Supervised Learning-enhanced Multi-Group Actor Critic for Live Stream Allocation in Feed

Jingxin Liu*
Kuaishou Technology
Beijing, China
liujingxin05@kuaishou.com

Xiang Gao
Kuaishou Technology
Beijing, China
gaoxiang12@kuaishou.com

YiSha Li
Kuaishou Technology
Beijing, China
liyisha@kuaishou.com

Xin Li
Kuaishou Technology
Beijing, China
lixin05@kuaishou.com

Haiyang Lu
Kuaishou Technology
Beijing, China
luhaiyang@kuaishou.com

Ben Wang
Kuaishou Technology
Beijing, China
benwang177@gmail.com

Abstract

In the context of a short video & live stream mixed recommendation scenario, the live stream recommendation system (RS) decides whether to allocate at most one live stream to the video feed for each user request. The inappropriate policy which ignores the long-term negative impact of live stream allocation can significantly affect app usage duration and user retention. To maximize long-term user engagement, it is crucial to determine an optimal policy for accurate live stream allocation. Recently, reinforcement learning (RL) has been widely applied in recommendation systems to capture long-term user engagement. However, traditional RL algorithms often face divergence and instability problems, which restricts application and deployment in large-scale industrial recommendation systems, especially in the aforementioned challenging scenario. To address these challenges, we propose a novel **S**upervised **L**earning-enhanced **M**ulti-**G**roup **A**ctor **C**ritic algorithm (SL-MGAC). Specifically, we introduce a supervised learning-enhanced actor-critic framework that incorporates *variance reduction* techniques, where multi-task supervised reward learning helps restrict bootstrapping error accumulation during critic learning. Additionally, we design a multi-group state decomposition module for both actor and critic networks to reduce prediction variance and improve model stability. We also propose a novel reward function to prevent overly greedy live stream allocation. Empirically, we evaluate the SL-MGAC algorithm using offline policy evaluation (OPE) and online A/B testing. Experimental results demonstrate that the proposed method not only outperforms baseline methods under platform-level constraints, but also exhibits improved stability in online recommendation scenarios.

*Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737264

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Reinforcement learning*.

Keywords

Reinforcement Learning, Recommendation System, Variance Reduction

ACM Reference Format:
Jingxin Liu, Xiang Gao, YiSha Li, Xin Li, Haiyang Lu, and Ben Wang. 2025. Supervised Learning-enhanced Multi-Group Actor Critic for Live Stream Allocation in Feed. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737264

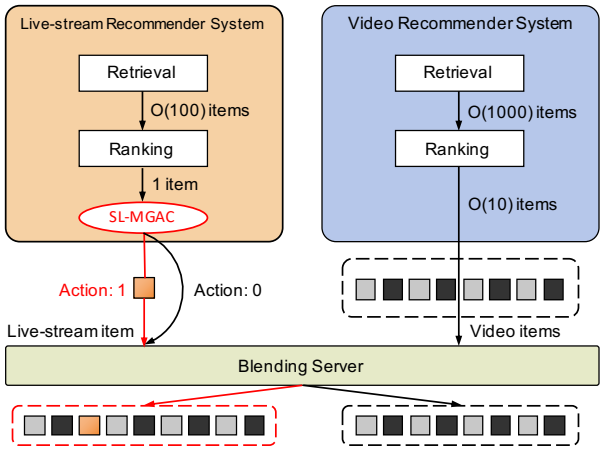


Figure 1: Structure of a short video & live stream mixed recommendation system(RS). The decision making of SL-MGAC takes place in the final stage of live stream RS.

1 Introduction

We consider a challenging sequential decision making scenario in a short video & live stream blended recommendation system, as shown in Fig. 1. The system consists of three components: a live streaming recommendation system, a short video recommendation

system, and a blending server. For each user request with timestamp t, the live stream recommendation system decides whether to inject the recommended live stream into the video feed, while the short video recommendation system suggests B (with $B < 10$) videos. The blending server then mixes and rearranges the single live stream with the B videos to form the final recommendation list. Our objective is to find a personalized optimal live stream allocation policy that not only maximizes user long-term engagement with live streams, but also meets the constraint of not causing a negative impact on app usage duration and user retention, which can be modeled as an infinite request-level Constrained Markov Decision Process (CMDP) [2].

Currently, deep reinforcement learning has shown great potential in learning policies to maximize long-term rewards in various research domains, such as computer vision, robotics, natural language processing, gaming, and recommendation systems [1, 25]. In the context of recommendation systems, reinforcement learning is applied to optimize long-term user engagement [50] and improve user retention [5]. Numerous RL applications have been proposed for real-world recommendation systems, including slate recommendation [9, 19, 28], personalized search ranking [30], and advertisement allocation [26].

However, the intrinsic issues of divergence and instability associated with traditional reinforcement learning algorithms (RL) [8, 11, 13, 24] are significantly exacerbated in the short video & live stream mixed recommendation system. A primary possible reason for this is the drastic fluctuations in both the live stream supply scale and user interaction behaviors over time, as shown in Fig. 7 in Appendix B. RL models often struggle to learn effective policies from data that exhibit such high variance across both time scales and user scales. In practice, we observe that RL models frequently encounter issues of *policy deterioration* or *model collapse* [10]. Most importantly, the live-stream allocation agent is **the final module of the live-stream RS**, as shown in Fig 1. If the RL agent becomes highly unstable or collapses in the online environment, it may lead to an excessive injection of live-stream content into the short-video feed. In turn, this can negatively impact the exposure of advertisements and E-commerce videos, potentially causing severe system malfunctions and significant platform losses. Hence, in the industrial RS, this RL application is **more risky** than existing RL methods [5, 46, 47], which are only applied in the multi-rank score aggregation of the ranking stage.

Furthermore, like advertisement, live stream is not welcomed by every user. Allocating too many live streams for a user will significantly interrupt the user's short video interest and lead to a decrease in the user's app usage duration, which will eventually affect the user engagement and user retention. Therefore, we cannot merely focus on the user feedback for a single request, but rather should optimize the long-term user experience.

To address the aforementioned problems, we propose a novel **S**upervised **L**earning-enhanced **M**ulti-**G**roup **A**ctor **C**ritic algorithm, SL-MGAC. Given the significant variation in users' interest in live stream content, which introduces high variance at both the feature and feedback levels, we incorporate separate **M**ulti-**G**roup **S**tate **D**ecomposition (MG-SD) modules within both the actor and critic networks. Additionally, we combine multi-task supervised reward learning with traditional critic learning to not only restrict temporal difference (TD) error accumulation during model training but also improve the accuracy of Q-value estimation. Through experiments, we compare the SL-MGAC method with competitive baselines using offline policy evaluation (OPE) [39] and online A/B tests to demonstrate its effectiveness and stability.

The main contributions of this paper are as follows.

- We introduce SL-MGAC, a novel RL model that outperforms existing methods in maximizing the user engagement of live stream while satisfying the platform-level constraints on app usage duration and user retention.
- We propose new *variance reduction* [3, 15, 24, 29, 34] techniques, including multi-group state decomposition, distribution discretization for reward learning, reward normalization, and Q value normalization.
- We integrate multi-task supervised reward learning with traditional critic learning to alleviate TD error accumulation and improve the accuracy of Q-value estimation.
- We successfully deploy the SL-MGAC model in a challenging short video and live stream mixed recommendation scenario for *Kwai*, a short video app with over 100 million users.

