Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082845

ARTICLE

Hybrid-RL: An Incremental Deep Clustering Framework with Reinforcement Learning for Adaptive Customer Segmentation

Anh Thi Diem Nguyen[1,2,#], **Tham Vo**[1] **and Vinh Truong Hoang**[3,*]

[1]Faculty of Information Technology, Nguyen Tat Thanh University, 300A Nguyen Tat Thanh Street, Xom Chieu Ward, Ho Chi Minh City, Vietnam

[2]Faculty of Information Technology, Van Lang School of Technology, Van Lang University, 69/68 Dang Thuy Tram Street, Binh Loi Trung Ward, Ho Chi Minh City, Vietnam

[3]Faculty of Information Technology, Ho Chi Minh City Open University, 35–37 Ho Hao Hon Street, Cau Ong Lanh Ward, Ho Chi Minh City, Vietnam

*Corresponding Author: Vinh Truong Hoang. Email: vinh.th@ou.edu.vn

[#]Email: 2400009706@nttu.edu.vn

Received: 24 March 2026; Accepted: 29 April 2026; Published: 15 June 2026

ABSTRACT: Keeping customers engaged remains a major challenge in appointment-based services, where user behavior continuously shifts due to seasonal, market, and social factors. These dynamic changes often cause concept drift, rendering traditional deep clustering models unreliable because they assume stable data distributions. Most existing approaches handle representation learning, parameter optimization, and model updating as separate components, limiting their adaptability in real-world streaming environments. This study proposes Hybrid-RL, a novel adaptive clustering framework that unifies incremental deep representation learning, multi-head reinforcement learning for joint hyperparameter optimization (number of clusters, latent dimension, and clustering method), incremental model updating, bandit-based decision making, surrogate-model explainable artificial intelligence (XAI), and continuous Gini-based fairness monitoring within a single closed-loop pipeline. The framework updates incrementally via autoencoder fine-tuning and MiniBatchKMeans partial_fit without requiring full retraining, enabling efficient adaptation to evolving customer behavior. Experiments conducted on real proprietary appointment data (10,212 records collected from 2021 to 2025) with natural concept drift demonstrate that Hybrid-RL achieves superior clustering quality, recording a Silhouette score of 0.7542, Davies–Bouldin Index (DBI) of 0.3150, and Calinski–Harabasz (CH) index of 1810.34, while maintaining an ultra-low inference time of 0.0001 s per sample. The model significantly outperforms 13 baseline methods. Under controlled synthetic drift, Hybrid-RL exhibits only a 6.1% drop in Silhouette score, confirming strong robustness. Additional validation on the public UCI Online Retail dataset further confirms the framework's generalizability. Fairness analysis reports an average Gini coefficient of 0.49 across clusters, indicating balanced action distribution.

KEYWORDS: Incremental clustering; reinforcement learning; adaptive segmentation; concept drift; explainable AI; customer retention optimization

1 Introduction

The rapid growth of digital platforms has led to a large increase in data across service industries, especially in appointment-based services such as personal care, healthcare, and beauty services. In these systems, transaction logs capture information such as booking time, spending, cancellation frequency, return intervals, and service types. These patterns do not stay fixed. They change with seasons, market conditions,

and social factors, so customer behavior tends to shift over time. Because of this, the data distribution is often unstable, and models trained on historical data can lose performance when concept drift occurs [1]. Most appointment platforms still focus mainly on scheduling and operational management. Tools for analyzing customer behavior are still limited. The main issue is not the lack of data, but the ability to adapt to changes in data distribution. In this setting, clustering is often used to explore hidden customer structures and support decision making. However, its effectiveness depends on how well the model can learn meaningful representations and remain stable as the data changes. Deep clustering has become a common approach for high-dimensional behavioral data. Models such as DEC learn a latent space using an autoencoder and optimize cluster assignments through divergence-based objectives [2,3], IDEC extends this idea by adding a reconstruction constraint to improve stability [4,5], VaDE models the latent space as a Gaussian mixture using variational inference [6–8]. These models usually perform well when the data distribution is stable. In practice, this assumption often does not hold, which limits their performance under concept drift.

To deal with this issue, several studies have explored reinforcement learning for dynamic parameter adjustment. The CRL model uses multi-agent reinforcement learning to handle clustering in changing environments [9,10]. Q-DE-KMeans combines Q-learning, Differential Evolution, and K-Means to search for better hyperparameters [11]. The Clustering with Attention-based Temporal Sequence (CATS) model integrates clustering with GRU and attention mechanisms to process sequential data [12]. These approaches improve parameter tuning or temporal modeling, but they still do not provide a stable way to update the model when new data continues to arrive.

At the same time, other work focuses on incremental clustering for streaming data. Methods such as MiniBatchKMeans and UIClust can process data streams efficiently [13], DynamicSupervisedClustering manages cluster evolution using weight decay and aging mechanisms [14]. These approaches can detect drift and update clusters, but parameter tuning is still largely based on heuristics. In addition, multi-objective optimization is not tightly integrated into the clustering process.

Recent studies also point out the growing importance of explainable AI, transfer learning, and data protection in customer analytics [15]. In many practical systems, explainability is added after clustering. Fairness is often not considered within the main pipeline. This creates a gap between research methods and real-world deployment.

Taken together, these research directions show clear limitations. Deep clustering provides strong representation learning but does not adapt well to streaming data. Incremental clustering can handle concept drift but does not fully benefit from deep representations. Reinforcement learning methods mainly focus on parameter tuning and are not deeply integrated into clustering architectures. As a result, there is still no unified framework that can handle representation learning, parameter optimization, and cluster updating at the same time.

To address this gap, this study proposes Hybrid-RL, an adaptive clustering framework that not only unifies deep representation learning, reinforcement learning-based optimization, and incremental updating in a single pipeline, but also introduces several key technical improvements over existing methods. The main contribution of this work is Hybrid-RL—the first unified pipeline that jointly performs incremental deep representation learning, multi-head reinforcement learning for joint hyperparameter optimization (K, latent dimension, and clustering method), incremental model updating, bandit-based decision making, surrogate-model explainable artificial intelligence, and continuous Gini-based fairness monitoring in a single closed-loop system. Instead of treating these components separately, the proposed approach allows them to work together under changing data distributions. A multi-head reinforcement learning network is used to adjust the number of clusters, the latent dimension, and the clustering method. This reduces the need for manual tuning. The model is updated incrementally as new data arrives, so it does not need to be retrained

from scratch. Beyond clustering, the framework also supports decision making through a multi-armed bandit mechanism and provides explanations through a surrogate model. Fairness is monitored throughout the pipeline. With this design, Hybrid-RL provides a more practical solution for customer behavior analysis in dynamic environments.

2 Relevant Literature

Deep clustering is often used to find structure in high-dimensional data. It combines representation learning with clustering in the same framework. Early models such as DEC, IDEC, and VaDE follow this idea. DEC learns a latent space and refines cluster assignments by minimizing Kullback-Leibler (KL) divergence [2,3], IDEC keeps a reconstruction term to preserve local structure [4,5], VaDE models the latent space as a Gaussian mixture using variational inference [6–8]. Recent studies have attempted to extend deep clustering to more realistic settings. For example, data-driven customer segmentation frameworks incorporating behavioral signals such as cart abandonment have been proposed to better capture dynamic user intent [16]. These approaches highlight the importance of integrating behavioral patterns into clustering, especially in e-commerce and service platforms. These models are typically developed under the assumption of static data distributions. As a result, they often struggle to maintain clustering quality when data evolves over time, limiting their applicability in real-world scenarios with concept drift.

Some studies try to include temporal information. Models such as CATS use GRU (Gated Recurrent Unit) and attention to capture changes in user behavior over time [12]. While this approach helps when the data clearly follows a sequence, it heavily depends on strong temporal patterns that are often absent in appointment-based service data, limiting its practical applicability. Another direction focuses on incremental clustering. The goal is to update clusters as new data arrives. Methods such as MiniBatchKMeans and UIClust follow this approach [13], DynamicSupervised extends it by introducing aging and merging mechanisms for clusters [14]. These methods can react to drift and update cluster structures efficiently, yet they often rely on simple heuristic rules for parameter tuning and do not fully exploit deep nonlinear representations, resulting in limited cluster quality under complex behavioral data. Reinforcement learning has also been applied to clustering. The idea is to learn how to select better parameters automatically. Models such as CRL and Q-DE-KMeans (Q-learning with Differential Evolution KMeans) use policy-based learning to search for configurations [9,11]. This reduces manual tuning. However, most of these methods are still built on top of traditional clustering algorithms. They are not tightly connected with deep representation learning.

Recent advances have explored adaptive clustering under concept drift in dynamic environments. Methods such as FedCCFA [17] introduce classifier clustering with feature alignment for distributed drift, while FedDAA [18] proposes dynamic client clustering to address heterogeneous multi-source drift in federated settings. In streaming scenarios, TEDA-based approaches [19] offer real-time clustering through typicality–eccentricity analysis combined with drift detection, and interpretable online clustering frameworks FURAKI [20] integrate transparency into adaptive pipelines. Despite these contributions, most methods rely on predefined update rules or explicit drift detection, which may limit flexibility in complex, non-stationary data.

