EFfECT-RL: Enabling Framework for Establishing Causality and Triggering engagement through RL

Debanjan Sadhukhan
Games24x7
Bangalore, Karnataka, India
debanjan.sadhukhan@games24x7.com

Deepanshi Seth
Games24x7
Bangalore, Karnataka, India
deepanshi.seth@games24x7.com

Sanjay Agrawal
Games24x7
Bangalore, Karnataka, India
sanjay.agrawal@games24x7.com

Tridib Mukherjee
Games24x7
Bangalore, Karnataka, India
tridib.mukherjee@games24x7.com

Abstract

Skill-based games offer an exceptional avenue for entertainment, fostering self-esteem, relaxation, and social satisfaction. Engagement in online skill gaming platforms is however heavily dependent on the *outcomes* and experience (e.g., wins/losses). Understanding the factors driving increased engagement is crucial within skill gaming platforms. In this study, we aim to address two key questions: (1) *"What factors are driving users to increase their engagement?"* and (2) *"How can we personalize users journey accordingly to further optimize their engagement?"*. In skill gaming platforms, the impact of causal relationships often manifests with a delay, which varies significantly as users' personas evolve. Without a detailed information on treatments (such as timing and frequency), estimating the impact of a causal-treatment-effect in a highly volatile game-play data becomes exceedingly challenging. This work proposes a framework called EFfECT-RL that establishes causal discovery by integrating change-point detection and explainable K-means clustering, while leveraging users' game-play and transactional-data. Unlike existing methods which were unable to detect causal-effects in extremely volatile-data, EFfECT-RL generates threshold-trees (~ 79% accuracy) elucidating causal-relationships. Once the causal relationship is established, we personalize treatments by developing a novel offline deep reinforcement learning-based approach. Our online recommendations show a 3% improvement in user engagement (platform-centric) with 70% relevancy (user-centric).

CCS Concepts

• **Computing methodologies → Sequential decision making**.

Keywords

Causal analysis, offline reinforcement learning, personalization, time series segmentation

ACM Reference Format:
Debanjan Sadhukhan, Deepanshi Seth, Sanjay Agrawal, and Tridib Mukherjee. 2024. EFfECT-RL: Enabling Framework for Establishing Causality and Triggering engagement through RL. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3627673.3680058

1 Introduction

In online skill-gaming platforms [5, 34], users play to achieve self-esteem, relaxation, and social gratification. Games are ranked ordinally (0 to 9) based on intensity. Players maintain a wallet, which decreases with each game based on its intensity and outcome. Once depleted, more deposits are required to continue playing. User engagement is measured by the duration (or frequency) and intensity of gameplay. Increasing one or the other may not only be insufficient to improve engagement, but also, may have detrimental effect overall (e.g., short-term increase in engagement via intensity enhacement may actually derogate the duration, and vice-versa). This study aims to address two questions: (1) *What factors drive increased user engagement?* and (2) *How can we personalize the user journey to optimize engagement?*

Optimizing engagement is more challenging than on other consumer centric platforms (like OTT [10] or e-commerce [31]) due to: (i) diverse user intents (exploration, challenge, social gratification) requiring adaptation to unknown preferences; (ii) highly volatile preferences that change based on outcomes; (iii) the transient interplay between ordinality choice, gameplay, outcomes, and wallet balance, affecting subsequent intent; and (iv) unique longitudinal gameplay behavior [30, 38, 48].

Engagement is influenced by various platform campaigns, but this work assumes limited or no details about these campaigns (e.g., timing and frequency). This is practical when multiple teams run numerous campaigns, and the objective is to create a resilient framework to identify impacts using only users' gameplay or transactional data which is extremely volatile in nature. In skill-gaming platforms, treatment effects are delayed (e.g., the delay in engagement drops after a loss) and vary as users evolve (mature users take more time before getting affected by losses). User personas are shaped by outcomes and experiences (wins, losses, bonuses).

Existing literature in this regard typically uses either fixed cause-effect interval [36] or DTW-based segmentation [3] of time-series observations. However, in outcome-based platforms like online skill gaming—(a) the cause-effect interval can be dynamic across personas as well as across time; and (b) DTW segmentation is inadequate to capture the user contexts as part of the segments. *We introduce EFfECT-RL, a framework using change-point detection [20] and explainable K-means clustering [29] to establish and explain multi-hop causal relationships with extremely volatile game-play and transactional data.* Once we identify the factors affecting user engagement, our goal is to tailor treatments to optimize the engagement, unlike campaign-based analyses that use A/B testing [18]. We achieve this through an innovative offline method grounded in deep reinforcement learning (DRL) [4].

DRL-based strategies can be online or offline. The online agents interact with the environment to collect experience, while the offline agents train on pre-collected data [4]. Online exploration can lead to irrelevant recommendations, negatively impacting engagement. Offline techniques usually generate sub-optimal recommendations from historical data, minimizing the opportunity costs during deployment. Recent work however suggests offline strategies can learn the optimal policy using importance-sampling estimation techniques [23]. The **contributions** of EFfECT-RL framework are:

- optimizing user engagement in skill-gaming platforms by establishing a link between *dynamic* causal relationships and DRL based recommendations;
- discerning causal relationships in volatile user behavior despite variable-lagged delays and the absence of treatment information via both change-point detection (CPD) and explainable K-means clustering techniques;
- establishing an offline DRL solution with an S-network to capture user dynamics, and a double-DQN-based Q-network to recommend an optimal policy.

The S-network uses auto-encoder and encoder-decoder structures to embed the current and next user states, while the Q-network interacts with the S-network to find the optimal policy. Unlike existing causal-effect estimation techniques that struggle with volatility and dynamic nature, EFfECT-RL establishes causality using threshold-trees (with 79% accuracy) and increases user engagement by 3% with a 6% increase in extent-of-win during deployment.