2 Problem Formulation

As shown in Fig. 1, in the short video & live stream mixed recommendation system, the live stream recommendation system is treated as an agent that interacts with various users and receives user feedback over time. However, an agent considering instant user feedback of a single request may allocate live streams greedily in a short video feed and eventually affect user retention. Therefore, the optimal live stream allocation control problem is modeled as an infinite request level CMDP to maximize the cumulative live stream reward and satisfy the platform-level constraint, for example, live stream allocation cannot reduce the total app usage duration.

Formally, we define live stream allocation CMDP as a tuple of six elements $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, C)$:

- **State space** $\mathcal{S}$: This is the set of user interaction states s, which includes user static features (e.g., user ID, location, gender, country), user history features (e.g., live stream watch history, short video watch history) and item features (e.g., live stream ID, author ID, author gender, etc). We limit the length of user live stream and short video history lists by timestamp order, keeping only the top 50 items.
- **Action space** $\mathcal{A}$: The action $a \in \mathcal{A}$ represents the decision of whether to inject a recommended live stream in response to a user's request. We define the action a as a binary variable, where $a = 1$ means injecting a live stream.
- **Transition Probability** $\mathcal{P}$: The transition probability is denoted as $p(s_{t+1}|s_t, a_t)$, determined by the environment.
- **Reward Function** $\mathcal{R}$: The reward function $\mathcal{R}$ is a mapping from state s_t and action a_t at timestamp t, which can be formulated as $r(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- **Discount Factor** γ: The discount factor $\gamma \in [0, 1]$ is used in the calculation of the cumulative reward. Typically, we set $\gamma < 1$ during model training.
- **Constraint** C: The platform-level constraint is an instantaneous constraint, that is, the live stream watch time should be longer than the average video watch time per request as

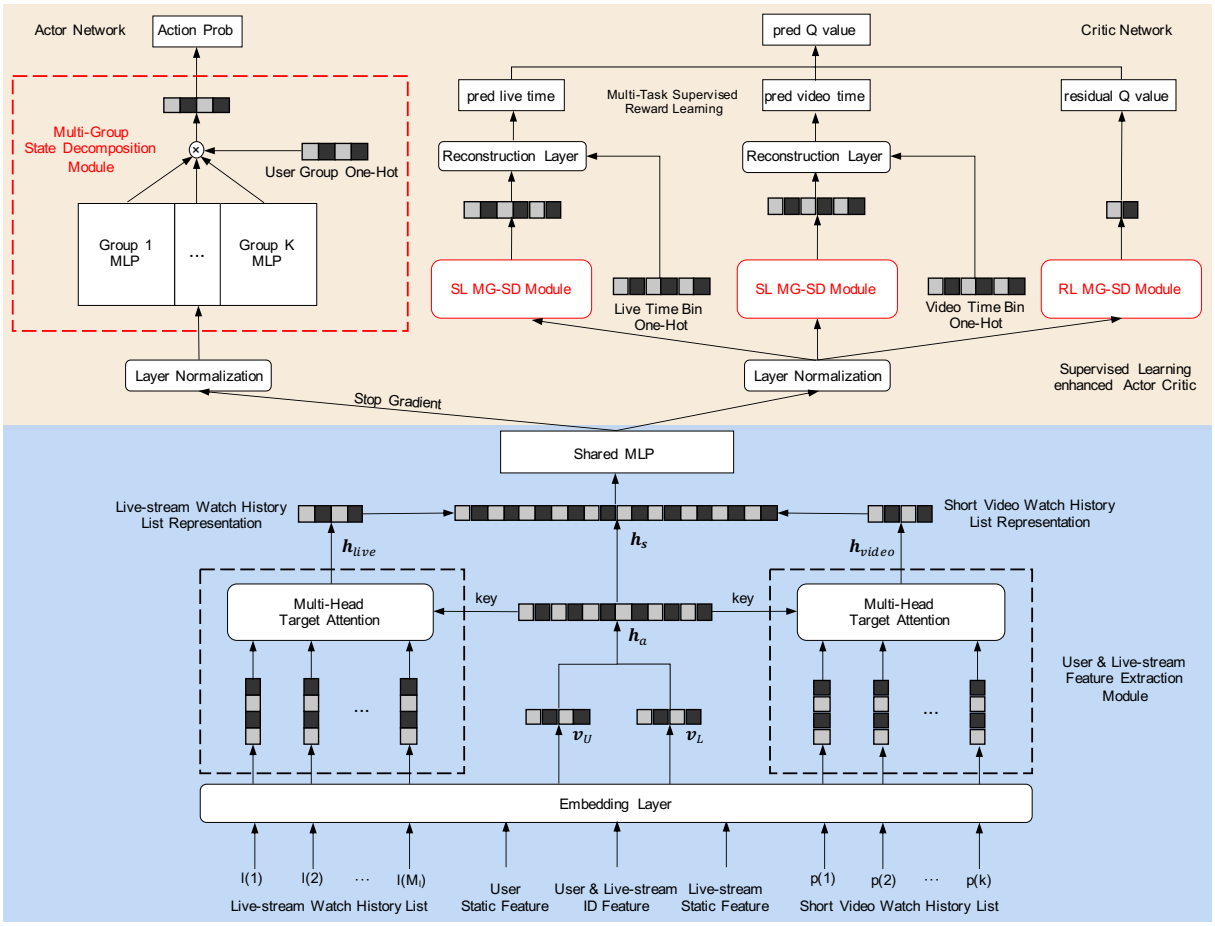


Figure 2: Overall framework of the SL-MGAC algorithm. The SL (RL) MG-SD Module is short for the Multi-Group State Decomposition Module for supervised reward learning and critic learning.

much as possible. This prevents a negative impact on app usage duration, which could happen if the live stream watch time is shorter. By meeting this constraint, we ensure that live streams boost users' long-term app engagement instead of reducing their time spent on it.

Overall, the concrete reward function $r(s_t, a_t)$ and constraint function $c(s_t, a_t)$ is shown below:

$$\begin{aligned} r(s_t, a_t) &= y_l \\ c(s_t, a_t) &= \frac{1}{B} y_v - y_l \end{aligned} \tag{1}$$

where y_l is the live stream watch time, y_v is the total video watch time, and B is the number of videos in a request.

The optimization objective of aforementioned optimal live stream allocation control problem is shown as follows:

$$\begin{aligned} &\max_{\pi} J_R(\pi) \\ &s.t. \quad J_C(\pi) \leq \epsilon \end{aligned} \tag{2}$$

where $J_R(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, $J_C(\pi) = c(s_t, a_t), \forall t \in [0, \infty)$, $\pi(s_t, a_t)$ is the live stream allocation policy to be optimized.