Reinforcement learning (RL) has been introduced to improve adaptability. Approaches such as Q-DE-KMeans [11] combine Q-learning with evolutionary optimization. More recent studies have explored adaptive value-based clustering and drift-aware decision policies [21], aiming to enhance automation in parameter tuning and decision-making processes. However, they typically operate on fixed feature representations and focus on optimizing clustering parameters, without jointly adapting the representation space and clustering structure, which may reduce effectiveness under evolving data distributions.

Besides performance, explainability has become more importantin customer analytics. Some studies use SHapley Additive exPlanations (SHAP) or Explainable Boosting Machines to interpret model outputs [15]. However, these techniques are usually applied as a post-hoc step rather than being embedded into the training pipeline, and fairness considerations are rarely monitored during model design.

Overall, current methods can be grouped into three directions. Deep clustering focuses on representation learning. Incremental clustering focuses on handling streaming data. Reinforcement learning focuses on parameter optimization. Each direction addresses part of the problem, but they are often developed separately. As a result, no existing approach successfully unifies incremental deep representation learning, multi-head RL joint optimization, incremental updating, bandit-based decision making, and built-in XAI/fairness in a single closed-loop system. To better illustrate these differences, Table 1 summarizes several recent approaches and compares them with the proposed Hybrid-RL framework across key aspects, including hybrid design, drift handling, incremental learning and explainability.

Table 1: A comparative evaluation of Hybrid-RL and more recent sophisticated approaches (2021–2025).

Model	Problem Formulation	Action Space	Reward Design	Update Mechanism	Integration with Deep Rep.	Incremental Learning	XAI/Fairness	Application
Q-DE-KMeans [11]	Parameter tuning + K-means	Single (K only)	Single metric	Q-learning + Differential Evolution	No (PCA)	No	Limited	Consumer Market
CRL [9]	Multi-agent clustering	Cluster assignment	Task-specific	Multi-agent RL	Partial	Partial	No	General
CATS/ CAGRU [12]	Temporal clustering + GRU + Attention	Temporal sequence modeling	Prediction accuracy	GRU + Attention	Partial	No	No	Purchase intention prediction
Hybrid churn + AI [15]	LSTM/GRU + LightGBM + SHAP/EBM	Feature-level	Churn prediction loss	Supervised ensemble	Yes (LSTM/GRU)	No	Yes (post-hoc XAI)	Streaming service churn prediction
UICluster [13]	Incremental clustering + drift detection	Cluster assignment + drift detection	Clustering quality	Concept drift detection + incremental update	No	Yes	No	Streaming-data clustering (multi-domain)
Dynamic-Supervised [14]	Hierarchical supervised clustering	Cluster aging + merging	Supervised loss + aging	Weight-decay + activity-based pruning	No	Yes	No	Real-time data stream segmentation
BIDQN-VADA [21]	Incremental deep RL + data balancing	RL action selection	Credit scoring reward	Incremental RL + augmentation	Yes	Yes	No	Customer credit scoring
FURAKI [20]	Q-learning + DE + K-means	Single (K + evolution params)	Segmentation quality	Differential Evolution + Q-learning	No	No	No	Digital marketing

(Continued)

Table 1 (continued)

Model	Problem Formulation	Action Space	Reward Design	Update Mechanism	Integration with Deep Rep.	Incremental Learning	XAI/Fairness	Application
Hybrid-RL (Ours)	Incremental deep clustering	Joint: K, latent_dim, clustering_type	Multi-objective (Sil + CH − DBI)	Multi-head Q-network + incremental AE fine-tuning + Bandit	Yes incremental AE	Yes	Yes (Gini 0.49 + XAI)	Appointment data, retention

Most existing methods handle one or two aspects, such as drift, incremental learning, or explainability. Few methods combine all of them in a single framework. This becomes a limitation in real applications, where data changes over time and model transparency is required.

Based on these observations, Hybrid-RL is proposed to address this gap. The framework combines deep representation learning, reinforcement learning, and incremental updating in one pipeline. The model updates its latent space as new data arrives. A multi-head Q-network is used to adjust clustering parameters during training. A bandit-based mechanism supports decision making. Explainability and fairness are included as part of the pipeline. This design makes the model easier to apply in dynamic customer behavior analysis.

3 Proposed Approach

Hybrid-RL is proposed as an adaptive clustering framework designed to address three related challenges: learning nonlinear representations, adapting to concept drift, and turning clustering results into actionable decisions. In many existing methods, these components are handled separately. In contrast, Hybrid-RL brings them together in a single pipeline. The overall architecture is shown in Fig. 1. It consists of four main parts: representation learning, reinforcement learning-based optimization, incremental clustering, and a decision layer that includes explainability and fairness. These components are connected in a closed learning loop. This allows the model to update continuously as new data arrives, rather than relying on static training.

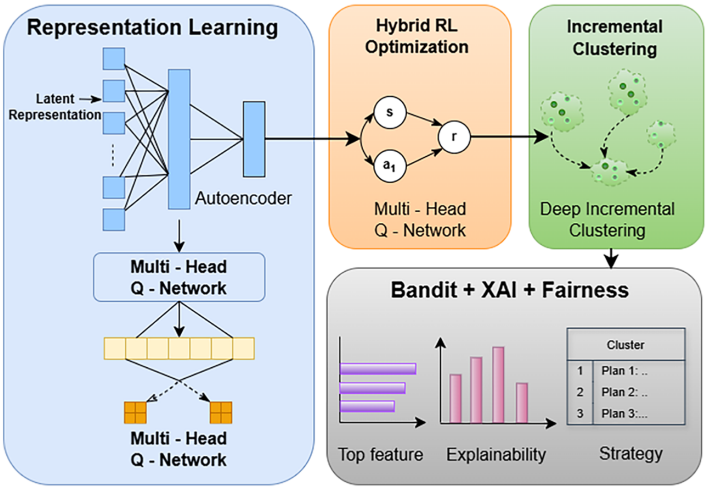

Figure 1: The Hybrid-RL architecture.

Fig. 1 illustrates how these components interact. The representation learning module maps input data into a latent space. The reinforcement learning module then adjusts clustering parameters based on the current state. The incremental clustering module updates cluster structures over time. The final layer links clustering outputs to decisions, while also providing explanations and monitoring fairness.

To clarify the technical contributions, Table 2 summarizes the main improvements of Hybrid-RL compared to earlier approaches. The table highlights how each component is modified and what it contributes to the overall system.

Table 2: Key technical improvements of Hybrid-RL compared with existing methods.

Component	Improvement over previous methods	Benefit
Deep Representation	Replaces linear representations (e.g., PCA in Q-DE-KMeans) with nonlinear latent features learned through an incremental autoencoder with continuous fine-tuning	Captures complex relationships in high-dimensional behavioral data and improves adaptation to drift
Incremental Drift Adaptation	Combines MiniBatchKMeans (partial_fit) with autoencoder fine-tuning instead of simple cluster updates as in UIClust or DynamicSupervisedClustering	Provides more stable cluster updates and preserves historical information while adapting to new data
Hybrid Reinforcement Learning (Multi-head)	Extends Q-learning to a multi-head network (HybridQNet) to jointly optimize multiple parameters (K, latent_dim, clustering type) using a multi-objective reward	Improves parameter optimization, reduces reliance on heuristics, and supports more stable convergence
Bandit Layer for Business Application	Integrates an ε-greedy bandit directly into the pipeline to learn decision policies based on cluster behavior and real-time feedback	Enables personalized strategies (e.g., reminders, deposits, bundle upgrades) and improves business applicability
XAI & Fairness Integration	Embeds a Random Forest surrogate model and Gini-based fairness monitoring into the training pipeline instead of using post-hoc explanation	Provides continuous transparency and fairness tracking, improving model reliability
Multi-objective Reward	Defines the reward as Silhouette + (Calinski–Harabasz/scaling factor) − Davies–Bouldin Index to balance cluster compactness and separation	Optimizes multiple clustering criteria simultaneously and maintains stable cluster structure in dynamic data

(Continued)

Table 2 (continued)

Component	Improvement over previous methods	Benefit
Practical Application	Applied to real appointment system data instead of general public datasets	Addresses real business needs, including dynamic segmentation and actionable recommendations

Table 2 shows that Hybrid-RL has improved the platform components in a more rational way, enabling the model to process action data efficiently, maintain stable segment quality, and support practical deployment in online scheduling systems with the concept of continuous drift. These improvements focus on enhancing adaptability, computational efficiency, and business applicability, while ensuring transparency and continuous work tracking.

3.1 Overall Pipeline and Algorithm

The Hybrid-RL framework is organized as a closed learning pipeline. The main flow can be described as: Representation → RL Optimization → Incremental Update → Decision. The detailed procedure is shown in Algorithm 1.

The initial dataset W_1 is used to learn both the latent representation and the reinforcement learning policy. During deployment, new data W_2 arrives in a streaming form. The model updates itself over time instead of being retrained from scratch.