2 Related Works

2.1 Causality

Understanding causality in user behavior involves estimating causal treatment effects [44] and causal discovery [46]. Causal treatment effect estimation techniques are categorized into time-invariant treatment effect, time-varying treatment effect, and dynamic regimes. Time-invariant techniques, such as Differences-in-Differences (DiD) Analysis [32], Propensity Score Matching [8], and Instrumental Variables (IV) Analysis [39], estimate the impact of treatments where the treatments are specified by treatment variable. Time-varying methods, such as Marginal Structural Models (MSM) [17] and G-estimation [6], and dynamic methods, such as Sequential Multiple Assignment Randomized Trials and Structural Nested Models (SNM), assume interventions change over time [7]. However, these methods are not applicable when both the number of treatments and the timing of these treatments are unknown. Causal discovery involves identifying and inferring causal relationships or dependencies between variables, often determining the direction of influence between variables. Methods based on Granger Causality and Conditional Independence assume a fixed lag between cause and effect variables [36]. Recently, a novel method combining Dynamic Time Warping (DTW) with Granger causality was proposed to identify variable lagged-based causality in time series. However, the performance of this method highly depends on the DTW-distance metric, which is very challenging to set and hence could not be applicable for skill-gaming platforms [3].

2.2 Time series sub-sequence clustering

Time series sub-sequence clustering breaks down long time-series data into smaller segments based on similarity or defined criteria. Typically, fixed-length sliding windows are used to extract these subsequences from the original data [49]. However, in this work, we consider the presence of variable-lagged causal relationships, making traditional approaches less applicable. After segment acquisition, clustering methods like Dynamic Time Warping (DTW) or alternative distance metrics are employed, which remains a significant challenge due to the need for suitable distance metrics [2] in our application and lacks provable interpretibilty.

2.3 Reinforcement-learning

In the realm of Deep Reinforcement Learning (DRL), initial efforts focused on processing image pixels to train agents for Atari games, primarily utilizing the Deep-Q-network (DQN) method, as demonstrated in [28]. Various extensions of DQN have been proposed to address its overestimation issue [41, 42]. Conversely, policy gradient methods directly optimize the policy without associating a value function with each action, employing a parameterized function over state features to obtain the policy distribution [37, 43]. Actor-critic, a hybrid approach, integrates both value function and policy gradient methodologies [14, 15]. Transitioning to DRL-driven recommendation systems (RS), state representation techniques vary, encompassing feature extraction for larger item sets and embedding techniques to condense user, item, and context information into a continuous latent space [1, 47]. RNN [47], CNN [13], and GAN [13]-based models are also leveraged to build such embedding. In this work, we employ an encoder-decoder structure for latent state representation. Policy optimization methods in DRL-based recommendations span value-based, policy gradient, and actor-critic approaches, each tailored to different network architectures and action spaces, with considerations for variance, learning speed, and exploration strategies [1, 27, 45].

3 Causal discovery to user-behavior

3.1 Preliminaries

Assume $d_1, d_2, ..., d_T \in D$ denotes the duration and $g_1, g_2, ..., g_T \in G$ intensity of the game-play of a user. We define H (h_i where $i \in T$) as a harmonic mean of $\frac{1}{D}$ and G. H ensures if either metric (such as $\frac{1}{D}$ or G) goes down, the overall metric goes down. This is analogues to the situation when the user lose interest in the platform and the duration increases, or decreases gameplay-intensity. We define the current state-of-engagement of the user using H and our objective of this work is to identify the reasons behind any change in H and attach any treatments that potentially land the users into higher state (or higher H). The objective can be generalised when $h_i \in R^d$ where $d >= 1$ denotes the dimension and we aim to generate treatments to increase long-term h_is. Without loss of generality we assume that the signal h_i is piece-wise stationary, means some characteristics of the process change abruptly at some unknown instants $t_1, t_2, ..., t_K \in T$, where K is unknown. Each of the change point segment in the above process is denoted as $[0, t_1]$, $[t_1 + 1, t_2]$, $[t_2 + 1, t_3]$, ..., $[t_{K-1} + 1, t_K]$ such that $t_1 < t_2 < \ldots < t_K$ and is denoted as $S_1, S_2, \ldots, S_K$.

3.2 Objective

Under the assumption that the strength of an impact of an independent or confounding variable sustains across multiple segments, each of these time-segments can be clustered into C clusters such that all the segments $S_i \in C_j$ have similar distribution and patterns. Hence, our objective is to segment the trajectories such that: (i) if there is any abrupt change in distribution, the corresponding points should belong to different segments (ii) the homogeneous segments should belong to same clusters (iii) the number of segments and number of clusters should be as less as possible. In other words, we aim to find the change points t_i for $i \leq K$ such that

$$\min_{T,C} \Bigg[\lambda_1 \sum_{k=0}^{K} c(h_{t_k}, \ldots, h_{t_{k+1}}) + \lambda_2 \ \text{pen}(K) + \lambda_3 \ \text{pen}(C) + \lambda_4 \sum_{i=1}^{K} \sum_{j=1}^{C} w_{i,j} d(S_i, C_j) + \ \lambda_5 \ \text{expl}(\cdot) \Bigg] \tag{1}$$

such that $0 < C \leq K$, where $h_{t_0} = 0$, $c(\cdot)$ denotes the goodness of fit within the data points present in the corresponding segmentation and pen(K) denotes the complexity of the segmentation which increases if more number of change points are present. Moreover, $d(S_i, C_j)$ denotes the distance between sequence S_i and cluster centroid C_j that defines the clusters. pen(C) denotes the complexity of clustering. Moreover, the constraint ensures that the number of clusters can never be more than the number of segments. expl$(\cdot)$ denotes the cost associated with explainability, and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ denote the corresponding stationary constants.

3.3 Algorithm

The objective function given in Eq. 1 is non-convex as clustering (such as K-means) is an iterative algorithm, and directly optimizing the above function is not possible. In order to convert the objective function tractable, we separately solve the CPD problem and explainable-clustering. In order to find the change points we utilize Pruned Exact Linear Time (PELT) method [20, 40] which dynamically segments H by iteratively extending or creating segments with minimal cost, employing pruning techniques to efficiently explore the solution space.

