Improved Real Time User Interaction in Extended Reality Systems Using the Deployment of Adaptive Intelligent Technologies

[1,2]Hayder M A Ghanimi, [3]Sathvik Bagam, [4]Shaishav Shah, [5]Vedaraj M, [6]Manjunath TC and [7]Manoranjan Kumar Sinha

[1]Department of Information Technology, College of Science, University of Warith Al-Anbiyaa, Karbala, 56003, Iraq.
[2]Department of Computer Science, College of Computer Science and Information Technology, University of Kerbala, Karbala, 56003, Iraq.
[3]Software Development Team Lead at Paycom, Masters in Computer Science, Oklahoma Christian University, Edmond, Oklahoma, 73013, United States.
[4]Department of Computer Engineering, Government Engineering College, Dahod, Gujarat, India.
[5]Department of Computer Science and Engineering, R.M.D. Engineering College, RSM Nagar, Kavaraipettai, Thiruvallur, Tamil Nadu, India.
[6]Department of Computer Science and Engineering, Dean Research (R&D), Rajarajeswari College of Engineering, Bengaluru, Karnataka, India.
[7]Department of Electronics Engineering, Medi-Caps University, Indore, Madhya Pradesh, India.
[1,2]hayder.alghanami@uowa.edu.iq, [3]sathvikbagam7@gmail.com, [4]shaishav999@gmail.com, [5]vedaraj84@gmail.com, [6]tcmanju@iitbombay.org, [7]sinhamanoranjan18@gmail.com

Correspondence should be addressed to Sathvik Bagam: sathvikbagam7@gmail.com

Article Info
Journal of Machine and Computing (https://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202505074
Received 14 March 2024; Revised from 29 June 2024; Accepted 26 February 2025.
Available online 05 April 2025.

Abstract - The Human-Computer Interaction (HCI) field has seen rapid growth in various industries due to the introduction of Extended Reality (XR) environments. These environments require improved interface methods, real-time processing, low latency, and integrated User Experience (UE) servicing. This work aims to improve user interactions in real-time XR environments and introduces a new Hierarchical Adaptive System (HAS) to address these challenges. This study presents a Real-Time Adaptation Model (RTAM) for XR interfaces, which combines adaptive optimized performance, Deep Reinforcement Learning (DRL), and Fuzzy Logic (FL). The system addresses unpredictability, Dynamic Resource Allocation (DRA), and parallel processing pipelines. HAS did better than the best methods by 46.3% in terms of Faster Learning Integration (FLI), 63.2% in terms of Lower Error Rates (LER), and 37.4% in terms of Reduced Task Completion Times (RTCT) in a study with 60 users in different interactive settings. Despite maintaining low adaptation latency, the system achieves a score of 0.86 for resource utilization efficiency. The study also identified improvements in system responsiveness and overall satisfaction. The results support that HAS effectively solves RTAM issues in XR settings, laying the basis for next-generation immersive apps with more responsive and user-centered communication models.

Keywords – Extended Reality, User-Centric Interaction, Hierarchical Adaptive System, Human-Computer Interaction, Latency, Deep Learning.

I. INTRODUCTION

The domain of Human-Computer Interaction (HCI) is currently experiencing a significant increase in the application of Extended Reality (XR) environments, which integrate multiple types of Augmented Reality (AR) and Virtual Reality (VR) [1-3]. Conventional and Adaptive Interface Systems (AIS) are in substantial demand due to the development of AIS in sectors such as healthcare, educational institutions, design for manufacturing, and professional training [4-5]. Suboptimal User Experiences (UE), reduced task performance, and the risk of user dissatisfaction result from the reasons that most modern XR uses static interface models that cannot adapt to changing user knowledge, environmental factors, and the complexity of tasks [6,7].

Although AIS has recently rendered significant progress in several domains, numerous challenges must be solved when applying it to XR interactions [8–10]. The numerous demands of real-time multimodal input processing, ultra-low latency

maintenance of submerging, and adaptation to highly dynamic 3D interface spaces are the primary reasons for these challenges. While traditional methods can provide some adaptability, they are typically unequipped to manage XR settings' complexity and computational speed demands. Several types of research conducted [11-12] indicate that currently developed systems have essential problems, specifically when it comes to adapting to different user behaviors, maintaining performance consistently under many environments, and mitigating present interruptions over adaptation.

Intelligent systems that can manage multiple input sources simultaneously, learn from active user interactions and adapt in real-time while maintaining the system's reliability and user-friendliness are important tasks to solve. Previous research has used methods like rule-based systems, Fuzzy Logic controllers, and Reinforcement Learning to tackle XR interactions, but these methods often focus on discrete adaptation elements, leading to incorrect adaptation or resource overuse due to inadequate management [14-16].

Traditional HCI faces challenges due to the complexity of XR interactions, which require accurate monitoring and decoding of user gestures, posture, gaze direction, and speech. Environmental factors like lighting, space limits, and interruptions make AIS more challenging [17-19]. Additionally, user needs and abilities may revolutionize significantly, demanding adaptable, adapted methods [20-21].

Implementing a Hierarchical Adaptive System (HAS) that involves Deep Reinforcement Learning (DRL), FL, and adaptive optimization into a predictive model is the main focus of the present research. This is achieved to address the problems that have been highlighted.

The recommended method proposes significant support to the domain.

- This scalable model uses an innovative parallel processing pipeline optimized for low-latency response, effectively analyzing and adapting to real-time multimodal XR interactions.
- A new multilevel learning algorithm combines short-term optimization with long-term local adaptations, resulting in rapid responses and continuous enhancement.
- Consistent performance over changing computational loads is provided by an intelligent resource management system that optimizes and allocates resources periodically.
- A complete evaluation paradigm that integrates objective performance measures and qualitative UE factors to measure the value of adaptation in XR environments.
- A reliable uncertainty behavior method that maintains the complete system reliability despite the conditions or user behavior.

Significant user studies demonstrate that the proposed model substantially improves upon current methods, with task completion times that are 37.4% faster and error rates that are 63.2% lower. The improvements stand out in complex interaction circumstances with several input modalities, and the environment evolves constantly. When compared to state-of-the-art solutions, the proposed model presented faster learning integration and higher Resource Utilisation Efficiency (RUE) [22].

Here is how each section of the paper is organized: Details of the system design, algorithms, and execution problems will be provided in Section 2 of the proposed methodology. The team of novel testing model is one of the evaluation metrics and experimental design details provided in Section 3. Section 4 presents a complete analysis and results of performance dimensions; Section 5 ends with recommendations for further research and prospective extensions of this work.

II. METHODOLOGY

System Architecture

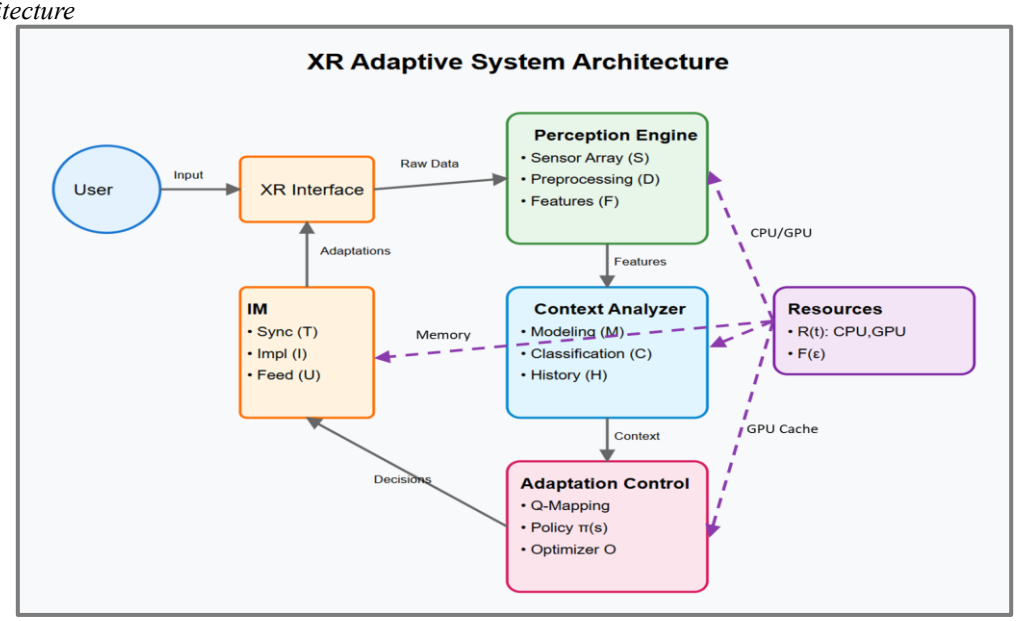

Fig 1. Proposed XR-AIS Model.

The present research describes a complete model for developing Artificial Intelligence (AI) systems for XR environments that address the essential problems of improving user interaction in real-time. **Figs 1 and 2** implement a Closed-Loop Adaptive Model (CLAM) that processes user inputs, analyses context and delivers real-time adaptations while maintaining optimal system performance. The model **Fig 1** starts a primary workflow where user interactions flow into a chain of focused processing modules by the XR interface. This high-level organization demonstrates the system's core data route, with compacted arrows representing direct data flow and dashed lines indicating RUE. The system employs four principal components that work in concert to deliver adaptive responses: the Perception Engine (PE), Context Analyzer (CA), Adaptation Controller (AC), and Interaction Manager (IM).

