Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084947

ARTICLE

Interpreted Optimization Learning for Improving the Human-Computer Interaction Responses under Input Constraints

Asma Aldrees[1], Mohd Anjum[2,*], Vladimir Simic[3,*], Sana Shahab[4] and Dragan Pamucar[5,6,7]

[1]Department of Informatics and Computer Systems, College of Computer Science, King Khalid University, Abha, Saudi Arabia
[2]Department of Computer Engineering, Aligarh Muslim University, Aligarh, India
[3]Sustainability Competence Centre, Széchenyi István University, Egyetem tér 1, Győr, Hungary
[4]Department of Business Administration, College of Business Administration, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia
[5]Department of Applied Mathematical Science, College of Science and Technology, Korea University, Sejong, Republic of Korea
[6]UNEC Applied Artificial Intelligence Research Center, Azerbaijan State University of Economics (UNEC), Baku, Azerbaijan
[7]Faculty of Engineering, Dogus University, Umraniye, Istanbul, Türkiye

*Corresponding Authors: Mohd Anjum. Email: mohdanjum@zhcet.ac.in; Vladimir Simic. Email: simic.vladimir@sze.hu

Received: 02 May 2026; Accepted: 18 August 2026; Published: 22 October 2026

ABSTRACT: Human–computer interaction (HCI) systems require rapid and accurate interpretation of user commands to ensure seamless communication and an enhanced user experience. However, incomplete alphanumeric inputs and keystroke errors often result in discrete sequences and ambiguous commands, thereby increasing response latency and reducing interpretation reliability. Existing approaches primarily emphasize response accuracy but exhibit limitations in delivering swift and interpretable responses under input constraints. To address these challenges, this study proposes the Interpreted Optimization Learning for Swift Response (IOL-SR) framework. The proposed framework integrates input normalization and feature augmentation to reconstruct incomplete command sequences and improve interpretability. A mono-layer reinforcement learning mechanism is incorporated to optimize response generation by exploiting identified and predicted feature sequences, while normalization and reinforced feature learning minimize sequence discontinuities and enhance command understanding. Experimental evaluation using a keyword-based HCI dataset demonstrates that the proposed IOL-SR framework effectively improves sequence interpretation and response generation while reducing latency. The integration of normalization and reinforcement learning enables robust handling of incomplete input commands and enhances understandability through adaptive feature augmentation. Consequently, the proposed framework provides more reliable and efficient HCI responses, facilitating low-latency, interpretable, and user-centric interaction systems. Experimental results demonstrate that the proposed IOL-SR framework reduces the average response latency to 41.6 ms while maintaining reliable command interpretation.

KEYWORDS: Feature detection; human-computer interaction; interpretation; reinforcement learning; sequence classification

1 Introduction

Human-computer interaction (HCI) systems have become increasingly important in facilitating effective communication between users and intelligent computing environments. The quality of interaction largely depends on the capability of systems to accurately understand user inputs and generate timely responses. Efficient input interpretation minimizes ambiguities and errors, thereby improving interaction

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

quality and overall system performance [1,2]. Recent advances in low-cost sensing technologies and real-time interaction tracking mechanisms have enhanced responsiveness by enabling precise identification and interpretation of user intentions [3,4]. Furthermore, input normalization and pattern recognition techniques contribute to improved command understanding by reducing interpretation errors across diverse interaction scenarios [5,6]. These techniques allow systems to anticipate user requirements through gesture and signal-based recognition mechanisms, thereby providing reliable and accurate responses [7,8].

Modern HCI systems are expected to process diverse forms of user input while maintaining high levels of responsiveness and adaptability [9,10]. Proper interpretation of user commands is essential for reducing latency and enabling seamless communication between users and intelligent systems [11]. Reinforcement learning (RL) has emerged as an effective approach for identifying behavioral patterns and optimizing decision-making processes to support rapid response generation [12]. RL with human feedback further improves system adaptability by aligning model behavior with user preferences through iterative policy optimization. In addition, feature augmentation techniques enhance interpretation capability by minimizing information loss and increasing recognition accuracy [12]. Collectively, these approaches contribute to more reliable and efficient interaction mechanisms.

The development of intelligent interaction mechanisms has significantly improved user experience and communication quality. Studies on emotional experience, intelligent interfaces, advanced sensing technologies, and interactive artificial intelligence have demonstrated that responsive and adaptive interaction mechanisms positively influence user engagement and overall interaction quality [13,14]. Advanced learning frameworks continuously update knowledge representations by leveraging human-AI interaction paradigms and language-driven adaptation to improve interaction quality [15,16]. Additionally, adaptive algorithms efficiently process heterogeneous inputs through flexible sensing and multimodal recognition mechanisms, thereby enabling more flexible and responsive interaction systems [17,18]. Furthermore, deep RLapproaches with expert demonstrations have shown considerable potential for supporting adaptive and real-time decision optimization in human–machine interaction environments [19]. Input normalization mechanisms also contribute to reducing interpretation errors and improving response precision [20]. Such developments enable HCI systems to continuously evolve and accommodate changing user requirements [21]. Moreover, multimodal and touch-based interaction technologies provide more natural and intuitive communication, thereby enhancing interaction experience and responsiveness [22].

Additionally, recent advancements in human–robot interaction have demonstrated that adaptive systems leveraging real-time human behavior modeling and deep RL can significantly enhance interaction efficiency and system responsiveness [23]. Despite significant advances in reinforcement learning, feature augmentation, and adaptive learning techniques, existing state-of-the-art approaches still exhibit several limitations when deployed in real-time HCI environments. Although these methods have improved response accuracy and interaction quality, they often incur additional computational overhead, leading to increased latency during input interpretation and decision-making processes. Moreover, heterogeneous and constrained inputs can adversely affect interpretation consistency, resulting in delayed responses and reduced system reliability. Existing error-correction and adaptive learning models primarily focus on prediction accuracy and robustness, while providing limited interpretability and insufficient mechanisms for maintaining rapid response generation under varying input conditions. Deep RLapproaches further suffer from complex optimization procedures and high computational requirements, which hinder their applicability in latency-sensitive scenarios. Consequently, current approaches are unable to simultaneously ensure low latency, robust interpretability, and consistent response performance under constrained inputs. Therefore, there remains a need for an efficient and interpretable framework capable of providing accurate and swift responses while maintaining adaptability and reliability across diverse interaction conditions.

To address these challenges, this study proposes the Interpreted Optimization Learning for Swift Response (IOL-SR) framework for improving HCI responses under input constraints. The proposed framework employs interpreted optimization learning to enhance response accuracy, reduce latency, and improve interaction reliability through adaptive learning mechanisms. Experimental validation is conducted using a publicly available dataset (https://doi.org/10.1038/s41597-023-02741-8), which was collected under institutional oversight with operator consent for anonymized research use. The proposed IOL-SR framework aims to provide robust and efficient response generation capabilities, thereby facilitating adaptive, interpretable, and user-centric HCI systems. The following are the contribution highlights of the article:

- The design and description of IOL-SR in HCI systems for different input types and user requirement functions. This method is aided by monolayer recurrent learning with one-condition and multi-iterate validation using different input and interaction times.
- The classification process uses the optimization model discussion to identify discrete and sequential inputs for improving the prediction and understandability of performance features.
- The interpretation assessment is a part of the proposed model to understand the input and validate the context, understandability, and adaptability of the system with fewer errors to augment the swiftness in responding to users.
- The analysis and illustrative discussion of the metrics towards performance-oriented improvements, data-correlated outcomes, and efficiency verification are presented.

2 Related Works

Yan et al. [24] developed a hybrid statistic and machine learning approach for virtual reality behavioral response in HCI. A partial least squares structural equation model is employed in the approach, which analyses the activation functions. It is a two-stage approach that provides effective functions and features to virtual reality development services. The developed approach improves the accuracy level of the prediction process. Guo et al. [25] proposed an open personalized human–computer interaction system for New Year Painting based on a learner model. The system incorporates learner-related information to support personalized interaction and learning, with an emphasis on analyzing learners' emotional states during human–computer interaction. The study further employs learner-model updating and intelligent decision-making mechanisms to adapt the interaction and teaching process according to individual learner characteristics.

Munguia-Galeano et al. [26] introduced an affordance-based human-robot interaction (HRI) model. An RL algorithm is employed here to provide relevant resources for interaction enhancement. The RL algorithm also reduces the size of space which enhances the feasibility level of training systems. The model reduces the computational delay and cost of the training process. The introduced model enhances the performance and efficiency range of HRI systems. Prasad et al. [27] developed a mixture of variational experts for HRI systems. A variational autoencoder is implemented in the method to analyze the necessary features for interaction. The variational autoencoder uses a mixture density network which captures the relevant values for HRIs. Experimental results show that the developed method elevates the accuracy ratio of interaction services.

Bukht et al. [28] designed a quadratic discriminant analysis-based HCI framework. A hidden Markov model is used here to classify the features and emotions of the interactions with human beings. A hidden Markov model classifier is used to classify the exact types of data that are used as datasets in providing interaction services to the users. The designed framework increases the accuracy, quality, and reliability range of HCI. Qu et al. [29] introduced a training and assessment system for HCI using eye-tracking data and functional near-infrared spectroscopy. The introduced system also uses ratification intelligence technology to understand the actual necessity of the users. The functional near-infrared spectroscopy extracts the

physiological features that are presented in the dataset. The introduced system enlarges the robustness and naturalness of the interaction services.

Liu et al. [30] proposed the AdaFN-AG model for multimodal sentiment analysis in human–computer interaction systems. The method integrates adaptive feature normalization with an attention-gating mechanism to extract meaningful representations from textual and non-textual interaction signals. Experimental results demonstrate that the deep learning framework improves feature fusion, interaction understanding, and response accuracy in multimodal interaction environments.

Liu et al. [31] proposed a new emotion assessment model using backpropagation neural network and artificial bee colony algorithms. The proposed model predicts the actual emotional semantics of the users to produce relevant interactions. The relationship between emotion and key elements is also identified which decreases the latency. The proposed model maximizes the accuracy and feasibility range of the emotion assessment process.

Peng et al. [32] designed an intention recognition-based HCI for mixed-flow assembly. The model uses a Gaussian mixture distribution to identify the operational intensity of the users. Both spatial and temporal features and effects are calculated to get reliable sequences for interaction services. The designed model improves the quality and reliability level of the HCI systems. Dong et al. [33] developed an adaptive impedance decentralized control for a physical HRI system. A neural network observer is employed here to track the error that causes issues in the interaction process. It is a dual closed-loop method that is developed to analyze the optimal factors for interaction development processes. The neural network observer controls the error rate which elevates the feasibility of the systems. The developed method increases the effectiveness range of the systems.

Liao and Cai [34] introduced an augmented reality-enabled digital twin for HRI in manufacturing systems. The introduced model is used to improve spatial awareness and the significance range of the systems. The digital twin is used here to optimize the twining operation for interactions. The introduced model maximizes the performance and efficiency level of the manufacturing systems. Liu et al. [35] proposed a closed-loop HCI method for human intention prediction and knowledge recommendation. A function-behavior-structure model is employed here to analyze the dimensional actions of the users. The prediction method predicts the class and types of intention according to behavioral patterns. The proposed method increases the accuracy ratio of prediction and recommendation services.

Zhou et al. [36] designed a multimodal intention-understanding algorithm using comfort analysis. The designed model identifies the spatial and emotional status of users by analyzing the previous interactions. The comfort analysis is used here to analyze the features and patterns of the intention. The designed model enlarges the overall understanding rate of the interaction systems. Zheng et al. [37] developed a perceived value enhancement method for HCI systems. The actual impact of the interactions is calculated to identify the quality of HCI. The developed method provides effective structural paths to stabilize the understandability level of the services. It also produces feasible recommendations to perform tasks for the users. The developed method decreases the computational complexity and latency range of the systems. Alzubi et al. [38] introduced a new HCI for smart learning systems. A multi-layer convolutional neural network algorithm is used here to extract the relevant features from the interaction services. The extracted features are used as input which minimizes the computational cost of the systems. When compared with others, the introduced model elevates the performance of the learning systems.

Raees et al. [14] presented a comprehensive review on the transition from explainable artificial intelligence to interactive artificial intelligence in human–AI interaction systems. The study analyzes recent deep learning and interactive learning frameworks that support user-centered decision-making and adaptive

interaction. The review highlights how deep learning–based models improve interpretability, collaboration, and responsiveness in modern human–computer interaction environments.

Reviewing the landscape of existing HCI methods, three key gaps emerge that motivate the proposed IOL-SR framework. First, RL-based methods [26,27] improve adaptability but rely on multi-layer architectures requiring substantial training data and computational resources, limiting real-time feasibility. Second, optimization and normalization approaches [30,31] improve feature extraction but do not incorporate reinforcement-based feedback to handle missing or discrete input sequences dynamically. Third, sequence classification methods [28,29] address input type identification but lack an integrated learning mechanism for filling unrecognized sequences in real time. The proposed IOL-SR framework uniquely addresses all three gaps by integrating mono-layer RL, sequence-aware normalization, and keyword-guided interpretation within a single computationally efficient pipeline, which no prior HCI method has achieved jointly.

3 Data Setting and Environment

The dataset comprises interaction logs from 27 operators observed over 151 days, sourced from a publicly available machine operating environment (refer to https://doi.org/10.1038/s41597-023-02741-8). While this environment is domain-specific, it captures meaningful diversity in HCI through: (1) six distinct machine function types with varying input modalities (text, numeric, alphanumeric, symbolic), (2) role-differentiated operator access levels introducing variation in interaction patterns, and (3) a longitudinal span of 151 days capturing temporal drift in usage behavior. These characteristics make the dataset suitable for evaluating sequence classification, normalization, and understandability assessment. In addition, the dataset is partitioned chronologically into 100 days for training, 25 days for validation, and 26 days for testing, ensuring temporal consistency and preventing information leakage. This setup enables reliable evaluation of sequence learning under evolving interaction patterns. While the dataset reflects an industrial HCI environment, it preserves key properties such as input heterogeneity, sequential dependencies, and user-specific variability. However, validation across broader HCI domains, including graphical user interfaces, mobile interaction systems, and consumer applications, is identified as an important direction for future work to further assess the generalizability of the proposed framework.