In Algo 1, line number $3-4$ we initialize all the required parameters. Line number $5-9$ applies gradient descent to estimate the optimal change points T. Line number 6 applies PELT [20, 40] for a given $pen(k)$ which segments H. Note that, if $pen(k)$ increases less number of segments generated and vice-versa. Line number 7 set the cost for change point detection where $c(.)$ denotes the sum squared error with respect to the mean $\mu_{[t_k, t_{k+1}]}$ as $\sum_{\forall i \in n_k} (h_i - \mu_{[t_k, t_{k+1}]})^2$ within each segment and n_k denotes the number of points within

Algorithm 1 Causal discovery

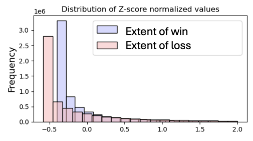
```
1: Input: $D, H$: where $D$ contains independent variables and $H$: dependent variable
2: Output: Threshold-trees $X$
3: Set $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ as hyper-parameters, $\alpha$ as learning rate
4: $pen(K)$ = random(0,1), $\nabla$ =0
5: while repeat until convergence do
6:     $S$ = PELT($H, pen(K)$)
7:     Set cost=$\lambda_1 \sum_{k=0}^{K} c(h_{t_k} \ldots h_{t_{k+1}}) + \lambda_2 \frac{log(pen(k))}{n}$
8:     Estimate finite difference gradient at current point as $\nabla$
9:     $pen(k) = pen(k) - \alpha \ \nabla$
10: end while
11: $S$ = PELT($H, pen(K)$) and $|S| = m$
12: Apply $f_{cl}(S \rightarrow \mathbb{R}^d)$ to obtain $V \in \mathbb{R}^d$
13: return $X$ = Clustering with Threshold trees [29] $(D, S, V)$
```

k^{th} segment. Here, we utilize $\frac{log(pen(k))}{n}$ (where $n = \sum_{\forall i}^{K} n_i$ denotes total number of points) to add normalized penalties over the number of change-points $pen(k)$. In line number 8, we utilize finite difference method [21, 22] to approximate the gradient at the current point. Line number 9 updates the current $pen(k)$ using the gradient and learning-rate.

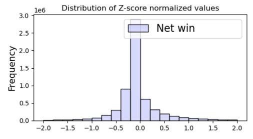
Algorithm 2 Clustering with Threshold trees

```
1: Input: $D$: containing independent variables, $S$: set of segments, $V \in \mathbb{R}^d$ for each segment in $S$
2: Output: Root of the threshold tree $X$
3: Set hyper-parameters : $\nabla_c$ for gradient clipping, $\alpha \in \mathbb{Z}$ for learning rate
4: Set $n_c = 1$ for number of clusters and $n'_c = 1 + \alpha$
5: while repeat until rate of decrease of error <= $\nabla_c$ do
6:     Apply K-means on $V$ to obtain cluster means $\mu_1, \ldots, \mu_{n_c} \leftarrow$ k-Means($V, n_c$) and $\mu'_1, \ldots, \mu_{n'_c} \leftarrow$ k-Means($V, n'_c$)
7:     Assign each data point $v_j \in V$ to it's closets cluster center $y_j$ from K-means($V, n_c$) to estimate MSE($v_j - y_j$) $\forall v_j \in V$ as MSE($V, n_c$)
8:     Assign each data point $v_j \in V$ to it's closets cluster center $y'_j$ from K-means($V, n'_c$) to estimate MSE($v_j - y'_j$) $\forall v_j \in V$ as MSE($V, n'_c$)
9:     Estimate rate of decrease of error as $\frac{1}{\alpha}(MSE(V, n_c) - MSE(V, n'_c))$
10:    $n_c = n'_c$ and $n'_c = n'_c + \alpha$
11: end while
12: Set $\{\mu_1, \ldots, \mu_{n'_c}\}, \{C_1, \ldots, C_j\} \leftarrow$ k-Means($V, n'_c$)
13: return threshold tree $X$ = build_tree($\{v_j\}_{j=1}^{m}, \{C_j\}_{j=1}^{m}, \{\mu_j\}_{j=1}^{n'_c}$)
```

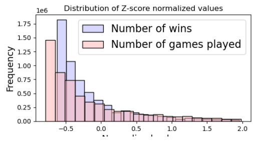
Once we obtain the segments after applying PELT with optimal parameter $pen(k)$ in line number 11, we aim to cluster the segments in S so that they can be explainable. In order to cluster the segments, we utilize unique signature of each segments such as convexity, length, variation, mean, and more. The function $f_{cl}(S \rightarrow R^d)$ maps each segment into d-dimensional signature (V) as shown in line number 12 (refer Sec 3.4). Next we utilized a modified iterative minimization Algo [29] to simultaneously cluster and create threshold-trees for explanability in line number 13. This method combines decision trees with clustering objectives to achieve both interpretability and clustering efficiency. By iteratively applying threshold cuts based on features, the algorithm aims to achieve a good approximation on the optimal clustering while maintaining interpretability [29]. Line number 5 to 11 in Algo. 2 increases the number of clusters (at an interval of $\alpha \in \mathbb{Z}$) until the rate of decrease of error is less than ∇_c to find the optimal number of clusters. The details of the function build_tree($\{v_j\}_{j=1}^{m}, \{y_j\}_{j=1}^{m}, \{\mu_j\}_{j=1}^{n'_c}$) is given in [29]. The build-tree function provides an $O(k^2)$ approximation to the optimal k-means with the threshold-trees for explanability (refer [29]).