Fig 2 demonstrates each component's internal layout and connectivity based on this core. The PE initiates the processing pipeline with three key elements: a Sensor Array (S) for capturing multimodal inputs, a Data Preprocessing module (D) for signal conditioning, and a Feature Extraction unit (F) that distills relevant interaction features. This HAS ensured the efficient processing of raw user inputs into meaningful feature sets.

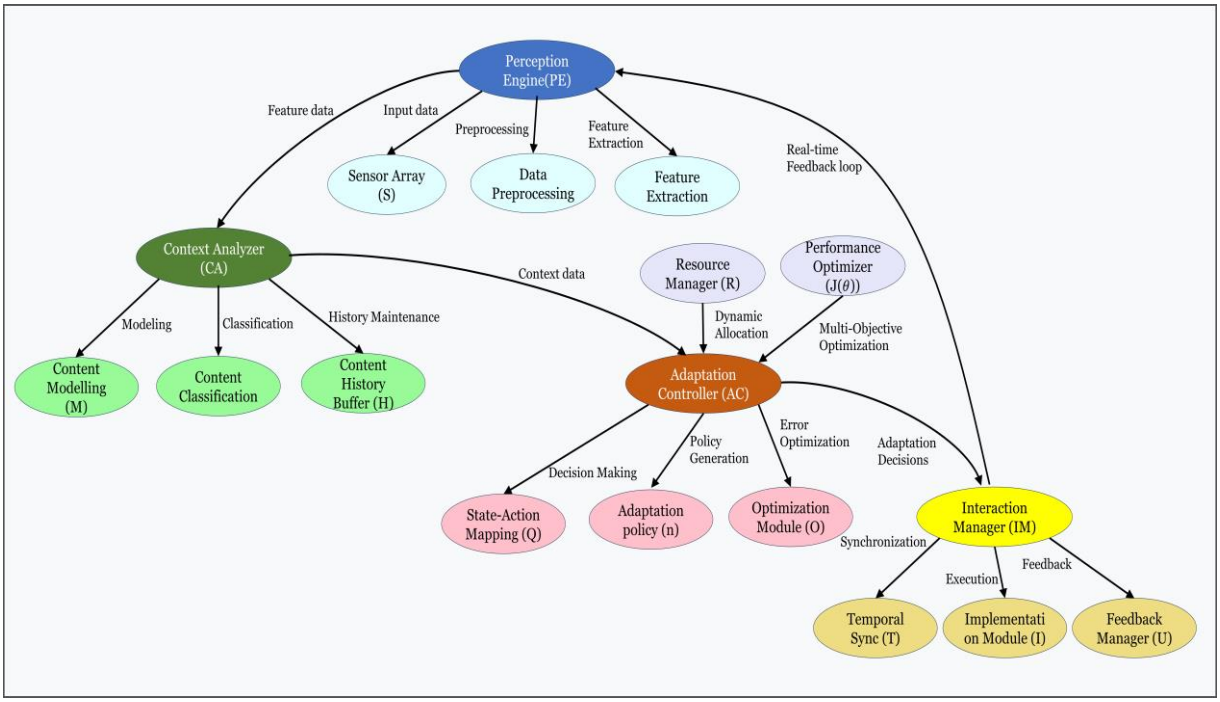

Fig 2. Component-Level Model and Processing Pipeline.

The Details of Each Component are Presented Below

The PE, defined as $PE = \{S, D, F\}$, acts as the primary input processing unit. Here, S represents the sensor array responsible for collecting multimodal input data, including user gestures $G(t)$, gaze vectors $V(t)$, and spatial positioning $P(t)$ at time t. The data preprocessing pipeline D performs signal filtering and normalization to ensure data quality, while the Feature extraction module F employs a sliding window approach with window size w and stride s to extract meaningful features from the input stream. The CA uses a HAS, represented as Eq. (1).

$$CA = \{M, C, H\}, \tag{1}$$

to derive contextual data. The context modeling function M maps the raw feature space X to a higher-level context space Y, while the context classification module C employs a hybrid method combining rule-based heuristics R and ML as θ, trained on historical interaction data. To maintain continuity, a temporal context log buffer H stores h frames for constant evaluation.

The AC, the core Decision-Making System (DMS), is represented as Eq. (2)

$$AC = \{Q, A, O\} \tag{2}$$

The state-action mapping $Q(s, a)$ determines optimal AC, governed by an adaptation policy $\pi(s)$ using a DRL with learning rate α and discount factor γ. To minimize the adaptation error function $E(\theta)$, the optimization module O adjusts system parameters through gradient descent with motion β. The Interaction Manager (IM) coordinates the execution of adaptation decisions and is defined as Eq. (3)

$$IM = \{T, I, U\} \tag{3}$$

It ensures temporal synchronization of adaptations within a maximum latency constraint λ. The implementation module I translates high-level decisions into concrete modifications within the XR environment, while the user feedback manager U collects feedback and computes performance metrics at a sampling rate r.
The interaction among these components follows a defined protocol Eq. (4).

$$P = \{\delta, \tau, \omega\} \tag{4}$$

Where,
$\delta \rightarrow$ the data flow pathways
$\tau \rightarrow$ synchronization mechanisms
$\omega \rightarrow$ communication protocols.
A global state vector Eq. (5).

$$\Psi(t) = [\psi_1(t), \psi_2(t), \dots, \psi_n(t)] \tag{5}$$

It tracks the status of all subsystems at any given time. The procedure uses Dynamic Resource Allocation (DRA) to optimize RUE, Eq. (6).

$$R(t) = \{CPU(t), GPU(t), MEM(t)\} \tag{6}$$

This section determines it by the complexity of the link and the present load. The consistent operation under numerous scenarios is provided by a fault-tolerance mechanism F(ε) with an error threshold 'ε'.
Eq. (7) is a multi-objective function that optimizes performance.

$$J(\theta) = \alpha_1 L_{\text{adapt}}(\theta) + \alpha_2 L_{\text{latency}}(\theta) + \alpha_3 L_{\text{resource}}(\theta) \tag{7}$$

The design allows for component-wise updates and upgrades by an adaptable model, with the relative significance of objectives determined by weights $\{\alpha_1, \alpha_2, \alpha_3\}$. It also features a setup database C = $\{c_1, c_2, \dots, c_k\}$ that stores all adapted factors, enabling proactive system adaptation to deployment requirements and hardware features. The integrated design enhances real-time user interaction in XR environments, ensuring its success, robustness, and adaptability.

Data Flow Design
The XR-AIS data flow design, consisting of PE, CA, AC, and Interaction Manager (IM), efficiently transfers data among its four primary components, ensuring synchronization, resilience, real-time processing, effective communication, and adaptive DMS. The data collection process starts at the PE, where the sensor array transmits the multimodal input data S= $\{G(t), V(t), P(t)\}$. D is assigned to preprocessing the raw sensor data, encompassing features like spatial positioning, gaze vectors, and user gestures. This module normalizes and filters the inputs to reduce noise and secure interoperability for the following steps. Processed signals are then passed to the feature extraction module F, where meaningful features are identified using a sliding window mechanism. Extracted features X_f serve as the primary inputs for context analysis. The extracted features X_f are transmitted to the CA, where the context modeling function M maps the features into the context space Y. The hybrid context classification module C, leveraging rule-based heuristics R and ML as θ, evaluates the user's state and the XR environment conditions. The temporal context history H stores recent context states $\{Y_{t-1}, Y_{t-2}, \dots, Y_{t-h}\}$, enabling a continuous evaluation loop. The resulting contextual information Y_c is then sent to the AC.

Upon receiving Y_c, the AC uses the state-action mapping $Q(s, a)$ to identify the optimal AC. The current system state s and context-derived action a are evaluated by the adaptation policy $\pi(s)$, implemented through a reinforcement learning model. Real-time adjustments to adaptation parameters are performed by the optimization module O, ensuring that the system minimizes adaptation errors $E(\theta)$ while meeting latency constraints. The final adaptation decisions A_d are then passed to the Interaction Manager (IM). The Interaction Manager (IM) receives A_d and translates these high-level adaptation decisions into concrete XR environment modifications. The temporal synchronization module T ensures that the adaptations are aligned with the user's current interaction timeline, adhering to the latency constraint λ. The implementation module I executes the modifications, such as adjusting visual elements, altering haptic feedback, or triggering auditory cues. Simultaneously, the feedback manager U collects user feedback U_f and system performance metrics, sampled at rate r, for evaluation.

User Feedback U_f and performance metrics are returned to the CA and AC as part of a feedback loop. The CA uses this data to refine its classification models θ and context history H, while the AC adjusts its policy $\pi(s)$ and optimization parameters O. The feedback loop ensures continuous learning and system improvement, adapting to changing user behavior and environmental conditions.

The interaction between components follows a defined communication protocol $P = \{\delta, \tau, \omega\}$:

- δ : Specifies the data flow pathways, ensuring efficient transmission between subsystems.
- τ : Manages synchronization to align data processing across the closed-loop system.
- ω : Standardizes communication protocols to maintain compatibility and reduce latency.

The data flow design causes the use of DRA as $R(t)= \{CPU(t), GPU(t), MEM(t)\}$ to ensure that data processing remains continuous regardless of the changing computational loads. The reliable operation within an error threshold 'ε' has been preserved by the fault-tolerance mechanism $F(\varepsilon)$, which monitors and resolves data flow problems. A real-time image of the condition of all subsystems can be attained by the framework's global state vector $\Psi(t)=[\psi_1(t),\psi_2(t),\ldots,\psi_n(t)]$. Optimal performance is assured with the help of this state vector, enabling the entire system to monitor and adaptive improvements.