A common method to solve the above CMDP is to transform the problem into a min-max optimization by introducing the Lagrange multiplier λ:

$$(\pi^*, \lambda^*) = \arg\min_{\lambda \geq 0} \max_{\pi} J_R(\pi) - \lambda(J_C(\pi) - \epsilon) \tag{3}$$

Inspired by the RCPO [38] algorithm, the primal-dual optimization problem can be transformed into an equivalent unconstrained problem with penalized reward function by substituting into $r(s_t, a_t)$ and $c(s_t, a_t)$ of Eq. 1:

$$\begin{aligned} \hat{r}(\lambda, s_t, a_t) &= r(s_t, a_t) - \lambda c(s_t, a_t) \\ &= y_l - \lambda(\frac{1}{B} y_v - y_l) \\ &= (1 + \lambda) y_l - \frac{\lambda}{B} y_v \end{aligned} \tag{4}$$

In practice, we solve the unconstrained MDP problem with a simplified reward function as follows:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(y_l - \frac{\lambda}{B} y_v\right)\right] \tag{5}$$

where λ is a hyper-parameter to carefully control the final live stream allocation ratio. From the perspective of constrained reinforcement learning, the term $\frac{\lambda}{B} y_v$ can be viewed as an implicit cost constraint. Note that we leave the further discussion of the above reward function in the Appendix C.

3 Proposed Framework

To address the intrinsic divergence and instability issues of reinforcement learning and successfully deploy the RL agent in our mixed short video & live stream recommendation system, we propose a novel Supervised Learning-enhanced Multi-Group Actor-Critic algorithm (SL-MGAC), as shown in Fig. 2. SL-MGAC incorporates a supervised learning-enhanced actor-critic model with a shared user & live stream feature extraction module, along with independent Multi-Group State Decomposition (MG-SD) modules. For clarity, only one critic network is shown in Fig. 2, while the remaining three critic networks, which use the same architecture, are omitted.

3.1 User & live stream Feature Extraction Module

The user & live stream feature extraction module aims to generate non-linear embedding representations for the state s_t. First, we use a unified embedding layer to map the user's static features, live stream features, historical live stream list features, and short video-list features into low-dimensional dense representations. We denote $\mathbf{v}_U$, $\mathbf{v}_L$, $\{\mathbf{e}_1, \cdots, \mathbf{e}_{M_l}\}$, $\left\{\mathbf{e}'_1, \cdots, \mathbf{e}'_{M_v}\right\}$ as the corresponding embedding vectors or sets of embedding vectors, respectively. We then define $\mathbf{h}_a = [\mathbf{v}_U, \mathbf{v}_L]$ as the concatenation of $\mathbf{v}_U$ and $\mathbf{v}_L$.

To aggregate historical live stream (short video) representations, we introduce the *target attention* mechanism [41, 49], which is defined as follows:

$$\begin{aligned} \mathbf{h}_{live} &= \sum_{i=0}^{M_l} f_l(\mathbf{h}_a, \mathbf{e}_i)\mathbf{e}_i \\ \mathbf{h}_{video} &= \sum_{j=0}^{M_v} f_v(\mathbf{h}_a, \mathbf{e}'_j)\mathbf{e}'_j \end{aligned} \tag{6}$$

where f_l, f_v are different target attention functions, such as a feed-forward network whose output is a normalized score.

After aggregating the historical live stream and short video features via two separate attention networks, we concatenate all of the above embedding vectors to form $\mathbf{h}_s = [\mathbf{v}_U, \mathbf{v}_L, \mathbf{h}_{live}, \mathbf{h}_{video}]$. We then use a shared multi-layer perceptron (MLP) for both the actor and critic networks to obtain the latent representation of state s_t, i.e. $\mathbf{h}'_s = f_{MLP}(\mathbf{h}_s)$. To ensure training stability, we stop the gradient flow through $\mathbf{h}'_s$ for the actor network and only use the more complex critic networks to optimize $\mathbf{h}'_s$, as we find that sharing the feature extraction module and the f_{MLP} network causes interference between the policy and critic gradients.

3.2 Multi-Group State Decomposition Module

Since live stream content is not equally appealing to all users of the short video app *Kwai*, it is crucial to inject live streams selectively into the short video feed. Otherwise, excessive live stream exposures may disrupt a user's interest in short videos, leading to a decrease in overall app usage duration. In practice, user interaction data for live streams are sparser and noisier compared to that for short videos, making it challenging for RL algorithms to learn an optimal live stream allocation policy for each individual user. Directly learning a policy from sparse and noisy user feedback is often infeasible, since it will inadvertently inject excessive live streams in the short video feed and highly affected the user engagement.

We find that prior information on diverse user groups is essential to improve the accuracy of the decision making of RL models in our scenario, which provides additional user preference information to the policy. A natural approach to distinguish between different user behaviors is to partition users into distinct groups based on their historical activity. Specifically, we categorize users into K disjoint groups according to their activity level of the live stream, which is determined by the cumulative watch time of the live stream during the past 3 weeks. Users with a higher historical watch time are assigned to groups with higher activity levels.

Empirically, we show the effectiveness of this module in enhancing the stability of the RL model in subsequent experiments. We find that SL-MGAC without the MG-SD module tends to be more unstable during training, with a larger variance in its Q-values. Furthermore, the MG-SD module is flexible and transferable, allowing the user-group partitioning strategy to be easily adapted to various scenarios in advertising and E-commerce.

3.3 Supervised Learning-enhanced Actor Critic

To alleviate the impact of drastic changes in the data distribution that could lead to divergence or instability of RL models, we propose a supervised learning-enhanced actor critic framework to prevent critic networks from being trapped in model collapse due to large cumulative errors in the critic learning process.

3.3.1 Layer Normalization. A recent work on RL divergence and instability [45] shows that offline critic learning exhibits a *self-excitation* pattern. Specifically, iteration of the Q value can inadvertently cause the target Q value $Q(s_{t+1}, a^*)$ to increase even more than the increment of $Q(s_t, a_t)$, amplifying the TD error and trapping the critic learning process in a positive feedback loop until the model collapse. Therefore, normalization techniques, such as *Layer Normalization* [4] can be utilized to alleviate divergence and instability problems.

From the proof in Appendix D of [45], we know that for any input $\mathbf{x}$ and any direction $\mathbf{v}$, if a network applies Layer Normalization to the input, then we have $k_{\text{NTK}}(\mathbf{x}, \mathbf{x} + \lambda\mathbf{v}) \leq C$, where $\lambda > 0$, C is a constant, and k_{NTK} is the *Neural Tangent Kernel* (NTK) [20]. This theoretically indicates that a network with Layer Normalization is less sensitive to input variations and can maintain stable gradients despite perturbations during model training. Hence, we apply Layer Normalization to the inputs of both the actor and critic networks, as shown in Fig. 2.

3.3.2 Critic Network. Let $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}) \in \mathcal{D}$ be a training sample from our real-time dataset $\mathcal{D}$. To address the *maximization bias* problem [40], we employ four critic networks: two current Q-networks Q_{ϕ_1}, Q_{ϕ_2} and two corresponding target Q-networks Q'_{ϕ_1}, Q'_{ϕ_2} in Clipped Double Q-Learning [13]. The critic learning

objective is as follows:

$$\mathcal{L}_{Critic} = \sum_{i=1}^{2} \mathbb{E}_{(s,a)\in\mathcal{D}} \left[Q_{\phi_i}(s_t, a_t) - Q_{label}(s_{t+1}) \right]^2 \\ Q_{label}(s_{t+1}) = r(s_t, a_t) + \gamma \max_{a_{t+1}} \hat{Q}'(s_{t+1}, a_{t+1}) \tag{7}$$

where $\hat{Q}'(s_{t+1}, a_{t+1}) = \min_{i=1,2} Q'_{\phi_i}(s_{t+1}, a_{t+1})$. We use the Huber loss [17] to optimize the above objective.