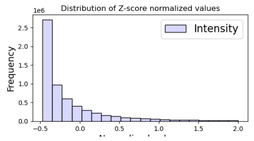
(a) Extent of win vs extent of loss



(b) Net win



(c) Number of games played



(d) Intensity of gameplay

Figure 1: Distribution of data

In this work, we utilize the signatures V for clustering objective whereas the features in D for interpretability. Finally, the output is generated as small decision trees (also known as threshold-trees) with k leaves to achieve explainable cluster assignments. Note that, the cost function given in line number 7 as $\lambda_1(\sum_{\forall i \in n_1}(h_i - \mu_{t_1})^2 + \sum_{\forall i \in n_2}(h_i - \mu_{t_2})^2 + \ldots + \sum_{\forall i \in n_k}(h_i - \mu_{t_k})^2) + \lambda_2 \frac{log(pen(k))}{n}$ is convex as the corresponding hessian matrix $\begin{bmatrix} 2n_1 & 0 & 0 & 0 \\ 0 & 2n_2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 2n_k \end{bmatrix}$ (where n_i denotes the corresponding number of points in segment i) is positive semi-definite. Similarly, it can be shown that the Algo 1 is also convex except line number 13 which shown to approximate k-means clustering with provable convergence guarantee [29].

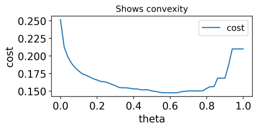
3.4 Data

The H-metric increases and decreases at certain times and the objective of the work is to identify the reasons behind the changes. Overall autocorrelation is less without any notable significant to prove that the H-metric is not correlated to itself (refer Fig. 3).

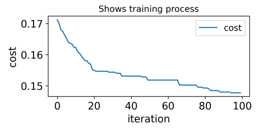
Table 1: Feature description

Feature Name	Rationale	Type
Number of games played	Increases skill	
Play propensity	Advances skill / loses interest if early	
Extent of loss	Discourages player / leads to deposit	
Skill		
Bonus Given		
Extent of win	Increases platform stickiness	
Number of wins		
Net win		Independent variables
Intensity	Captures engagement	
Duration	Captures if loses interest	Dependent variables

Table 1 shows various features utilized for the analysis and the rationale behind including the same. We collected 9 months of game-play, transactional and demographic data ($\sim 2M$ across $200K$ users). The distribution of the data is shown in Fig. 1. The winning cash amount and lost amount follows exponential-decay distribution because the probability of a user winning a very high amount decreases exponentially (refer 1a). Similar trend follows with Fig. 1c and Fig. 1d for the number of winning-games and intensity. The net win is obtained by subtracting the extent of lost from the extent of win, and hence Fig. 1b follows normal distribution.



(a) Cost vs $pen(k)$



(b) Cost vs iteration

Figure 2: Training cost

Table 2: Hyperparameters

Causality	Value	Engagement Max	value
Learning rate α	0.05	Learning rate	0.00005
max-iteration	500	Optimizer	Adam
tolerance for convergence	$1 \times e^{-3}$	action size	4
∇_c	0.05	state feature size	~120
$\lambda_1, \lambda_2, \lambda_3, \lambda_4$	1	#iterations	500
quadratic func	$a x^2 + bx +c	#epochs in each iteration	5
PELT-jump	5	epsilon_min	0.01
min-seg-length	2	epsilon_decay	0.05
$pen(k)$	0.58	update target weights	once in every 10 iterations
		time-lag	6

3.5 Results

The column 'causality' in Tab. 2 ('causality') shows the hyper-parameters utilized in the work. In order to show the Algo 1 follows convexity, we vary $pen(k)$ (as *theta*) and plot the corresponding cost in Fig. 2a (line number 7 in Algo 1). As we can see that the cost initially decreases and reaches the optimal value at 0.15 with corresponding $pen(k)$ as 0.58. The optimal $pen(k)$ value is captured in Fig. 2b.

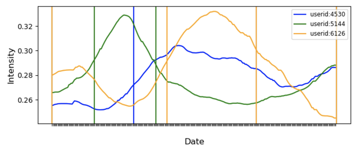
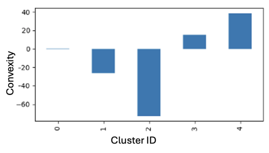
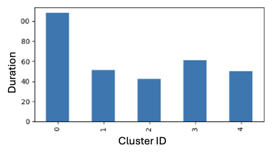


Figure 3: Shows optimal change points on users' trajectories

Fig. 3 shows the corresponding optimal change points for 3 different users where the y-axis denotes the H-metric and x-axis denotes the dates, and the corresponding vertical lines show the change-points. In test set the optimal cost is 0.61. Note that each segment S_i within each trajectory possess unique signature. Either the overall mean value of the metric is high (means users engagement is high), or the variability is low (users do not deviate in terms of engagement), or length of the segment is high with low deviation (user maintains the engagement-autograph stationary for long-time). In this work, we consider *convexity* (when the user gradually increasing vs when the user is decreasing) and *duration* as the signatures for clustering.



(a) Convexity

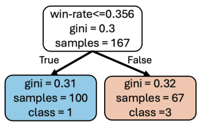


(b) Duration

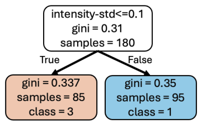
Figure 4: Signature of approximate clusters

Convexity is determined by fitting a quadratic function: if the value is less than zero, the segment is concave; if more than zero, it is convex; and if close to zero, it is neither. When a segment is more convex, the fluctuating intensities of engagement or H-metric increases as well. Figure 4 shows the signatures of the optimal clusters after applying Algo. 2. Among the five optimal clusters, clusters 1 and 3 exhibit distinct long-duration signatures. We focus on long-duration segments to address the correlation versus causation dilemma. Our goal is to determine if an event at the start of a segment causes a distinct user behavior during the segment, making the impact of causation more evident in long segments. To generate threshold trees, we use dataset D, which contains independent variables (and their derivatives) listed in Table 1, emphasizing values at the beginning of each segment.