Algorithm 1: Hybrid-RL deep incremental clustering with Bandit & XAI

Input:
Initial dataset W_1
Streaming dataset W_2
Latent embedding Z from Autoencoder + DEC-lite trained on W_1
Output:
Optimized clusters C on W_2
Recommendation policy (Bandit policy)
Explanations (XAI)
1: Pre-train Autoencoder on W_1 to learn latent representation Z
2: Encode behavioral features into latent space Z
3: Initialize Reinforcement Learning agent (multi-head Q-network)
4: For each RL episode do
5: Observe clustering state S(distance statistics + current parameters)
6: Select action $A = \{K, latent_dim, clustering_type\}$using epsilon-greedy
7: Perform clustering on Z with parameters A
8: Compute reward:

$$reward = Silhouette + \frac{CH}{500} - DBI \tag{1}$$

9: Update Q-network parameters via Q-learning (multi-head loss)

(Continued)

Algorithm 1 (continued)

10: end for
11: Select optimal parameters from trained policy
$$(K^*, latent\_dim^*, clustering\_type^*) \quad (2)$$
12: Perform clustering on W_1 using optimal parameters
13: For each incoming batch in W_2 do
14: Fine-tune Autoencoder incrementally
15: Update clusters using MiniBatchKMeans (partial_fit) or retrain Spectral clustering if required
16: End for
17: Apply Bandit policy to recommend actions for each cluster (reminder, voucher, bundle, deposit)
18: Generate explanations using RandomForest surrogate model with local interpretation (XAI)
19: Return clusters C, recommendation policy, explanations

Algorithm 1 describes the main workflow of Hybrid-RL. The process starts with training an autoencoder on the initial dataset to learn a latent representation of customer behavior. A reinforcement learning agent is then used to explore the parameter space of clustering and identify a suitable configuration based on a multi-objective reward. Once the policy is learned, the model performs initial clustering on W_1. During deployment, new data is processed in batches from W_2. The model updates its representation through incremental fine-tuning of the autoencoder and adjusts cluster structures using MiniBatchKMeans. This avoids full retraining and keeps the system responsive to changes in data.

The final stage links clustering results to decisions. A bandit-based policy is used to recommend actions for each cluster. At the same time, a surrogate model provides explanations for the clustering outcomes. This makes the results easier to interpret and apply in practice.

3.2 Data Protocol and Incremental Setting

To evaluate the ability of the model to adapt over time, the dataset is divided into two parts: an initial training set W_1 and an incremental dataset W_2. The autoencoder and reinforcement learning components are first trained on W_1. This stage provides a baseline representation and an initial policy. After that, W_2 is introduced as a stream of new data. The model updates its parameters gradually using incremental learning instead of retraining from scratch. This setup reflects how data arrives in real systems.

This approach reduces computational cost and makes it possible to observe how the model behaves under changing data distributions. It also helps limit overfitting, since the model is exposed to new patterns over time.

3.3 Layer 1—Deep Representation Learning

The first layer focuses on learning a nonlinear representation of customer behavior. An autoencoder is used to map the input data into a latent space. This helps reduce noise and makes cluster structure easier to identify. The learned representation is then passed to the next stage for parameter optimization.

In this study, the autoencoder follows a simple architecture (128 → 64 → latent_dim). It compresses 17 behavioral features, including booking frequency, cancellation rate, and average spending. The loss function combines reconstruction, clustering, and regularization terms:

$$L = \beta_1 \cdot MSE + \beta_2 \cdot KL + \lambda_1 \cdot L\_compact + \lambda_2 \cdot L\_separation \tag{3}$$

where $\beta_1 = 1.0$, $\beta_2 = 0.3$, $\lambda_1 = 0.15$, and $\lambda_2 = 0.08$. These hyperparameters were determined via grid search on the initial dataset W_1 to achieve a good balance between reconstruction fidelity and cluster compactness/separation. *L_compact* is defined as the average squared Euclidean distance between each data point and its assigned cluster center (measuring intra-cluster compactness), and *L_separation* is defined as the mean Euclidean distance between all pairs of cluster centers (measuring inter-cluster separation). The KL term is defined as:

$$KL = \sum p_{ij} \log(p_{ij}/q_{ij})$$

and q_{ij} is computed using a Student-t distribution. The additional terms encourage compact clusters while keeping them separated in the latent space.

Training is done in stages. The model is first trained on W_1, followed by a refinement step. After that, small updates are applied when new data arrives. Adam is used for optimization with a learning rate of 3×10^{-4} and batch size 128 [3]. This setup provides a stable latent space without increasing computational cost too much.

3.4 Layer 2—Hybrid Reinforcement Learning Hyperparameter Optimization

Layer 2 is responsible for dynamically adjusting clustering hyperparameters using a hybrid reinforcement learning mechanism, formulated as a Markov Decision Process (MDP). Rather than relying on fixed settings or heuristic tuning, the model learns to explore and refine parameter configurations as the data evolves over time.

At each time step t, the state S_t is represented as a five-dimensional vector. It reflects both the current clustering quality and the stability of the cluster structure. The components include the average intra-cluster distance, the standard deviation of distances, cluster inertia, the reward from the previous step, and a stability indicator (e.g., ARI between consecutive label assignments). This representation captures not only the current condition but also short-term changes in the data. The action space is defined as a combination of clustering hyperparameters:

$$A_t = \{K, \text{latent\_dim}, \text{clustering\_type}\} \tag{4}$$

where $K \in [3, 10]$, latent_dim $\in [5, 20]$, and clustering_type $\in$ {KMeans, Spectral}. These parameters are selected jointly, so both the representation space and the clustering method are adjusted together instead of being handled in separate steps.

The range [5, 20] for the latent dimension was chosen based on domain knowledge and empirical sensitivity analysis. With 17 behavioral features in the dataset, a very low dimension (below 5) risks significant information loss, while an excessively high dimension (above 20) may introduce redundancy and increase the risk of overfitting. The selected range provides a practical balance between representation capacity, clustering quality, and computational efficiency. This choice was further validated through sensitivity experiments presented in Section 4.7.

The reward function is designed in a multi-objective form:

$$R_t = \text{Silhouette} + \frac{\text{Calinski–Harabasz}}{\alpha} - \text{Davies–Bouldin} \tag{5}$$

where α is a scaling factor used to align the metrics. This formulation avoids over-optimizing a single criterion and helps maintain a more balanced cluster structure when the data distribution shifts.

To learn the policy, a multi-head Q-network is used. Each head corresponds to one component of the action space (K, latent_dim, clustering_type). This design keeps the action space manageable and improves convergence compared to standard Q-learning.

Training is carried out over 150 episodes using an epsilon-greedy strategy. The exploration rate decreases from 0.9 to 0.05 with a decay factor of 0.99, and the discount factor is set to $\gamma = 0.98$. This setup keeps a balance between exploring new configurations and reusing those that have shown good performance.

One implementation detail is worth noting. When a new latent dimension is selected by the multi-head Q-network, a completely new autoencoder is instantiated with the updated latent_dim while retaining the same overall architecture (128 → 64 → latent_dim). The newly created autoencoder is then pre-trained for 20 epochs on the current data batch using the Adam optimizer with a learning rate of 1×10^{-3}. This approach avoids catastrophic forgetting of previously learned features while allowing the representation space to adapt flexibly to the new dimension chosen by the RL policy. The weights from the previous autoencoder are not directly transferred; instead, the model learns a fresh latent representation suitable for the selected dimensionality.

With this setup, representation learning and clustering parameters are optimized within the same loop. As a result, the reinforcement learning component not only reduces the time needed to reach a good configuration, but also produces more stable parameter choices than static optimization methods, especially when concept drift is present.

3.5 Layer 3—Deep Incremental Clustering

Layer 3 handles model updates in a dynamic data setting. When new data arrives, the model is adjusted through incremental fine-tuning of the representation and updates to the cluster structure, instead of retraining from scratch.

The Hybrid-RL framework uses MiniBatchKMeans (batch size = 256) as the main method for incremental clustering. Spectral clustering can also be used at the initial stage when needed. Cluster stability is evaluated using the ARI index across both static and incremental settings [13]. Unlike conventional approaches, the framework updates latent representations and cluster centers together. The autoencoder is fine-tuned in small steps, while cluster assignments are updated at the same time. This keeps the representation space and clustering structure aligned as new data is introduced.

The computational complexity is reduced from $O(n^3)$ to $O(n \cdot k \cdot d)$, which makes the method more scalable. Only new data batches are processed during updates, while previously learned information is retained. This reduces retraining cost and supports continuous operation in real-time behavioral segmentation tasks.

3.6 Layer 4—Decision-Theoretic Bandit, Explainability, and Fairness-Aware Optimization

The final layer connects clustering results to decision making. Instead of treating clustering as a descriptive step, the framework uses behavioral clusters as inputs to an adaptive decision system. A multi-armed bandit is used to learn action strategies, while a surrogate model provides explanations. Fairness

is monitored at the same time through the distribution of actions across clusters. These components are designed to work together, so that decisions remain effective while still being interpretable.

3.6.1 Bandit-Based Decision Optimization

For each cluster C_k, the system learns a policy that aims to improve business outcomes, such as increasing retention or reducing cancellations. At time t, an action a_t is selected using an epsilon-greedy strategy:

$$a_t = \begin{cases} \text{random action}, & \text{with probability } \varepsilon \\ \arg\max_a Q(a), & \text{otherwise} \end{cases} \tag{6}$$

The action value is updated incrementally:

$$Q_{t+1}(a) = Q_t(a) + \eta\,(r_t - Q_t(a)) \tag{7}$$

where η is the learning rate and r_t is the observed reward from real outcomes, such as completed services or return visits.