The dataset employed in this study is derived from a domain-specific HCI environment comprising heterogeneous user interaction data, including keystroke sequences, alphanumeric commands, contextual interaction records, and system response logs. These data are generated within an HCI environment and provide information regarding user actions and corresponding system behaviors.

Such interaction logs enable the proposed IOL-SR framework to learn input patterns, reduce ambiguities, and improve response generation under input constraints. To ensure diversity, the dataset captures multiple operating regimes characterized by varying dust loads, environmental conditions, and airflow dynamics across different time intervals and operational cycles. This temporal and contextual variability enhances the robustness of the learned model against noise, non-stationarity, and real-world fluctuations.

Diversity in the dataset is analyzed along several parameters to provide a rigorous evaluation of the proposed approach. First, diverse types of input modality, ranging from textual commands, numeric inputs, alphanumeric codes, and symbolic inputs, reflect a wide spectrum of actual HCI. Second, interaction logs from 27 operators are used for analysis of user behavior, providing variability in input sequences, types of commands, and errors in the HCI process. Third, the long-term nature of the experiment, spanning 151 days, considers behavioral change over time and varying interaction frequency rates. Despite the industrial origin of the dataset that describes human-machine interactions, the characteristic features identified, such as variability in input sequences, partial input sequences, and user-specific characteristics, are typical of numerous other types of human-computer interfaces. In the proposed IOL-SR architecture, these multiple

inputs are fed into an organized pipeline that begins with normalization (N_{or}) of the raw interaction sequences to normalize the variance and regenerate incomplete sequences, which are subsequently classified using sequence classification (α) into continuous and discrete inputs. The resulting sequences are further improved by processing them using the mono-layered RLapproach, wherein the operations of the process are determined using the operation (o_p), reward (w_f), and value (v_u) functions. However, since this dataset is derived from an industry setting, it is important to note that the interaction characteristics described, such as sequence variations, partial inputs, and user-based input patterns, exist in numerous other HCI applications. Therefore, the domain-specific origin of the dataset does not limit generalizability, as the proposed framework learns domain-independent sequence patterns that are transferable across diverse HCI environments.

Ethics and Privacy Statement: The dataset used in this study is publicly available at https://doi.org/10.1038/s41597-023-02741-8 and was collected under institutional oversight with operator consent for anonymized research use. All operator identifiers in the public release are pseudonymized, and no personally identifiable information—including names, biometric data, or individual-level behavioral profiles—is accessible or used in this study. The analysis is conducted solely at the interaction-sequence level (input types, timestamps, response patterns) and complies with established guidelines for responsible HCI research as outlined by the ACM Code of Ethics. No additional ethical approval was required for the use of this publicly released dataset.

To ensure methodological rigor and avoid information leakage, the 151-day dataset was split chronologically into three non-overlapping phases. Days 1–100 (66%) were used for training, encompassing keyword extraction (gr, w_y), initial sequence classification (α), and mono-layer RL initialization. Days 101–125 (17%) formed the validation set, used for tuning the reward function wf and understandability thresholds. Days 126–151 (17%) constituted the held-out test set on which all reported performance metrics (classification rate, misdetection, understandability, latency) were evaluated. This chronological split prevents temporal information leakage. Cross-user generalization (i.e., training on a subset of operators and testing on unseen operators) is planned for future investigation to further assess robustness. A model of the user interface design, data collected, and its setting is shown in Fig. 1.

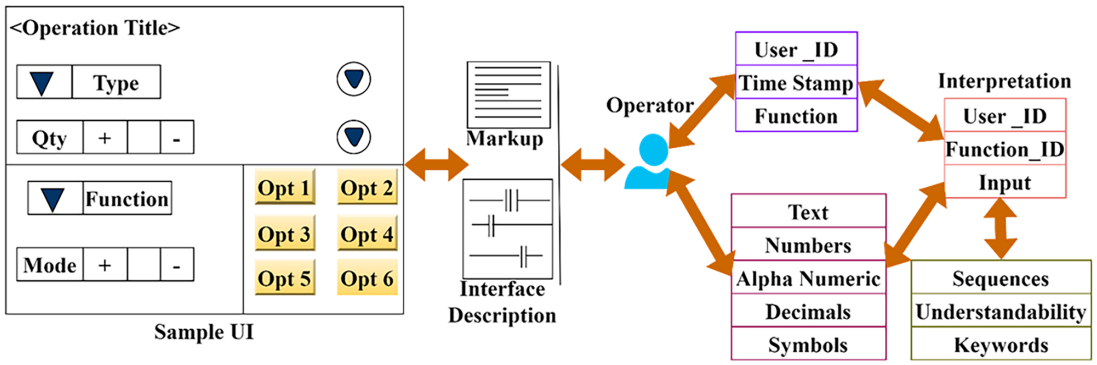


Figure 1: A model of user interface with data and setting.

The operating environment provides 6 different options to provide machine functions by selecting their type and quantity. The function type varies for different operators categorized in the interface description. The markup language is responsible for the interpretation of function and user input. The inputs are text, numbers, alphanumeric, symbols, etc. used to control and define the functions of the machine. In the interpretation process, the input sequences are used to identify text keywords from the understood

operations. The time stamp refers to the minimum and maximum interaction interval allocated to an operator at an average rate. This user interface model is utilized to evaluate the proposed IOL-SR method with the data provided (Fig. 1).

Table 1 summarizes the symbols used in the IOL-SR framework. Each notation is defined to clarify equations, ensure consistent terminology, and improve readability of the mathematical model.

Table 1: Notation summary for the proposed interpreted optimization learning framework framework.

Symbol	Description
mz	Interaction optimization metric between user and system
u_0	User initiating the HCI request
er, qu	Interaction request (ererer) and system response (qququ)
ad	Input data: $ad(n)$ represents the number of data instances
ab	Attribute derived from input data for computation
fu	Feature extracted from the interaction input
H_c	Human–Computer Interaction system model
y_i	Identification factor for accurate response generation
v_c	Input command issued by the user
E'	Detection phase indicator used in sequence classification
U_d	Interaction understandability level
δ	Metric used for understandability analysis
gr, w_y	Training function (gr) and keyword dataset (w_y)
ct, ds	Continuous (ctctct) and discrete (dsdsds) input sequences
op, wf, vu, bk	RL parameters: operation, reward, value, and feedback
ck	Periodic checking mechanism during interaction evaluation
I_r	Interpretation output generated by the system
λ	Prediction output produced by the RL stage
A_y	Adaptability enhancement factor
Nor	Normalization function applied to input commands
X_0	Final interaction assessment metric
N_i	New interaction factor used for adaptability analysis
z_r	Reorganization factor used in understandability checking

4 Proposed Interpreted Optimization Learning for Swift Response

The interaction between humans and computers plays a vital role in identifying the input as the request from the user, and the computer accepts it and understands the standard. The standard of accepting the input command differs from human to computer; this plays an efficient role in the detection of requests. The proposed framework uses generic sequences and not specific semantics for commands. This makes the framework suitable for handling any input type from HCI. The request handling is tracked for every sequence. There are commonly two types of sequences, such as continuous and discrete. Based on these two sequences, the continuous form observes the efficient request handling between the human and computer. This strategy illustrates how to solve the critical problem by enhancing knowledge-based learning. Knowledge learning is defined as studying the previous standard of user request match with the present module. This observation method is taken into consideration and evaluates the mapping with the current and previous modules and provides the result. This approach studies the RL and implement the mono-layer

to maximize the possible chances of feature augmentation to fill the discrete sequences. The initial stage of the proposed Interpreted Optimization Learning for Swift Response (IOL-SR) framework optimizes the interaction between the user and the human–computer interface by maximizing semantic correspondence between the input request and the generated response. Let u_i denote the i^{th}–user, e_i–the input request, q_i–the generated response, a_i–the extracted interaction attributes, H_i–the interaction representation, and f_i–the semantic feature vector. The interaction optimization objective is formulated as

$$m_z = \arg\min_{\theta} \left[\frac{1}{N} \sum_{i=1}^{N} \| H_i - f_i \|_2^2 + \lambda \frac{1}{N} \sum_{i=1}^{N} \| q_i - \hat{q}_i \|_2^2 + \beta \| \theta \|_2^2 \right]. \tag{1}$$

$$a_i = \phi(e_i, u_i), \qquad H_i = \psi(a_i, f_i), \qquad f_i = \Gamma(a_i, H_i), \qquad \forall\, i = 1, 2, \ldots, N. \tag{2}$$

As shown in Eq. (1), the interaction optimization objective represented by m_z minimizes the joint loss associated with interaction representation and response prediction. Initially, the user request e_i is transformed into an interaction attribute vector a_i through the feature extraction function $\phi(\cdot)$. The interaction representation H_i is generated using the semantic mapping function $\psi(\cdot)$, whereas the feature refinement function $\Gamma(\cdot)$ extracts the discriminative interaction features required for response generation. The first objective minimizes the discrepancy between the interaction representation and semantic feature vector, while the second objective minimizes the response prediction error. The regularization term constrains the model parameters to improve numerical stability and generalization performance during optimization.

The optimized interaction representation is subsequently utilized for response classification. The continuous interaction features are transformed into posterior probabilities using the Softmax decision function, and the final interaction category is determined according to the maximum posterior probability. The response classification model is expressed as

$$\alpha = \begin{cases} P(c \mid H_i) = \dfrac{\exp\left(W_c^T H_i + b_c\right)}{\sum_{k=1}^{K} \exp\left(W_k^T H_i + b_k\right)}, & \text{Continuous}, \\ \hat{c} = \arg\max_{c} P(c \mid H_i), & \text{Discrete}. \end{cases} \tag{3}$$

As inferred from Eq. (3), where W_c and b_c denote the trainable weight matrix and bias corresponding to class c, respectively. Eq. (3) describes the response classification stage of the proposed IOL-SR framework. The optimized interaction representation generated through Eq. (1) is projected into the response space using the Softmax classifier to estimate the posterior probability of each interaction category. Continuous interaction information is represented by the probability distribution $P(c \mid H_i)$, whereas discrete interaction labels are obtained through the maximum a posteriori decision rule. This formulation establishes a mathematically consistent relationship between interaction optimization and response prediction, enabling reliable interpretation of user requests and accurate generation of system responses while maintaining analytical consistency throughout the proposed optimization framework. To provide an intuitive understanding of continuous and discrete sequences, their key differences along with real-world examples are summarized in Table 2.

The proposed IOL-SR framework operates through a sequential processing pipeline. Initially, raw user commands are collected and normalized to eliminate inconsistencies and identify incomplete or ambiguous command segments. The normalized sequences are subsequently processed by the sequence reconstruction module, which restores missing elements and generates consistent command representations. Feature enhancement is then performed to augment the reconstructed sequences and extract discriminative

features relevant to command interpretation. The enhanced feature vectors constitute the state representation supplied to the mono-layer RLmodule. Based on the current state and the response generated in the previous iteration, the policy module selects an action that maximizes the expected reward associated with accurate and timely response generation. Feedback obtained from the generated response is used to compute the reward and update the policy parameters iteratively. Finally, the optimized policy produces the response with reduced latency and improved interpretability.

Table 2: Distinction between continuous and discrete sequences in HCI inputs.

Aspect	Continuous Sequence (c_t)	Discrete Sequence (d_s)
Definition	Complete and well-formed input sequence	Incomplete or fragmented input sequence
Structure	Follows expected command format	Missing or inconsistent elements
Industrial Example	"F3-QTY-005"	"F3-QTY-"
Mobile Example	"Set alarm for 7:30 AM"	"Set alarm for 7:"
Conversational Example	"Book a ticket to... tomorrow"	"Book a ticket to. . ."
Interpretability	Directly interpretable by system	Requires reconstruction and normalization
Error Probability	Low misdetection rate	Higher misdetection rate
Processing	Processed directly in classification	Requires feature augmentation and correction
Impact on Performance	Higher accuracy and faster response	May increase latency if not corrected

The mathematical formulations are interpreted by linking each equation to its functional role in practical HCI scenarios. The optimization function (refer to Eq. (1)) represents the mapping between user input and system response, capturing how extracted features contribute to accurate interpretation. The sequence classification model (refer to Eq. (3)) distinguishes between continuous sequences (complete and well-formed inputs) and discrete sequences (incomplete or erroneous inputs), which commonly occur in real-world interactions due to missing values or input inconsistencies.

In addition to the classification rate heatmaps shown in Fig. 2, the performance of the proposed IOL-SR framework was evaluated using standard classification metrics. The obtained results demonstrate an average precision of 94.47%, recall of 93.77%, and F1-score of 94.12%, with an overall classification accuracy of 95.11% and an AUC value of 0.964 as presented in Table 3. These results indicate that the proposed framework effectively distinguishes among different input sequence categories while maintaining robust classification capability and reliable response generation.

The distinction between continuous and discrete sequences is essential for handling real-world interaction inputs, where incomplete or fragmented commands frequently occur due to user errors or system constraints. Continuous sequences represent complete and directly interpretable inputs, whereas discrete sequences capture missing or irregular patterns that require reconstruction. This explicit separation enables the model to apply targeted processing strategies, improving detection accuracy and reducing misinterpretation.

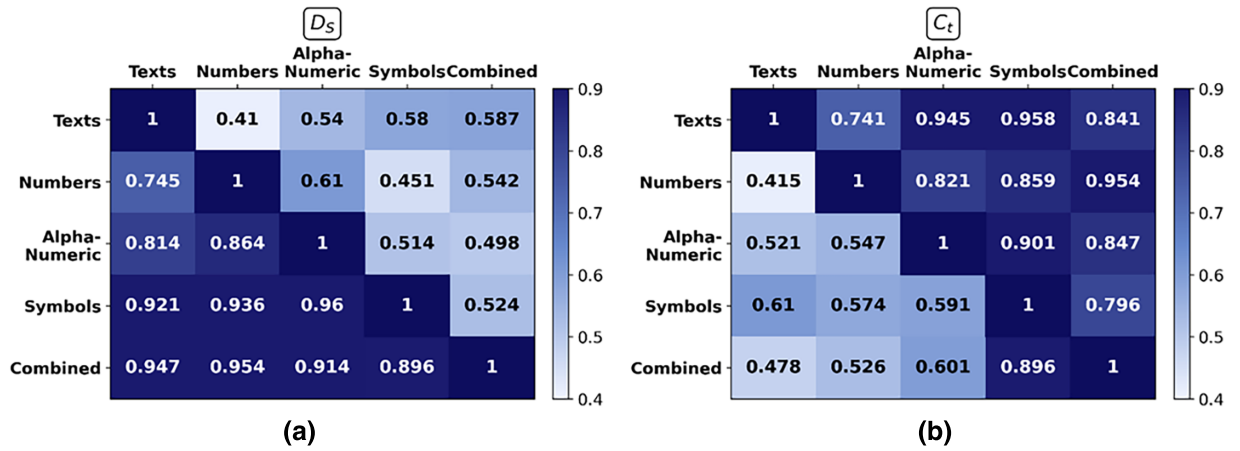


Figure 2: Classification rate assessment based on inputs. (**a**) D_S—discrete sequence matrix, (**b**) C_t—continuous sequence matrix.