Integration Model

A unified framework can be found by the integrating model of the recommended XR-AIS, which ensures subsystems interface effectively while maintaining performance, reliability, and adaptability at their optimal levels. This model, illustrated by **Figs 1 and 2**, combines components, manages resources dynamically, synchronizes, and deals with errors securely; it can be used in real-time XR environments because of its multilayer design.

The fundamental part of the model is the component connectivity layer, which measures the interface among the four primary subsystems: the PE, CA, AC, and IM. A feature-transformed data interface, IF(X_f), allows the PE to transmit the extracted features to the CA in a reliable and practical approach, processing the sensory inputs throughout the process. The CA generates contextual data, Y_c, and sends it to the AC via the interface IC(Y_c), where the AC is specified. Through the IA(A_d) input interface, the AC later transmits recommendations for adaptation to the IM. To complete the feedback loop, the IM transmits the user's feedback data (U_f) back to the PE by the interface IU(U_f). Modular interaction is made probable by providing these interfaces, meaning subsystems can be updated periodically without impacting the overall system's functionality.

A focused resource management layer monitors the integration of resources and is responsible for the real-time allocation and monitoring of memory, graphics processing unit (GPU), and CPU resources. This real-time demand and load mapping, $RM(t)$, allocates computing resources to subsystems. The Performance Optimiser $J(Q)$ manages the process by monitoring performance metrics like adaptation accuracy, latency, and RUE. It adjusts resource allocation dynamically while maintaining system functionality within predefined constraints. The model also features a global state management mechanism for synchronization and coherence.

This is represented by the state vector Eq. (8)

$$\Psi(t) = [\psi_{PE}(t), \psi_{CA}(t), \psi_{AC}(t), \psi_{IM}(t)], \tag{8}$$

which is updated at 120 *Hz* to ensure optimal synchronization.

An internal priority queue divides system functions into critical, standard, and background forms, enabling better integration of events and allowing the system to prioritize dynamic tasks while delaying less important ones, ensuring responsiveness in high-demand environments.

The model's robust performance is backed by an advanced fault management system, which involves local elements addressing problems individually before sending them to system-level processors. Recovery mechanisms, including retry protocols, fallback methods, and system reset, ensure system security and minimize downtime in unpredictable situations.

The model's adaptability is achieved through a module-based framework, allowing new elements, adaptation rules, and techniques to be implemented. All modules are tested for compatibility and version control, ensuring compatibility without impacting current activities. This design allows for iterative updates and experimentation. Security measures, such as encrypting communications and operating within restricted trust boundaries, are implemented using a multi-layered approach. The system filters potential risks with automated integrity checks and real-time security event monitoring.

Implementation of AIS

Learning Algorithms

The XR system's AIS implementation uses a multi-layered approach, including optimization, uncertainty management, and core AC, to quickly tune system parameters, adapt in real-time, and endure changing conditions, focusing on user interaction, environmental unpredictability, and resource management.

Core Adaptation Algorithms: DRL

A DRL, including Proximal Policy Optimisation (PPO) or Deep Q-Networks (DQN), is a key tool for adaptive behavior.

The rule sets π(s) for state-action mappings $Q\ (s,\ a)$ are optimized dynamically by these algorithms.

where:

- $s \in S$→Represents the system's state,
- $\{G(t), V(t), P(t)\}$ →encompassing user inputs
- $\psi_{PE}(t), \psi_{CA}(t), \psi_{AC}(t), \psi_{IM}(t)$→ subsystem statuses

- $a \in A$→Adaptive actions may include modifying video and audio.
- $Q(s,a)$→ Computes the cumulative reward

$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where r_t is the immediate reward and $\gamma \in [0,1]$ is the discount factor.
The PPO refines the policy $\pi(s)$ by maximizing the clipped surrogate objective Eq. (9)

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\ (r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{9}$$

Where,

$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$→probability ratio, $\hat{A}_t$→reward function, ϵ→ clipping parameter to prevent massive updates. Alternatively, the DQN approximates the Q-value function $Q_\theta(s,a)$ using a DNN, updated iteratively with the Bellman Eq. (10)

$$Q_\theta(s,a) = r_t + \gamma \max_{a'} Q_\theta(s',a') \tag{10}$$

where s'→is the next state.

This setup ensures dynamic adaptation to user behaviors and environmental shifts.

Uncertainty Management Algorithms: FL and Bayesian Networks

To handle uncertainties in user behavior and environmental conditions, the system incorporates FL and Bayesian Networks.

FL

FL represents imprecise inputs and outputs through membership functions $\mu: X \to [0,1]$, where $x \in X$ (*e.g.,* gesture speed, gaze angle). Fuzzy rules, such as:

IF x_1 is A_1 AND x_2 is A_2, THEN y is B, infer appropriate adaptive actions y. These rules ensure smooth transitions and robust handling of overlapping or vague user inputs.

Bayesian Networks

Bayesian Networks model probabilistic relationships among system variables. For a set of variables $V = \{v_1, v_2, \ldots, v_n\}$, the joint probability is expressed as:

$$P(V) = \prod_{i=1}^{n} P\left(v_i \mid \text{Parents}\ (v_i)\right) \tag{11}$$

Optimization Algorithms: Evolutionary Algorithms (GA and PSO)

Optimization of parameters, such as network weights θ, adaptation policies $\pi(s)$, and resource allocations $R(t)$, is achieved through Evolutionary Algorithms (EA), including Genetic Algorithms (GA) and Particle Swarm Optimization (PSO).

GA

GA evolves a population of candidate solutions $\{x_1, x_2, \ldots, x_n\}$ through selection, crossover, and mutation.
The fitness function is Eq. (12)

$$\text{Fit}\ (x) = \alpha_1 L_{\text{adapt}}\ (x) + \alpha_2 L_{\text{latency}}\ (x) + \alpha_3 L_{\text{resource}}\ (x), \tag{12}$$

where L_{adapt} , $L_{\text{latency,}}$ and L_{resource} measure adaptation accuracy, system latency, and resource efficiency, respectively. The weights $\alpha_1, \alpha_2, \alpha_3$ balance these objectives.

Particle Swarm Optimization (PSO)

PSO optimizes a set of particles $\{p_1, p_2, \ldots, p_n\}$ in a search space. Each particle updates its velocity and position as follows: Eq. (13) and Eq. (14)

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (p_{\text{best}} - p_i) + c_2 r_2 (g_{\text{best}} - p_i) \tag{13}$$

$$p_i(t+1) = p_i(t) + v_i(t+1) \tag{14}$$

where v_i is the particle velocity, p_{best} and g_{best} are the personal and global best positions, respectively, and ω, c_1, c_2 control the convergence behavior. This ensures efficient fine-tuning of system parameters.

Algorithm: Adaptive Intelligence Implementation in XR Systems

Input:

- User inputs $S = \{G(t), V(t), P(t)\}$ (Gestures, Gaze, Position).
- Environmental context data $C(t)$.
- Initial system parameters θ_0, R_0 (network weights, resource allocation).

Output:

- Adaptation decisions A_d.
- Updated parameters θ, R.

1. **Initialize System:**
   Set initial states $\Psi(0) = [\psi_{PE}(0), \psi_{CA}(0), \psi_{AC}(0), \psi_{IM}(0)]$.
   Load pre-trained policy $\pi(s)$, Q-values $Q(s, a)$, and system configurations $C = \{c_1, c_2, \dots, c_k\}$.
2. **Input Data Processing:**
   Step 2.1: Collect user inputs S and environmental context $C(t)$.
   Step 2.2: Perform signal filtering and normalization on S and $C(t)$.
   Step 2.3: Extract features X_f using a sliding window approach (window size w, stride s).
3. **Context Analysis:**
   Step 3.1: Map extracted features X_f to context space Y using $M: X \to Y$.
   Step 3.2: Use FL to compute membership values $\mu: X \to [0,1]$ and infer possible actions A_{fuzzy} based on fuzzy rules.
   Step 3.3: Apply Bayesian Networks to estimate the likelihood of contextual states $P(Y \mid X)$ and refine possible actions A_{bayesian} .
4. **Core Adaptation:**
   Step 4.1: Retrieve the current state s from the global state vector $\Psi(t)$.
   Step 4.2: Select action a based on the policy $\pi(s)$:
   $a = \arg\max_{a'} Q(s, a')$ (DQN) or $a \sim \pi_\theta(a \mid s)$ (PPO)
   Step 4.3: Execute action a and observe the reward r_t and next state s'.
5. **Update Core Algorithm:**
   Step 5.1: Update Q-values (for DQN): $Q(s, a) \leftarrow r_t + \gamma \max_{a'} Q(s', a')$.
   Step 5.2: Update policy (for PPO):
   Maximize Clipped Surrogate Objective: $L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\ (r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$.
6. **Optimization of Parameters:**
   Step 6.1: Optimize resource allocation $R(t)$ using PSO:
   Update Velocity and Position: $v_i(t+1) = \omega v_i(t) + c_1 r_1 (p_{\text{best}} - p_i) + c_2 r_2 (g_{\text{best}} - p_i)$,
   $p_i(t+1) = p_i(t) + v_i(t+1)$
   Step 6.2: Evolve adaptation parameters θ using GA:
   - Select parents based on fitness fit $(\theta) = \alpha_1 L_{\text{adapt}}(\theta) + \alpha_2 L_{\text{latency}}(\theta) + \alpha_3 L_{\text{resource}}(\theta)$.
   - Perform crossover and mutation to generate new offspring.
7. **Decision Execution and Feedback Collection:**
   Step 7.1: Send adaptation decisions A_d to the Interaction Manager (IM).
   Step 7.2: Collect user feedback U_f and performance metrics (e.g., latency, accuracy).
8. **Feedback Integration and System Update:**
   Step 8.1: Update the CA (fuzzy rules R and Bayesian probabilities $P(Y \mid X)$) using feedback U_f.
   Step 8.2: Adjust resource mappings $R(t)$ based on observed system performance.
   Step 8.3: Refine adaptation policy $\pi(s)$ and Q-values using updated reward signals.
9. **Repeat:**
   Go to Step 2 for continuous adaptation in real-time.