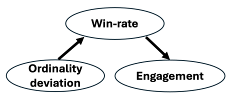
Fig. 5a provides the corresponding threshold tree (with 79% as test accuracy in 169 segments), showing that the feature *win-rate* causes the segments into class-1 or class-3. The *win-rate* is calculated by dividing the number of win-games by the number of games played. If the win-rate is affected, this eventually impacts the psychology of the players that governs interest in the platform. Next, we aim to determine the causes of the win-rate being affected while setting win-rate as the H-metric, revealing that ordinality standard deviation has the most impact (refer Fig.5b). Higher ordinal games involve higher skills, attracting skillful players, while novice players engage in low ordinal games. As users develop skills, their ordinality increases. Excessive losses may lead to user churn, as inappropriate intensity or ordinality selection reduces platform interest, negatively impacting engagement. In contrary, a slow and gradual increase in ordinality leads to a healthy engagement. Thus, ordinality selection impacts win-rate which impacts user-engagement (refer Fig.5c). Hence, in section 4 we aim to develop a personalized ordinality recommendation system using RL to optimize the user engagement.



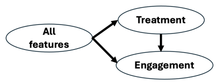
(a) H: engagement



(b) H: win-rate



(c) Causal impact flows



(d) di-graph

Figure 5: Threshold-trees and causal graphs

3.6 Ablation study

We apply regression [35], propensity score stratification [16], weighting [24], and matching [33] techniques while setting the dependent column as engagement and independent columns as shown in Tab. 1 to estimate the impact of causal-effect. Further, we do not have any treatment information available, and hence initially we set a random treatment (which generates a random number $\in [0,1]$) to conduct the study (setting a significance threshold of 0.05). The corresponding di-graph is shown in Fig. 5d. Next, instead of setting a random treatment, we set the treatment as the features given in Tab 1 while varying the techniques and show the most significant results in Tab 3. However, the game-play data is extremely volatile and the methods fails to capture the causality (refer significant-test columns which shows 'FALSE') as very weak causal relationship is present. Similarly, instrumental variable test [25] can not detect any weak-causal relations as well (coefficient 0 in Tab 4).

Table 3: Propensity Score-based causal effect estimation

	Point estimate	**Standard Error**	**Significance Test**	**Treatment**
Regression	-0.0001	0.0001	**FALSE**	Random
Propensity Score - Stratification	-0.0007	0.0002	**FALSE**	Win-rate
Propensity Score - Matching	0.0020	0.0004	**FALSE**	Intensity-std
Propensity Score - Weighting	0.0002	0.0002	**FALSE**	loss-extent

Table 4: Instrumental variable

	Coefficient	Standard Error	t-value	p-value
win-extent	**0.00**	0.00	-4.86	0.00
games	**0.00**	0.00	0.33	0.74
bonus	**0.00**	0.00	2.09	0.04
win rate	**0.00**	0.00	-3.72	0.00
intensity std	**0.00**	0.00	12.91	0.00

We apply Granger causality based techniques while varying the lag between 1 to 5 and conducted several tests such as liklihood ratio, chi-square and F-test, and compare the statistics obtained along with corresponding p-value in Tab. 5. As we can see Granger test establishes causality with low-confidence (p-value: 0.09) between intensity standard deviation and engagement with lag = 4.

Table 5: Granger Test

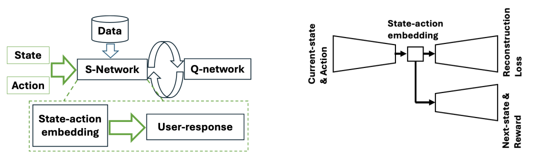
Independent Column	**Lag**	**Test Type**	**Statistic**	**p-value**
win-extent	5	ssr-chi2test	3.47770822	0.62676331
win-extent	5	lrtest	3.47762306	0.62677622
loss-extent	1	ssr-ftest	1.97473551	0.15995048
loss-extent	1	lrtest	1.97479148	0.15994019
loss-extent	1	params-ftest	1.97473551	0.15995048
bonus	5	ssr-chi2test	5.60298143	0.34678568
bonus	5	lrtest	5.60276039	0.34680935
intensity std	**4**	**ssr-ftest**	**1.96584088**	**0.09672607**
intensity std	**4**	**ssr-chi2test**	**7.86436026**	**0.09667514**
intensity std	**4**	**lrtest**	**7.86392481**	**0.09669192**
intensity std	**4**	**params-ftest**	**1.96584088**	**0.09672607**

4 Engagement Maximization

In this section, we aim to create a personalized, automated ordinality recommendation agent for a skill-gaming platform. The agent aims to maximize: (i) user-centric objectives, providing the most relevant ordinality for the user's persona and intent (or relevancy), and (ii) platform-centric objectives, increasing user engagement.

4.1 Preliminaries

We use a Markov decision process (MDP), defined as $M = (S, A, T, r, \gamma)$, where S is the set of states, A is the set of actions, $T(s_{t+1}|s_t, a_t)$ is the transition probability, $r : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The agent aims to learn the optimal policy π_* that maximizes returns. The value of a state-action pair, $q_\pi(s, a) = \mathbb{E}_\pi[R_t \mid s_t = s, a_t = a]$, is the expected return from state s with action a under policy π. The optimal Q-value,



(a) Architecture (b) State-action embedding

Figure 6: Collaborative S and Q-netowrk based learning

$q_*(s,a) = \mathbb{E}_{s'}\,[r_{t+1} + \gamma\, \max_{a'} q_*(s',a')]$, represents the maximum achievable expected return from state s while taking action a under the current policy. Specifically, M is defined as follows:

State (s_t): State $s_t \in S$ includes the user's current wallet balance, previous ordinality-distribution, user's demographic information, win-rate, lost-rate, number of practice games, and many more (refer Table. 1).

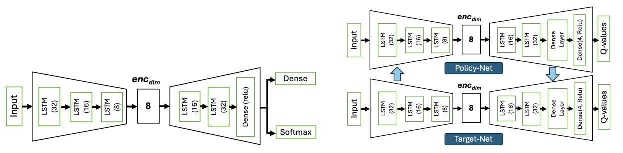
Action ($a \in [0,1,2,3]$): The action $a_t^{rec} \in A$ denotes the recommended ordinality at time t. Instead of recommending the ordinality (ranges from 0 to 9) directly, we recommend the jump from the last ordinality where 0 means no-jump and 3 means 3-jumps.