In practice, $Q_{label}(s_{t+1})$ may be dominated by $\hat{Q}'(s_{t+1}, a_{t+1})$ during critic learning, which can affect the performance of critic networks. Specifically, due to instability and divergence issues, target networks often predict inaccurate $\hat{Q}'(s_{t+1}, a_{t+1})$ values, which is much larger than the $r(s_t, a_t)$ term. Then the term $r(s_t, a_t)$ could not provide any information to guide the learning of the critic. Therefore, we seamlessly introduce supervised learning (e.g. multi-task learning [48]) into the critic learning procedure. Specifically, we divide the critic network Q_{ϕ_i} (or Q'_{ϕ_i}) into two components: a Reward Prediction Network (RPN) and a Q Residual Network (QRN), as follows:

$$Q_{\phi_i}(s_t, a_t) = R_{\theta_i}(s_t, a_t) + \gamma T_{\xi_i}(s_t, a_t), \quad i = 1, 2 \\ Q'_{\phi_i}(s_{t+1}, a_{t+1}) = R'_{\theta_i}(s_{t+1}, a_{t+1}) + \gamma T'_{\xi_i}(s_{t+1}, a_{t+1}), \quad i = 1, 2 \tag{8}$$

where R_{θ_i} and T_{ξ_i} are RPN and QRN, R'_{θ_i} and T'_{ξ_i} are target RPN and QRN, respectively. We use separate MLPs to model the RPN and QRN.

Since the distribution of live stream (short video) watch time changes drastically over time, the time gain reward r in Eq. 5 becomes difficult to learn. In this work, we propose a *distribution discretization* method to improve reward learning.

Formally, we divide the live stream and short video watch time distribution into N_l and N_v non-overlapping bins, where each bin represents an interval of live stream (short video) watch time. Let y denote the actual live stream or short video watch time, which falls in the time bin $[y_{st}, y_{end}]$. The proportion of y within this bin is $\delta = \frac{y - y_{st}}{y_{end} - y_{st}} \in [0, 1]$. Then, we have a *linear reconstruction layer* to reconstruct y from δ:

$$y = y_{st} + \delta \cdot (y_{end} - y_{st}) \tag{9}$$

which resembles the structure of linear reward shifting [35].

Let $o(t)_l \in \mathbb{R}^{N_l+1}$, $o(t)_v \in \mathbb{R}^{N_v+1}$ be one-hot vectors representing the time bins in which the real live stream (short video) watch time of sample (s_t, a_t, r_t) falls. Note that a separate bin is set for the case $a_t = 0$. Then, the RPN R_{θ_i} can be modeled by multi-task neural networks:

$$R_{\theta_i}(s_t, a_t) = sigmoid\left(F_{\Gamma_i}(s_t, a_t) - G_{\Theta_i}(s_t, a_t)\right) \\ F_{\Gamma_i}(s_t, a_t) = o(t)_l{}^T \left(y_l^{st} + f_{\Gamma_i}(s_t, a_t) \cdot \left(y_l^{end} - y_l^{st} \right) \right) \\ G_{\Theta_i}(s_t, a_t) = o(t)_v{}^T \left(y_v^{st} + g_{\Theta_i}(s_t, a_t) \cdot \left(y_v^{end} - y_v^{st} \right) \right) \tag{10}$$

where $y_l^{st}, y_l^{end} \in \mathbb{R}^{N_l+1}$ predefined left (right) boundary vectors for the live stream watch time bin, $y_v^{st}, y_v^{end} \in \mathbb{R}^{N_v+1}$ are predefined left (right) boundary vectors for the short video watch time bin. Note that we introduce the posterior one-hot vectors $o(t)_l$ and $o(t)_v$ in Eq. 10 will not affect the calculation of $Q'_{\phi_i}(s_{t+1}, \cdot)$, because we have already reserved r_{t+1} in the dataset $\mathcal{D}$. Therefore, $o(t+1)_l$ and $o(t+1)_v$ can be easily obtained from r_{t+1} to compute $R'_{\theta_i}(s_{t+1}, \cdot)$.

As shown in Eq. 10, we employ two separate MLPs, $f_{\Gamma_i}(s_t, a_t)$ and $g_{\Theta_i}(s_t, a_t)$, to predict the proportions within a time bin. We then reconstruct the predicted live streaming and short video watch times, $F_{\Gamma_i}(s_t, a_t)$ and $G_{\Theta_i}(s_t, a_t)$, respectively. Finally, we obtain the predicted normalized reward, $R_{\theta_i}(s_t, a_t)$. To optimize the predicted watch-time proportions, we use Huber loss as follows:

$$\mathcal{L}_{SL} = \sum_{i=1}^{2} Huber\_Loss(\delta_l, F'_{\Gamma_i}) + Huber\_Loss(\delta_v, G'_{\Theta_i}) \tag{11}$$

where δ_l, δ_v are the actual proportion labels for live stream and short video watch times, respectively.

The reason for using the Huber loss on the time ratio δ instead of the actual watch time y is that the time ratio is within the range $[0, 1]$, resulting in smaller gradients in the neural network. Furthermore, the variance of δ is much smaller than that of the original watch time y. Consequently, the outputs of the learned reward networks (f_{Γ_i} and g_{Θ_i}) will exhibit smaller variances. Hence, reward learning with the discretization of watch time distributions can be viewed as a novel *variance reduction* technique.

3.3.3 Actor Network. We also incorporate the multi-group state decomposition module into the actor network. The loss function of the actor network is shown below:

$$\mathcal{L}_{Actor} = \mathbb{E}_{(s,a)\in\mathcal{D}} \left[-\hat{Q}(s_t, a_t) \log p(s_t, a_t) \right] \tag{12}$$

where $\hat{Q}(s_t, a_t) = \min_{i=1,2} Q_{\phi_i}(s_t, a_t)$, and $p(s_t, a_t)$ is the action probability output of the actor network.

Moreover, we observe that high values of $\hat{Q}(s_t, a_t)$ can cause instability in actor training, which can ultimately lead to *policy deterioration*, where $\forall t$, $\pi(a_t = 1|s_t) > \pi(a_t = 0|s_t)$ or $\pi(a_t = 1|s_t) < \pi(a_t = 0|s_t)$. To address this, we apply softmax normalization to $\hat{Q}(s_t, a_t)$ to obtain $\hat{Q}_{norm} = softmax(\hat{Q}(s_t, a_t))$, and propose a modified actor loss function:

$$\mathcal{L}_{Actor} = \mathbb{E}_{(s,a)\in\mathcal{D}} \left[-\hat{Q}_{norm}(s_t, a_t) \log p(s_t, a_t) \right] \tag{13}$$

A similar loss function can be found in AWAC [33], and its theoretical results demonstrate that the above loss is equivalent to an implicitly constrained RL problem:

$$\pi_{k+1} = \arg\max_{\pi\in\Pi} \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})} \left[\hat{Q}^{\pi_k}(s_t, a_t) \right] \\ \text{s.t. } D_{\text{KL}}\left(\pi(\cdot|\mathbf{s}) \| \pi_o(\cdot|\mathbf{s})\right) \le \epsilon \tag{14}$$

where D_{KL} is the Kullback-Leibler (KL) divergence, and π_o is the behavior policy derived from the dataset $\mathcal{D}$.