This setup allows the system to learn directly from operational data. Over time, clusters are linked to actions that perform better, rather than being defined only for analysis.

3.6.2 Surrogate-Based Explainability

To make clustering results easier to interpret, a surrogate model based on Random Forest is trained to approximate the mapping from behavioral features to cluster labels:

$$\hat{y} = f_{RF}(x) \tag{8}$$

Feature importance is computed using the average impurity reduction:

$$I_j = \frac{1}{T}\sum_{t=1}^{T} \Delta Impurity_{j,t} \tag{9}$$

where I_j measures the contribution of feature j across the decision trees.

Features with higher importance values indicate stronger influence on cluster formation. This provides a direct way to interpret which behavioral patterns define each segment.

3.6.3 Fairness-Aware Regularization and Monitoring

Fairness is not treated as a post-processing step. Instead, it is monitored continuously during decision making. The imbalance of action distribution across clusters is measured using the Gini coefficient:

$$G = 1 - \sum_{i=1}^{K} p_i^2 \tag{10}$$

where p_i is the proportion of customers in cluster i receiving a given action.

To track changes over time, the variation is defined as:

$$\Delta G_t = \left| G_t - G_{t-1} \right| \tag{11}$$

This helps detect shifts in decision policies that may lead to bias when the data distribution changes.

Fairness can also be linked to the reward signal:

$$r_t^{\star} = r_t - \lambda \cdot G_t \tag{12}$$

where λ controls the sensitivity to imbalance. In this study, this term is used as a guiding principle rather than a strict implementation.

This formulation keeps the system from focusing too heavily on a small subset of customer groups. In this work, fairness is defined in behavioral terms, meaning that customers with similar patterns are treated consistently. This is suitable when demographic attributes are not available.

3.6.4 Integrated Closed-Loop Learning

The proposed framework operates as a closed-loop learning system, where all components are continuously connected through the following cycle: Clustering → Decision → Feedback → Policy Update.

In this loop, clustering defines the current behavioral state of customers, which serves as the input for decision making. Based on this state, the bandit module selects actions for each cluster. The outcomes of these actions generate feedback in the form of new data, which is then used to update the decision policy over time. This iterative process enables the system to adapt continuously to changing data conditions. Rather than stopping at descriptive segmentation, the framework incorporates feedback from real outcomes and gradually improves decision quality as new observations become available.

For practical deployment, inference per sample only requires a forward pass through the encoder and a lightweight cluster assignment (0.0001 s on CPU for batch size 128). However, periodic updates are still needed when new data arrives. Updating the autoencoder and clusters on a batch of 500 samples takes approximately 12–15 s on a standard CPU and consumes about 1.2 GB of additional memory. We recommend performing this update once every 7 days or after every 2000–3000 new appointments. The bandit policy uses an ε-greedy strategy with ε starting at 0.1 and decaying over time. To handle non-stationary reward distributions, a sliding window of the most recent 30 days of feedback is applied when updating action values.

4 Experimentation

The effectiveness of Hybrid-RL is evaluated using both proprietary real-world data and a widely used public benchmark to demonstrate robustness and generalizability. The primary dataset was collected from the internal management system of Moon Lashes & Brows, a beauty service chain in California. The dataset contains 10,212 appointment records from January 2021 to December 2025. Each record includes appointment ID, customer ID, status (Completed, Confirmed, Cancelled), net revenue, service duration, timestamps (created, booked, cancelled), as well as service category and type. To comply with California Consumer Privacy Act (CCPA) requirements, all personally identifiable information is removed. Customer IDs are replaced using pseudonymization. Sensitive features such as total_spending and recency_days are perturbed with random noise at a level of 0.05 times their standard deviation.

Descriptive statistics reflect typical patterns in this domain. The average revenue is 85.72 ± 45.61 USD. The cancellation rate is 7.2% ± 12.4%. Customer lifetime is around 105.8 ± 98.2 days, and the revisit frequency is 2.1 visits per 30 days ± 5.3. Some variables are strongly right-skewed (e.g., total_spending = 3.12, freq_30d = 4.56). To reduce the impact of outliers, winsorization is applied at the 1%–99% range, followed by log transformation for variables with skewness greater than 1.5.

To reflect real-world changes in customer behavior, the data is split over time. The first part, W_1 (80%, covering 2021–2024), is used for initial training. The remaining part, W_2 (20%, year 2025), is treated as incoming data for incremental updates. To ensure reproducibility and fair comparison, all experiments

(including all baseline methods) were conducted using the same random seed (SEED = 42) and the identical temporal data split. The nature and magnitude of the distribution shift between W_1 and W_2 were quantified using Kullback-Leibler (KL) divergence on key behavioral features (cancel_rate, total_spending, freq_30d), yielding a KL divergence of 0.28, indicating a moderate concept drift. Additionally, Kolmogorov-Smirnov tests on individual features confirmed statistically significant shifts ($p < 0.01$ for several key variables).

For initialization, the number of clusters is set to K = 5 as a baseline reference. The latent dimension is not fixed and is instead selected within the range [5, 20] using reinforcement learning. The autoencoder is pre-trained for 60 epochs with a batch size of 128. Reinforcement learning is trained over 150 episodes. The reward function combines Silhouette, Calinski–Harabasz (scaled), and Davies–Bouldin Index to balance compactness and separation.

The optimization process is carried out jointly in the latent space and clustering parameter space. Each change in latent_dim triggers a short fine-tuning step of the autoencoder. This keeps the representation aligned with the clustering configuration. The autoencoder is trained with a learning rate of 3×10^{-4} and is fine-tuned during reinforcement learning when the latent dimension changes. The implementation is based on Python, using scikit-learn and PyTorch.

The comparison includes three groups of baselines: traditional clustering methods (KMeans, MiniBatchKMeans, Agglomerative, Birch, GMM, Spectral), static deep clustering models (DEC, IDEC, VaDE, CRL) and adaptive or incremental approaches (Q-DE-KMeans, UICluster, DynamicSupervisedClustering, and Hybrid-RL). These baselines are selected to cover the main directions in the literature, including conventional clustering, deep clustering, and adaptive methods for dynamic data. Each model is run 10 times to ensure statistical stability. All experiments are conducted under the same computational environment to avoid hardware-related variation.

4.1 Generalizability Evaluation on Public Dataset

To address generalizability concerns and support claims beyond the proprietary dataset, we additionally evaluated Hybrid-RL on the widely used UCI Online Retail II benchmark (396,000 transaction records).

Table 3 presents the clustering performance comparison on the Online Retail dataset. Hybrid-RL achieves the best overall results, with a Silhouette score of 0.8455, DBI of 0.1777, and CH index of 14,368.19. These results significantly outperform both traditional methods (e.g., KMeans Silhouette = 0.2108) and deep clustering baselines such as DEC (0.8380) and IDEC (0.7179). In terms of inference efficiency, Hybrid-RL maintains near-constant runtime of 0.0002 s, demonstrating excellent scalability.

Table 3: Comparative clustering performance (Silhouette, DBI, CH, and runtime) over 10 runs.

Algorithm	Silhouette	DBI	CH	Runtime
KMeans	0.2485	1.1263	159.9326	0.0890
MBKMeans	0.1819	1.8764	55.3617	0.0575
Birch	0.2270	1.2519	145.9615	0.1280
Agglomerative	0.2153	1.2819	143.8108	0.1803
Spectral	0.2304	1.5458	48.6612	0.0537
GMM	0.2382	1.1981	143.5139	0.3503
VaDE	0.4824	1.2995	404.2984	0.4262
IDEC	0.4030	0.7514	877.7537	0.8713
DEC	0.4937	0.6117	1036.7593	1.1872

(Continued)

Table 3 (continued)

Algorithm	Silhouette	DBI	CH	Runtime
Q-DE-KMeans	0.2944	1.0747	143.3902	0.3186
CRL	0.4486	0.6505	693.0562	0.0238
UIC	0.2221	1.2234	126.2152	0.0181
DynamicSupervised	0.2830	1.0989	140.7828	0.0146
Hybrid-RL	**0.7542**	**0.3150**	**1810.3384**	**0.0001**

Note: Bold values indicate the best performance among all compared methods.

4.2 Comprehensive Performance Assessment

This section examines the overall clustering performance of the proposed model across several complementary criteria, including separation, compactness, and computational efficiency. The evaluation is carried out against a range of baseline methods covering traditional clustering, static deep clustering, and adaptive approaches. The results are summarized in Table 4, which reports the average Silhouette, DBI, CH, and runtime over 10 independent runs. The bold row indicates the best performance across all methods. Hybrid-RL achieves the strongest results on all four metrics.

Table 4: Clustering performance comparison on the UCI online retail dataset.