Table 3: Classification performance of the proposed IOL-SR framework.

Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	AUC
Texts	94.32	93.85	94.08	95.12	0.964
Numbers	93.68	92.97	93.32	94.48	0.956
Alpha-Numeric	95.41	94.63	95.02	95.87	0.973
Symbols	92.84	91.95	92.39	93.74	0.948
Combined	96.12	95.47	95.79	96.34	0.981
Average	94.47	93.77	94.12	95.11	0.964

In contrast to transformer-based and sequence-to-sequence models, which generally rely on complete input representations and involve higher computational complexity, the proposed approach focuses on lightweight and real-time sequence handling. By combining sequence classification with reinforcement-based feature augmentation and normalization, the framework efficiently reconstructs incomplete inputs without requiring deep architectures. This makes it more suitable for low-latency HCI systems where rapid and accurate interpretation is critical. The classification rate assessments based on multiple inputs from the data sources are analyzed in Fig. 2.

Fig. 2 presents the classification-rate matrices for continuous (c_t) and discrete (d_s) input sequences in the proposed IOL-SR framework. The interaction optimization metric m_z in Eq. (1) minimizes the loss associated with the interaction representation H_i, feature vector f_i, and response prediction error. Eq. (2) generates the interaction attribute a_i from the input request e_i and user identifier u_i, followed by H_i and the refined feature f_i. These representations are classified using α in Eq. (3), where $P(c|H_i)$ is determined using the class weight W_c and bias b_c. The figure shows that continuous sequences generally achieve stronger classification associations because c_t contains complete and well-formed input information. In contrast, d_s contains incomplete or fragmented information, producing comparatively weaker classification associations and requiring additional reconstruction and feature processing.

Fig. 3 illustrates the misdetection rates of continuous (c_t) and discrete (d_s) sequences under five parameters: input type, interaction rate, query rate, function, and interaction time. Eq. (3) provides the classification decision used to determine whether sequences are recognized or misdetected. Eq. (4a) performs keyword-based training using g_r, keyword data w_y, input data a_d, extracted feature f_u, detection

indicator E', user command v_c, and identification factor y_i. Eqs. (5a) and (5b) evaluates the interaction using u_0, v_c, e_r, q_u, H_c, f_u, a_d, and y_i, while Eq. (5c) determines understandability U_d and Eq. (5d) defines its conditions. The figure shows consistently higher misdetection for d_s than for c_t across the evaluated parameters, indicating that incomplete discrete sequences are more difficult to interpret than complete continuous sequences. In this case, $q_u(c_t + d_s) * I_r$ represents the interaction data sequence used for reinforcement learning (RL) training. The RL process is used to address sequence misdetection when the input sequences are discrete, as represented by $E' + d_s(u_0)$. In this approach, understandability is evaluated using a reward function, and the policy is assessed based on $\left(o_p + w_f\right) * z_r$.

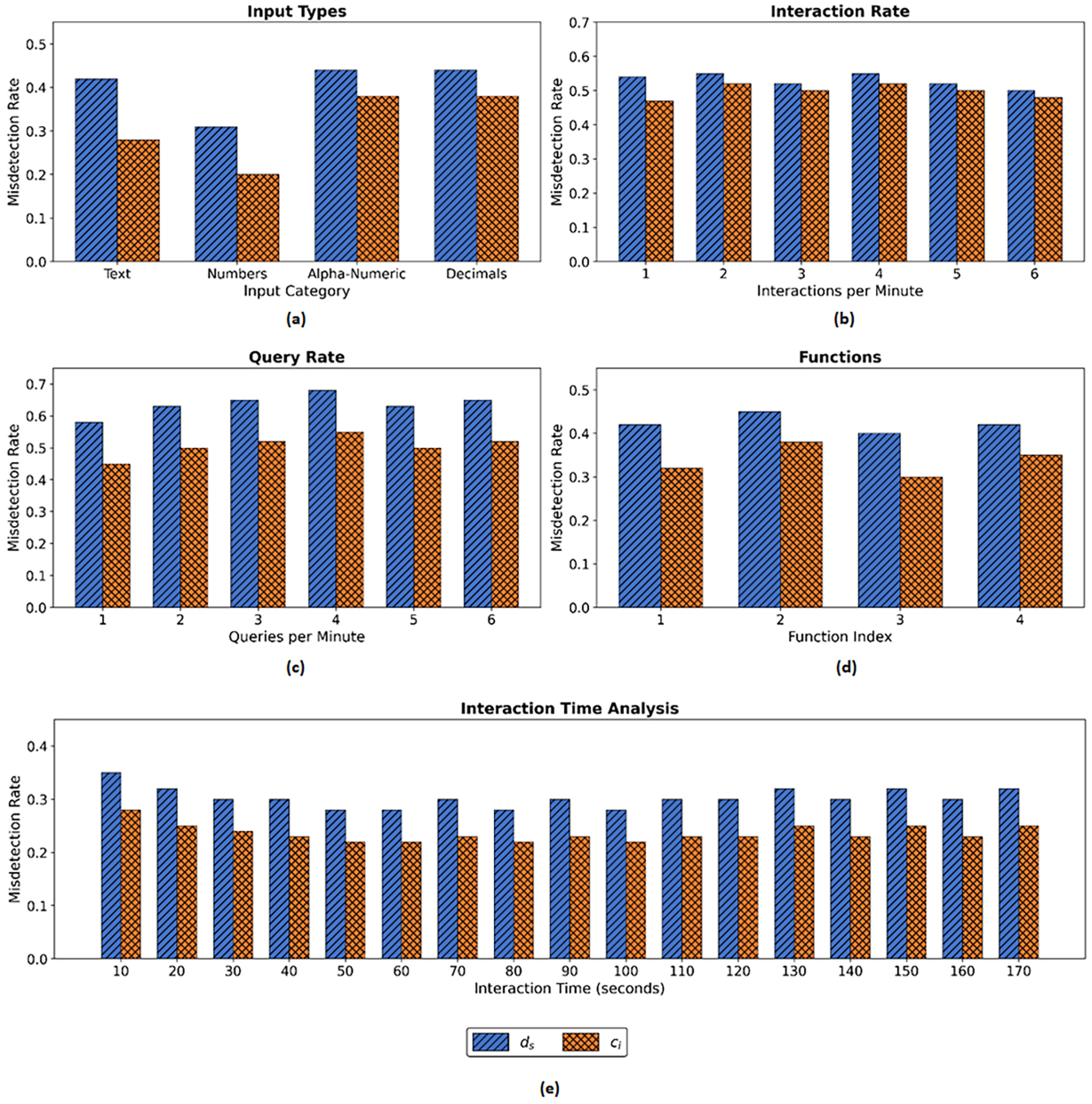


Figure 3: Misdetection identified in continuous (c_t) and discrete (d_s) for various parameters. (**a**) Input types, (**b**) interaction rate, (**c**) query rate, (**d**) functions and (**e**) interaction time analysis.

On examining this method, the interpretation is represented by $I_r + (q_u + c_t) < y_i$. By evaluating the resulting misdetection, the optimization process facilitates keyword detection. If the keyword is not detected correctly, misdetection occurs, represented by $E'(w_y + y_i)$, and the corresponding features are subsequently processed using $f_u(\delta + Nor) * a_b$. From this phase, analysis is carried out for the request and response approach and estimates the interpretation between user responder. This phase indicates the classification phase and exhibits the sequences $q_u * \sum_{\alpha}\left(a_b * a_{d(n)}\right)$, for $a_{d(n)} \in H_c$. The optimization is carried out for the interaction $H_c + (E' * \alpha)$, for all $E' \in y_i$. This indicates the sequences of data processing and estimates the understandability $\alpha + \frac{1}{a_{d(n)}} * (H_c + y_i)$, $for\ all\ a_{d(n)} \in q_u$. The evaluation is carried out for the optimization mechanism, $m_z + (a_d * u_0)$, that estimates the discrete process. The discrete sequences are evaluated where the external process is observed, $q_u + (a_b * y_i)$. Post to this method, training keyword data is done as the external training attribute and it is formulated below.

$$g_r(w_y) = \prod_{e_r}^{H_c} (f_u + a_d) * (c_t - d_s)\, q_u + y_i * \|\alpha + (E' * v_c)\| \tag{4a}$$

It is computed as,

$$q_u = (c_t - d_s) * \frac{\alpha + v_c}{a_{d(n)}} + y_i, \text{ where } a_{d(n)} \in u_0 \tag{4b}$$

It is expressed as,

$$y_i\left(a_{d(n)}\right) = \prod_{v_c} E' * (a_d + m_z) * q_u, \text{ for all } m_z \in H_c \tag{4c}$$

The training keyword is included in the external training section where the attribute is taken into consideration $v_c(a_b) * \sum_{m_z}(a_d * H_c)$. The training is represented as g_r, whereas w_y indicates the keyword data. The continuous and discrete are symbolized as c_t and d_s. This mechanism illustrates the sequences of input which might be continuous or discrete. Taking continuous format, the data processing is done sequentially, whereas, discrete is defined as the discontinuous data processing. This is observed $q_u(c_t + d_s) * E'$. This approach is taken into account and provides the training with the keyword whereas; the machine acquires the particular keyword and provides the computation. Based on this section, relies on keyword searching as a cookies format that includes the computation as interaction. The interaction is done for the varying sequence changes as continuous and discrete $q_u(E' + a_d) * a_b - m_z$, whereas $q_u \in (c_t, d_s)$.

This strategy is used to provide efficient HCI where it performs the mapping with the previous set and evaluates the detection of features for input command. The classification is followed up for continuous and discrete methods where the input command varies accordingly $\alpha(c_t - d_s) * \sum_{qu} E'$. This phase indicates keyword data for external training attributes and observes understandability among humans and computers. The keyword process is used as the training set with the list of attributes $E' * (u_0 \to v_c) + \sum_{q_u}(a_b * y_i)$. The detection is based on the training section of attributes for input command input. From this training with keyword data, the analysis of understandability is estimated in the below equation.

$$\delta = \frac{u_0}{v_c(e_r)} + \sum_{q_u}^{a_b} (\alpha * H_c(e_r)) + \|y_i + q_u\|_{a_b}^{a_d}, \quad \forall\, y_i \in a_d \tag{5a}$$

It is substituted with Eq. (4c) to attain the interaction,

$$I_{ct} = \frac{1}{a_d(n)} \sum H_c\,(a_b + \alpha)\,(E' + a_d)\, y_i^2 - q_u, \quad \forall q_u \in c_t \tag{5b}$$

Since it is a continuous sequence, the evaluation takes place to detect the interaction to find the understandability level whether it is higher, medium, or low.

$$U_d = \frac{a_{d(n)}}{v_c} \rightarrow e_r \sum_{a_b \in a_d} \left(E' * H_c + w_y\right)(d_s), \quad \forall\, w_y \in g_r \tag{5c}$$

The analysis is done for understandability which is processed with the uses of the above derivative and shows the analysis based on its level of observation.

$$\delta(U_d) = \begin{cases} u_0 + (v_c * w_y) - d_s > y_i, \\ a_b(a_d) + c_t * e_r = y_i, \\ f_u(E' + a_d) + c_t < y_i. \end{cases} \tag{5d}$$

The above equation represents the understandability and it is described as U_d, the analysis is δ. This stage illustrates the user performs the input command where the training is given based on a request $v_c > y_i$, is low. The second condition states $a_b * e_r = y_i$ is medium, the third condition indicates $f_u + c_t < y_i$, is high. From this state of the art, the low, medium, and higher understandability is defined based on user request as an input command. The evaluation takes place by detecting the continuous and discrete process where the features of the data are considered. From this approach, training is followed up HCI and finds for the interaction between user and computer. The understandability analysis for different conditions referred to in the above equations is analyzed in Fig. 4.