Real-Time Processing Methods and DMS

The proposed XR-AIS integrates advanced real-time processing methods and hierarchical DMS to deliver contextually appropriate adaptations while meeting strict temporal constraints. Although the system is in a continual state of change, these methods ensure reliability and optimal adapting.

Real-Time Processing Pipeline

Real-time processing begins with parallel data streams that handle multimodal inputs through a multi-threaded architecture. Input data is processed using a sliding window approach $W(t, \Delta t)$, where Δt (window size) is dynamically adjusted based on input complexity and current system load. Critical operations are prioritized through a preemptive scheduling mechanism $S(p, \tau)$, where p represents priority levels, and τ denotes timing constraints. This mechanism guarantees

bounded processing times such that $T_{\text{process}} \leq T_{\max}$ with $T_{\max}$ determined by the frame rate and the system's real-time requirements.

Parallel data sets that analyze multimodal inputs employing a multi-threaded framework are the basis elements of real-time processing. A sliding window method ($W(t, \Delta t)$ is employed for analyzing the input data, with the size of the window (Δ_t) being dynamically adapted based on the input complex and the current system load. The scheduling preemptive mechanism *S(p,τ)*→essential tasks), with 'p'(priority levels) and τ (time limitations). This system provides that processing times are limited so that $T_{Process} \leq T_{Max}$, where T_{Max}→ the system's real-time demands and the frame rate.

The Data Preprocessing Stage Employs an Adaptive Filtering Scheme

$$F(x,t) = \alpha(t)F(x,t-1) + (1-\alpha(t))x(t) \tag{15}$$

where $F(x,t)$→filtered signal at time t; $\alpha(t)$→ signal's noise features. Feature extraction is carried out on these filtered data streams through a HAS as $H = \{h_1, h_2, \ldots, h_n\}$. Each level h_i extracts features of increasing abstraction, ensuring temporal constraints are met with a total processing budget $\sum_i T(h_i) \leq T_{\text{budget}}$.

Hierarchical DMS

To control proactive and reactive adaptations, the DMS uses a model defined as the Hierarchical Markov Decision Process (HMDP) system.

Strategic Decisions

At the top level, high-level method decisions D_s are made based on a policy function:

$$\pi_s(s) = \text{Arg\_Max}_a Q(s,a) \tag{16}$$

Where,

s→system state, incorporating the immediate context and historical patterns.

These decisions are computationally intensive but provide long-term adaptation methods.

Tactical Decisions

At a lower level, tactical decisions D_t are derived from a faster, reactive policy:

$$\pi_t(s_{\text{local}}) = \text{Arg\_Max}_a Q(s_{\text{local}}, a) \tag{17}$$

where,

s_{local} →localized state information for rapid response.

Reduced approach timeframes focus actions towards maximization of instant. The levels are incorporated in the end-point adaption decision $D_{final}(t)$ using dynamic weights $\{\omega_s(t),\ \omega_t(t)\}$ which are modified depending on decision probability and timing limits:

$$D_{\text{final}}(t) = \omega_s(t)D_s(t) + \omega_t(t)D_t(t) \tag{18}$$

Validation and Execution

Decisions involving real-time adaptation are rapidly examined for practicality before being put into action. Availability of resources, timely delivery, and operational security are all validated by the feasibility check by Eq. (19).

$$V(d) = \{\ \text{resource\_check, timing\_verify, constraint\_validate}\ \}. \tag{19}$$

A predictive execution model $E(d,t)$ is employed by the model to prevent recurrent or uncertain behaviors by predicting the result of adaptation 'd' at the time 't'. Estimated computing methods have been used to perform efficient but reliable analyses when full validation requires too much time.

Decision Execution Follows a III-Stage Commit Protocol

- *Preparation (P):* Resource allocation and system setup for the decision.
- *Validation (V):* Confirmation of the feasibility and consistency of the decision.
- *Execution (E):* Implementation of the adaptation in the XR environment.

The Total Execution Time Constraint Bounds These Stages

$$T_P + T_V + T_E \leq T_{\text{frame}} \tag{20}$$

Execution priorities are managed through a dynamic queue $Q(t) = \{q_1, q_2, \ldots, q_k\}$, where each q_i represents a decision-execution pair with associated timing and resource requirements. For concurrent adaptations, a conflict resolution mechanism $C(D_1, D_2)$ ensures decision compatibility. If decisions $\{D_1, D_2\}$ are compatible, they are merged into a composite decision:

An iterative list $Q(t)=\{q_1, q_2,\ldots,q_k\}$ is employed to control the order of execution, where each 'q_i' → a decision-execution pair that has equal time and resource requirements. A system for addressing conflicts, denoted as $C(D_1, D_2)$, provides decision consistency for parallel adaptations. A unified decision is made by integrating decisions $\{D_1, D_2\}$ if they are acceptable:

$$R(D_1, D_2) \rightarrow D_{\text{Hybrid}} \tag{21}$$

If conflicts exist, the system applies a prioritization function:

$$P(D_1, D_2) \rightarrow D_{\text{Priority}} \tag{22}$$

to prioritize adaptations and select the one that will maintain the system functioning without interruption.

III. EXPERIMENTAL DESIGN

Test Environment Setup

This study conducted our XR-AIS test results in a monitored lab environment that matched real-world use cases. The hardware configuration consisted of a high-end XR headset (Meta Quest Pro) with integrated eye-tracking capabilities (90 Hz Refresh Rate, 96° Horizontal FOV) paired with precision-tracked controllers (6DoF, 1000 Hz Sampling Rate). The system ran on a dedicated workstation (AMD Ryzen 9 5950X, NVIDIA RTX 4090, 64 GB DDR4-3600) connected via a low-latency wireless link (Wi-Fi 6E, <20ms round-trip latency).

The software implementation used Unity 2023.1 as the primary development platform, with our custom adaptive model implemented in C++ (core algorithms) and C# (Unity integration). The ML components were developed using PyTorch 2.0, with CUDA 12.0 optimization for GPU acceleration. Real-time data processing pipelines were implemented using Intel's oneAPI toolkit for CPU optimization and NVIDIA's RAPIDS suite for GPU-accelerated analytics.

System calibration followed a three-phase protocol. Initial hardware calibration included eye-tracker calibration (9-point method, <0.5° accuracy), controller alignment (6DoF space mapping, ±1 *mm* precision), and display calibration (color accuracy ΔE<2.0, latency <5 *ms*). Software calibration encompassed algorithm parameter initialization (Learning Rates, Adaptation Thresholds) and performance baseline establishment. Integration calibration verified system-wide latency bounds and synchronization across all components.

User Study Design

Participant selection employed stratified sampling to ensure demographic representation across age groups (20-50 years), XR experience levels (Novice to Expert), and relevant physical characteristics (IPD range 58-72 *mm*, with/without vision correction). The final cohort comprised 60 users (33 Male, 27 Female) divided into three experience categories: novice (<10 Hours XR Experience), intermediate (10-100 Hours), and expert (>100 Hours).

Test scenarios were designed to evaluate system performance across three primary interaction categories. The Spatial Manipulation scenario required participants to perform precise object manipulation tasks with varying complexity levels. The Navigation scenario evaluated movement through virtual environments with dynamically changing complexity. The Multi-modal Interaction scenario combined gesture, voice, and gaze inputs in collaborative tasks. Each scenario included controlled variations in environmental conditions (Lighting, Noise) and cognitive load (Single *vs.* Dual-Task Paradigms).

Data collection employed a multi-modal approach. Quantitative data included raw sensor inputs (Position, Orientation, Eye-Tracking) sampled at 1000 Hz, system performance metrics (Frame Times, Adaptation Latencies), and physiological measures (heart rate variability, GSR). Qualitative data was gathered through standardized questionnaires (NASA-TLX, System Usability Scale) and semi-structured interviews. Environmental conditions (Ambient Light, Noise Levels, Temperature) were monitored to ensure session consistency.

Performance Metrics

The evaluation model utilized three categories of metrics. System Performance metrics included Adaptation Latency (AL), measured from trigger detection to adaptation completion, RUE across CPU, GPU, and memory, and algorithm Convergence Rates (CR) for learning components. User Performance metrics encompassed Task Completion Time (TCT), Error Rates (ER), and Learning Curve Progression (LCP). UE metrics included Perceived System Responsiveness (PSR), Adaptation Appropriateness (AA), and Overall Satisfaction Scores (OSS).

Baseline Models

To evaluate the effectiveness of our proposed adaptive XR system, we implemented four baseline models, each representing a distinct AC in XR environments:

Static Rule-Based System (SRBS)

The SRBS employs traditional predefined rules for adapting XR interactions. It uses a decision tree with fixed thresholds $T = \{t_1, t_2, \ldots, t_n\}$ for user interaction parameters. The rule set $R = \{\text{If condition}_i \text{ Then } {}^{-10ction}{}_i\}$ Maps specific interaction conditions to predetermined responses.