Algorithm	Silhouette	DBI	CH	Runtime (s)
KMeans	0.2108	1.4124	268.5202	0.0719
MBKMeans	0.1531	1.7042	144.5248	0.0252
Agglomerative	0.1577	1.4325	227.1589	0.1134
Birch	0.2980	1.1693	221.3645	0.0713
GMM	0.3319	1.3302	248.4282	0.0540
DEC	0.8380	0.2665	2119.8310	1.4745
CRL	0.0782	3.2704	85.7418	0.0335
IDEC	0.7179	0.4584	1945.6460	1.4925
VaDE	0.5130	1.3047	354.5202	1.0287
Spectral	0.2090	1.3992	59.3366	0.2337
Q-DE-KMeans	0.2214	1.4405	205.5314	0.6220
UICluster	0.1697	1.5391	152.2354	0.0520
DynamicSupervised	0.1911	1.4613	177.3692	0.0318
Hybrid-RL	**0.8455**	**0.1777**	**14,368.19**	**0.0002**

Note: Bold values indicate the best performance among all compared methods.

It is important to note that the reported runtime corresponds only to the inference stage. The training of the autoencoder and the reinforcement learning optimization are performed offline and are not included. During deployment, inference consists of a forward pass through the encoder to obtain latent representations, followed by cluster assignment using pre-optimized parameters. This keeps latency nearly constant and makes the model suitable for real-time use. Baseline methods are evaluated under comparable inference conditions whenever possible. Looking first at cluster separation, centroid-based methods such as KMeans, MiniBatchKMeans, and Birch show relatively low Silhouette values, ranging from 0.18 to 0.25. This suggests that distance-based methods have difficulty capturing nonlinear behavioral patterns. In comparison,

Hybrid-RL reaches a Silhouette score of 0.7542, which is higher than all other methods, including deep clustering models such as DEC (0.4937), IDEC (0.4030), and VaDE (0.4824). The improvement indicates that combining representation learning with adaptive parameter selection leads to clearer separation in the latent space.

A similar trend appears in DBI, where lower values indicate better compactness. Hybrid-RL achieves the lowest DBI at 0.3150. Traditional methods such as KMeans (1.1263) and MBKMeans (1.8764) remain considerably higher. Deep clustering approaches, including DEC (0.6117) and CRL (0.6505), reduce this gap but still fall short of the proposed method. This suggests that clusters produced by Hybrid-RL are both tighter and more stable.

For CH, which reflects the degree of separation between clusters, Hybrid-RL again records the highest value at 1810.3384. This is noticeably higher than DEC (1036.7593), IDEC (877.7537), and CRL (693.0562). The result indicates that cluster boundaries remain well defined, even under changing data conditions.

The most pronounced difference is observed in inference runtime. Hybrid-RL achieves a runtime of 0.0001 s, which is the lowest among all methods. This is substantially faster than KMeans (0.0890), as well as adaptive approaches such as DynamicSupervisedClustering (0.0146) and UICluster (0.0181). The gap becomes even larger when compared with deep clustering models such as DEC (1.1872) and IDEC (0.8713). This behavior can be attributed to the separation of the optimization stage from inference, together with the use of incremental updates rather than repeated retraining.

The Friedman test was applied to the Silhouette scores across 10 runs, yielding a test statistic of $\chi^2 = 128.45$ with $p < 0.001$, indicating statistically significant differences among the compared methods. Post-hoc analysis further confirms that Hybrid-RL outperforms both traditional and deep clustering baselines in terms of clustering quality and computational efficiency.

Taken together, these results suggest that Hybrid-RL not only improves clustering quality across multiple metrics but also maintains very low inference cost. This combination is particularly useful in settings where data evolves over time and timely responses are required.

4.3 Fairness Analysis

Fairness is integrated into the framework not as a post-processing step but as a continuous monitoring mechanism during decision making. The imbalance of action distribution across clusters is quantified using the Gini coefficient.

$$G = 1 - \sum_{i=1}^{K} p_i^2 \tag{13}$$

where p_i is the proportion of customers in cluster i receiving a given action. A Gini value of 0 indicates perfect equality, while 1 indicates perfect inequality.

On the proprietary dataset, the average Gini coefficient across all clusters and actions is 0.49, which is close to 0.5 and considered acceptable in this context (indicating relatively balanced action allocation). Cluster-level Gini values range from 0.47 to 0.50, showing no significant disparity. To assess stability over time, we computed the variation Delta Gini between consecutive batches in W_2; the maximum Delta Gini was 0.03, indicating that fairness remains stable under distribution shift.

Although fairness is currently used as a guiding principle rather than a hard constraint ($r_t^* = r_t - \lambda G_t$), incorporating it during training leads to only a minor trade-off: the Silhouette score decreases by approximately 0.02 while improving action equity across behavioral groups. No fairness constraints were violated in any cluster or time period.

These results substantiate the claim that fairness is meaningfully monitored throughout the pipeline and support the practical applicability of Hybrid-RL in real-world deployment.

4.4 Ablation Study

To better understand the contribution of each component, we conducted a systematic ablation study comparing four variants: (1) AE + KMeans (static baseline), (2) AE + Incremental (without RL), (3) AE + RL (without incremental update), and (4) the full Hybrid-RL model. Performance was evaluated under both static (W_1 test split) and dynamic (W_2) conditions to clearly separate the effects of each module.

Table 5 reports the Silhouette score, DBI, and CH index for all variants under static and dynamic settings. Statistical significance was assessed using the Wilcoxon signed-rank test (with Bonferroni correction) comparing each variant against the full Hybrid-RL model.

Table 5: Ablation study results under static (W_1) and dynamic (W_2) conditions.

Variant	Silhouette (W_1—Static)	Silhouette (W_2—Dynamic)	DBI (W_2)	CH (W_2)
AE + KMeans (static)	0.8851	0.8441	0.3878	1009.47
AE + Incremental (no RL)	0.8851	0.7341	0.4060	1016.40
AE + RL (no incremental)	0.8851	0.8441	0.3878	1009.47
Full Hybrid-RL	0.8279	0.7542	0.3150	1810.34

The results show that removing the incremental update causes a noticeable performance drop on the dynamic dataset W_2 (from 0.7542 to 0.7341 for AE + Incremental). Similarly, removing the RL component leads to significantly worse clustering quality under both static and dynamic conditions. The full Hybrid-RL model consistently achieves the best performance across both settings, particularly under distribution shift. These findings demonstrate that the combination of incremental learning and reinforcement learning is essential for maintaining high clustering quality in dynamic environments.

4.5 Evaluating the Clustering Outcomes

After the quantitative evaluation, clustering results are further examined through visualization to assess their practical meaning. Fig. 2 shows the latent space projected into two dimensions using PCA. The clusters are clearly separated, with large inter-cluster distances and limited overlap. This observation is consistent with the quantitative results in Table 4 (Silhouette = 0.7542 and DBI = 0.3150), suggesting that the learned embedding preserves cluster structure even after dimensionality reduction.

A closer look at the clusters reveals distinct behavioral patterns. Cluster 0 includes 11 customers with low spending (24.24 USD) and no cancellations, representing a low-activity group. Cluster 3 contains 9 customers with higher spending (167.78 USD) but a very high cancellation rate (72%), indicating a high-risk segment. Clusters 1 and 4 show moderate and stable behavior, with relatively low cancellation rates. Cluster 2 consists of a small group of high-value customers with consistently high spending. These patterns suggest that the clusters are not only geometrically separated but also meaningful in terms of behavior,

including spending, cancellation, and visit frequency. The alignment between data structure and business interpretation is particularly important in practical applications, where clustering results need to support decision making.

