing MMoE task grouping strategies. All the delayed targets use a 14-day window. Single MMoE has no viewer segment targeting; all other variants include VST. Bold indicates best performance for each metric.

	LMP			Spend		
Architecture	**E**	**D**	**All**	**E**	**D**	**All**
Single MMoE (All targets)	0.1961	0.2395	0.2347	0.0276	**0.0797**	**0.0741**
FSM(SMP) + MMoE(Delayed only)	0.2091	0.2501	0.2456	0.0280	0.0791	0.0736
MMoE(SMP+LMP) + MMoE(Delayed)	0.2094	0.2505	0.2460	0.0282	0.0794	0.0739
FSM(SMP) + MMoE(LMP+Delayed)	**0.2095**	**0.2507**	**0.2462**	**0.0284**	0.0794	0.0739

experiment, which showed weaker results: overall capped ARPU +0.19%, DAV +0.06% ($p = 0.10$), LMP +0.06%, and follows +0.15%, none reaching statistical significance, confirming MMoE as the stronger MTL backbone in our online setting.

Meanwhile, the MMoE consolidation reduces the delayed-target modeling component from 26.7 million parameters (multiple independent DSMs) to approximately 15.5 million parameters (an MMoE), a 41.9% reduction that significantly decreases training costs while maintaining serving latency. This demonstrates significant efficiency gains at scale for a real-time inference ranking model.

6 Ablation Studies

We conduct ablation studies to validate our architectural design decisions for the MMoE component. While Table 1 compares across architecture families, Table 3 focuses on how targets should be grouped when using the FSM + MMoE design.

The Single MMoE approach that jointly models all five targets shows a significant trade-off: while achieving the highest overall spend NDCG@6 (0.0741), it severely degrades LMP performance (-4.7% from 0.2462 to 0.2347). This confirms our hypothesis that including SMP in joint modeling hurts engagement metrics due to its dominant signal overwhelming sparser targets.

Among FSM-based architectures, we compare three strategies. Excluding LMP from MMoE (row 2) achieves moderate performance. The dual-MMoE approach (row 3) that groups SMP and LMP in one MMoE while keeping delayed targets in another shows slight LMP improvement but increased system complexity with minimal gains. Our final architecture (row 4) that includes LMP with delayed targets in MMoE achieves the best LMP performance (0.2462) while maintaining competitive spend NDCG (0.0739). This supports our hypothesis that LMP, despite being more frequent, shares underlying engagement patterns with delayed actions that benefit from joint modeling.

7 Discussion

7.1 Signal Separation vs. Architecture Choice

A recurring finding across our experiments is that the architectural separation of fresh and delayed signals matters more than the choice of MTL backbone. All three MTL architectures (MMoE, Shared-Bottom, CGC) achieved comparable offline performance when paired with an independent FSM, while all single unified models severely degraded engagement regardless of backbone sophistication. We attribute this to the fundamental differences between shallow engagement (SMP) and deeper engagement and monetization targets: they differ in both engagement depth and target density, with SMP being orders of magnitude denser than actions like follow and spend. When modeled jointly, the dense shallow signal dominates gradient updates and overwhelms sparser targets. This suggests that when targets differ substantially in density and engagement depth, practitioners can prioritize signal-level architectural separation over MTL backbone selection.

7.2 Practical Design Lessons

We initially explored training-time interventions to address viewer segment bias, including up-weighting E samples in the loss function. This did not improve performance. In contrast, inference-time segment weighting with zero additional parameters proved effective for E engagement without degrading D performance. This framework naturally extends to finer-grained segmentation, though automating weight configuration through online learning remains an open challenge. For the delayed window, extending the delayed window improves target density up to a point but introduces trade-offs: longer windows increase data aggregation costs and risk capturing stale patterns. Our analysis showed diminishing returns beyond 14 days, with D engagement metrics declining at longer windows. Additionally, longer windows increase pipeline dependency, as the training data cannot be finalized until Δt days after exposure.

8 Conclusion and Future Work

We present a multi-objective optimization framework for live-streaming recommendations that addresses target sparsity, delayed feedback, and viewer segment bias through: (1) delayed window aggregation for sparse targets, (2) multi-model architecture balancing fresh and delayed signals, (3) segment-aware targeting across viewer lifecycle stages, and (4) MMoE integration that improves performance while reducing parameters. Through staged experiments on Twitch, we achieved +0.09% Daily Active Viewers and +0.56% D ARPU with the delayed window and multi-model approach, with segment targeting further improving E DAV by +0.15% and MMoE adding +0.08% overall DAV, achieving sub-110ms p99 ranking latency at scale. The approach also generalized to Twitch's mobile livefeed, achieving +1.12% positive interactions in a 14-day A/B test ($p < 0.001$). Future directions include automating weight optimization through online learning, exploring fine-grained segmentation beyond E/D categories, and investigating adaptive delayed windows.

Acknowledgments

We gratefully acknowledge Edgar Chen, whose early work on the delayed window concept, initial experimentation, and data curation formed a cornerstone of the multi-objective framework presented in this paper. His guidance on the system architecture, as well as his leadership in the early MMoE exploration through mentorship, were instrumental to this work.