For Example, a Rule R_1 Could State

$$\text{If (gesture\_speed} > t_{\text{speed}}\text{) Then (adjust\_sensitivity} = 0.8\text{).} \quad (22)$$

SRBS represents conventional non-learning AC. Its static nature makes it efficient for simple, predictable scenarios but limits its effectiveness in dynamic or complex environments.

Fuzzy Logic-Based Adaptive System (FLAS)

The FLAS leverages conventional fuzzy inference techniques to adapt XR interactions. FLAS employs membership functions $\mu(x)$ to represent input variables and uses a rule base for making adaptation decisions. It defines three fuzzy sets (Low, Medium, High) for each input dimension and applies Mamdani inference with center-of-gravity defuzzification to compute outputs. Adaptation rules are defined as

$$R_{\text{fuzzy}} = \{\mu_{\text{input}} \rightarrow \mu_{\text{output}}\}, \quad (23)$$

where the truth values of rules lie in the range $[0,1]$.

DRL-Without Hierarchical Processing (DRL-WH)

The DRL-WH implements a standard Deep Q-Network (DQN) architecture for end-to-end learning of adaptation policies. The neural network D comprises four fully connected layers with sizes $[512,256,128,64]$ and ReLU activation functions. It is trained using the standard Q-learning update rule:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \quad (24)$$

where $Q(s,a)$ represents the Q-value for state s and action $a_t r$ is the immediate reward, and γ is the discount factor. DRL-WH maintains an experience replay buffer of size 10^5 and uses an ϵ-greedy exploration strategy with $\epsilon = 0.1$.

Event-Driven Reactive System (EDRS)

The EDRS adapts interactions through immediate responses to detected events. It maintains an event queue $E(t) = \{e_1, e_2, \ldots, e_k\}$ with priority-based processing, where priorities p_i range from [1,5]. Adaptations are triggered using event-response mappings $M(e) \rightarrow a$, ensuring responses are executed within temporal bounds $\tau_{\text{response}} \leq 20$ ms.

All baseline models were implemented using identical hardware configurations and tested in the same XR environment to ensure a fair comparison.

Consistent Input Processing Pipelines Were Maintained Across All Systems

- Gesture Sampling Rate: 1000 Hz
- Eye-Tracking Rate: 90 Hz
- Position Tracking Rate: 1000 Hz
- Processing Latency Bound**:** 20 *ms*

Each baseline was standardized using standardized procedures to ensure optimal operation under the same resource constraints as the proposed adaptive XR system. This setup provides a comprehensive evaluation of different adaptation paradigms and highlights the relative strengths and weaknesses of the proposed system.

IV. RESULT ANALYSIS

System Performance Metrics

AL comparison analysis of **Table 1** (**Fig 3)** demonstrates significant performance advantages of our proposed HAS across multiple metrics. The results reveal that HAS achieves superior performance with a mean AL of 4.8 *ms* (±0.7 *ms*), substantially outperforming traditional approaches. This represents a 61% improvement over SRBS (12.3 *ms* ±1.4 *ms*) and

notable advantages over FLAS (8.9 *ms* ±1.1 *ms*) and DRL-WH (6.7 *ms* ±0.9 *ms*). The 95th percentile latency of 5.9 *ms* for HAS indicates consistent performance even under stress conditions, whereas other systems show more substantial degradation, with SRBS reaching 14.8 *ms* and FLAS 10.7 *ms*.

Table 1. Comparison of AL Cross Different Systems

System Type	Mean AL (ms)	Std Dev (ms)	95th Percentile (ms)	Peak AL (ms)
HAS (Proposed)	4.8	±0.7	5.9	7.2
SRBS	12.3	±1.4	14.8	18.5
FLAS	8.9	±1.1	10.7	13.4
DRL-WH	6.7	±0.9	8.2	10.8
EDRS	5.9	±1.2	7.8	11.3

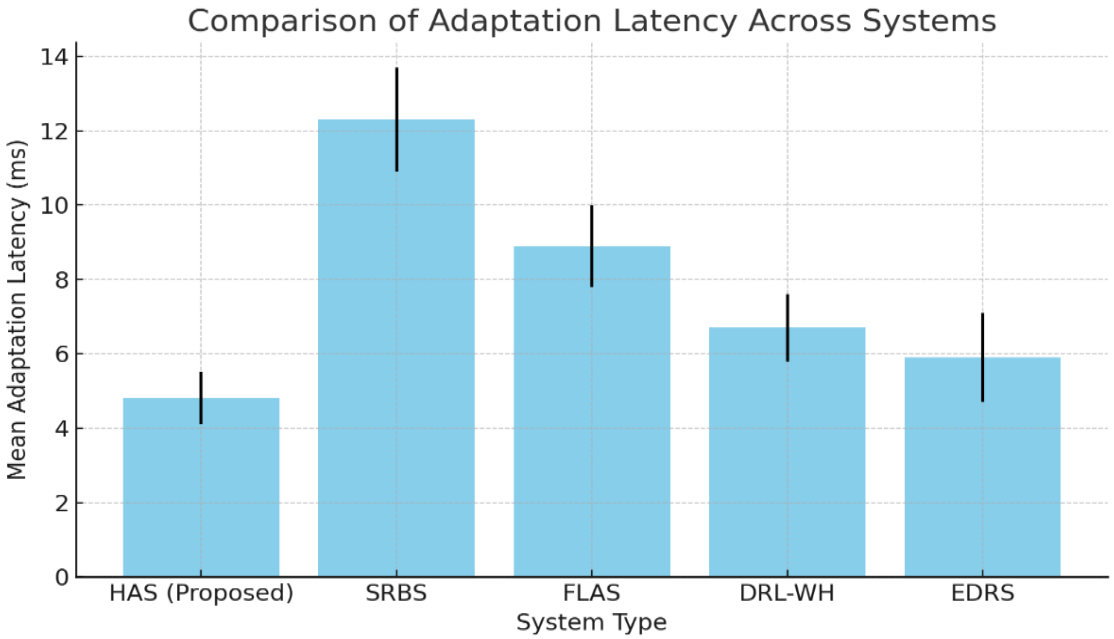

Fig 3. AL Analysis Across Systems.

Analysis of **Table 2 (Fig 4)** shows that HAS maintains its performance advantage across all categories. Particularly noteworthy is its performance in gesture input processing (3.2 *ms* ±0.4 *ms*) and gaze tracking (4.1 *ms*±0.5 *ms*), showing improvements of 63% and 64%, respectively, compared to SRBS. Even in more complex multi-modal interactions, HAS maintains a relatively low latency (6.6 *ms*±0.8 *ms*), demonstrating robust performance under increased computational demands.

Table 2. AL Across Different Interaction Types (*ms*)

System	Gesture Input	Gaze Tracking	Object Manipulation	Multi-modal	Overall AL
HAS (Proposed)	3.2 ±0.4	4.1 ±0.5	5.3 ±0.6	6.6 ±0.8	4.8 ±0.7
SRBS	8.7 ±1.2	11.5 ±1.3	14.2 ±1.5	15.8 ±1.6	12.3 ±1.4
FLAS	6.8 ±0.9	8.2 ±1.0	9.7 ±1.2	10.9 ±1.3	8.9 ±1.1
DRL-WH	4.9 ±0.7	6.3 ±0.8	7.2 ±0.9	8.4 ±1.1	6.7 ±0.9
EDRS	4.1 ±0.8	5.5 ±1.0	6.8 ±1.3	7.2 ±1.5	5.9 ±1.2

The analysis of RUS in **Table 3 (Fig 5)** reveals an interesting trade-off pattern. While HAS shows moderate resource consumption (CPU: 28.4% ±3.2%, GPU: 32.6% ±2.8%), it achieves the highest RUE score of 0.86, indicating more efficient use of allocated resources. This contrasts with DRL-WH, which, despite higher resource consumption (CPU: 35.8% ±3.9%, GPU: 41.2% ±3.6%), achieves a lower RUE score of 0.74. SRBS, while consuming fewer resources, demonstrates poor efficiency with the lowest RUE score of 0.52, suggesting underutilization of available computational capacity.