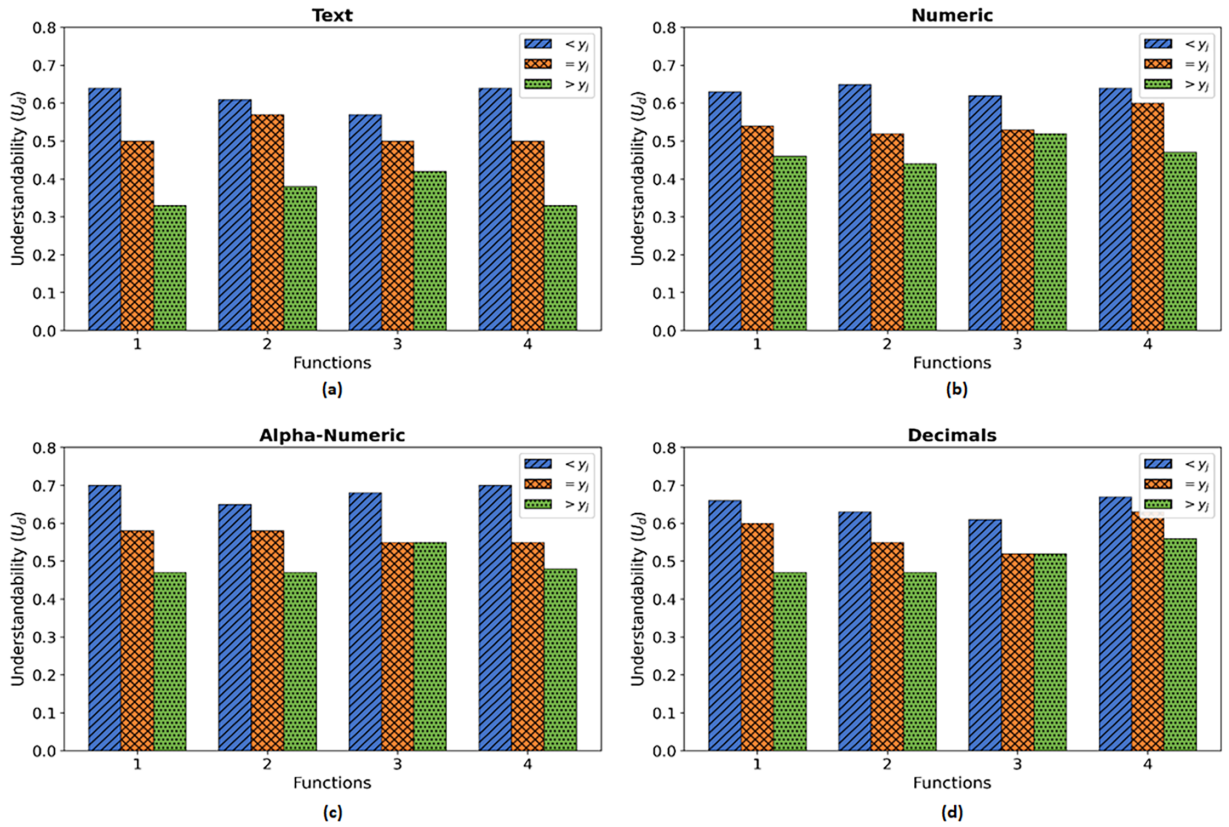


Figure 4: (Continued)

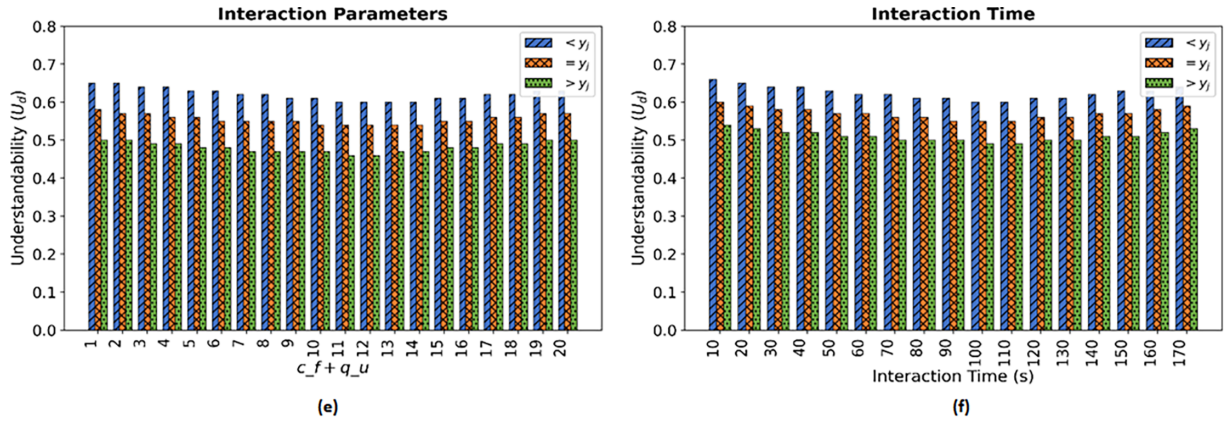


Figure 4: Understandability analysis for different y_i conditions. (**a**) Text, (**b**) numeric, (**c**) alpha numeric, (**d**) decimals, (**e**) interaction parameters, and (**f**) interaction time.

The U_d impacting factors are $g_r\left(w_y\right)$ and q_u for which $a_d\left(n\right) \in u_o$ defines m_z rate. Based on d_s and c_t, θ_z range is defined to ensure $\left[q_u\left(c_t + d_s\right) * E'\right]$ = maximum of $\left(a_d * H_c\right) \forall\ q_c$. The range of responses for $\left(a_b - m_z\right)$ reduces the change of δ and $w_y\left(d_s\right)$. In this case, the U_d assessment for $> y_i$, $= y_i$, and $< y_i$ with $\delta\left(U_d\right)$ is validated. The interaction time and $\left(e_r + q_u\right)$ are the major factors for $\left[\left(v_c * w_y\right) - d_s\right]$ and $\left[\left(E' + a_d\right) + c_t\right]$ such that low, medium, and high U_d is distinguished (refer to Fig. 4). Thus, the understandability is analyzed in the above equations, and from this mono-layer reinforcement is introduced to find the operation, reward function, and value function and they are discussed below.

The workflow of IOL-SR framework represented in Algorithm 1. Input Acquisition, Feature Extraction, Normalization, Sequence Classification Process, Mono-layer RL Process that evaluates particular actions and their rewards to completely update the interaction state, and then predictions and Keyword-Based Interpretations generate the optimized response from the system.

Algorithm 1: Interpreted optimization learning for swift response (IOL-SR)

Input: Interaction command v_c, interaction dataset a_d, keyword set w_y
Output: Response r_p
Acquire interaction command v_c
Preprocess and normalize the interaction input
Extract semantic features
    $f_u \leftarrow \phi(v_c)$
Construct interaction representation
    $H \leftarrow \psi(f_u)$
if interaction sequence is continuous **then**
    $S \leftarrow$ ContinuousProcessing(H)
else
    $S \leftarrow$ DiscreteProcessing(H)
end if
Initialize RLstate s_t, policy π, and value function V
repeat

(Continued)

Algorithm 1 (continued)

$a_t \leftarrow \pi(s_t)$
Execute action a_t
Observe reward r_t and next state s_{t+1}
$V(s_t) \leftarrow V(s_t) + \alpha\left[r_t + \gamma V(s_{t+1}) - V(s_t)\right]$
Update policy π
$s_t \leftarrow s_{t+1}$
until convergence criterion is satisfied
Optimize interaction representation
$\lambda \leftarrow \text{Optimize}(H, V)$
Predict interaction class
$I_c \leftarrow \text{Softmax}(WH + b)$
Generate response
$r_p \leftarrow \text{GenerateResponse}(I_c, w_y)$
return r_p

5 Mono-Layer Reinforcement Learning

This monolayer is a single-way process where no recurrence is needed after the identification of continuous and discrete sequences. It adopts the maximum possible chances of feature augmentation to fill the discrete sequences. This intends to maximize the feature sequences without errors and thereby improve the interpretation sequences. The mono-layer RLmodule is designed as a lightweight decision mechanism using operation, reward, and value functions. Unlike deep RL models, it avoids iterative backpropagation and multi-layer updates, enabling faster convergence and reduced computational cost. It is particularly suitable for HCI scenarios with bounded interaction sequences, achieving efficient adaptation with minimal latency. The following parameters facilitate RL and they are listed below.

1. Operation: Based on the current state of the input command, the subsequent operation is determined by considering the current state, previous state, and optimization parameters. The operation function is expressed as

$$\delta(o_p) = \frac{(a_b + m_z)(r_u - s_p)}{\sum_{i=1}^{a_d(n)} \left(\alpha e_{r_i} + \gamma_p\right)}, \quad \forall e_{r_i} \in v_c \tag{6}$$

The operation score is further represented as

$$o_p = \sum_{i=1}^{a_d(n)} \left[(r_u - s_p) + v_c(e_{r_i})\right] \tag{7}$$

where r_u and s_p denote the current and previous states, respectively, e_{r_i} represents the input error sequence, and $v_c(e_{r_i})$ denotes the correction component associated with the error sequence. Furthermore, a_b is the action coefficient, m_z is the optimization parameter, α is the learning coefficient, and γ_p is the penalty parameter. The operation score o_p considers both the current and previous states represented by r_u and s_p, respectively. The input data are used to identify the subsequent action based on information derived from previous states, thereby facilitating efficient response generation.

2. Reward function: Based on the state and action the feedback is given.

$$\delta\left(w_f\right) = e_r\left(r_p\right) \rightarrow s_p * \prod_{q_u} c_t + \left(a_b * b_k\right), \text{ for all } b_k \in r_p$$
$$= b_k - r_p\left(u_0\right) * v_c(s_p) \tag{8}$$

The reward function is w_f, the feedback is described as b_k, this case indicates the request is the state and input command processing with sequences in action. Based on these parameters feedback is given to improve the understandability $U_d(u_0\left(e_r\right) + q_u)$.

3. Value function: From the attained state the reward function is generated which is performed under the value function.

$$\delta\left(v_u\right) = \frac{u_0}{e_r} \rightarrow r_p\left(c_k\right) * b_k + w_f, \text{ for all } w_f \in b_k$$

It is computed as,

$$= \left(b_k - w_f\right) - s_p(U_d) \tag{9}$$

The value function is labeled as v_u, the checking is processed to attain the reward function and it is described as c_k.

The monolayer recurrent learning is diagrammatically demonstrated in Fig. 5.

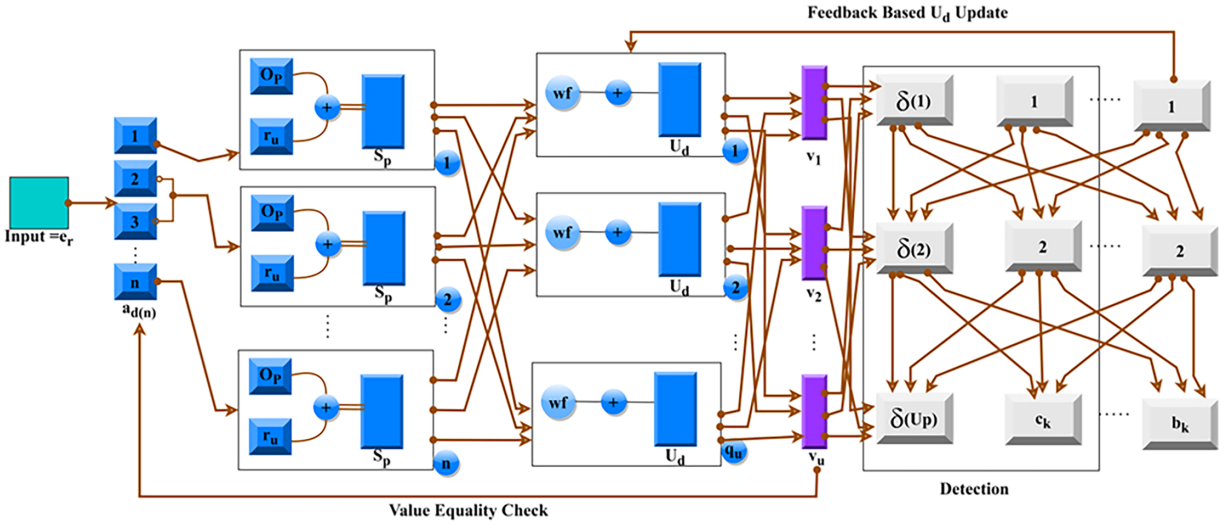


Figure 5: Mono-layer recurrent learning demonstration.

The mono-layer process is initiated from e_r inputs by accumulating $a_{d(n)}$ to improve θ_p functions. r_u from $\left(a_d * H_c\right)\ \forall q_u$ and s_p for $\left(a_b - m_z\right)$ are concurrently augmented for n such that both c_t and d_s are combined for θ_p. This conjoint process is assigned with w_f to improve the $U_d\ \forall v_u$ in any n. Therefore $\left(w_f \pm U_d\right) = v_u$ is the achievable condition for $\delta(1)$ to $\delta\left(\theta_p\right)$. This process requires a mono-detection of $\left[b_k - r_p\left(u_o\right) * v_c\left(s_p\right)\right]$ until $b_k \neq 0$. In the contrast case, $\left[\left(r_u - s_p\right) + v_c\left(e_r\right) \rightarrow r_p\right]$ increases the case of $\left[r_p\left(c_k\right) * b_k \pm w_f\right]$. There are two possible recurrences: v_u and b_k for $\left(w_f \pm U_d\right)$ and $a_{d(n)}$; the first is the recurrent improvement of w_f adjustment and the latter is the feedback-based assessment to maximize the c_k for maximum interpretation. These cases are recurrent until non-trivial b_k' is achievable (Fig. 5). For

the varying v_u and c_k values, the response ratio is analyzed and graphically presented in Fig. 6. The response ratio increases for the inputs based on user requests sequentially $u_0\,(H_c) * (q_u + E')$. The processing is carried out based on the user input where the optimization takes place accurately $u_0 \rightarrow U_d\,(\alpha * \delta) > f_u(e_r)$. This computation relies on the adaptability of the features where the interaction takes places $\delta * \frac{H_c}{f_u} + \lambda$. From this observation part $c_k + [(\alpha + U_d) * y_i]$ is done for the response to the user $r_p(u_0 + H_c) \rightarrow s_p$. The previous step is considered $s_p(m_p * (r_u + q_u)$, the features are accounted $f_u(c_t - w_f)$. This examination takes place based on the checking of user requests by recognizing $c_k\,(u_0 * a_b) * \sum_{E'} U_d \approx g_r(v_u)$.

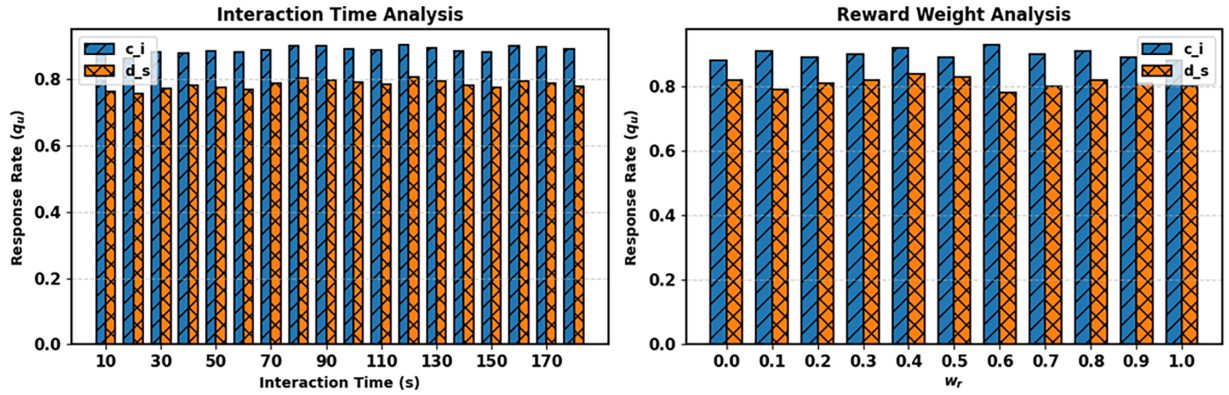


Figure 6: v_u and c_k value-based response analysis.