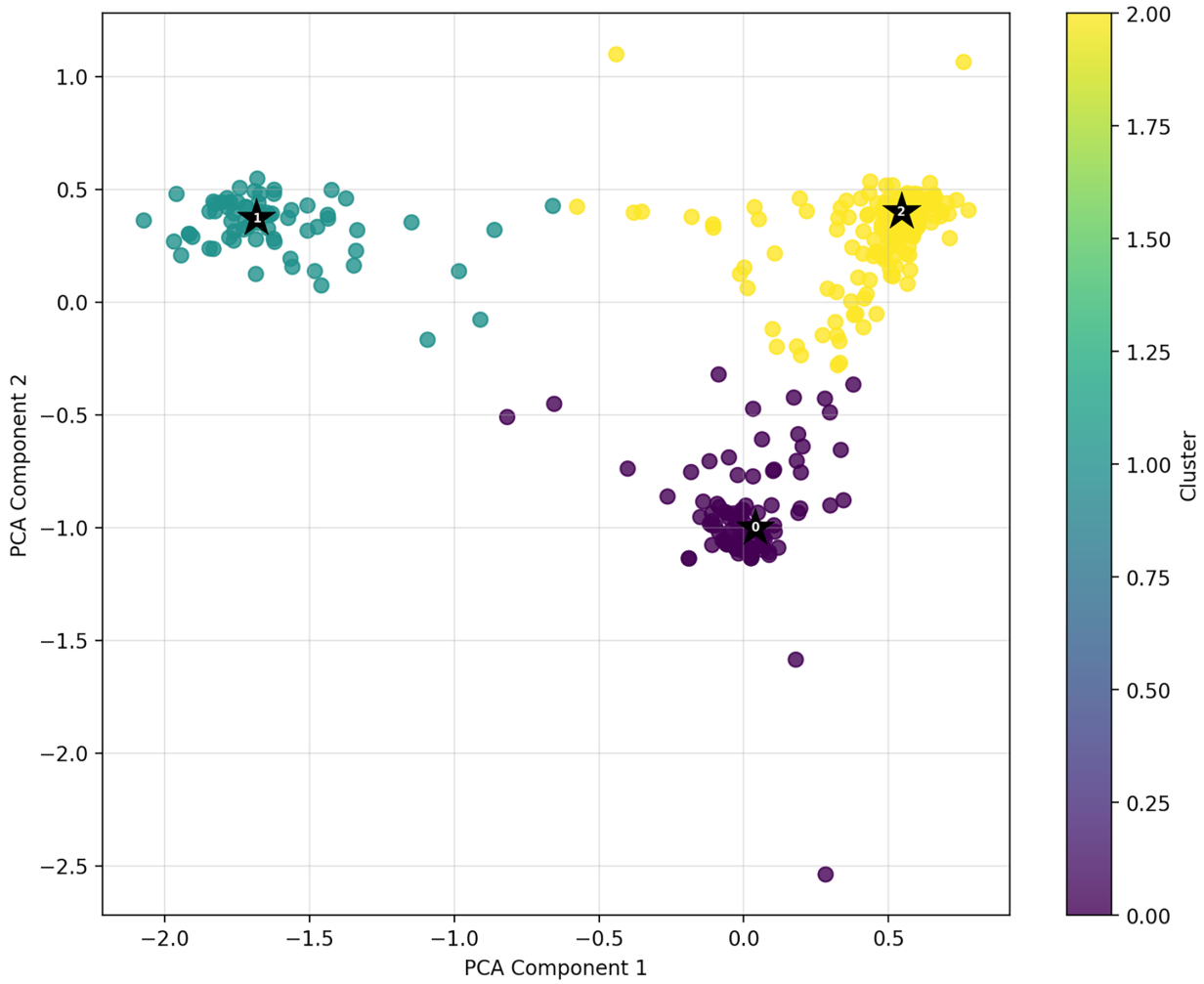

Figure 2: Visualization of customer clusters using PCA projection (Star markers indicate cluster centroids, highlighting cluster compactness and separation).

Fig. 3 presents the average inference runtime on the test set. Hybrid-RL achieves 0.0001 s, which is the lowest among all methods and at least one order of magnitude faster than the others. Static deep clustering models such as DEC (1.1872), IDEC (0.8713), VaDE (0.4262), and Q-DE-KMeans (0.3186) require significantly more time. Adaptive models such as UICluster (0.0181) and DynamicSupervisedClustering (0.0146) are faster, but still noticeably slower than Hybrid-RL.

This difference is related to the design of the model. The reinforcement learning optimization and representation learning are handled offline, while the inference stage only involves a forward pass and lightweight cluster updates. As a result, there is no need to recompute embeddings or re-optimize parameters during deployment. This leads to lower latency and makes the model suitable for applications that require near real-time response, such as appointment-based service platforms.

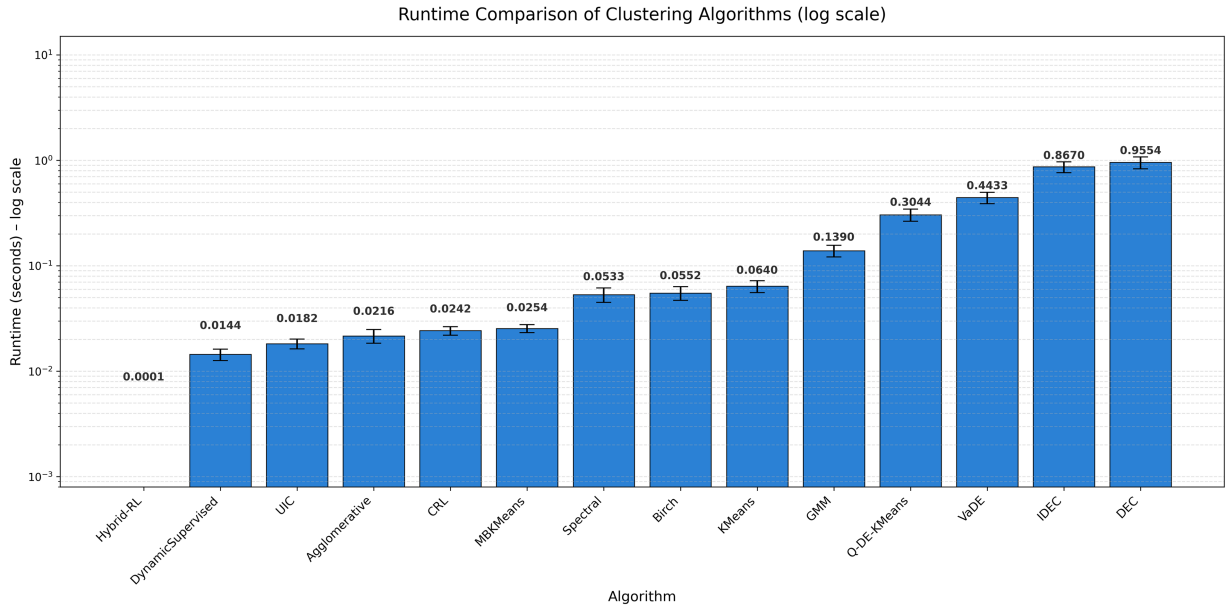

Figure 3: Runtime comparison of clustering algorithms on a logarithmic scale.

4.6 Scalability and Runtime Analysis

The scalability of Hybrid-RL is examined by separating the analysis into two stages: training and inference. This distinction reflects how the model is used in practice.

Training is performed once, or updated periodically when needed. The reported runtime of 0.0001 s corresponds only to the inference stage, after the model has been fully trained. During deployment, the processing pipeline is simple. It consists of a forward pass through the encoder to obtain the latent representation, followed by cluster assignment using previously optimized parameters. No reinforcement learning step or model retraining is involved at this stage. As a result, inference latency remains nearly constant and is suitable for real-time applications.

To ensure a fair comparison, runtime measurements are collected under the same computational environment for all methods. The training phase of Hybrid-RL includes three main components: autoencoder pre-training (2.14 s), reinforcement learning optimization (1.87 s), and incremental updates on W_2 (0.38 s). The total training time is 4.39 s. This cost is incurred only once and does not repeat when new data arrives.

Inference time is measured separately after training is completed. It corresponds to the time required to assign a cluster to a new data sample. The average runtime on W_2 is 0.0001 s (as reported in Table 4), indicating negligible delay in practice.

To further examine scalability, the model is evaluated on datasets of different sizes (n = 1000; 5000; 10,000). The inference time increases gradually from 0.0068 to 0.0095 s and 0.0128 s. This trend is close to linear. The main computational cost follows $O(n * k * d)$, driven by MiniBatchKMeans and the forward pass of the autoencoder.

In contrast, Agglomerative and Spectral clustering scale less favorably due to $O(n^2)$ and $O(n^3)$ complexity. Static deep clustering models such as DEC, IDEC, and VaDE become more expensive when the embedding needs to be retrained under distribution changes. Methods like Q-DE-KMeans and CRL also show slower behavior at larger scales, as their optimization steps are repeated.

Table 3 summarizes the runtime and complexity across different data sizes. Hybrid-RL maintains near-linear behavior and remains among the fastest methods across all settings. While some approaches, such as DynamicSupervisedClustering, show comparable runtime at smaller scales, Hybrid-RL still provides stronger clustering quality. This suggests a better balance between performance and computational cost.

Overall, the results indicate that Hybrid-RL is not only effective in terms of clustering quality but also practical to deploy as data size increases.

4.7 Application of Bandit Policy and Analysis of Explainable Artificial Intelligence (XAI)

Beyond clustering, the study applies the learned bandit policy to derive practical business actions. For example, reminder strategies are used for low-activity customers, while deposit requirements are introduced for high-risk groups to reduce cancellation rates. Unlike conventional approaches that stop at cluster analysis, Hybrid-RL connects clustering outputs directly to decision making. The bandit mechanism allows these actions to be adjusted over time based on observed feedback.

A Random Forest surrogate model is used to explain the clustering results. The most influential features include avg_spending (0.2039), total_spending (0.1781), avg_duration (0.1550), num_appointments (0.1067), and tenure_days (0.0775). These variables are consistently associated with cluster assignment.

The results suggest that spending behavior and interaction frequency play a central role in shaping cluster structure. In contrast, cancellation-related features are more closely linked to customer risk. This pattern aligns with the objective of improving retention in appointment-based services.

Fig. 4 illustrates the average values of five key features (avg_spending, cancel_rate, freq_30d, recency_days, unique_services) across clusters. Clear differences can be observed along both value-related dimensions (e.g., spending) and behavioral dimensions (e.g., frequency and cancellation). This supports the consistency between the learned embedding and real customer behavior.

Table 5 presents detailed cluster profiles based on the XAI analysis. One cluster is labeled as “unsafe” due to a noticeably higher cancellation rate. This type of labeling improves interpretability and makes the results easier to use in practice, especially when decisions need to be explained rather than inferred from abstract metrics.