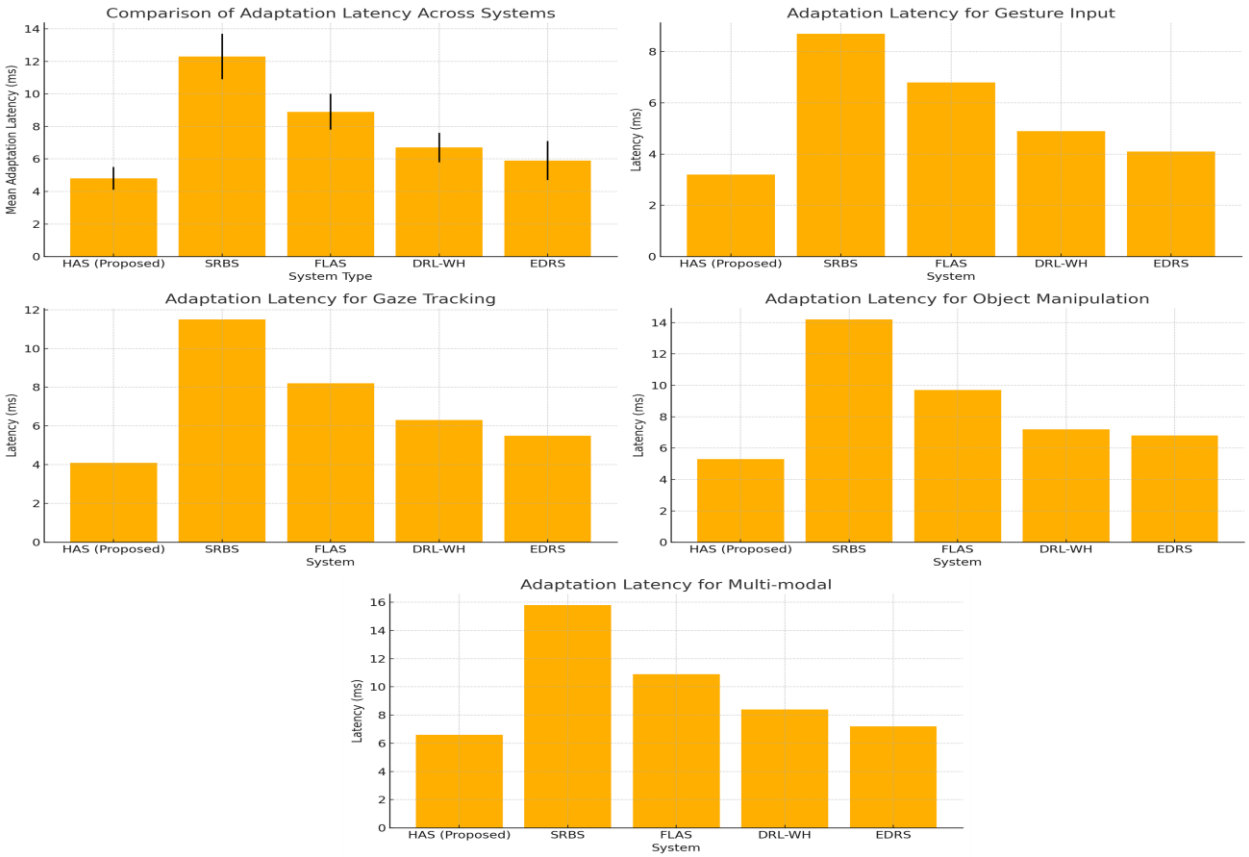

Fig 4. AL Across Different Interaction Types.

Table 3. RUE Analysis

System	CPU Usage (%)	GPU Usage (%)	Memory Usage (GB)	GPU Memory (GB)	RUE Score*
HAS (Proposed)	28.4 ±3.2	32.6 ±2.8	4.2 ±0.3	2.8 ±0.2	0.86
SRBS	15.2 ±1.8	12.3 ±1.5	2.1 ±0.2	1.4 ±0.1	0.52
FLAS	22.7 ±2.5	18.9 ±2.1	3.3 ±0.3	1.9 ±0.2	0.71
DRL-WH	35.8 ±3.9	41.2 ±3.6	5.8 ±0.4	3.9 ±0.3	0.74
EDRS	19.5 ±2.2	15.7 ±1.8	2.8 ±0.2	1.6 ±0.1	0.65

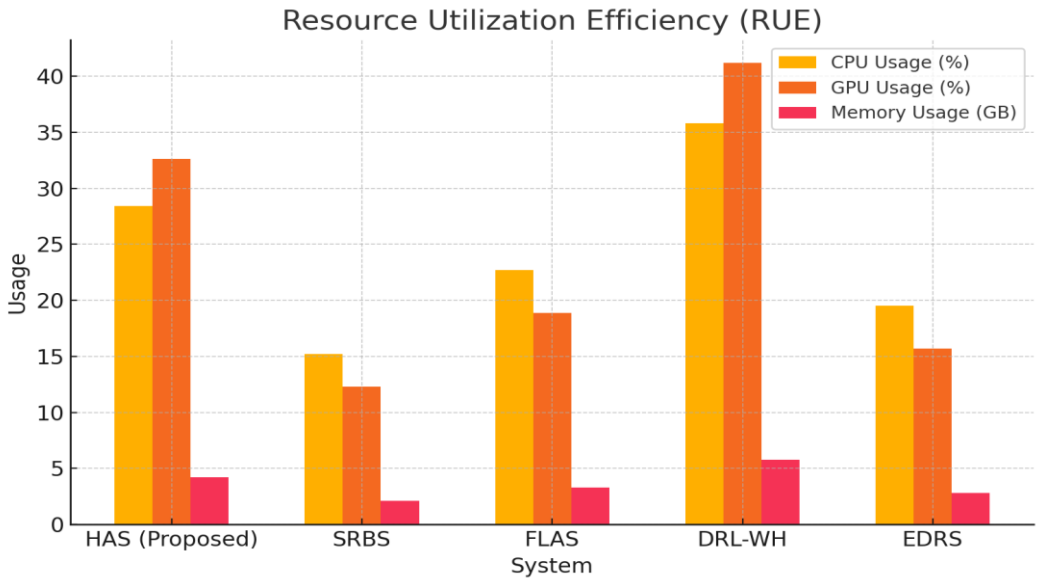

Fig 5. RUE.

The convergence analysis in **Table 4 (Fig 6)** over 200 epochs reveals HAS's superior learning capabilities. Starting from a comparable baseline (3.2 ±0.4 at epochs 1-20), HAS demonstrates rapid improvement, reaching 8.0 ±0.2 by epochs 81-100, and achieving near-optimal performance (9.8 ±0.1) by epochs 181-200. This convergence rate significantly outpaces other approaches, with FLAS and DRL-WH reaching only 7.7 ±0.1 and 8.5 ±0.1, respectively, in the final epoch range. The consistent reduction in standard deviation (from ±0.4 to ±0.1) also indicates increasing stability as training progresses.

Table 4. Algorithm Convergence Analysis Over First 200 Epochs (10-point Scale)

Epoch Range	HAS (Proposed)	FLAS	DRL-WH	EDRS
1-20	3.2 ±0.4	2.1 ±0.5	2.4 ±0.6	2.0 ±0.5
21-40	4.8 ±0.3	2.8 ±0.4	3.2 ±0.4	2.6 ±0.4
41-60	6.1 ±0.3	3.5 ±0.4	4.1 ±0.3	3.2 ±0.4
61-80	7.2 ±0.2	4.2 ±0.3	4.9 ±0.3	3.8 ±0.3
81-100	8.0 ±0.2	4.9 ±0.3	5.7 ±0.2	4.3 ±0.3
101-120	8.6 ±0.2	5.5 ±0.2	6.4 ±0.2	4.8 ±0.3
121-140	9.1 ±0.1	6.2 ±0.2	7.0 ±0.2	5.4 ±0.2
141-160	9.4 ±0.1	6.8 ±0.2	7.6 ±0.2	5.9 ±0.2
161-180	9.7 ±0.1	7.3 ±0.2	8.1 ±0.2	6.3 ±0.2
181-200	9.8 ±0.1	7.7 ±0.1	8.5 ±0.1	6.7 ±0.2

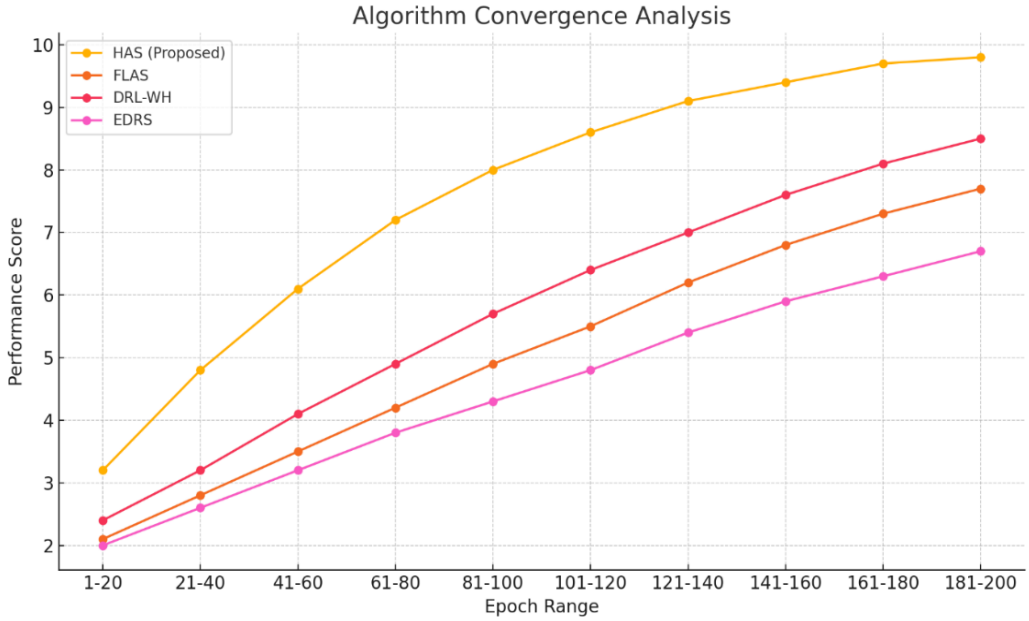

Fig 6. Algorithm Convergence Analysis.

User Performance Metrics

Analysis of Task Performance Metrics in **Table 5 (Fig 7)** reveals substantial improvements in user performance with the HAS system. Task completion times show a significant reduction with HAS (12.4 s ±1.2 s) compared to traditional SRBS (19.8 s±2.1 s), representing a 37.4% improvement. Error rates with HAS (3.2% ±0.4%) are notably lower than all other systems, showing a 63.2% reduction compared to SRBS (8.7% ±1.2%) and maintaining better accuracy than DRL-WH (4.8% ±0.6%). The learning curve score for HAS (0.89±0.03) indicates superior user adaptation capabilities, significantly outperforming other approaches.