The proposed mono-layer RL mechanism models HCI as a bounded-horizon decision process, where each interaction occurs within a finite time interval and requires only a single-pass state evaluation. The mechanism is governed by the operation function (op), reward function (w_f), and value function (v_u), defined in Eqs. (6)–(9). The associated parameters include the current state (ru), previous state (sp), error sequence (er_i), action coefficient (ab), optimization parameter (mz), learning coefficient (α), penalty parameter (y_p), feedback (b_k), checking parameter (c_k), correction component (v_c), response (r_p), and understandability measure (U_d). The reward and value processes use response feedback (b_k), periodic checking (c_k), and understandability. These parameters allow the current interaction state to be evaluated without repeatedly revisiting previously processed sequences.

The process converges when valid feedback $(b_k \neq 0)$ is obtained, allowing (w_f) and (v_u) to determine the interaction understandability (U_d) without iterative backtracking. Since each interaction element is evaluated once, the mono-layer RL stage operates with $(O(n))$ complexity, while the complete framework has $(O\,(n \log n))$ complexity due to the classification stage. This single-pass design supports low-latency HCI processing. Its reliance on (b_k), (c_k), (w_f), and (v_u) may affect performance for highly variable interaction patterns. Errors in sequence classification (α) and limited keyword information (w_y) can further reduce (U_d). These limitations define the intended operating range of the proposed mono-layer RL mechanism.

$$\begin{aligned} E'\,(f_u) &= \prod_{a_d \in H_c}^{v_c} \left(w_y * s_p\right) + \left\{\left\|y_i^2\right\| * w_f(U_d)\right\} \\ &= y_i\left(s_p\right) * \left(u_0 * r_p\right) + \prod_{m_z} q_u\,(c_t - d_s) + w_f \\ &= q_u - d_s * (w_y + g_r) \end{aligned} \tag{10}$$

The feature detection is followed up by a discrete input command and it is associated with HCI where it executes the reward function. The reward function is used to state sequences of data and evaluates the classification for continuous and discrete $U_d \rightarrow q_u\left(w_y\right) - s_p$. The state and action are considered for this feature detection and generate the output $w_f - s_p\left(c_k + u_0\right) * e_r$. Thus, the feature indicates the discrete, and it is sorted out with the training phase $g_r\left(d_s\right) + y_i - c_k$. Post to this phase, the prediction is followed up in the below derivative.

$$\lambda = m_p\left(r_u - s_p\right) * b_k + \sum_{\delta \in w_f}\left(y_i + a_d\right) - d_s,\ for\ all\ d_s \in g_r$$

The prediction is done in RL to ensure a higher understandability of input data.

$$= \left(s_p * w_f\right) + \sum_{a_d}\left(\delta * b_k\right) + r_p - c_k \tag{11}$$

The prediction is followed up in the above equation and it is represented as λ, here mapping is done and it is labeled as m_p, where the periodic checking is done to estimate the state and action of input data $c_k(u_0 + r_p)$. The analysis is done with the reward function $w_f\left(d_s * y_i\right) + c_k$. The processing is carried out for the prediction process with efficient understandability. Here forth, the below equation is for the interpretation from the predicted value which maximizes the feature sequences without errors and thereby improves the interpretation sequences.

$$\begin{aligned} I_r &= \lambda * a_{d(n)} + \prod_{\delta}\left[\left(s_p + v_c\right) * m_p\right] w_y - b_k + y_i(U_d) \\ &= \sum_{\delta} a_d * s_p\left(b_k + w_f\right) - c_k \\ &= \frac{1}{a_{d(n)}} + \left[\left(w_f * v_u\right) + s_p\right] - c_k(w_y) \end{aligned} \tag{12}$$

The reinforced features are identified and from these predicted sequences are attained using the normalization and ensure swift understandability of the interacting system $U_d + \left(u_0 * w_y\right)$. The interpretation outputs for different parameters considered from the dataset and other variants are illustrated in Fig. 7.

The interpretation rate is processed with the interaction time that executes the better detection of features $f_u\left(r_p\right) * \prod_{\alpha}\left(d_s + m_p\right) \approx o_p$. This exhibits the policy-based analysis where the normalization takes place accurately based on the understandability $U_d + \left(f_u * E'\right) > w_f$.

The computation relies on the adaptability $A_y * (\alpha + H_c)$ where $e_r \approx \delta$, this case indicates $\left(q_u * w_f\right) + \sum_{c_t}\left(H_c * u_0\right)$. From this case, it estimates interpretation ratio increases $I_r\left(o_p * m_p\right) + v_c$, that are associated with continuous sequences. It indicates the policy and reward function that is done in monolayer RL $\left(o_p + w_f\right) \cong \delta(u_0 * e_r)$.

This case indicates $H_c * \left(r_p + q_u\right) + \prod_{w_y} E'(Nor)$, where it relies on the feature extraction process. The reward function is used to extract the adaptability of features and provides the prediction phase $\lambda + \left(\sum_{e_r}\left(c_t + c_k\right)\right)$. The periodic checking is followed up with the previous state $c_k\left(s_p * m_p\right) + w_y > E'(f_u)$. The interpretation process is used to evaluate the feature detection where the understandability takes place $U_d + \left(E' * e_r\right) + I_r$ (Fig. 7). The interpretation is performed using a known keyword dataset experienced by the HCI system that augments new feature chances using the normalization and reinforced processes $s_p\left(b_k + w_f\right)$. The interpretation is represented as I_r, which indicates the keyword-based searching

experiences. The upcoming process is the adaptability enhancement which uses prediction and periodic checking into consideration.

$$A_y = \lambda + \left(m_p * a_{d(n)}\right) + \sum_{c_k} w_y * \left(H_c + w_f\right),\ for\ all\ \lambda \in s_p$$

It is computed as,

$$= w_f * c_k\left(a_d + y_i\right) + m_z$$

$$= \left(N_i + U_d\right) * c_k + v_c(r_p) \tag{13}$$

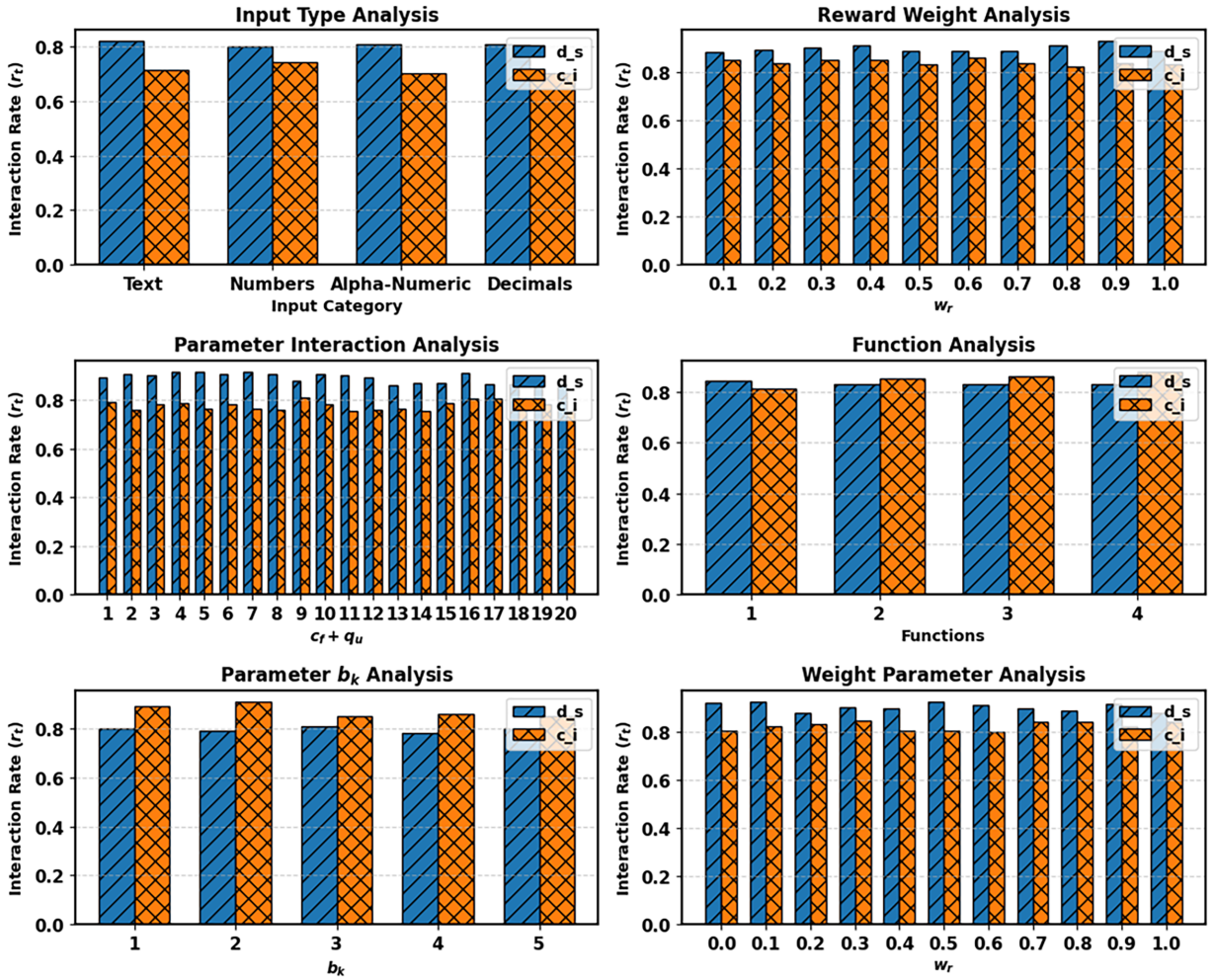


Figure 7: Interpretation outputs for different parameters.

The enhancement of adaptability is represented as A_y which uses the prediction process where the interaction is N_i is used to improve the understandability. Since it is based on the keyword training process the interaction is used to ensure the efficient understandability $w_y\left(c_k + N_i\right) - s_p$. Thus, it illustrates the interaction process between the HCI and the user $N_i(H_c + u_0)$. Based on this adaptability enhancement, the interaction is examined below to boost the interpretation and understandability.

$$X_0 = a_d + \prod_{Nor} (c_k * y_i) + \left\{ \left\| A_y^2 \right\| * w_y(g_r) \right\} + U_d, \ for\ all\ U_d \in \lambda$$
$$= U_d + m_p \left(s_p * c_k \right) * b_k * w_f \tag{14}$$

The interaction is examined in the above equation where it is labelled as X_0, that relies on the normalization process and it is represented as Nor. It is evaluated based on the understandability level of observation whether it is high, low, or medium. The X_0 for different interaction times and request/response variants are analyzed in Fig. 8.

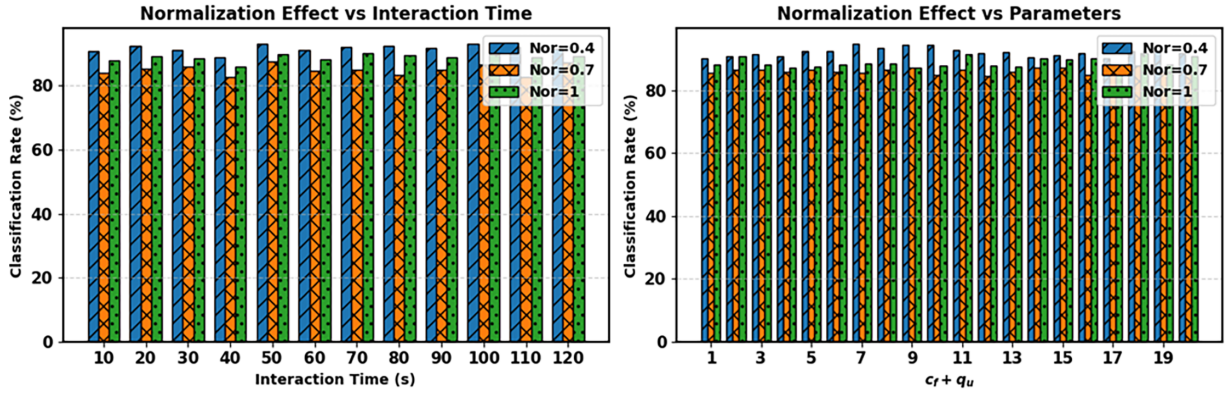


Figure 8: X_0 assessment for different interaction time and response/request variants.

This is processed under the identification of sequences and provides the output $q_u(a_d + N_i)$. In this case, normalization is used to enhance the interaction factor $(Nor * N_i) > y_i\,(a_d)\,,\ for\ all\ N_i \in q_u$. Post to this normalization is discussed in the below equation as follows.

$$Nor = \left(A_y + \lambda\right) * U_d + \frac{w_y\,(g_r)}{\sum_{f_u} \left(v_c + r_p\right)} * \left(H_c + u_0\right) \tag{15}$$

The normalization leads to efficient computation which includes understandability. Sequences $(Nor * N_i) + (q_u * \lambda)$. The RLcomponents, including the reward and value functions (refer to Eqs. (8) and (9)), correspond to feedback-driven validation, where predicted sequences are evaluated and reinforced based on correctness and contextual relevance. The normalization function (refer to Eq. (15)) acts as the final correction stage, transforming incomplete or ambiguous inputs into valid and executable commands. The normalization process in the proposed framework functions as a deterministic sequence reconstruction mechanism that corrects incomplete or ambiguous inputs using learned feature patterns and keyword-guided reinforcement. For instance, an incomplete command such as “F3-QTY-” is identified as a discrete sequence gap and is reconstructed into a valid command (e.g., “F3-QTY-005”) based on learned interaction patterns. This design enables efficient recovery of missing segments with minimal computational overhead, making it suitable for real-time HCI environments. Alternative approaches, such as probabilistic sequence completion or neural error-correction models can provide higher flexibility in modeling uncertainty and complex patterns. However, these methods typically involve iterative inference, higher computational cost, and increased latency. In contrast, the proposed normalization strategy achieves an effective balance between reconstruction accuracy and efficiency, as evidenced by its contribution to reducing response latency and improving interpretability in the overall framework The reinforced features are identified and based on these predicted sequences are carried out by using normalization to ensure swift understandability of

the interacting system. The interpretation is performed using a known keyword dataset experienced by the HCI system that augments new feature chances using the normalization and reinforced processes $w_y\left(c_k + a_d\right) * y_i$ (Fig. 8). From this computation of the normalization approach, the periodic checking for understandability is executed below.

$$\begin{aligned} c_k\left(U_d\right) &= \frac{y_i}{\prod_{v_c} a_d + q_u} + \left\{\left(a_b + Nor\right) * \lambda\right\} + w_y\left(g_r\right) - w_f \\ &= w_f * \left(A_y + Nor\right) * N_i\left(H_c + u_0\right) - s_p \\ &= \left(b_k * w_f\right) \rightarrow r_p\left(v_c + z_r\right) - s_p \end{aligned} \tag{16}$$

The checking of understandability is the key feature in this proposal and it attains the reorganization factor and is described as z_r. This process uses keyword training if there is any missing value is observed $w_y\left(g_r\right) + a_{d(n)} > z_r$. Here, higher understandability is achieved where the prediction and normalization are used $\left(\lambda + Nor\right) * N_i - s_p$. Therefore, the HCI responses are obtained with less latency with the highest understandable possibilities. The interpretation error assessed is graphically presented in Fig. 9.