Reward(r): The instantaneous engagement (which is a combination of both the duration (or frequency) and intensity or ordinality of the game-play) is denoted as the current reward (r).

The agent finds the optimal policy (π_*) that maximizes the expected achievable return $E_\pi[R_t|s_t = s]$ for all state-action pair that follows Bellman equation: $Q_*(s,a) = E_{s'}[r + \gamma\, max_{a'} Q_*(s',a')]$. As the state-action space is huge, deep neural network based approximators ($Q_*(s,a^{rec}) \approx Q_*(s,a^{rec};\theta)$) are used by minimizing the following loss function [9, 47]

$$L(\theta) = E_{s,a^{rec},r,s'}\,[(y - Q_*(s,a^{rec};\theta))^2], \tag{2}$$

where $y = E_{s'}[r + \gamma\, max_{a'} Q_*(s',a')]$ is the target for the current iteration. This technique is known as deep Q-network (DQN) [9, 47]. Further, to increase the stability of the network, current updated weights are kept fixed and fed into a second (duplicate) network (known as target network) whose outputs are used as Q-learning target. In other words, $max_{a'} Q_*(s',a';\theta)$ comes from the target network (with weights θ') and replaced by $max_{a'} Q_*(s',a';\theta')$. In order to alleviate the overestimation problem induced in DQN, Double-DQN [41] replaces the target value as $y = E_{s'}[r + \gamma \times Q_*(s', arg\,max_{a'} Q_*(s',a';\theta);\theta')]$. Further to avoid the over-fitting to narrow peaks in the value function, y is replaced by $y = E_{s'}[r + \gamma\, Q_*(s', arg\,max_{a'} Q_*(s',a';\theta);\theta') + \epsilon]$ where $\epsilon \in \text{clip}(\mathcal{N}(0,\ \sigma))$ [12].



(a) Encoder-decoder (b) Policy and Target Network

Figure 7: Architecture of Encoder-decoder and Q-network

4.2 Algorithm

Overall offline framework is shown in Fig. 6 which consists of S-network and Q-netowrk [50]. The S-network is responsible to capture the user-dynamics and assist the Q-network during training. The S-network consists of state-action embedding to capture the hidden representation of the user-dynamic. The input to the S-network is state-action embedding whereas the output is next-state and reward. The Q-network interacts with the S-network to find the optimal policy that generates maximum engagement.

To include current-state information, we use the reconstruction loss in an auto-encoder (*ae*) as shown in Fig.6b. The encoder-decoder (*ed*) captures user dynamics by generating the next-state and reward pair. Algo.3 illustrates the collaborative training of the S-network and Q-network. The *build_ed*(...) procedure trains the S-network, while the *DDQNNC*(...) procedure handles the collaborative training of the S and Q-network.

Algorithm 3 Offline training

```
1:  $build\_ed(enc_{dim}, epochs, D)$ :
2:  $ae$= create instance of autoencoder with random weights, $ed$= create instance of
    encoder-decoder, $enc_{dim}$= dimension of the embedding
3:  for $e = 1, 2, \ldots, epochs$ do
4:      while for every batch of $(s, a, s', r) \in D$ do
5:          Train $ae$ with $(s, a)$ using reconstruction loss
6:          Copy weights from $ae$ to $ed$
7:          Train $ed$ using Eq. 3
8:          Copy weights from $ed$ to $ae$
9:          return $ae, ed$
10:     end while
11: end for
12:
-------------------------------------------------------------------------
1:  $DDQNNC(ae, ed, D)$ :
2:  $ae, ed = build\_ed(enc_{dim}, epochs, D)$
3:  Initialize policy and target network with $ed$ as $Q_{net}$
4:  while do until convergence do
5:      Collect batches of offline data from $D$
6:      for $batch = 1, 2, \ldots, N$ do
7:          for each new training sample $(s)$ in $batch$ do
8:              for $i = 1, 2 \ldots, T$ do
9:                  Set $a^{rec}$ = action from explore/exploitation
10:                 $(s', intensity, dur) = ed(s, a)$
11:                 if $dur > d_{th}$ then $break$
12:                 estimate $r$ from $dur$ and $intensity$
13:                 Store $(s, a, s', r)$ in buffer $M$
14:                 update $s = s'$
15:             end for
16:         end for
17:         Train policy network ($Q_{net}$) using Data $M$ and with Eq. 2
18:         After some iteration, copy weights from policy network to target network
19:         $ae, ed = build\_ed(enc_{dim}, epochs, lr, \alpha, batch\_size, M)$
20:     end for
21: end while
```

The *ae* (auto-encoder) and *ed* (encoder-decoder) are initialized with random weights in the *build_ed*(...) procedure. Both share the same architecture and use the state-action embedding dimension (enc_{dim}). Fig. 7a shows the encoder-decoder network architecture, consisting of LSTM layers followed by a dense layer to predict the next state and reward. The S-network predicts the step-increase from the last ordinality, ranging from 0 to 3, where 0 and 1 denote no increase and a 1-step increase, respectively. The cost function $Cost(ed)$ is:

$$Cost(ed) = \sum_{\forall v} MSE(v_{act}, v_{pred}) + \sum_{\forall u} Softmax(u_{act}, u_{pred}) \tag{3}$$

where $v \in \{wallet\ balance, duration\}$, $u \in \{step\ increase, win\_loss, practice\ \ games\}$. MSE is used for floating-point values (wallet-balance, duration), and Softmax is used for probabilities (step-increase, win-loss) since only one step-increase can be chosen, and a game can be won or lost. The number of practice games ranges from 0 to 3. The policy/target network is similar to the encoder-decoder but includes an additional dense layer that produces Q-values for four actions (recommend the user play with the last ordinality (LO) or a 1/2/3-step increase; see Fig.7b). The agent cannot recommend less than the last ordinality. During each epoch, a batch of samples is collected from the dataset D. In the $build_ed$ procedure in Algo 3, samples of (s, a, s', r), representing current state, action, next state, and reward (estimated from dur and $intensity$), are collected to train the auto-encoder and encoder-decoder in the S-network. The auto-encoder uses reconstruction loss, while the encoder-decoder uses Eq. 3. After each training of the ae, the encoder weights are copied to the ed network (line 6) to ensure the hidden representation includes both current and next state information (line 8).