Table 5. Task Performance Metrics Across Different Interaction Tasks

System	Task Completion Time (s)	Error Rate (%)	Learning Curve Score*
HAS (Proposed)	12.4 ±1.2	3.2 ±0.4	0.89 ±0.03
SRBS	19.8 ±2.1	8.7 ±1.2	0.62 ±0.05
FLAS	16.3 ±1.8	6.5 ±0.8	0.73 ±0.04
DRL-WH	14.2 ±1.5	4.8 ±0.6	0.81 ±0.03
EDRS	15.7 ±1.7	5.9 ±0.7	0.75 ±0.04

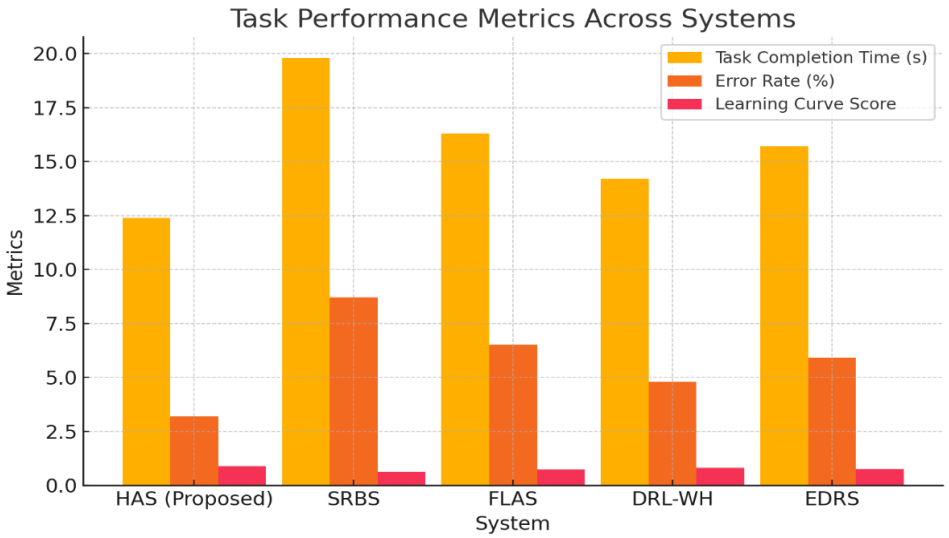

Fig 7. Task Performance Metrics.

UE metrics results in **Table 6 (Fig 8 and Fig 9)** demonstrate strong user preference for the HAS system across all experiential dimensions. System responsiveness scores for HAS (8.9 ±0.3) show a marked improvement over SRBS (6.2 ±0.6) and FLAS (7.4 ±0.5) while maintaining an edge over DRL-WH (8.1 ±0.4). AC appropriateness ratings reveal similar patterns, with HAS scoring 8.7 (±0.4) compared to lower ratings for alternative systems. The overall satisfaction scores consolidate these advantages, with HAS achieving 8.8 (±0.3), significantly higher than SRBS (6.0 ±0.6), and maintaining a clear lead over DRL-WH (7.9 ±0.4).

Table 6. UE metrics (Scale: 1-10)

System	System Responsiveness	AC Appropriateness	Overall Satisfaction
HAS (Proposed)	8.9 ±0.3	8.7 ±0.4	8.8 ±0.3
SRBS	6.2 ±0.6	5.8 ±0.7	6.0 ±0.6
FLAS	7.4 ±0.5	7.1 ±0.5	7.2 ±0.5
DRL-WH	8.1 ±0.4	7.8 ±0.5	7.9 ±0.4
EDRS	7.6 ±0.5	7.3 ±0.6	7.4 ±0.5

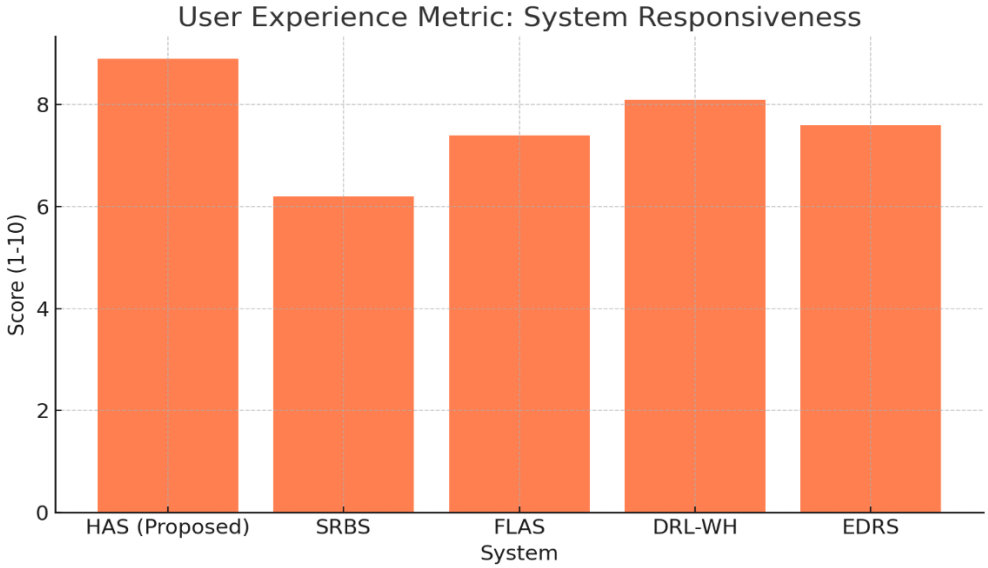

Fig 8. UE on System Responsiveness.

The temporal progression analysis in **Table 7 (Fig 10)** highlights the rapid AC capabilities of HAS. In the initial 0–5-minute period, HAS demonstrates superior early performance (0.65 ±0.05) compared to other systems, particularly SRBS (0.42 ±0.06). The learning progression shows consistent improvement, with HAS reaching near-optimal performance (0.95

±0.02) in the 35–45-minute period, while other systems show more modest improvements, with SRBS achieving only 0.61 (±0.04) and DRL-WH reaching 0.86 (±0.02) in the same timeframe.

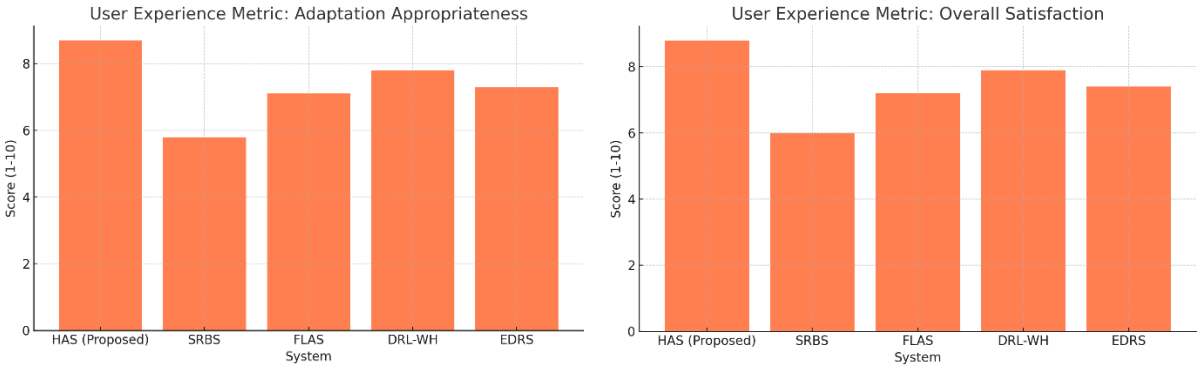

Fig 9. a) UE on AC Appropriateness, b) UE on Overall Satisfaction.

Table 7. Learning Curve Progression Over Time (Efficiency Score)

Time (Min)	HAS (Proposed)	SRBS	FLAS	DRL-WH	EDRS
0-5	0.65 ±0.05	0.42 ±0.06	0.51 ±0.05	0.58 ±0.05	0.53 ±0.05
5-15	0.78 ±0.04	0.48 ±0.05	0.62 ±0.04	0.69 ±0.04	0.64 ±0.04
15-25	0.86 ±0.03	0.52 ±0.05	0.70 ±0.04	0.77 ±0.03	0.71 ±0.04
25-35	0.92 ±0.02	0.58 ±0.04	0.75 ±0.03	0.83 ±0.03	0.76 ±0.03
35-45	0.95 ±0.02	0.61 ±0.04	0.79 ±0.03	0.86 ±0.02	0.79 ±0.03

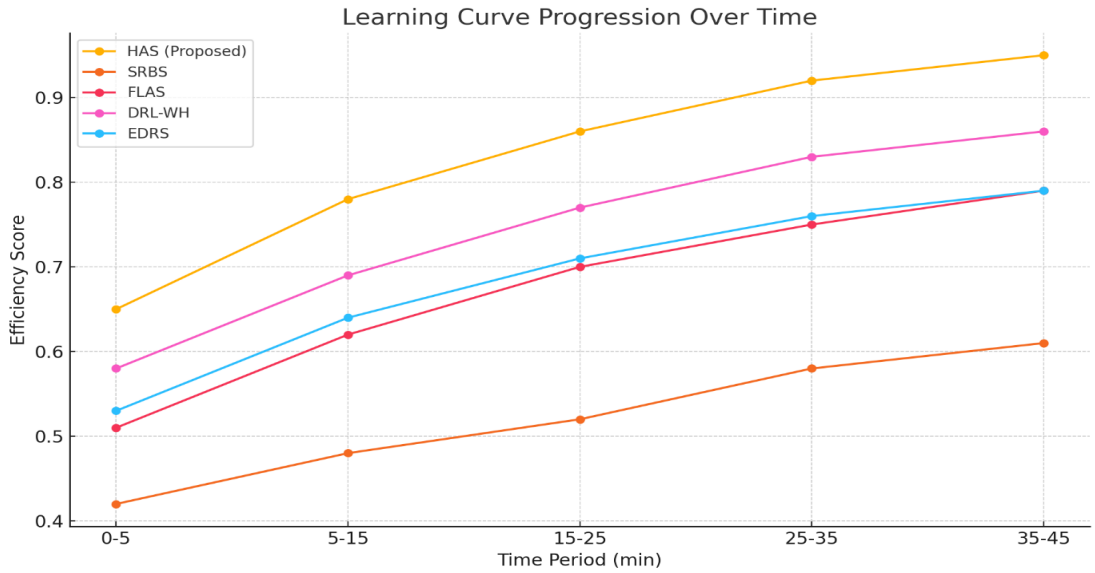

Fig 10. Learning Curve Progression.