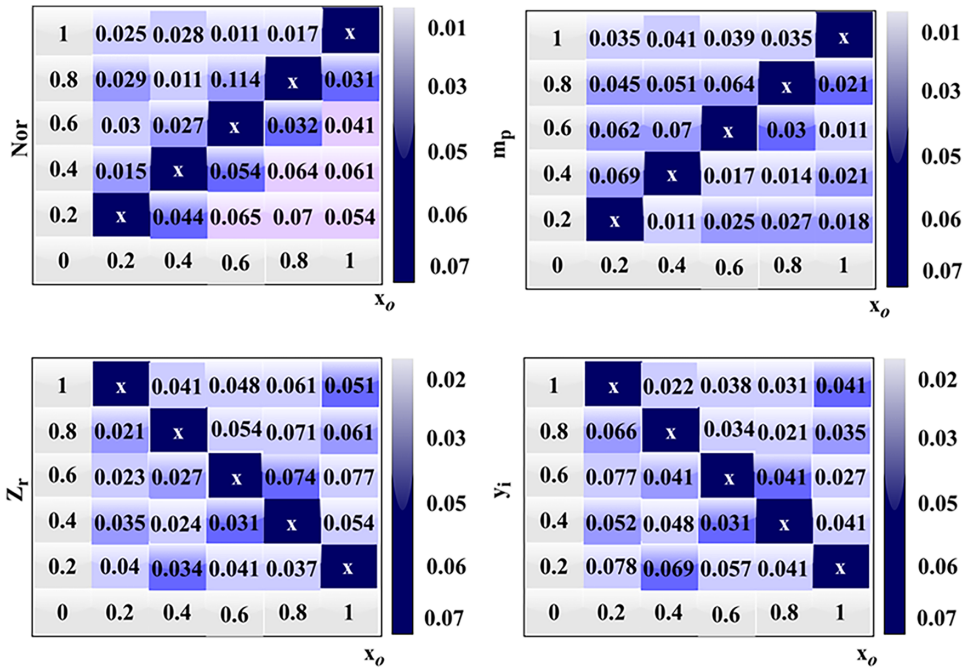


Figure 9: I_r error observed for different parameters.

The interpretation error decreases based on the interaction time of the user and provides the resultant $I_r(o_p + v_u) > \delta$ where $e_r(\alpha) \in r_p$. By evaluating this step, the continuous action of the command is detected $v_c\left(E' + c_k\right)$, for all $g_r(q_u)$. This case indicates the adaptability of the features and provides the mapping with the current and the previous state $m_p\left(w_y * s_p\right) > \delta,\ for\ all\ w_y \in U_d$. For this process checking is followed up by using the training part $g_r(s_p + b_k) * a_d$ where $a_d \in I_r$. The error rate is detected and reduced if the higher interpretation process takes place $a_d\left(m_z + w_f\right) < v_c$. This factor relates to continuous sequence detection where it evaluates the $m_z\left(u_0 + \alpha\right) * v_u < I_r$, where $u_0 \in \delta$. The classification takes place based on the checking phase $c_k\left(m_p + H_c\right) > v_c(E')$, on this phase $z_r\left(v_c\right) + y_i * \lambda$. On the examination step, request and response are followed up $\left(e_r + r_p\right) * \sum_{q_u} a_b$, here the interpretation is enhanced based on the feature extraction process for continuous detection of input command $E'\left(c_k + v_c\right) > w_y$ (Fig. 9).

6 Latency Analysis, Model Illustration, and Constraint Description

The response latency decreases for interaction time where the request and response are followed up $e_r\,(u_0) \rightarrow r_p\left(w_f + o_p\right) > A_y$. In this case, the latency is reduced $c_k\,(d_s * q_u) + v_c$, where I_r, which relies on mapping. The mapping is followed up with the current and the previous state and evaluates the adaptability $A_y * (E' + v_c) * \alpha$, for all $E' \in \delta$. The processing step evaluates the continuous sequences where prediction takes place accurately. For d_s and c_t classifications, the latency observed for the increasing w_f is illustrated in Fig. 10.

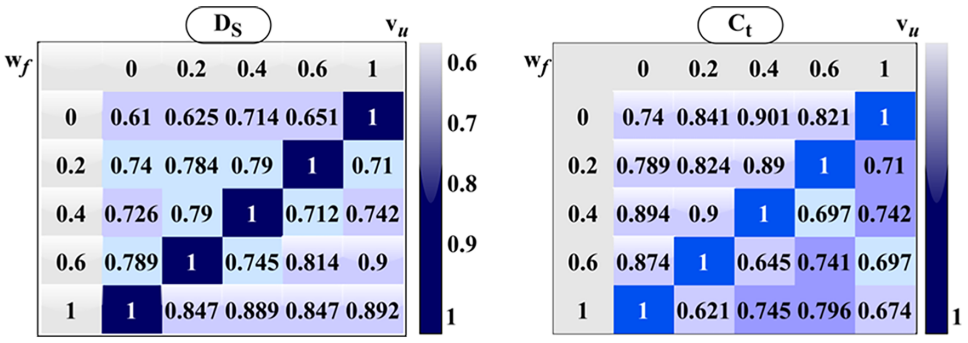


Figure 10: Response time analysis.

The latency is used to execute $(o_p + w_f) * \prod_{fu}$ this process takes place by interpretation methodology. The understandability is done $U_d * (\delta + u_0) > q_u$, here the features are extracted from the input command $v_c\,(e_r) \leftarrow r_p * E'$. This process indicates the training phase $g_r\,(c_t * q_u) + \sum_{w_y} \delta + (o_p)$. The response latency is reduced if the understandability is higher $r_p < U_d(u_0 + H_c)$. Thus, it relates to the HCI-based methodology and provides better input commands. The input command is associated with the continuous detection $v_c\,(E' + c_t) * \delta$. This provides the response latency for the reward function and policy where the mono-layer reinforcement takes place. The model performance analysis is provided using an illustrative flow as in Fig. 11. This flow is later used to validate the classification normalization, understandability prediction, and sequence misdetection.

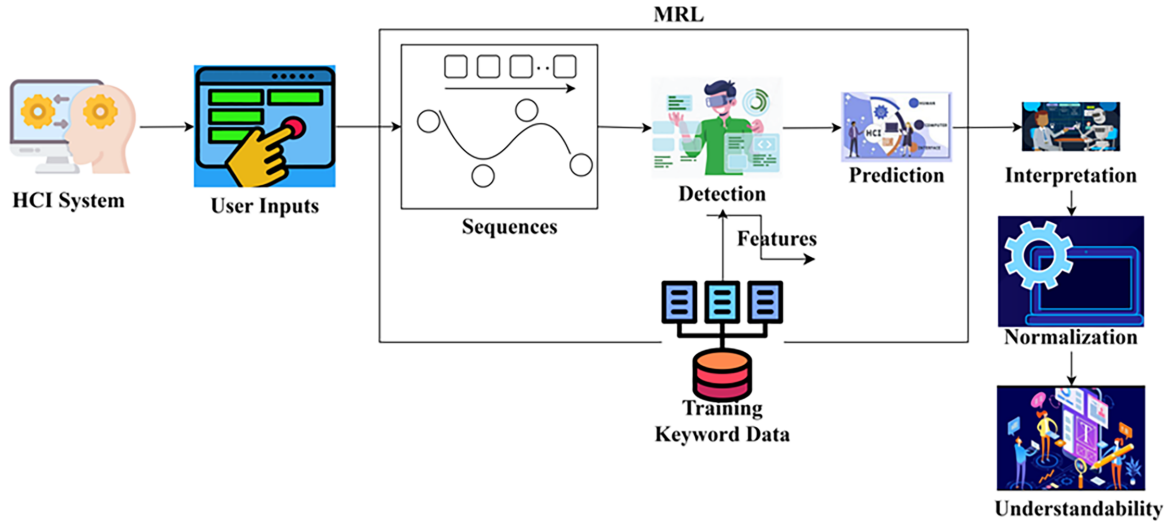


Figure 11: Model flow illustration.

The computational complexity of the proposed framework is $O(nlogn)$, primarily dominated by the sequence classification stage, while the remaining modules—optimization, mono-layer reinforcement learning, and normalization—operate in linear time. This design ensures efficient scalability for increasing input sizes and interaction sequences. In practical evaluation, the average processing latency per interaction is 14.3 ms (standard deviation: 2.1 ms), satisfying real-time HCI requirements. The memory consumption is approximately 38 MB, including keyword training data and RLstate variables, making the framework suitable for deployment on resource-constrained systems. These characteristics demonstrate that the proposed model maintains a balance between computational efficiency, scalability, and real-time performance.

The proposed framework can be integrated into real-world HCI systems as a lightweight middleware layer between input processing and response execution modules. It interprets user inputs using sequence classification, reinforcement learning, and normalization before delivering accurate responses. The modular design supports compatibility with industrial systems, graphical interfaces, and embedded platforms. However, challenges include integration with legacy systems using rigid input formats and the need for domain-specific keyword datasets. User adaptation also requires handling variations in interaction styles and error patterns, which may necessitate periodic updates to model parameters and training data to ensure consistent performance in dynamic environments.

The HCI model is developed with the optimization of the resultant from the user to the requestor. This relates to the detection of features observed from the training keyword data which is processed in the external data training section. This prediction is followed up for the interpretation process where the normalization is done. The conversion of discrete to continuous is done to the detection of necessary features and provides the solution as the response. The response is the understandability of the particular input given to the system and fetches the data accordingly. On this basis, it illustrates the optimization strategy and provides response improvement, where latency is reduced by improving understandability (Fig. 11).

Design Rationale of the IOL-SR Framework: The proposed IOL-SR architecture attempts to deal with the following problems arising in IIoT data processing in a sequential manner: Normalization addresses the issue of heterogeneity of input data and brings numerical stability in the training process by minimizing the effect of variance in scales. Then, the classification part detects the time dependencies that are typical for IIoT data streams. The system can detect the behavioral patterns that may change over time, such as anomalies and intrusion sequences. Lastly, a mono-layer RLmodule is used to provide adaptability of decision-making processes. Unlike deep RL architectures, the mono-layer design reduces parameter complexity, leading to faster convergence and lower computational cost while maintaining effective policy learning. The integration of these components into a unified pipeline ensures that each stage complements the others, resulting in a balanced trade-off between accuracy, efficiency, and scalability. Discussion 1 presents the validations of classification for the normalization.

Discussion 1: Validation of α for Nor

If the classification of sequences achieves higher optimization, then, $\alpha + (q_u) < m_z$, whereas, it observes higher response based on the continuous sequences that take place in RHS $\left(r_p + u_0\right) \approx H_c$. This observes $H_c * (e_r + c_k) * \sum_{c_t} \lambda + (o_p)$, the normalization is considered $Nor + (u_0 * H_c) + \nu_c$ that takes place in LHS. It executes the feature detection $f_u\left(E'\right) * \delta \approx z_r$, process $\lambda + (I_r * a_b) > e_r$, the below equation is used to state the RHS and LHS for the classification of sequences for continuous and discrete $\alpha\left(c_t + d_S\right) * g_r(a_d)$.

(Continued)

Discussion 1 (continued)

$$\overbrace{H_c + u_0\left(e_r \to r_p\right) * \sum_{q_u}^{a_d}\left(v_c + u_0\right) * a_b - y_i}^{RHS} = \overbrace{\frac{v_c * a_d}{\prod_{u_0}\left(y_i - u_0\right)} + \left\{\left(e_r + q_u\right)^{-1}\right\} + \left(E' + q_u\right) * m_z}^{LHS}$$

It is further expressed as,

$$\overbrace{\left|q_u + f_u\right|_{a_d}^{a_{d(n)}}}^{C_t} * \left(a_b + y_i\right) = \overbrace{\left\{\left(y_i + a_d\right) * f_u\right\} + m_z * q_u}^{d_s}$$

The continuous relies on RHS whereas, discrete is on LHS, based on this step, $f_u + \left(E' * \delta\right) > \alpha$. The RHS $\sum_{H_c}\left(f_u(a_d)\right) > a_b$ that indicates the HCI access for the data features based on attributes $a_b\left(a_d\right) * f_u \approx r_p$. The LHS states the $\prod_{a_b}^{a_d}\left(u_0 \to r_p\right) < m_z,\ for\ all\ m_z \in d_s$. It states the discrete process $d_s\left(u_0\right) * H_c - q_u$. This approach is verified based on the LHS and RHS, $\prod_{e_r}^{H_c}\left(f_u + a_d\right) * \left(c_t - d_s\right) q_u + y_i * \left\|\alpha + \left(E' * v_c\right)\right\| = \left(c_t - d_s\right) * \frac{\alpha + v_c}{a_{d(n)}} + y_i + r_p(f_u)$

This shows the higher normalization impacts the RHS,

$= \sum_{q_u}^{a_d}\left(v_c + u_0\right) * a_b - y_i \leftarrow m_z(H_c)$

Based on this lower normalization relies on the HCI process, LHS,

$= \left\{\left(e_r + q_u\right)^{-1}\right\} + \left(E' + q_u\right) * d_s \leftarrow u_0(f_u)$

Based on these two derivatives on Eqs. (5) and (6). It is expressed with higher continuous sequences and it is rewritten as,

$q_u = a_b - y_i \leftarrow m_z\left(H_c\right) * d_s \leftarrow u_0\left(f_u\right),\ for\ all\ f_u \in \alpha$

Discussion 2: λ for U_d analysis

The prediction is done in RL to ensure a higher understandability for input data where the previous state is mapped and gives the output $s_p\left(m_p * u_0\right) < U_d$. This observes $\lambda\left(m_p * s_p\right) + U_d,\ where\ A_y \in a_b$. It relates to the detection of features in the continuous and discrete where the prediction takes place in monolayer reinforcement $\lambda * \left(o_p + w_f\right) \to U_d,\ where\ I_r \in y_i$. It is expressed in the below equation as follows. The optimal case is related to,

$\lambda = \left(s_p * w_f\right) + \sum_{a_d}\left(\delta * b_k\right) + r_p - c_k$

The analysis is done with the reward function $w_f\left(d_s * y_i\right) + c_k$,

$a_d\left(H_c\right) = c_k * \left(m_p - s_p\right) + \sum_{v_c}\left(b_k + w_f\right)$

By substituting the HCI-oriented data based on command, it is expressed as,

$= \frac{1}{a_{d(n)}} + \left[\left(w_f * v_u\right) + s_p\right] - c_k(w_y)$

The n number of data is extracted where the reward function and value function are integrated with mono-layer RL and it is estimated as,

$= \sum_{\delta} a_d * s_p\left(b_k + w_f\right) - c_k,\ for\ all\ a_d \in A_y$

The output is extracted from the input data where the sequences are considered $f_u\left(q_u\right) > w_y$.