We note that the actor loss in Eq. 13 is a standard cross-entropy loss function. From the perspective of knowledge distillation [18], the actor distills policy knowledge from more complex critic networks. The teacher critic networks guide the more lightweight student actor network to adjust and converge to an optimal policy. In practice, we only need to deploy the actor network in the online

live stream recommendation system, which significantly reduces computational complexity and improves real-time response speed.

Overall, the final loss function of our proposed SL-MGAC algorithm is as follows:

$$\mathcal{L} = \mathcal{L}_{Actor} + \mathcal{L}_{Critic} + \mathcal{L}_{SL} \tag{15}$$

3.4 Online Exploration and Deployment

For online exploration, we adopt the commonly used ϵ-greedy [42] strategy:

$$\pi_{online}(s_t) = \begin{cases} \text{random action from } \mathcal{A}(s_t), & if\ \psi < \epsilon \\ \arg\max_{a \in \mathcal{A}(s_t)} \pi(a_t|s_t), & \text{otherwise} \end{cases} \tag{16}$$

where ψ is a random number, and ϵ is maximal exploration probability.

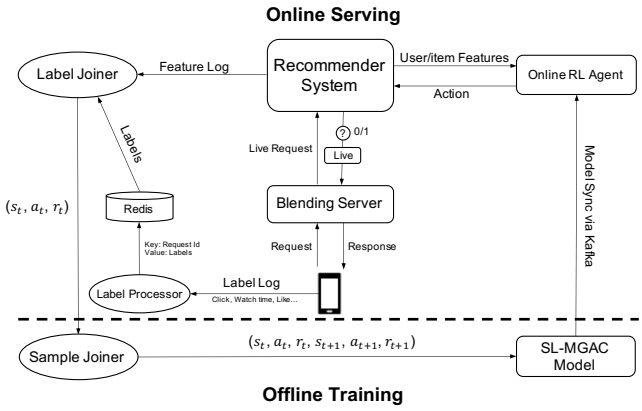


Figure 3: System Architecture of the SL-MGAC algorithm.

We implemented the SL-MGAC algorithm in our recommendation system, and the overall architecture of the system is shown in Fig. 3. The online RL agent collects real-time user interaction logs, while the offline model trainer optimizes the SL-MGAC model in an off-policy manner using streaming training data. Moreover, the offline trainer sends the latest model parameters to update the online deployed actor network in real-time.

Most importantly, we focus on reducing computational costs for successful deployment in two key aspects. First, for real-time training, we avoid using user-item cross features, which are commonly employed in industrial ranking models, thereby reducing the sparse embedding parameters. Additionally, the network parameters of the Actor part in SL-MGAC are much smaller than those of the Critic, with the complex Critic acting as a teacher to guide the Actor's learning, as explained in Section 3.3.3. As a result, the total parameters of SL-MGAC are under 100 million in our recommendation system, significantly smaller than the parameter scale of typical ranking models, which usually exceed 1 billion parameters.

For fast online inference, the smaller Actor network enables fast inference, processing each request in under 20ms, even with over 10,000 requests per second during peak times. Furthermore, since the complex Critic is not used during online inference, this further reduces computational costs, contributing to faster response time in the online environment.

4 Experiments

We perform both offline evaluation on a real-world dataset collected from our recommendation system and online A/B test experiments with SL-MGAC and the baselines.

4.1 Dataset

Due to the lack of publicly available datasets for recommendation decisions involving live streaming allocations in short video feeds, we collect an offline dataset from our recommendation system with 100,000 users through random sampling over a full day for offline evaluation, ensuring that the proportions of different user groups are similar to those in real online data. In total, we have over 1,800,000 samples, which are divided into training and test sets with a 4:1 ratio. In addition, the online environment contains more than 100 million app users for AB-Tests.

4.2 Compared Methods

4.2.1 Baselines. We compare our approach with existing non-reinforcement learning and reinforcement learning methods.

- **Learning to Rank (L2R)** [6]. A supervised learning framework that predicts the reward for each action. The action with the maximum reward is selected as the agent's action.
- **DQN** [32]. A deep neural network algorithm for Q-learning that introduces the technique of updating the target network.
- **BCQ** [14]. A widely used offline reinforcement learning algorithm that adds action restrictions and encourages the agent to behave similarly to the behavior policy.
- **SAC** [7, 16]. A classic off-policy reinforcement learning method that maximizes the trade-off between cumulative reward and policy entropy. We use the discrete version of SAC in later experiments.
- **TD3** [13]. A modified version of DDPG [27] addresses function approximation errors using three techniques: Clipped Double Q-Learning, Delayed Policy Updates, and Target Policy Smoothing.
- **TD3-BC** [12]. An offline reinforcement learning variant of the TD3 algorithm, with behavior cloning (BC) regularization to constrain policy learning.
- **IQL** [23]. An offline reinforcement learning algorithm that leverages expectile regression in Q-Learning.
- **RLUR** [5]. An RL model that aims to optimize the long-term user retention.

Note that in our optimal live stream allocation control problem, the action is discrete. Therefore, we introduce the Straight-Through Gumbel Softmax [21] technique in TD3 and TD3-BC.

4.2.2 Variations of our model. We also compare the SL-MGAC algorithm with several variants in an ablation study to illustrate the effectiveness of multi-group state decomposition, supervised learning for critic learning, distribution discretization (DD), and other techniques.

- **SL-MGAC (w/o MG)**: An SL-MGAC variant without the Multi-Group State Decomposition module.

- **SL-MGAC (w/o MG & DD)**: An SL-MGAC variant without the Multi-Group State Decomposition module and the distribution discretization technique in reward learning.
- **SL-MGAC (w/o MG & DD & SL)**: An SL-MGAC variant without the Multi-Group State Decomposition module, the distribution discretization technique in reward learning, and the supervised learning procedure.
- **SL-MGAC (w/o LN)**: An SL-MGAC variant without the Layer Normalization technique.
- **SL-MGAC (w/o SG)**: An SL-MGAC variant without the Stop Gradient technique for hidden state representation in the actor network.
- **SL-MGAC (w/o Q-norm)**: An SL-MGAC variant without Q-value normalization in actor loss.
- **SL-MGAC-0**: An SL-MGAC variant with the discount factor $\gamma = 0$.
- **SL-MGAC-sep**: An SL-MGAC variant with separate optimization of the actor network. Hence, $\hat{Q}_{norm}$ in Eq. 13 is detached from the computation graph.
- **SL-MGAC-vanilla**: An SL-MGAC variant with the vanilla Q label in the loss of the critic, as shown below, where $p(s_{t+1}, \cdot)$ is the probability of action of the output of s_{t+1} by the actor network.

$$Q_{label}(s_{t+1}) = r(s_t, a_t) + \gamma \sum_{a_{t+1}} p(s_{t+1}, a_{t+1}) \hat{Q}(s_{t+1}, a_{t+1}) \tag{17}$$

4.3 Implementation Details

To ensure fairness among all compared methods, we use a consistent feature extraction module with a 5000-size embedding layer. In addition to the embedding layer, the feature extraction module includes a 2-layer MLP with hidden sizes of [256, 128]. The batch size and number of epochs for all methods are 2048 and 500, respectively. The learning rate for the embedding layer is set to 1e-5, while the learning rate for other hidden parameters is 1e-3. The discount factor γ is set to 0.9 for all methods and λ in Eq 5 is 0.1. Detailed network architecture and parameter settings for SL-MGAC are provided in Table 4 of the Appendix A. And the code of SL-MGAC method is available at https://github.com/frankg1/SL-MGAC-torch.