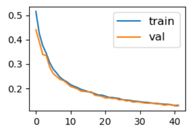
The Double-DQN-with-noise-Clipping (DDQNNC) procedure handles collaborative training of the S and Q networks using the auto-encoder (ae) and encoder-decoder (ed). In each iteration, offline data containing the current state (s) and user information is collected. The S-network simulates user dynamics for each s and action (a) from exploration or exploitation, producing the next state and reward for training the Q-network. The ed returns the next state, $duration$, and $intensity$ for a given s and a, and the reward is estimated from dur and $intensity$. If the user is churning, dur becomes very high, breaking the loop (line 11). The new state and action are stored in buffer M for training both the Q (line 17) and S (line 18) networks. To debias logged data, importance sampling ($\frac{\pi(a_i|s)}{\pi_b(a_i|s)}$, where π is the Q-net policy and π_b is the base policy from data D) is used during S-network training.

4.3 Offline results

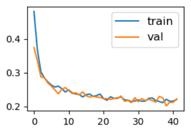
Along with the features in Tab.1, the sequential data includes current wallet information and past intensity of game-play (refer Fig. 1d) to train the S and Q-networks. Users typically prefer to play the last ordinality (LO), with selections more than 3 steps from the last being very rare (less than 1%). Therefore, we consider action $a \in [0, 1, 2, 3]$. Fig.8a and Fig.8b illustrate the training processes for the auto-encoder and encoder-decoder, respectively. The hyper-parameters are shown in Tab2 under 'engagement max'. The X-axis represents the number of iterations (line 4 in DDQNNC), while the Y-axis shows the reconstruction error for the auto-encoder (Fig.8a) and the error from Eq.3 for the encoder-decoder (Fig.8b). Both errors decrease over time in validation and training sets, indicating convergence of Algo3. By iteration 50, ae and ed training converges, and the Q-network begins learning the optimal policy. Fig.8 shows mean reward per epoch (engagement) increasing after iteration 150 as the agent spends 100 more iterations learning the optimal policy. Fig.8d shows mean cumulative reward increasing over time, with the amber line indicating the agent's recommendations outperform random recommendations.

4.4 Online results

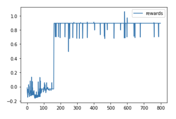
The deployed architecture comprises components like front-end for UI generation, tech service for event triggers, data commission layer for data flow management, online model training using AWS and Sagemaker [11], and ML life-cycle platforms for real-time inference. Tools like KAFKA [26], Millow, Yellowbrick, and Tableau support monitoring [19], while AWS/S3 stores logs and facilitates model updates.



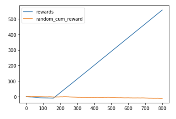
(a) Auto-encoder error



(b) Encoder-decoder error



(c) Mean reward

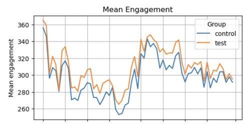


(d) Mean cumulative reward

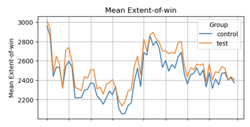
Figure 8: S-network based results

Table 6: Online performance

	Daily mean engagement	Avg engagement per user	Daily mean extent-of -win	Avg extent-of-win per user	Relevancy for played <= LO	Relevancy for played >LO
Control	0.304	0.435	0.249	0.356	–	–
Test	0.310	0.448	0.253	0.377	0.7	0.770
% change	1.974	3.057	1.782	5.854	–	–



(a) Mean engagement



(b) Mean extent-of-win

Figure 9: Increasing engagement

Fig 9 displays the mean daily engagement of both a control and a test path, with the test path exhibiting higher engagement. This increase is achieved by elevating the mean extent-of-win (see Fig. 9b). As the mean extent-of-win rises, user interest in the platform grows, leading to enhanced engagement over time. Tab. 6 shows a 2% increase in daily mean engagement and a 3% rise in average engagement per user in the test path, due to an increase in the average extent-of-win per user. This is achieved by improving relevancy for potential users who wants to play with higher ordinality. Relevancy, measured by the percentage of users following the recommended ordinality, is achieved by 77% among users naturally inclined to increase their gameplay intensity.

5 Conclusion and Future Work

We personalize and recommend game intensity to make a gaming platform more interesting. This results into 3% higher average engagement and corresponding higher extent-of-win. Future work involves adding more explainability for each recommendation given to a user.

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] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information systems* 53 (2015), 16–38.

[3] Chainarong Amornbunchornvej, Elena Zheleva, and Tanya Y Berger-Wolf. 2019. Variable-lag granger causality for time series analysis. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 21–30.

[4] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[5] Surajit Chakrabarty, Rukma Talwadker, and Tridib Mukherjee. 2021. ScarceGAN: Discriminative Classification Framework for Rare Class Identification for Longitudinal Data with Weak Prior. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 140–150.

[6] Arthur Chatton, Florent Le Borgne, Clémence Leyrat, Florence Gillaizeau, Chloé Rousseau, Laetitia Barbin, David Laplaud, Maxime Léger, Bruno Giraudeau, and Yohann Foucher. 2020. G-computation, propensity score-based methods, and targeted maximum likelihood estimator for causal inference with different covariates sets: a comparative simulation study. *Scientific reports* 10, 1 (2020), 9219.

[7] Linda M Collins, Susan A Murphy, and Victor Strecher. 2007. The multiphase optimization strategy (MOST) and the sequential multiple assignment randomized trial (SMART): new methods for more potent eHealth interventions. *American journal of preventive medicine* 32, 5 (2007), S112–S118.

[8] Rajeev H Dehejia and Sadek Wahba. 2002. Propensity score-matching methods for nonexperimental causal studies. *Review of Economics and statistics* 84, 1 (2002), 151–161.