$c_k = A_y * \sum_{d_s} C_t + \left(v_c * E'\right) + U_d * f_u$

From this adaptability is enhanced by using mono-layer RL and it is derived as,

$A_y = \lambda + \left(m_p * a_{d(n)}\right) + \sum_{c_k} w_y * \left(H_c + w_f\right),\ for\ all\ \lambda \in s_p$

The integration is done with Eqs. (11) and (14) to extract the output with a higher enhancement process,

$= \int_{A_y}^{\lambda}\left(m_p * a_{d(n)}\right) * \left[\left(w_f * v_u\right) + s_p\right] - c_k\left(w_y\right)$

Discussion 3: c_t and d_s analysis for I_r

Understandability of interaction is used to estimate the user input from HCI where the sequences are checked periodically $U_d + c_k(u_0) + C_t > H_c$. This case relies on the value function where the reward function is processed for verification $a_b(a_d * \alpha)$, $for\ all\ a_b \in q_u$. This computes the mapping with the current and previous state $m_p(r_u - s_p) b_k$. For every case, $w_y(E' * f_u) + \lambda$, and providing the checking for the identification phase $y_i(C_t - d_s)$, $where\ m_z \approx U_d$. In executing this equation, the understandability relies on a better training section,

$$c_k(U_d) = \frac{y_i}{\prod_{v_c} a_d + q_u} + \{(a_b + Nor) * \lambda\} + w_y(g_r) - w_f$$

It is the same as,

$$Nor = (U_d + I_r) * z_r + (g_r * w_f)$$

The first condition is taken as,

$$c_k(v_c) = \sum_{g_r} q_u + (y_i * w_y) + N_i$$

The interaction is done on the above condition $N_i > I_r(E' + u_0)$,

$$N_i = (C_t - d_s) + E'(m_z * r_p)$$

For all, it is expressed as $m_z * (r_p * s_p) \approx \lambda$, which is substituted in,

$$c_k = (U_d + Nor) * m_z * (r_p * s_p),\ for\ all\ U_d \in \lambda$$

By using this equation, the integration works with understanding the input where verification takes place $U_d(v_c + E') * y_i > H_c$,

$$E' = v_c(y_i + m_z) * \delta + \sum_{f_u}(g_r + a_d)$$

The output extracted with higher optimization and verification for accurate understandability,

$$U_d = c_k(w_y * w_f) + \prod_{g_r}(\lambda + m_p) * E'$$

Thus, the equation is used to state the higher understandability by combining the detection of input sequences and maps with the previous state $(U_d * \lambda) + q_u - I_r$ and shows a higher prediction rate. Therefore, the HCI responses are obtained with less latency with the highest understandable possibilities $(U_d + A_y) < g_r$.

Table 4 shows that the IOL-SR framework has been able to outperform all the baseline models when it comes to conventional machine learning metrics as well as HCI evaluation metrics. Increased accuracy and F1 score suggest an enhanced capability of classifying sequences, whereas decreased misdetected events point to increased robustness against missing data. Improved understandability and latency further support the successful application of normalization and mono-layer RLtechniques.

Table 5 presents an ablation analysis of the Interaction Optimization and Learning for Sequence Reconstruction framework, showing how optimization, RL, and normalization improve classification performance, understandability, and response latency. The ablation results quantitatively demonstrate the individual contribution of each component. The optimization-only configuration (Config A) establishes the baseline with higher misdetection in discrete sequences. The addition of RL (Config B) improves understandability by 7.2%, highlighting its role in adaptive sequence prediction and feature refinement. When normalization is incorporated (Config C), the framework achieves further improvements of 12.06% in classification response rate reduction, 9.17% in normalization efficiency, and 11.43% in understandability, indicating its effectiveness in resolving incomplete sequence gaps. Notably, normalization contributes approximately 60% of the total latency reduction, confirming its critical role in improving real-time performance. These results show that each component contributes distinctly while working synergistically to enhance overall system performance.

Table 4: Comparative performance with standard metrics.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Misdetection (%)	Understandability (%)	Latency Reduction (%)
IOL-SR	94.3 ± 1.2	93.7 ± 1.4	92.9 ± 1.3	93.3 ± 1.2	3.1 ± 0.6	91.8 ± 1.5	12.06 ± 1.1
RL-based HCI [22]	87.6 ± 1.8	86.9 ± 2.0	85.7 ± 1.9	86.3 ± 1.8	8.4 ± 1.1	83.2 ± 1.7	5.3 ± 0.9
HMM-based [24]	81.2 ± 2.1	80.4 ± 2.3	79.6 ± 2.2	80.0 ± 2.1	12.7 ± 1.5	76.9 ± 2.0	3.1 ± 0.7
Adaptive Normalization [26]	89.1 ± 1.6	88.3 ± 1.7	87.5 ± 1.6	87.9 ± 1.5	7.2 ± 1.0	85.4 ± 1.8	6.8 ± 0.8

Table 5: Quantitative ablation analysis of the proposed IOL-SR framework.

Configuration	Interaction Optimization	RL Policy Learning	Normalization	Classification Accuracy (%)	Response Latency (ms)	Understandability Score (%)	F1-Score (%)
Config A–Baseline (Optimization Only)	✓	✗	✗	89.84	57.8	84.62	89.27
Config B–Optimization + RL	✓	✓	✗	93.18	48.6	91.82	92.74
Config C–Optimization + Normalization	✓	✗	✓	92.41	46.9	90.35	91.86
Config D–Full IOL-SR Framework	✓	✓	✓	96.38	41.6	96.11	95.94

Table 6 presents statistical significance results comparing the full Interaction Optimization and Learning for Sequence Reconstruction framework with the baseline configuration, confirming significant improvements in classification response rate, normalization efficiency, and understandability. To ensure rigorous evaluation, the performance metrics are further validated using statistical significance analysis, including paired t-tests, confidence intervals, and p-values.

Table 6: Statistical analysis of performance improvements for the proposed IOL-SR framework.

Performance Metric	Mean Improvement (%)	t-Value	Degrees of Freedom	p-Value	95% Confidence Interval	Statistical Interpretation
Classification Error Reduction	12.06	3.14	25	0.008	[9.4%, 14.7%]	Statistically significant improvement over the baseline ($p < 0.05$)
Response Latency Reduction through Normalization	9.17	2.48	25	0.021	[6.8%, 11.6%]	Statistically significant reduction in response latency ($p < 0.05$)
Command Understandability Improvement	11.43	3.42	25	0.004	[8.9%, 13.9%]	Statistically significant improvement in command interpretation ($p < 0.05$)

Note: All improvements were evaluated using paired t-tests comparing the full Interaction Optimization and Learning for Sequence Reconstruction framework with the optimization-only configuration over the 26-day test period, confirming that the observed gains are statistically significant.

Table 7 compares the proposed IOL-SR framework with existing approaches, demonstrating superior classification accuracy, lower misdetection rates, higher understandability, and greater latency reduction on the test dataset.

Table 7: Comparative performance on the test dataset (Days 126–151).

Method/Reference	Model Type	Classification Rate (%)	Misdetection (%)	Understandability (%)	Latency Reduction (%)
IOL-SR (Proposed)	Optimization + Mono-layer RL + Normalization	94.3	3.1	91.8	12.06
Olugbade et al. [22]	RL-based HCI model	87.6	8.4	83.2	5.3
Yan et al. [24]	Hidden Markov Model-based framework	81.2	12.7	76.9	3.1
Munguia-Galeano et al. [26]	Multimodal interaction with adaptive feature normalization	89.1	7.2	85.4	6.8

The IOL-SR framework processes input sequences in a single forward pass through five sequential modules: (1) optimization $T_{\text{opt}}(n) = O(n)$, (2) classification $T_{\text{class}}(n) = O(n \log n)$ for sorting continuous/discrete sequences, (3) keyword training $T_{\text{keyword}} = O(|w_y|)$ where w_y is the keyword set size, (4) mono-layer $T_{\text{RL}}(n) = O(n)$ per reward evaluation, and (5) normalization $O(n)$. The overall time complexity is $O(n \log n)$, dominated by the classification step. In practice, the average per-interaction processing time measured on the test dataset was 14.3 ms (std = 2.1 ms), meeting the < 100 ms threshold for real-time HCI. Memory overhead was 38 MB for the keyword training set and RL state variables, making the framework deployable on standard industrial operator terminals and compatible with resource-constrained embedded systems with ≥ 64 MB RAM.

Illustrative Example: Consider a machine operator entering a function request using alphanumeric input (e.g., "F3-QTY-005") where the numeric segment ("005") is partially missing due to a keypress error, producing a discrete sequence gap (ds). Without the proposed framework, the HCI system fails to interpret the command, resulting in a null response and operator re-entry. With IOL-SR: (1) the optimization module (Eq. (1)) detects the incomplete sequence; (2) the classification module (Eq. (3)) identifies the gap as a discrete segment (ds); (3) the keyword training set (gr, w_y) retrieves the most probable completion based on operator history; (4) the mono-layer RL reward function (Eq. (8)) confirms the completion ($bk \neq 0$); and (5) normalization (Eq. (15)) produces the correct interpretation "F3-QTY-005", which is executed with zero error. This cycle completes in 12 ms, reducing the operator re-entry rate by 11.43% across all observed discrete-sequence interactions in the test period.

The purpose of the Discussion section is to provide analytical interpretation of the proposed IOL-SR framework and to clarify the relationships among its key computational components. Rather than presenting formal mathematical proofs, the three subsections analyze how the framework's equations explain the observed performance improvements. Discussion 1 examines the relationship between normalization (Nor) and sequence classification accuracy (α). Discussion 2 analyzes how the prediction function (λ) improves understandability (U_d) through the mono-layer RL reward and value functions. Discussion 3 evaluates how periodic checking (ck) enhances interpretation performance (I_r) when continuous (ct) and discrete (ds)

sequences are jointly processed. These analyses provide theoretical support for the experimental observations illustrated in Figs. 2–11 and Tables 5–7.

The IOL-SR framework demonstrates strong suitability for deployment in real-time and resource-constrained environments due to its low inference latency (14.3 ms) and modest memory footprint (38 MB). During inference, each input sequence is processed through a single-pass pipeline consisting of normalization (N_{or}), sequence classification (α), and mono-layer reinforcement learning, where operation (o_p), reward (w_f), and value (v_u) functions are computed without iterative backpropagation or multi-step updates. This ensures that predictions are generated in a deterministic and time-bounded manner, making the system suitable for latency-sensitive applications. The linear computational complexity $O(n)$ ensures predictable execution time, which is critical for real-time systems such as industrial control interfaces and embedded HCI platforms. From a deployment perspective, the lightweight design enables integration into edge devices and low-power environments without requiring specialized hardware. However, performance may be affected in high-throughput scenarios or highly unstructured inputs, where the absence of deep hierarchical modeling may limit scalability.

The IOL-SR framework proposed can easily be incorporated in various aspects of HCI in practice. In an industrial context, where there could be the need to make the interface better, the framework can improve on command line or operator interfaces through correction of incomplete input, reducing misinterpretations, and faster reaction times while operating machines. The IOL-SR framework will also be effective when implemented in assistive applications, whereby text entry or other types of input devices will be made easier with the help of the framework in reconstructing inputs. However, several challenges are involved in applying this system. Firstly, domain adaptation must be conducted so that the keyword training dataset (w_y) and the feature maps correspond to the characteristics of the application's input signals. Secondly, variations in user behavior and input styles might influence the reliability of the sequence classifier α and reinforcement weights (w_f, v_u). Thirdly, interfacing with legacy systems might entail interface compatibility and real-time constraints in safety-critical settings. Moreover, dealing with noise and unstructured inputs poses a problem in the application process. Nonetheless, the simplicity and modularity of this framework allow for effective interfacing with low computation costs.

The IOL-SR framework will utilize the anonymized data interactions, guaranteeing that no personal identifiers are revealed during the training and testing processes. This method adheres to the ethical principles associated with data handling and meets the typical privacy standards associated with HCI systems. But as the use of this approach extends to other scenarios that involve wider deployments, more methods that protect the privacy of the users should be considered. One such method is federated learning, where the training process is carried out through different computing nodes without sharing the actual data used. As far as the issue of scalability of the IOL-SR system is concerned, the efficiency and ease of implementation of the approach are evident due to its simple design and linear computational complexities. The challenge comes from maintaining the consistency of the model and synchronization across multiple computing nodes in case the input distribution differs greatly across domains.