4.4 Offline Policy Evaluation

We follow the approach in [44] and adopt a commonly used offline policy evaluation method, namely Normalized Capped Importance Sampling (NCIS) [37], to evaluate performance. The evaluation metric is the cumulative reward across all trajectories of test users.

The results of the offline policy evaluation are shown in Table 1. Compared to the supervised learning method L2R, most RL methods achieve higher cumulative rewards, except for DQN. SL-MGAC significantly outperforms all other methods, demonstrating that SL-MGAC can achieve higher long-term rewards in the complex live stream & short video mixed recommendation system with drastically changing data distributions. Furthermore, SL-MGAC outperforms SL-MGAC-vanilla, suggesting that incorporating the actor network in the Q label calculation of Eq. 17 may interfere with the learning of the critic to some extent, slightly affecting the performance of the SL-MGAC-vanilla.

Methods	Cumulative Reward
L2R [6]	413.49
DQN [32]	413.18
BCQ [14]	416.69
SAC [7]	435.41
TD3 [13]	432.08
TD3-BC [12]	430.23
IQL [23]	432.03
RLUR [5]	**443.01**
SL-MGAC-sep	435.19
SL-MGAC-vanilla	450.21
SL-MGAC	**458.49**

Table 1: Overall offline performance of compared methods.

4.5 Ablation Study

We compare the offline performance of different SL-MGAC variants. As shown in Table 2, SL-MGAC outperforms all other variants on our offline dataset, demonstrating the effectiveness of the proposed multi-group state decomposition module, supervised learning procedure, and other techniques. Moreover, we find that SL-MGAC (w/o Q-norm) achieves the lowest reward, indicating that the Q-normalization technique in the actor loss enhances the model's convergence and improves its performance.

Methods	Cumulative Reward
SL-MGAC (w/o MG)	454.62
SL-MGAC (w/o MG & DD)	452.59
SL-MGAC (w/o MG & DD & SL)	449.13
SL-MGAC (w/o LN)	453.61
SL-MGAC (w/o SG)	457.26
SL-MGAC (w/o Q-norm)	392.12
SL-MGAC	**458.49**

Table 2: Offline performance of SL-MGAC variants.

Next, we compare the training processes of SL-MGAC and SL-MGAC (w/o MG), as shown in Fig. 4. Note that the shaded area in yellow corresponds to the Q value std range of the SL-MGAC model, while the shaded area in purple corresponds to the Q value std range of the SL-MGAC model(w/o MG). We observe that the Q-value curve of SL-MGAC is more stable than that of SL-MGAC (w/o MG), and the Q-value variance of SL-MGAC is much smaller. This demonstrates the effectiveness and variance reduction effect of the MG-SD module.

4.6 Parameter Sensitivity

We analyze the impact of the number of user groups K on the performance of SL-MGAC. As shown in Fig. 5, the proposed SL-MGAC algorithm with $K = 6$ achieves the highest cumulative reward, demonstrating that increasing the number of user groups within a certain range improves model performance.

Methods	live stream DAU	live stream Watch Time	Video Watch Time	App Usage Duration	User Retention
L2R	+0.321%	-0.558%	-0.238%	-0.178%	-0.154%
Dummy	**+51.3%**	**+34.6%**	-2.216%	-1.062%	-0.439%
SL-MGAC-0	+3.928%	-2.934%	-0.153%	-0.117%	-0.102%
SL-MGAC	+2.616%	+7.431%	**+0.197%**	**+0.121%**	**+0.086%**

Table 3: Online A/B Test Performance of compared methods.

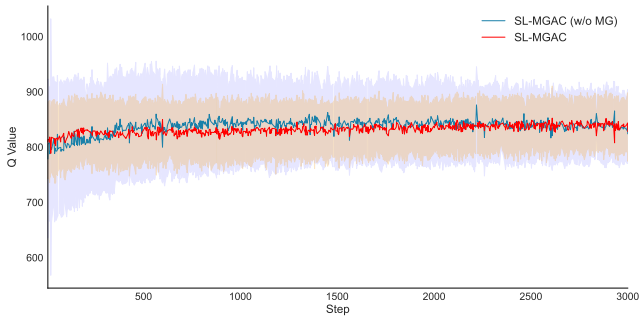


Figure 4: The Q value curves between SL-MGAC and SL-MGAC (w/o MG) over 10 rounds of training. The lines correspond to the means of Q-value and the shaded areas correspond to the standard deviations (std).

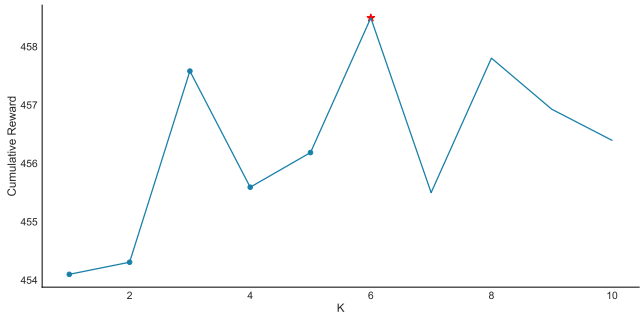


Figure 5: Performance of different numbers of user group K.

4.7 Online A/B Test Experiment

The online A/B Test is conducted over a 5-day period in 2024 September and we randomly choose 20% users (over 20 million) in Kwai app as the experimental group. Then we compare the improvements of different methods in terms of daily active live stream users (DAU), live stream watch time, video watch time, app usage duration, and user retention relative to the baseline. The baseline uses the SAC framework for live stream allocation. The results of the online A/B test are shown in Table 3.

Note that the Dummy method injects a live stream for each request. We can see that SL-MGAC achieves a significant improvement in live stream watch time while also increasing the app usage duration and user retention than other methods. This shows that SL-MGAC is more effective in maximizing long-term live stream rewards while avoiding the negative impact on app usage duration and user retention. Although SL-MGAC-0 shows a greater improvement in live stream DAU compared to SL-MGAC, it tends to be more greedy in injecting live streams, which may affect the long-term user experience for most users. Moreover, although the dummy method can greatly improve DAU and live stream viewing time, it significantly affects app usage duration and user retention.

4.8 Online Model Stability

We evaluate the online model stability between the SAC-based baseline and SL-MGAC. The live-stream injection ratio is chosen as the evaluation metric, calculated by dividing the number of live-stream injection requests by the total number of requests within a given timestamp. As shown in Fig. 6, the live-stream allocation ratio for both models fluctuates significantly throughout the day, from high-traffic hours (such as 6:00–11:00) to low-traffic hours (14:00–18:00). This variation is due to changes in the overall user scale during these periods, which in turn affect the data distribution. However, SL-MGAC exhibits greater stability than the baseline, with a smaller amplitude. Note that the live stream allocation ratio refers to the proportion of requests with action $a = 1$.