References

[1] Shaghayegh Agah, Shaun Schaeffer, Maria Peifer, Neeraj Sharma, Ankit Maheshwari, and Sardar Hamidian. 2025. Pareto-Optimal Solution: Optimizing Engagement and Revenue. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 1034–1037.

[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 385–394.

[3] Yimeng Bai, Yang Zhang, Fuli Feng, Jing Lu, Xiaoxue Zang, Chenyi Lei, and Yang Song. 2024. GradCraft: Elevating Multi-task Recommendations through Holistic Gradient Crafting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4774–4783.

[4] Jiangxia Cao, Pengbo Xu, Yin Cheng, Kaiwei Guo, Jian Tang, Shijun Wang, Dewei Leng, Shuang Yang, Zhaojie Liu, Yanan Niu, et al. 2025. Pantheon: Personalized multi-objective ensemble sort via iterative pareto policy optimization. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 5575–5582.

[5] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1097–1105.

[6] Edgar Chen, Mark Ally, Eder Santana, and Saad Ali. 2022. Weighing dynamic availability and consumption for Twitch recommendations. In *KDD Workshop on Online and Adaptive Recommender Systems (OARS)*.

[7] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

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

[9] Alex Deng, Ya Xu, Ron Kohavi, and Toby Walker. 2013. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 123–132.

[10] Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, et al. 2021. MSSM: a multiple-level sparse sharing model for efficient multi-task learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2237–2241.

[11] Yuval Dishi, Ophir Friedler, Yonatan Karni, Natalia Silberstein, and Yulia Stolin. 2025. Practical Multi-Task Learning for Rare Conversions in Ad Tech. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 1042–1045.

[12] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 69–78.

[13] Hui Gao and Yihan Yang. 2022. Multi-Head Online Learning for Delayed Feedback Modeling. *arXiv preprint arXiv:2205.12406* (2022).

[14] Olivier Jeunen, Jatin Mandav, Ivan Potapov, Nakul Agarwal, Sourabh Vaid, Wenzhe Shi, and Aleksei Ustimenko. 2024. Multi-objective recommendation via multivariate policy learning. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 712–721.

[15] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. 2019. Addressing delayed feedback for continuous training with neural networks in CTR prediction. In *Proceedings of the 13th ACM conference on recommender systems*. 187–195.

[16] Fengqi Liang, Baigong Zheng, Liqin Zhao, Guorui Zhou, Qian Wang, and Yanan Niu. 2024. Ensure timeliness and accuracy: A novel sliding window data stream paradigm for live streaming recommendation. *arXiv preprint arXiv:2402.14399* (2024).

[17] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, and Peng Jiang. 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.

[18] Langming Liu, Wanyu Wang, Chi Zhang, Bo Li, Hongzhi Yin, Xuetao Wei, Wenbo Su, Bo Zheng, and Xiangyu Zhao. 2025. Multi-task Offline Reinforcement Learning for Online Advertising in Recommender Systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4635–4646.

[19] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. 2019. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 216–223.

[20] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[21] 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.

[22] Sofia Maria Nikolakaki, Siyong Ma, Srivas Chennu, and Humeyra Topcu Altintas. 2025. SEMORec: A Scalarized Efficient Multi-Objective Recommendation Framework. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 1074–1077.

[23] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[24] Tobias Schnabel and Paul N Bennett. 2020. Debiasing item-to-item recommendations with small annotated datasets. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 73–81.

[25] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*. PMLR, 1670–1679.

[26] Chenglei Shen, Jiahao Zhao, Xiao Zhang, Weijie Yu, Ming He, and Jianping Fan. 2025. Paragon: Parameter Generation for Controllable Multi-Task Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 370–380.

[27] 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.

[28] 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.

[29] Shen Wang, Yusheng Huang, Ruochen Yang, Shuang Wen, Pengbo Xu, Jiangxia Cao, Yueyang Liu, Kuo Cai, Chengcheng Guo, Shiyao Wang, et al. 2026. OneLive: Dynamically Unified Generative Framework for Live-Streaming Recommendation. *arXiv preprint arXiv:2602.08612* (2026).

[30] Xu Wang, Jiangxia Cao, Zhiyi Fu, Kun Gai, and Guorui Zhou. 2025. Home: Hierarchy of multi-gate experts for multi-task learning at kuaishou. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 2638–2647.

[31] Yifan Wang, Peijie Sun, Min Zhang, Qinglin Jia, Jingjie Li, and Shaoping Ma. 2023. Unbiased delayed feedback label correction for conversion rate prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2456–2466.

[32] 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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3745–3755.

[33] Boyang Xia, Zhou Yu, Zhiliang Zhu, Hanxiao Sun, Biyun Han, Jun Wang, Runnan Liu, and Wenwu Ou. 2026. HarmonRank: Ranking-aligned Multi-objective Ensemble for Live-streaming E-commerce Recommendation. *arXiv preprint arXiv:2601.02955* (2026).

[34] Ruobing Xie, Yanlei Liu, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Personalized approximate pareto-efficient recommendation. In *Proceedings of the web conference 2021*. 3839–3849.

[35] Yuya Yoshikawa and Yusaku Imai. 2018. A nonparametric delayed feedback model for conversion rate prediction. *arXiv preprint arXiv:1802.00255* (2018).

[36] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in neural information processing systems* 33 (2020), 5824–5836.

[37] 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.