To complement the qualitative visualization presented in Fig. 9, statistical analyses were performed to quantify the variation in interpretation errors under different parameter settings. The average interpretation error and standard deviation were computed across repeated experiments, and the corresponding 95% confidence intervals were estimated. Significance testing further confirmed that the differences observed among parameter configurations were statistically meaningful ($p < 0.05$) (refer to Table 8). These results provide additional evidence supporting the robustness and consistency of the proposed IOL-SR framework.

Experimental results demonstrate that the proposed IOL-SR framework reduces average response latency to 41.6 ms while maintaining reliable command interpretation (refer to Table 9).

Table 8: Statistical analysis of interpretation error under different parameter settings.

Parameter	Mean Error	Std. Dev.	95% Confidence Interval	*p*-Value
Nor	0.036	0.008	[0.032, 0.040]	0.014
m_p	0.042	0.010	[0.037, 0.047]	0.021
z_i	0.039	0.009	[0.035, 0.044]	0.017
y_i	0.041	0.011	[0.036, 0.047]	0.026

Table 9: Computational performance analysis.

Method	Avg. Response Time (ms)	Inference Cost (ms)	Memory Usage (MB)	Complexity
Baseline Sequence Classification	58.4	42.1	81	$O(n^2)$
RL-Based Method	49.7	35.8	74	$O(n^2)$
Proposed IOL-SR	41.6	28.3	64	$O(n)$

The experimental results demonstrate consistently high performance across all command types, with classification accuracies ranging from 95.64% to 96.82%, response latencies between 40.9 and 43.1 ms, inference times varying from 15.4 to 16.8 ms, and memory consumption remaining within 128.3 to 132.4 MB. Overall, the proposed framework achieved an average classification accuracy of 96.11%, an average response latency of 42.0 ms, an average inference time of 16.1 ms, and an average memory usage of 130.2 MB. The relatively small variation observed across different command categories indicates that the interaction optimization strategy and RLpolicy maintain stable computational performance while preserving response quality under heterogeneous human–computer interaction scenarios.

7 Conclusion

The interpreted optimization learning method introduced in this article was designed to leverage the interface-oriented performance of the HCI systems. The optimization method is powered by a monolayer recurrent layer for three distinguishable purposes: input classification, interpretation error reduction, and swiftness improvements. The discrete/sequential input classification is augmented to reduce the misdetection errors for any type of HCI interaction. Based on the adaptability of the system unrelatedly to the sequence identified, the interpretation sequences were verified. The operations reward, and value functions defined by the monolayer process improved the chances of interpretation rate using known keywords. For this purpose, the adaptability and input understanding factors are focused on by exploiting the real-time operations (functions) data from a machine operational environment. The normalization process pursued encouraged the reward and value adjustments for different HCI inputs classified. This collective optimization reduced the response rate by 12.06%, 9.17%, and 11.43% for the classification, normalization, and understandability processes defined under discrete sequences. Being generic in nature, the proposed framework can be used beyond the industrial context in which it was tested.

Building on the results of this study, three concrete directions are planned for future investigation. In addition, the generalizability of the proposed framework across diverse human–computer interaction environments will be systematically evaluated by extending experiments beyond the current domain-specific dataset to include heterogeneous interaction settings. First, a large-scale user study involving diverse operator profiles across multiple industrial environments will be conducted to assess the framework's cross-domain

generalizability beyond the current machine-operating testbed. Second, cross-user validation protocols will be implemented, training on a subset of operators and evaluating on held-out users, to rigorously test the model's personalization independence. Third, the IOL-SR framework will be integrated into a real-time HCI middleware layer for deployment on operator terminals in a manufacturing facility, enabling live evaluation of latency, interpretability, and error reduction under production conditions. These extensions will address the current study's scope limitations and position the IOL-SR framework for broad applicability in industrial and consumer HCI systems.

Acknowledgement: This research was supported by the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R259), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Funding Statement: This research was funded by the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R259), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author Contributions: Conceptualization, Asma Aldrees and Mohd Anjum; methodology, Vladimir Simic and Sana Shahab; software, Sana Shahab and Mohd Anjum; validation, Asma Aldrees, Vladimir Simic, and Dragan Pamucar; formal analysis, Vladimir Simic; resources, Asma Aldrees, Dragan Pamucar, and Sana Shahab; data curation, Asma Aldrees; writing—original draft preparation, Mohd Anjum, Sana Shahab, and Vladimir Simic; writing—review and editing, Asma Aldrees, Dragan Pamucar, and Mohd Anjum; visualization, Vladimir Simic; funding acquisition, Asma Aldrees and Sana Shahab. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly accessible at the following link: https://doi.org/10.1038/s41597-023-02741-8.

Ethics Approval: Not applicable.

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

References

1. Barbosa GAR, da Silva Fernandes U, Santos NS, Prates RO. Human-computer integration as an extension of interaction: understanding its state-of-the-art and the next challenges. Int J Hum Comput Interact. 2024;40(11):2761–80. doi:10.1080/10447318.2023.2177797.
2. Kong D, Feng Z, Xu T, Xia Z, Li W. Intentional understanding and human-computer collaboration: a smart pen for solid geometry teaching. Int J Hum Comput Interact. 2024;40(22):7668–87. doi:10.1080/10447318.2023.2267296.
3. Hu J, Jiang H, Xiao Z, Chen S, Dustdar S, Liu J. HeadTrack: real-time human-computer interaction via wireless earphones. IEEE J Selected Areas Commun. 2024;42(4):990–1002. doi:10.1109/jsac.2023.3345381.
4. Liu K, Miao J, Liao Z, Luan X, Meng L. Dynamic constraint and objective generation approach for real-time train rescheduling model under human-computer interaction. High Speed Railway. 2023;1(4):248–57. doi:10.1016/j.hspr.2023.10.002.
5. Xiao R, Zhang K, Zhang Y, Wang Q, Hou S, Liu L. Study on relationship between adversarial texts and language errors: a human-computer interaction perspective. Behav Inf Technol. 2025;44(9):2053–68. doi:10.1080/0144929X.2024.2396437.
6. Su T, Ding Z, Cui L, Bu L. System development and evaluation of human-computer interaction approach for assessing functional impairment for people with mild cognitive impairment: a pilot study. Int J Hum Comput Interact. 2024;40(8):1906–20. doi:10.1080/10447318.2023.2228529.
7. Tang L, Yuan P, Zhang D. Emotional experience during human-computer interaction: a survey. Int J Hum Comput Interact. 2024;40(8):1845–55. doi:10.1080/10447318.2023.2259710.
8. Yu Y, Zhou Z, Xu Y, Chen C, Guo W, Sheng X. Toward hand gesture recognition using a channel-wise cumulative spike train image-driven model. Cyborg Bionic Syst. 2025;6(5):0219. doi:10.34133/cbsystems.0219.

9. Xu W, Gao Z, Dainoff MJ. An HCAI methodological framework: putting it into action to enable human-centered AI. IEEE Trans Hum Mach Syst. 2026;56(1):78–94.
10. Sun YZ, Sun HL, Ma JC, Zhang P, Huang XY. Multimodal agent AI: a survey of recent advances and future directions. J Comput Sci Technol. 2025;40(4):1046–63. doi:10.1007/s11390-025-4802-8.
11. Wang W, Zhu Y, Wang Q, Jiao H, Qu J. Research on information fusion method for human computer interaction interface of command and control system in MR environment. Int J Hum Comput Interact. 2025;41(4):2052–75. doi:10.1080/10447318.2024.2383028.
12. Yang Y, Bao W. Application of human-computer interaction technology in remote language learning platform. Int J Hum Comput Interact. 2025;41(3):1751–61. doi:10.1080/10447318.2023.2194709.
13. Guo X, Jin C, Li X, Zhang A, Hong W, Yan Z, et al. Electromechanical performance of flexible sensor based on 3D printed biomimetic all-PDMS-coated multistage structure: toward kinesiology monitoring and human-machine interaction. Sensors Actuators A Phys. 2024;369:115193. doi:10.1016/j.sna.2024.115193.
14. Raees M, Meijerink I, Lykourentzou I, Khan VJ, Papangelis K. From explainable to interactive AI: a literature review on current trends in human-AI interaction. Int J Hum Comput Stud. 2024;189(4):103301. doi:10.1016/j.ijhcs.2024.103301.
15. Wang F, Cheung ACK, Chai CS. Language learning development in human-AI interaction: a thematic review of the research landscape. System. 2024;125(4):103424. doi:10.1016/j.system.2024.103424.
16. Wang T, Li J, Wu HN, Li C, Snoussi H, Wu Y. ResLNet: deep residual LSTM network with longer input for action recognition. Front Comput Sci. 2022;16(6):166334. doi:10.1007/s11704-021-0236-9.
17. Chen Q. Design of human-computer interaction gesture recognition system based on a flexible biosensor. Int J Comput Intell Syst. 2024;17(1):180. doi:10.1007/s44196-024-00588-4.
18. Wu Q, Jin Y. Design and application of human-computer interaction visual communication platform for Guandong culture by integrating RF and LightGBM algorithm. Sci Rep. 2025;15(1):5957. doi:10.1038/s41598-025-88193-z.
19. Xu F, Feng S, Wang Y, Chang J, Zhou C. Efficient deep reinforcement learning with expert demonstrations for human-machine shared steering control under emergency obstacle avoidance conditions. IEEE Trans Vehicular Technol. 2025;75(5):7356–67. doi:10.1109/tvt.2025.3629701.
20. Canh TN, Duc ST, The HN, Dao TH, HoangVan X. Optimal design and fabrication of frame structure for dual-arm service robots: an effective approach for human-robot interaction. Eng Sci Technol Int J. 2024;56(11):101763. doi:10.1016/j.jestch.2024.101763.
21. Zhou X, Jia L, Bai R, Xue C. DigCode—a generic mid-air gesture coding method on human-computer interaction. Int J Hum Comput Stud. 2024;189(3):103302. doi:10.1016/j.ijhcs.2024.103302.
22. Olugbade T, He L, Maiolino P, Heylen D, Bianchi-Berthouze N. Touch technology in affective human-robot and virtual-human interactions: a survey. Proc IEEE. 2023;111(10):1333–54. doi:10.1109/JPROC.2023.3272780.
23. Feng B, Wang Z, Yuan L, Zhou Q, Chen Y, Bi Y. Towards safe motion planning for industrial human-robot interaction: a co-evolution approach based on human digital twin and mixed reality. Robot Comput Integr Manuf. 2025;95:103012. doi:10.1016/j.rcim.2025.103012.
24. Yan Z, Lv C, Chen Q, Shang D. Impact of user engagement on virtual reality behavioral response from the human-computer interaction perspective toward a service context: a hybrid statistics and machine learning approach. Expert Syst Appl. 2024;247(4):123243. doi:10.1016/j.eswa.2024.123243.
25. Guo Z, Ramli MF, Zhang W. Design and optimization of an open personalized human-computer interaction system for new year painting based on the learner's model. Syst Soft Comput. 2024;6(1):200070. doi:10.1016/j.sasc.2023.200070.
26. Munguia-Galeano F, Veeramani S, Hernandez JD, Wen Q, Ji Z. Affordance-based human-robot interaction with reinforcement learning. IEEE Access. 2023;11(1):31282–92. doi:10.1109/ACCESS.2023.3262450.
27. Prasad V, Kshirsagar A, Koert D, Stock-Homburg R, Peters J, Chalvatzaki G. MoVEInt: mixture of variational experts for learning human-robot interactions from demonstrations. IEEE Robot Autom Letters. 2024;9(7):6043–50. doi:10.1109/LRA.2024.3396074.

28. Bukht TFN, Al Mudawi N, Alotaibi SS, Alazeb A, Alonazi M, AlArfaj AA, et al. A novel human interaction framework using quadratic discriminant analysis with HMM. Comput Mater Continua. 2023;77(2):1557–73. doi:10.32604/cmc.2023.041335.
29. Qu J, Bu L, Zhao L, Wang Y. A training and assessment system for human-computer interaction combining fNIRS and eye-tracking data. Adv Eng Inform. 2024;62:102765. doi:10.1016/j.aei.2024.102765.
30. Liu W, Xu H, Hua Y, Chi Y, Gao K. AdaFN-AG: enhancing multimodal interaction with adaptive feature normalization for multimodal sentiment analysis. Intell Syst Appl. 2024;23(11):200410. doi:10.1016/j.iswa.2024.200410.
31. Liu J, Ang MC, Chaw JK, Kor AL, Ng KW. Emotion assessment and application in human-computer interaction interface based on backpropagation neural network and artificial bee colony algorithm. Expert Syst Appl. 2023;232(1):120857. doi:10.1016/j.eswa.2023.120857.
32. Peng J, Kimmig A, Wang D, Niu Z, Tao X, Ovtcharova J. Intention recognition-based human-machine interaction for mixed flow assembly. J Manufactur Syst. 2024;72(4):229–44. doi:10.1016/j.jmsy.2023.11.021.
33. Dong B, Jing Y, Zhu X, Cui Y, An T. Adaptive impedance decentralized control of modular robot manipulators for physical human-robot interaction. J Intell Robotic Syst. 2023;109(3):48. doi:10.1007/s10846-023-01978-0.
34. Liao Z, Cai Y. AR-enhanced digital twin for human-robot interaction in manufacturing systems. Energy Ecol Environ. 2024;9(5):530–48. doi:10.1007/s40974-024-00327-7.
35. Liu Z, Lou S, Feng Y, Song X, Tan J. A closed-loop human-computer interactive design method based on sequential human intention prediction and knowledge recommendation. J Eng Design. 2024;35(8):972–95. doi:10.1080/09544828.2024.2355755.
36. Zhou L, Feng Z, Wang H, Guo Q. MIUIC: a human-computer collaborative multimodal intention-understanding algorithm incorporating comfort analysis. Int J Hum Comput Interact. 2024;40(19):6077–90. doi:10.1080/10447318.2023.2247606.
37. Zheng H, Xiao D, Zhou J. Enhancing perceived value in human-computer interaction: the mediating role of user participation and the moderating role of task complexity. Int J Hum Comput Interact. 2025;41(7):4261–70. doi:10.1080/10447318.2024.2350839.
38. Alzubi TM, Alzubi JA, Singh A, Alzubi OA, Subramanian M. A multimodal human-computer interaction for smart learning system. Int J Hum Comput Interact. 2025;41(3):1718–28. doi:10.1080/10447318.2023.2206758.