Table 6 summarizes the business strategies assigned to each cluster. Price-sensitive customers are associated with discount vouchers, while long-term loyal customers are linked to service upgrade offers. More specifically, Cluster 0 (104 customers), characterized by low but stable spending, is assigned the voucher_10 strategy to encourage additional purchases. Cluster 1 (69 customers), which shows a higher cancellation rate, is assigned a deposit_required policy to reduce booking risk. Cluster 2 (228 customers), representing moderate users with low visit frequency, is associated with reminder actions to increase return visits.

These strategies are not manually defined. They are learned through an epsilon-greedy bandit process, which updates decisions based on observed outcomes. Over time, the system adjusts its actions according to actual performance rather than relying on fixed rules.

Taken together, these results show that Hybrid-RL can move from technical analysis to actionable decisions. The combination of clustering, XAI, and bandit learning forms a closed feedback loop, where decisions generate new data that is then used to refine future actions. This makes the system more suitable for dynamic environments where both behavior and optimal strategies may change over time.

Table 7 presents the detailed characteristics of each customer cluster based on the XAI analysis. The clusters are interpreted in terms of spending behavior, cancellation patterns, and visit frequency to provide actionable insights.

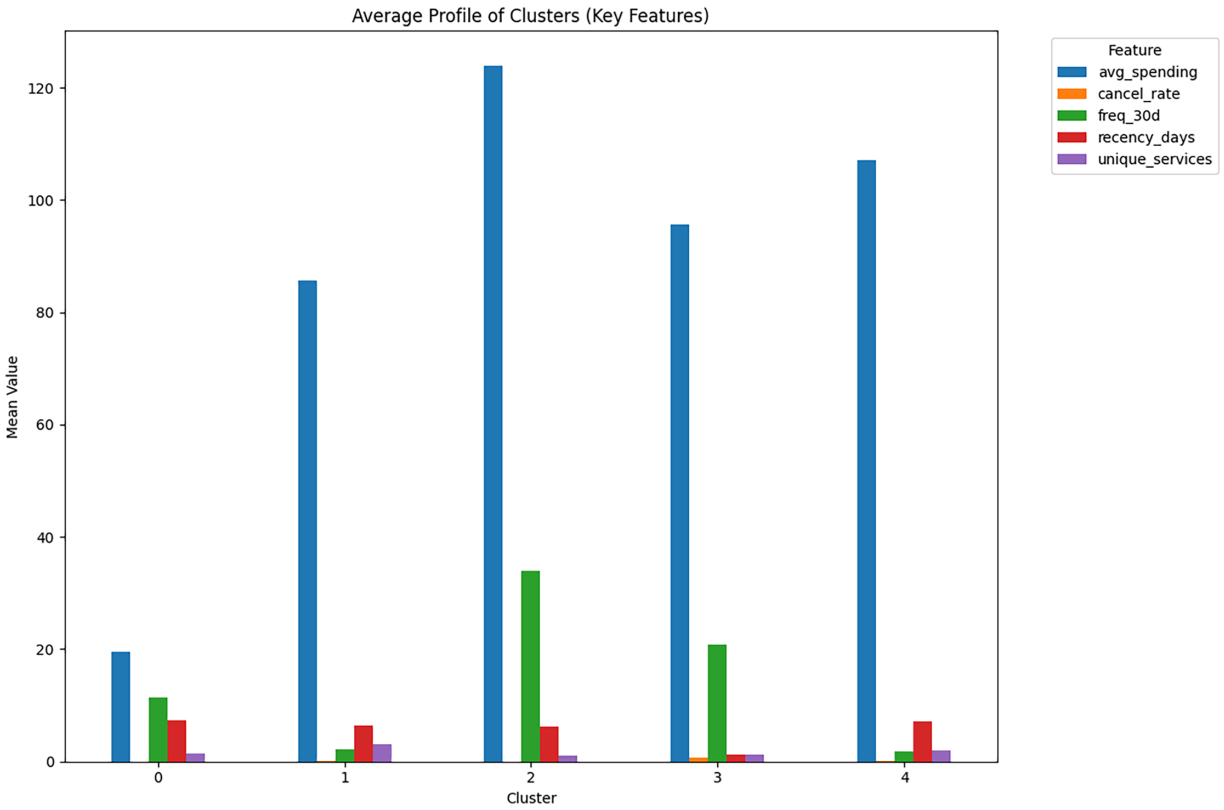

Figure 4: The average profile of the clusters (main traits).

Table 6: Runtime, computational complexity, and scalability analysis across different data sizes.

Algorithm	Complexity	Runtime n = 1000 (s)	Runtime n = 5000 (s)	Runtime n = 10,000 (s)
KMeans	O(n k d i)	0.0521	0.1333	0.1988
MBKMeans	O(n k d)	0.0111	0.0100	0.0109
Agglomerative	$O(n^2)$	0.0296	0.7252	3.2895
Birch	O(n)	0.0554	0.2696	0.4497
GMM	$O(n\ k\ d^2)$	0.1120	0.3333	1.3251
DEC	$O(n^2 + \text{epochs} \times n)$	0.2998	1.8080	2.7021
CRL	O(epochs × actions × n)	0.0601	4.1975	1.4873
IDEC	$O(n^2 + \text{epochs} \times n)$	1.8551	2.7113	3.3233
VaDE	$O(\text{epochs} \times n\ d^2)$	0.2520	1.1629	2.6723
Spectral	$O(n^3)$	0.2560	3.5350	2.4165
Q-DE-KMeans	$O(n^2 + \text{de\_iter} \times n\ k)$	0.2943	1.3751	3.2200
UICluster	O(n k d log n)	0.0274	0.7261	2.7999
DynamicSupervised	O(n k d + epochs)	0.0077	0.0148	0.0219
Hybrid-RL	O(n k d + epochs × actions)	0.0050	0.0132	0.0224

Table 7: Detailed characteristics of customer clusters based on XAI analysis.

Cluster 0 (104 customers)	- Average spending: 84.02 (total 5.21) - Cancel rate: 1.0% (canceled 0.0/1.4 appointments) - Frequency: 12.59 appointments/30 days, most recent: 9 days - Loyalty: Completion 1.0%, service variety: 1.8 - Key characteristic: Low-spending customers
Cluster 1 (69 customers)	- Average spending: 112.65 (total 6.15) - Cancel rate: 19.0% (canceled 0.7/1.7 appointments) - Frequency: 9.78 appointments/30 days, most recent: 8 days - Loyalty: Completion 0.0%, service variety: 2.4 - Key characteristic: Loyal customers
Cluster 2 (228 customers)	- Average spending: 90.91 (total 7.31) - Cancel rate: 7.0% (canceled 1.6/2.9 appointments) - Frequency: 1.97 appointments/30 days, most recent: 8 days - Loyalty: Completion 0.0%, service variety: 3.2 - Key characteristic: Average customers

Table 8 summarizes the recommended business strategies for each customer cluster, linking behavioral characteristics to actionable decisions.

Table 8: Recommended business strategies for each customer cluster.

Cluster	Behavioral Characteristics	Recommended Action	Rationale	Expected Impact
C0	Low spending (84.02), very low cancel rate (1.0%), recent activity (9 days)	voucher_10	Encourage additional spending for low-value but stable customers	Increase revenue and retention by 5%–15%
C1	Moderate spending (112.65), high cancel rate (19.0%), recent activity (8 days)	Deposit_required	Reduce booking cancellations for high-risk customers	Improve booking reliability and retention by 5%–15%
C2	Moderate spending (90.91), moderate cancel rate (7.0%), low visit frequency	Reminder	Increase return frequency for moderately active customers	Improve engagement and retention by 5%–15%

4.8 Synthetic Drift Test

To rigorously evaluate the robustness of Hybrid-RL under concept drift, we conducted a controlled synthetic drift experiment on the incremental dataset W_2. Moderate perturbations were applied, including

Gaussian noise with standard deviation σ = 0.03 and feature shift on 5% of the features by ±5% of their original mean.

The pre-trained model was evaluated on the drifted data without retraining. The Silhouette score decreased from 0.7341 to 0.6891, corresponding to a 6.1% drop. These results indicate that Hybrid-RL maintains relatively high stability under moderate distribution shifts, confirming the effectiveness of its incremental autoencoder fine-tuning and multi-head RL optimization.

4.9 Component-Wise Impact Analysis

To examine the role of each component, several ablation settings are tested by removing key modules from the Hybrid-RL pipeline, including AE + KMeans (basic clustering on learned embeddings), AE + Incremental Clustering (without reinforcement learning), AE + RL Optimization (without incremental updates), and the full Hybrid-RL model. Removing the reinforcement learning component leads to a clear drop in performance, with the Silhouette score falling to 0.2494, roughly 67% lower than the full model (0.7542). This change suggests that hyperparameter optimization is important for identifying suitable clustering configurations in the latent space.

A different behavior can be seen when the incremental mechanism is removed. In a static setting, the model reaches a higher Silhouette score of 0.8343, but this improvement does not carry over when the data distribution changes. Under dynamic conditions, the score decreases by about 11%, which is consistent with the effect of catastrophic forgetting. This indicates that optimization alone is not sufficient if the model cannot adapt to new data over time.

The full Hybrid-RL model shows a more balanced result, with a Silhouette score of 0.7542 that remains relatively stable across both static and dynamic scenarios. This points to the importance of combining representation learning, reinforcement learning–based optimization, and incremental updates, rather than relying on any single component.