[9] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. 2020. A theoretical analysis of deep Q-learning. In *Learning for Dynamics and Control*. PMLR, 486–489.

[10] Scott Fitzgerald. 2019. Over-the-top video services in India: Media imperialism after globalization. *Media Industries Journal* 6, 2 (2019), 00–00.

[11] Chris Fregly and Antje Barth. 2021. *Data Science on AWS*. " O'Reilly Media, Inc.".

[12] 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.

[13] Rong Gao, Haifeng Xia, Jing Li, Donghua Liu, Shuai Chen, and Gang Chun. 2019. DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1048–1053.

[14] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 1291–1307.

[15] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. 2020. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters* 5, 4 (2020), 6217–6224.

[16] Jason S Haukoos and Roger J Lewis. 2015. The propensity score. *Jama* 314, 15 (2015), 1637–1638.

[17] Miguel A Hernán, Babette Brumback, and James M Robins. 2001. Marginal structural models to estimate the joint causal effect of nonrandomized treatments. *J. Amer. Statist. Assoc.* 96, 454 (2001), 440–448.

[18] Daniel N Hill, Robert Moakler, Alan E Hubbard, Vadim Tsemekhman, Foster Provost, and Kiril Tsemekhman. 2015. Measuring causal impact of online actions via natural experiments: Application to display advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1839–1847.

[19] Bibhudutta Jena. 2019. An Approach for Forecast Prediction in Data Analytics Field by Tableau Software. *International Journal of Information Engineering & Electronic Business* 11, 1 (2019).

[20] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *J. Amer. Statist. Assoc.* 107, 500 (2012), 1590–1598.

[21] Tatiana Kossaczká, Matthias Ehrhardt, and Michael Günther. 2023. Deep FDM: Enhanced finite difference methods by deep learning. *Franklin Open* 4 (2023), 100039.

[22] Randall J LeVeque. 1998. Finite difference methods for differential equations. *Draft version for use in AMath* 585, 6 (1998), 112.

[23] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[24] Fan Li, Kari Lock Morgan, and Alan M Zaslavsky. 2018. Balancing covariates via propensity score weighting. *J. Amer. Statist. Assoc.* 113, 521 (2018), 390–400.

[25] Giampiero Marra and Rosalba Radice. 2011. A flexible instrumental variable approach. *Statistical Modelling* 11, 6 (2011), 581–603.

[26] Cristian Martín, Peter Langendoerfer, Pouya Soltani Zarrin, Manuel Díaz, and Bartolomé Rubio. 2022. Kafka-ML: Connecting the data stream with ML/AI frameworks. *Future Generation Computer Systems* 126 (2022), 15–33.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[28] 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.

[29] Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. 2020. Explainable k-means and k-medians clustering. In *International conference on machine learning*. PMLR, 7055–7065.

[30] Koyel Mukherjee, Deepanshi Seth, Prachi Mittal, Nuthi S Gowtham, Tridib Mukherjee, Dattatreya Biswas, and Sanjay Agrawal. 2022. ComParE: A User Behavior Centric Framework for Personalized Recommendations in Skill Gaming Platforms. In *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*. 186–194.

[31] Abiodun E Onile, Ram Machlev, Eduard Petlenkov, Yoash Levron, and Juri Belikov. 2021. Uses of the digital twins concept for energy services, intelligent recommendation systems, and demand side management: A review. *Energy Reports* 7 (2021), 997–1015.

[32] Stephen O'Neill, Noémi Kreif, Richard Grieve, Matthew Sutton, and Jasjeet S Sekhon. 2016. Estimating causal effects: considering three alternatives to difference-in-differences estimation. *Health Services and Outcomes Research Methodology* 16 (2016), 1–21.

[33] Justus J Randolph and Kristina Falbe. 2014. A step-by-step guide to propensity score matching in R. *Practical Assessment, Research & Evaluation* 19 (2014).

[34] Debanjan Sadhukhan, Sachin Kumar, Swarit Sankule, and Tridib Mukherjee. 2023. t-RELOAD: A REinforcement Learning-based Recommendation for Outcome-driven Application. In *Proceedings of the Third International Conference on AI-ML Systems*. 1–7.

[35] Yonadav Shavit, Benjamin Edelman, and Brian Axelrod. 2020. Causal strategic linear regression. In *International Conference on Machine Learning*. PMLR, 8676–8686.

[36] Ali Shojaie and Emily B Fox. 2022. Granger causality: A review and recent advances. *Annual Review of Statistics and Its Application* 9 (2022), 289–319.

[37] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[38] Rukma Talwadker, Surajit Chakrabarty, Aditya Pareek, Tridib Mukherjee, and Deepak Saini. 2022. Cognitionnet: A collaborative neural network for play style discovery in online skill gaming platform. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3961–3969.

[39] Zhiqiang Tan. 2006. Regression and weighting methods for causal inference using instrumental variables. *J. Amer. Statist. Assoc.* 101, 476 (2006), 1607–1618.

[40] Sam O Tickle, IA Eckley, Paul Fearnhead, and Kaylea Haynes. 2020. Parallelization of a common changepoint detection method. *Journal of Computational and Graphical Statistics* 29, 1 (2020), 149–161.

[41] 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.

[42] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.

[43] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[44] Liuyi Yao, Zhixuan Chu, Sheng Li, Yaliang Li, Jing Gao, and Aidong Zhang. 2021. A survey on causal inference. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 5 (2021), 1–46.

[45] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1201–1208.

[46] Alessio Zanga, Elif Ozkirimli, and Fabio Stella. 2022. A survey on causal discovery: Theory and practice. *International Journal of Approximate Reasoning* 151 (2022), 101–129.

[47] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.

[48] Jichen Zhu and Santiago Ontañón. 2020. Player-centered AI for automatic game personalization: Open problems. In *International Conference on the Foundations of Digital Games*. 1–8.

[49] Seyedjamal Zolhavarieh, Saeed Aghabozorgi, Ying Wah Teh, et al. 2014. A review of subsequence time series clustering. *The Scientific World Journal* 2014 (2014).

[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.