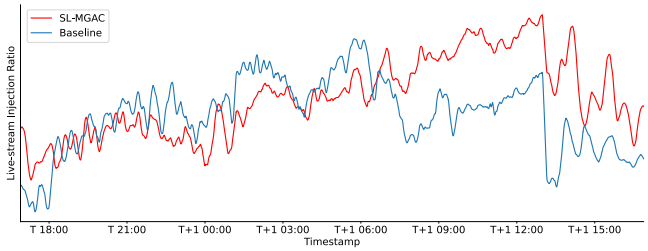


Figure 6: The trend curve of online live stream allocation ratio during a whole day.

We calculate the amplitudes of the live stream allocation ratio within sliding time windows of 20 minutes. The amplitude density is shown in Fig. 8 in Appendix B. The results indicate that SL-MGAC has a smaller mean amplitude and fewer outliers. In contrast, the baseline model exhibits larger amplitude outliers, as highlighted in the red dashed box, demonstrating the superior online model stability of SL-MGAC.

5 Related Work

5.1 RL in Recommendation Systems

Reinforcement learning (RL) aims to optimize cumulative long-term rewards over time, which has attracted significant attention in the research of recommendation systems in recent years [1]. Methods such as SLATEQ [19], GeMS [9], and HAC [28] use RL to recommend complete item lists, where the number of candidate items can

be large. BatchRL-MTF [47], RLUR [5], and UNEX-RL [46] leverage RL to model the multi-rank score aggregation process, optimizing the weights for score aggregation. The work in [30] explores the use of off-policy RL for personalized search ranking in multiple sessions. CrossDQN [26] introduces an RL-based approach to ad allocation in a feed, aiming to maximize revenue while improving the user experience. Furthermore, traditional RL methods such as DQN [31, 32], Double DQN [40], SAC [16], DDPG [27], and TD3 [13] serve as backbones in real-world RL applications for recommendation systems.

A similar approach, called self-supervised actor-critic [43], combines supervised learning with critic learning. Their supervised learning task focuses on predicting the next item recommendation probability, whereas our approach introduces supervised learning to restrict critic learning.

However, the aforementioned RL methods may encounter instability issues and could fail when faced with highly noisy or drastically changing data distributions. The proposed SL-MGAC method is more robust and has been successfully deployed in the "risky" final stage of a real-world industrial RS. We believe this work provides valuable insights not only into the model framework design for RL applications in industrial RS, but also offers practical experience in deploying robust and successful RL systems for other industrial companies.

5.2 RL divergence and instability

Recently, there has been a growing literature that focuses on RL divergence and instability. [40] introduces Double DQN to mitigate the *maximization bias* problem in DQN. [13] proposes the Clipped Double Q-Learning technique in TD3 to reduce the overestimation of Q values. [29] introduces a bias-free, input-dependent baseline for the policy gradient algorithm [36] to reduce variance and improve the stability of training. [10] investigates policy collapse in a 2-state MDP and finds that L2 regularization and the non-stationary Adam optimizer [22] are both effective in alleviating RL instability. [45] theoretically analyzes the causes of RL divergence and applies Layer Normalization to mitigate RL divergence and instability.

6 Conclusion

In the challenging context of a live stream & short video mixed recommendation system, we propose a novel Supervised Learning-enhanced Multi-Group Actor-Critic algorithm (SL-MGAC) to optimize request-level live stream allocation policies under the platform-level constraints on app usage duration and user retention. Specifically, we introduce a multi-group state decomposition module to reduce prediction variance and seamlessly integrate multi-task supervised reward learning with traditional critic learning to constrain Q-value estimation. Compared with existing RL methods for recommendation systems, our approach not only focuses on improving online performance, but also improves the stability and robustness of the RL model. In practice, we minimize the risk of *policy deterioration* or *model collapse*, thereby enabling the proposed SL-MGAC method to be successfully implemented in large-scale industrial recommendation systems. Currently, we are trying to apply the SL-MGAC methods to other similar recommendation scenarios, including E-commerce and advertising. We also aim to continuously improve the performance of SL-MGAC in further research. Learning live stream policy from a long sequence of user behaviors (s_t, a_t, r_t) and exploring more refined and complex user state representations would lead to our future work.

7 Acknowledgment

The authors acknowledge Wei Bai, Xiaoshuang Chen and anonymous reviewers for proposing detailed modification advice to help us improve the quality of this manuscript.

References

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.
[2] Eitan Altman. 2021. *Constrained Markov decision processes*. Routledge.
[3] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*. PMLR, 176–185.
[4] Jimmy Lei Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[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. In *Companion Proceedings of the ACM Web Conference 2023*. 421–426.
[6] Ming Chen and Xiuze Zhou. 2020. DeepRank: Learning to rank with neural networks for recommendation. *Knowledge-Based Systems* 209 (2020), 106478.
[7] Petros Christodoulou. [n. d.]. Soft actor-critic for discrete action settings. arXiv 2019. *arXiv preprint arXiv:1910.07207* ([n. d.]).
[8] Vibhavari Dasagi, Jake Bruce, Thierry Peynot, and Jürgen Leitner. 2019. Ctrl-z: Recovering from instability in reinforcement learning. *arXiv preprint arXiv:1910.03732* (2019).
[9] Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten De Rijke. 2023. Generative slate recommendation with reinforcement learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 580–588.
[10] Shibhansh Dohare, Qingfeng Lan, and A Rupam Mahmood. 2023. Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*.
[11] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. 2015. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011* (2015).
[12] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.
[13] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.
[14] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.
[15] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research* 5, 9 (2004).
[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.
[17] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer.
[18] Geoffrey Hinton. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).
[19] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets. (2019).
[20] Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* 31 (2018).
[21] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).
[22] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[23] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169* (2021).

[24] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems* 32 (2019).

[25] Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).

[26] Guogang Liao, Ze Wang, Xiaoxu Wu, Xiaowen Shi, Chuheng Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. Cross dqn: Cross deep q network for ads allocation in feed. In *Proceedings of the ACM Web Conference 2022*. 401–409.

[27] TP Lillicrap. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[28] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Peng Jiang, Kun Gai, Xiangyu Zhao, and Yongfeng Zhang. 2023. Exploration and regularization of the latent action space in recommendation. In *Proceedings of the ACM Web Conference 2023*. 833–844.

[29] Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. 2018. Variance reduction for reinforcement learning in input-driven environments. *arXiv preprint arXiv:1807.02264* (2018).

[30] Dadong Miao, Yanan Wang, Guoyu Tang, Lin Liu, Sulong Xu, Bo Long, Yun Xiao, Lingfei Wu, and Yunjiang Jiang. 2021. Sequential Search with Off-Policy Reinforcement Learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4006–4015.

[31] Volodymyr Mnih. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[33] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[34] Joshua Romoff, Peter Henderson, Alexandre Piché, Vincent Francois-Lavet, and Joelle Pineau. 2018. Reward estimation for variance reduction in deep reinforcement learning. *arXiv preprint arXiv:1805.03359* (2018).

[35] Hao Sun, Lei Han, Rui Yang, Xiaoteng Ma, Jian Guo, and Bolei Zhou. 2022. Exploit reward shifting in value-based deep-rl: Optimistic curiosity-based exploration and conservative exploitation via linear reward shaping. *Advances in neural information processing systems* 35 (2022), 37719–37734.