Pairwise t-tests indicate that the differences between Hybrid-RL and the baseline methods are statistically significant ($p \approx 0$, with $|t| > 100$ across comparisons), making it unlikely that the observed improvements are due to random variation. Sensitivity analysis also reveals a clear pattern. The best performance is obtained at K = 3 (Silhouette = 0.7549), while performance decreases at K = 5 (0.6331) and K = 7 (0.5257). A similar trend appears for the latent dimension, where a smaller latent space (latent_dim = 5) performs better than larger values (10 or 20), suggesting that limiting model complexity helps reduce overfitting.

Under synthetic drift, Hybrid-RL shows only a small performance drop, with the Silhouette score decreasing by around 6.1%. This is comparable to baseline methods, while still preserving cluster structure and allowing continuous updates without full retraining. The fairness analysis reports an average Gini coefficient of about 0.49 across clusters, indicating a relatively balanced distribution of actions and consistent decision behavior across different customer groups

5 Discussion

The experimental results demonstrate a consistent advantage of Hybrid-RL in clustering quality, particularly when the data distribution changes over time. Hybrid-RL achieves the highest Silhouette score (0.7542) and the lowest DBI (0.3150) among all compared methods on the proprietary dataset. This advantage becomes more evident in dynamic settings, where several baseline methods show clear performance degradation as new behavioral patterns emerge.

A key strength lies in the tight integration of incremental deep representation learning and reinforcement learning. Rather than treating these components separately, Hybrid-RL updates the latent space and

cluster structure simultaneously. This design helps avoid repeated full retraining and reduces sensitivity to concept drift compared to static deep clustering models such as DEC, IDEC, and VaDE. In addition, the multi-head Q-network enables joint optimization of multiple parameters, while the multi-objective reward prevents over-optimization on a single metric, leading to more compact and well-separated clusters.

From a practical perspective, the model also shows strong scalability. Inference time remains extremely low at 0.0001 s per sample, significantly faster than both traditional and deep clustering baselines. This separation of training and inference makes Hybrid-RL particularly suitable for real-time applications in appointment-based services.

To further validate generalizability, we evaluated Hybrid-RL on the public UCI Online Retail dataset. The model maintained superior performance (Silhouette = 0.8455) compared to all baselines, confirming that its advantages are not limited to the proprietary appointment data.

Overall, the strength of Hybrid-RL stems from the synergistic combination of its components rather than any single module. By linking representation learning, adaptive optimization, and incremental updates within a unified closed-loop pipeline, the framework remains effective even as both data and optimal actions evolve.

6 Conclusion and Future Directions

This study presents Hybrid-RL, an adaptive clustering framework designed for customer behavior analysis in dynamic environments where concept drift is common. By integrating incremental deep representation learning, multi-head reinforcement learning for joint hyperparameter optimization, and continuous model updating, Hybrid-RL effectively handles shifting data distributions without requiring full retraining.

Experimental results on both proprietary appointment data and the public UCI Online Retail dataset demonstrate that Hybrid-RL consistently outperforms traditional, deep, and adaptive baselines in clustering quality while maintaining extremely low inference latency. The framework also incorporates bandit-based decision making, surrogate-model explainability, and Gini-based fairness monitoring, moving beyond descriptive clustering to support actionable and interpretable business decisions.

Although the results are encouraging, several limitations should be noted. First, the current evaluation is based primarily on data from a single service domain (beauty appointments) and one public benchmark (UCI Online Retail). While the addition of the public dataset strengthens generalizability claims, further validation on larger and more diverse datasets from other industries is still needed. Second, the optimization objective currently focuses on clustering quality metrics. Business-oriented outcomes (e.g., actual retention rate or revenue uplift) are not directly incorporated into the learning process. This separation may limit the model's ability to align perfectly with real-world business goals.

Future work will explore extending Hybrid-RL to multi-source data by combining behavioral, transactional, and contextual information. We also plan to incorporate business impact metrics directly into the reward function and investigate more advanced reinforcement learning techniques to further enhance adaptability under severe concept drift. Overall, the unified closed-loop design of Hybrid-RL offers a promising direction for practical adaptive customer segmentation in streaming environments.

Acknowledgement: This research was supported by Nguyen Tat Thanh University and Van Lang University. The authors also thank Moon Lashes & Brows for enabling the application of the proposed algorithm to real customer behavioral data.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm their contributions to this study as follows: Conceptualization, model development, experimentation, data processing and analysis, and writing—original draft preparation: Anh Thi Diem Nguyen; research orientation, validation of results, and writing—review and editing: Vinh Truong Hoang; supporting contributions to analysis, discussion, and proofreading: Tham Vo. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data used in this study are proprietary and contain sensitive business information; therefore, they are not publicly available.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Xiang Q, Zi L, Cong X, Wang Y. Concept drift adaptation methods under the deep learning framework: a literature review. Appl Sci. 2023;13(11):6515. doi:10.3390/app13116515.
2. Xiao JW, Xie Y, Fang H, Wang YW. A new deep clustering method with application to customer selection for demand response program. Int J Electr Power Energy Syst. 2023;150:109072. doi:10.1016/j.ijepes.2023.109072.
3. Lee Y, Park C, Kang S. Deep embedded clustering framework for mixed data. IEEE Access. 2023;11(11):33–40. doi:10.1109/access.2022.3232372.
4. Guo X, Gao L, Liu X, Yin J. Improved deep embedded clustering with local structure preservation. Int Jt Conf Artif Intell. 2017;17:1753–59. doi:10.24963/ijcai.2017/243.
5. Ha NQ, Huyen NT, Uyen MT, Viet NQ, Quang NN, Thanh DN. Customer intent mining from service inquiries with newly improved deep embedded clustering. J Uncertain Syst. 2024;17(02):2440004. doi:10.1142/s175289092440004x.
6. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H. Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence 2017. Washington, DC, USA: AAAI Press; 2017. p. 1965–72.
7. Guo X, Liu X, Zhu E, Yin J. Deep clustering with convolutional autoencoders. In: Neural information processing. Cham, Switzerland: Springer International Publishing; 2017. p. 373–82. doi:10.1007/978-3-319-70096-0_39.
8. Guo J, Fan W, Amayri M, Bouguila N. Deep clustering analysis via variational autoencoder with Gamma mixture latent embeddings. Neural Netw. 2025;183(8):106979. doi:10.1016/j.neunet.2024.106979.
9. Li P, Gao J, Zhang J, Jin S, Chen Z. Deep reinforcement clustering. IEEE Trans Multimedia. 2023;25:8183–93. doi:10.1109/tmm.2022.3233249.
10. Likas A. A reinforcement learning approach to online clustering. Neural Comput. 1999;11(8):1915–32. doi:10.1162/089976699300016025.
11. Wang G. Customer segmentation in the digital marketing using a Q-learning based differential evolution algorithm integrated with K-means clustering. PLoS One. 2025;20(2):e0318519. doi:10.1371/journal.pone.0318519.
12. Kuang Y, Zhang T, Huang ZW, Zeng Z, Li ZY, Huang L, et al. CATS: clustering-aggregated and time series for business customer purchase intention prediction. [cited 2025 Sep 21]. Available from: https://dl.acm.org/doi/abs/10.1145/3701716.3717545.
13. Woodbright MD, Rahman MA, Islam MZ. A novel incremental clustering technique with concept drift detection. arXiv:2003.13225. 2020.
14. Nikpour S, Asadi S. A dynamic hierarchical incremental learning-based supervised clustering for data stream with considering concept drift. J Ambient Intell Humaniz Comput. 2022;13(6):2983–3003. doi:10.1007/s12652-021-03673-0.
15. Joy UG, Hoque KE, Uddin MN, Chowdhury L, Park SB. A big data-driven hybrid model for enhancing streaming service customer retention through churn prediction integrated with explainable AI. IEEE Access. 2024;12(4):69130–50. doi:10.1109/ACCESS.2024.3401247.

16. Chavhan R, Dutta P, Samant N, Kar S. Data-driven strategic customer segmentation considering cart abandonment behavior: insights from e-grocery delivery platforms. Inf Sci. 2025;718:122327. doi:10.1016/j.ins.2025.122327.
17. Chen J, Xue J, Wang Y, Liu Z, Huang L. Classifier clustering and feature alignment for federated learning under distributed concept drift. arXiv:2410.18478. 2024.
18. Peng F, Tang M. FedDAA: dynamic client clustering for concept drift adaptation in federated learning. arXiv:2506.21054. 2025.
19. Rezaei Z, Sajedi H. TEDA-driven adaptive stream clustering for concept drift detection. Data Knowl Eng. 2025;160(2):102484. doi:10.1016/j.datak.2025.102484.
20. Corradini F, Nucci V, Piangerelli M, Re B. Online clustering with interpretable drift adaptation to mixed features. Intell Syst Appl. 2025;26(6):200510. doi:10.1016/j.iswa.2025.200510.
21. Wang Y, Jia Y, Zhong Y, Huang J, Xiao J. Balanced incremental deep reinforcement learning based on variational autoencoder data augmentation for customer credit scoring. Eng Appl Artif Intell. 2023;122(6):106056. doi:10.1016/j.engappai.2023.106056.