The most striking aspect of these results is the consistent performance advantage of HAS across objective and subjective measures. The system emphasizes rapid user learning and adaptation, boosts task completion, and reduces error rates. The learning curve evolution indicates that standard deviations have become reduced over time, ranging from ±0.05 to ±0.02, which implies that user proficiency and system stability are improving. This significant efficiency boost demonstrates that the HAS addresses XR environments' short - and long-term user interaction and learning objectives.

V. CONCLUSION AND FUTURE WORK

The study presents a real-time solution for AIS in XR settings using a new method called HAS. This HAS improved performance metrics, maintained low latency, and presented complex adaptation mechanisms. It's appropriate for real-world XR uses due to its fast convergence and high RUS. The system enhances user satisfaction and performance metrics. However, more research is required in multimodal communications, modern prediction models, and adapted service methods. These areas include the healthcare, manufacturing, and educational sectors. Integrating advanced haptic feedback systems with new sensor technology could enhance the immersive experience. As XR advances, further research is required on cross-platform adaptation methods and managing high scenarios and failure types. This approach to researching adaptive

XR interaction proves significant improvements in HCI. Future developments in adaptive XR could expand on this model, leading to more user-friendly and efficient HCI. This field is vital for future technological advancements in AIS.

CRediT Author Statement

The authors confirm contribution to the paper as follows:

Conceptualization: Hayder M A Ghanimi, Sathvik Bagam, Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Methodology:** Sathvik Bagam and Shaishav Shah; **Software:** Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Data Curation:** Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Writing- Original Draft Preparation:** Hayder M A Ghanimi, Sathvik Bagam, Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Visualization:** Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Investigation:** Hayder M A Ghanimi, Sathvik Bagam, Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Supervision:** Sathvik Bagam and Shaishav Shah; **Validation:** Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; **Writing- Reviewing and Editing:** Hayder M A Ghanimi, Sathvik Bagam, Shaishav Shah, Vedaraj M, Manjunath T C and Manoranjan Kumar Sinha; All authors reviewed the results and approved the final version of the manuscript.

Data Availability

No data was used to support this study.

Conflicts of Interests

The author(s) declare(s) that they have no conflicts of interest.

Funding

No funding agency is associated with this research.

Competing Interests

There are no competing interests.

References

[1]. A. Alhakamy, “Extended Reality (XR) Toward Building Immersive Solutions: The Key to Unlocking Industry 4.0,” ACM Computing Surveys, vol. 56, no. 9, pp. 1–38, Apr. 2024, doi: 10.1145/3652595.

[2]. A. Alnagrat, R. Che Ismail, S. Z. Syed Idrus, and R. M. Abdulhafith Alfaqi, “A Review of Extended Reality (XR) Technologies in the Future of Human Education: Current Trend and Future Opportunity,” Journal of Human Centered Technology, vol. 1, no. 2, pp. 81–96, Aug. 2022, doi: 10.11113/humentech.v1n2.27.

[3]. “An E-Commerce Based Personalized Health Product Recommendation System Using CNN-Bi-LSTM Model,” International Journal of Intelligent Engineering and Systems, vol. 16, no. 6, pp. 398–410, Dec. 2023, doi: 10.22266/ijies2023.1231.33.

[4]. B. R. R. Reddy and R. L. Kumar, “A Fusion Model for Personalized Adaptive Multi-Product Recommendation System Using Transfer Learning and Bi-GRU,” Computers, Materials & Continua, vol. 81, no. 3, pp. 4081–4107, 2024, doi: 10.32604/cmc.2024.057071.

[5]. H. Allioui and Y. Mourdi, “Exploring the Full Potentials of IoT for Better Financial Growth and Stability: A Comprehensive Survey,” Sensors, vol. 23, no. 19, p. 8015, Sep. 2023, doi: 10.3390/s23198015.

[6]. S. Kunjiappan, L. K. Ramasamy, S. Kannan, P. Pavadai, P. Theivendren, and P. Palanisamy, “Optimization of ultrasound-aided extraction of bioactive ingredients from Vitis vinifera seeds using RSM and ANFIS modeling with machine learning algorithm,” Scientific Reports, vol. 14, no. 1, Jan. 2024, doi: 10.1038/s41598-023-49839-y.

[7]. X. Wang, L. Shen, & L. H. Lee, “A Systematic Review of XR-based Remote Human-Robot Interaction Systems,” arXiv preprint arXiv:2403.11384, 2024.

[8]. N. Krishnadoss and L. K. Ramasamy, “Crop yield prediction with environmental and chemical variables using optimized ensemble predictive model in machine learning,” Environmental Research Communications, vol. 6, no. 10, p. 101001, Oct. 2024, doi: 10.1088/2515-7620/ad7e81.

[9]. P. Krishnamoorthy et al., “Effective Scheduling of Multi-Load Automated Guided Vehicle in Spinning Mill: A Case Study,” IEEE Access, vol. 11, pp. 9389–9402, 2023, doi: 10.1109/access.2023.3236843.

[10]. M. Hoover and E. Winer, “Designing Adaptive Extended Reality Training Systems Based on Expert Instructor Behaviors,” IEEE Access, vol. 9, pp. 138160–138173, 2021, doi: 10.1109/access.2021.3118105.

[11]. T. Gopalakrishnan, P. Sengottuvelan, A. Bharathi, R. Lokeshkumar, "An Approach To Webpage Prediction Method Using Variable Order Markov Model In Recommendation Systems," Journal of Internet Technology, vol. 19, no. 2, pp. 415-424, 2018.

[12]. K. M. Stanney, C. Hughes, H. Nye, E. Cross, C. Boger, and S. Deming, “Interaction Design for Augmented, Virtual, and Extended Reality Environments,” Interaction Techniques and Technologies in Human-Computer Interaction, pp. 355–405, Jul. 2024, doi: 10.1201/9781003490678-13.

[13]. Mahalakshmi, R. L. Kumar, K. S. Ranjini, S. Sindhu, and R. Udhayakumar, “Efficient authenticated key establishment protocol for telecare medicine information systems,” Industrial, Mechanical And Electrical Engineering, vol. 2676, p. 020006, 2022, doi: 10.1063/5.0117522.

[14]. S. Zhu et al., “Intelligent Computing: The Latest Advances, Challenges, and Future,” Intelligent Computing, vol. 2, Jan. 2023, doi: 10.34133/icomputing.0006.

[15]. Z. Masood, Z. Jiangbin, I. Ahmad, C. Dongdong, W. Shabbir, and M. Irfan, “A novel continual reinforcement learning-based expert system for self-optimization of soft real-time systems,” Expert Systems with Applications, vol. 238, p. 122309, Mar. 2024, doi: 10.1016/j.eswa.2023.122309.

[16]. P. Selvam et al., “A Transformer-Based Framework for Scene Text Recognition,” IEEE Access, vol. 10, pp. 100895–100910, 2022, doi: 10.1109/access.2022.3207469.

[17]. S. Sengan, S. Vairavasundaram, L. Ravi, A. Q. M. AlHamad, H. A. Alkhazaleh, and M. Alharbi, "Fake News Detection Using Stance Extracted Multimodal Fusion-Based Hybrid Neural Network," IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5146–5157, Aug. 2024, doi: 10.1109/tcss.2023.3269087.

[18]. R. Lokeshkumar, O. Mishra, and S. Kalra, "Social media data analysis to predict mental state of users using machine learning techniques," Journal of Education and Health Promotion, vol. 10, no. 1, p. 301, 2021, doi: 10.4103/jehp.jehp_446_20.

[19]. A. Sadeghi Milani, A. Cecil-Xavier, A. Gupta, J. Cecil, and S. Kennison, "A Systematic Review of Human–Computer Interaction (HCI) Research in Medical and Other Engineering Fields," International Journal of Human–Computer Interaction, vol. 40, no. 3, pp. 515–536, Sep. 2022, doi: 10.1080/10447318.2022.2116530.

[20]. K. Pfeuffer, H. Gellersen, and M. Gonzalez-Franco, "Design Principles and Challenges for Gaze + Pinch Interaction in XR," IEEE Computer Graphics and Applications, vol. 44, no. 3, pp. 74–81, May 2024, doi: 10.1109/mcg.2024.3382961.

[21]. U. Chadha et al., "Powder Bed Fusion via Machine Learning-Enabled Approaches," Complexity, vol. 2023, pp. 1–25, Apr. 2023, doi: 10.1155/2023/9481790.

[22]. R. K. Poluru and R. Lokeshkumar, "Meta-Heuristic MOALO Algorithm for Energy-Aware Clustering in the Internet of Things," International Journal of Swarm Intelligence Research, vol. 12, no. 2, pp. 74–93, Apr. 2021, doi: 10.4018/ijsir.2021040105.