[36] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[37] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems* 28 (2015).

[38] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2018. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074* (2018).

[39] Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. 2022. A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355* (2022).

[40] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[41] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[42] Christopher John Cornish Hellaby Watkins. 1989. Learning from delayed rewards. (1989).

[43] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 931–940.

[44] Wanqi Xue, Qingpeng Cai, Zhenghai Xue, Shuo Sun, Shuchang Liu, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. 2023. PrefRec: recommender systems with human preferences for reinforcing long-term user engagement. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2874–2884.

[45] Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. 2024. Understanding, predicting and better resolving Q-value divergence in offline-RL. *Advances in Neural Information Processing Systems* 36 (2024).

[46] Gengrui Zhang, Yao Wang, Xiaoshuang Chen, Hongyi Qian, Kaiqiao Zhan, and Ben Wang. 2024. UNEX-RL: Reinforcing Long-Term Rewards in Multi-Stage Recommender Systems with UNidirectional EXecution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 9305–9313.

[47] Qihua Zhang, Junning Liu, Yuzhuo Dai, Yiyan Qi, Yifan Yuan, Kunlun Zheng, Fan Huang, and Xianfeng Tan. 2022. Multi-task fusion via reinforcement learning for long-term user satisfaction in recommender systems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 4510–4520.

[48] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering* 34, 12 (2021), 5586–5609.

[49] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.

[50] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2810–2818.

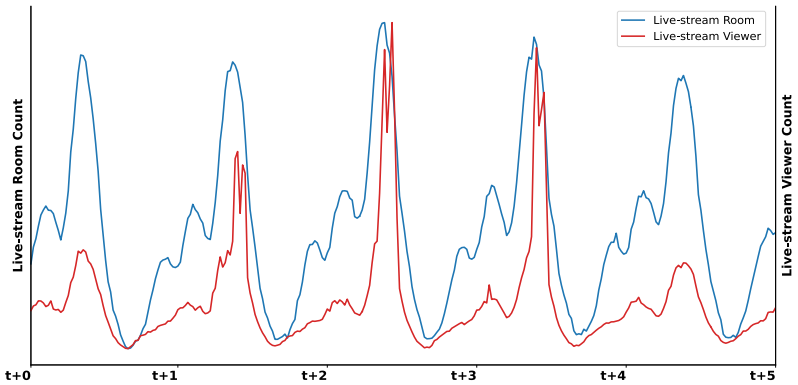
A Tables

Hyper-parameter	Value
MGN in Actor	[128, 63, 31, 2]
MGN in RPN	[128, 64, 32, 8]
MGN in QRN	[128, 64, 32, 2]
Live-Time DDB	[0s, 6s, 15s, 30s, 60s, 100s, 600s, 1200s]
Video-Time DDB	[0s, 3s, 10s, 25s, 50s, 100s, 600s, 1200s]
Optimizer	Adam
User Group Number K	6
Online Exploration ϵ	0.2

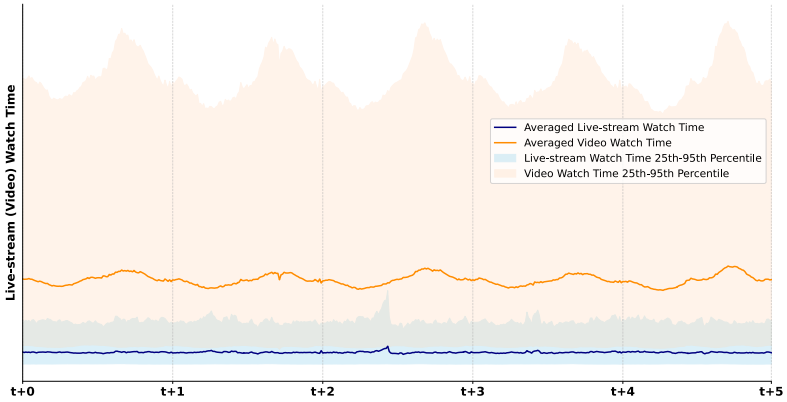
Table 4: Hyper-parameters of SL-MGAC.

We denote MGN as the multi-group network, RPN as the Reward Prediction Network, QRN as Q Residual Networks, and DDB as Distribution Discretization Bins. Both live and video watch time distribution have 7 time bins. We leave a separate time bin for the $a = 1$ case, and hence the output layer dimension of RPN is 8.

B Figures



(a) Distribution of Online Live-stream Room and Viewer Count



(b) Distribution of Averaged Watch Time

Figure 7: Data distributions of online live stream room (or viewer) count, live stream watch time and short video watch time.

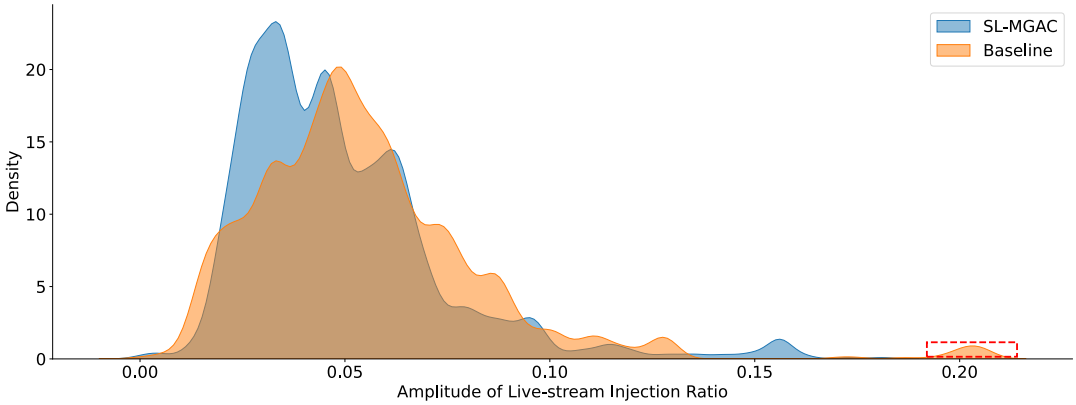


Figure 8: The amplitude distribution of live stream injection ratio.

C More Ablation Studies

We conduct further online AB experiments to demonstrate the performance of the SL-MGAC policy at different values of λ in Eq 5, as shown in Table 5. It indicates that when $\lambda > 0$, the live stream allocation policy of SL-MAGC behaves more conservatively to balance live stream watch time and video watch time, and finally optimize user's long-term engagement and retention. However, when $\lambda < 0$, the SL-MAGC behaves more greedily to allocate live streams to the video feed. Specifically, we find that the SL-MAGC allocates more live streams to the requests that occur earlier in a session due to the position bias on the reward, which would interrupt the user interest throughout the session and hence affect user retention.

λ	live stream DAU	live stream Watch Time	Video Watch Time	App Usage Duration	User Retention
0.2	-1.245%	-3.567%	+0.215%	+0.123%	+0.106%
0.1	+2.616%	+7.431%	+0.197%	+0.121%	+0.086%
-0.1	+4.123%	+6.564%	-0.248%	-0.362%	-0.112%
-0.2	+5.876%	+7.215%	-0.296%	-0.445%	-0.159%

Table 5: Online AB Test Performance of SL-MGAC under different values of λ.