International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Live Detection of AI-Agents Through Interaction Pattern Analysis

A Unified Behavioral-ML Architecture for Real-Time Bot Classification, Multi-Feature Anomaly Detection, and Confidence-Scored User Authentication

Dr. C. M. T. Karthigeyan • Jaya Sri S • Renuka M • Suganya T
Department of Computer Science and Engineering,
Government College of Engineering, Bargur – 635104, Krishnagiri, Tamil Nadu, India
Anna University, Chennai – 600 025

ABSTRACT - The proliferation of intelligent automated agents across digital platforms poses mounting threats to system security, service reliability, and user authenticity. Traditional verification mechanisms such as CAPTCHA and image-based filters are increasingly ineffective against sophisticated bots equipped with machine learning capabilities. This paper presents the *Live Detection of AI-Agents Through Interaction Pattern Analysis* — a fully passive, real-time behavioral classification system that continuously monitors user interactions without interrupting normal usage. The system integrates four synergistic components: (1) Multi-feature behavioral collection capturing mouse movements, click dynamics, typing rhythm, and drag behavior; (2) Statistical feature extraction computing 8 interaction attributes including velocity, acceleration, entropy, and variance; (3) Hybrid ML classification combining Isolation Forest, One-Class SVM, and Random Forest ensemble with LSTM/GRU sequential models; and (4) a four-category user taxonomy (Human, Basic Bot, Advanced Bot, Human-like Bot) with confidence scoring. Implemented on a Flask-based asynchronous backend with a sliding window monitoring mechanism, the system delivers end-to-end classification within 142–228 milliseconds. Experimental evaluation across 10,000 synthetic samples spanning four user categories demonstrates a classification accuracy of 97.3% on clean data, F1-scores ranging from 0.880 to 0.979 across categories, and full compatibility with privacy-preserving passive deployment. The system's open-source architecture, REST API, and documented Flask backend position it as a scalable, equitable, and extensible supplement to traditional bot-prevention systems.

Keywords: ***Bot Detection; Behavioral Biometrics; Anomaly Detection; Isolation Forest; One-Class SVM; LSTM; GRU; Random Forest; Real-Time Classification; Interaction Pattern Analysis; Flask Microservices; User Authentication***

1. INTRODUCTION

The modern digital ecosystem relies on billions of daily interactions between users and online services — from e-commerce transactions and social media engagement to cloud-based productivity tools and critical infrastructure dashboards. Yet a growing and increasingly sophisticated proportion of these interactions originates not from human users but from automated software agents commonly known as bots. According to recent cybersecurity industry reports, bot traffic constitutes approximately 40–47% of all internet traffic, with malicious bots alone accounting for over 27% of total web requests [Imperva, 2023].

The consequences of undetected bot infiltration are substantial and multidimensional. In e-commerce, credential stuffing attacks — where bots systematically test stolen username-password pairs — result in billions of dollars in annual account takeover losses. On social media platforms, coordinated inauthentic behavior by bot networks distorts public discourse, amplifies misinformation, and manipulates engagement metrics. In web scraping scenarios, bots systematically harvest proprietary content, pricing data, and personal information in violation of terms of service. In click fraud, automated agents generate fraudulent advertising impressions and conversions, siphoning advertiser budgets at scale.

The convergence of machine learning, lightweight deep learning architectures, and high-throughput web frameworks in the mid-2020s makes the proposed system feasible in ways not achievable even five years ago. Scikit-learn's ensemble models now achieve over 95% classification accuracy on behavioral datasets on commodity hardware. Sequential neural networks (LSTM, GRU) have matured sufficiently for sub-200ms inference on CPU-only deployments. Flask and asynchronous Python enable concurrent multi-user behavioral analysis without specialized infrastructure.

This paper makes the following original contributions to the field of AI agent detection:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- A unified, modular behavioral analysis pipeline integrating multi-feature extraction, unsupervised anomaly detection, and ensemble classification into a single cohesive web service.
- A four-category user taxonomy (Human, Basic Bot, Advanced Bot, Human-like Bot) providing clinically actionable diagnostic specificity beyond binary human/bot classification.
- Empirical evaluation across 10,000 synthetic behavioral samples spanning four user categories and five operating conditions, establishing performance baselines for future research.
- A fully documented, open-source Flask REST API enabling integration with enterprise security systems, mobile applications, and educational platforms.
- Demonstrated superiority over CAPTCHA, rule-based filtering, and isolated ML approaches across eight comparative dimensions, with particular differentiation in passive operation capability and multi-category classification.

Table 1. Feature Comparison of Bot Detection Approaches

Feature	CAPTCHA	Rule-Based	Isolated ML	Deep Learning	Proposed System
Real-Time Feedback	No	Partial	No	Partial	Yes (<200ms)
Multi-Feature Analysis	No	No	Partial	Yes	Yes — 8 features
Deep Learning	No	No	No	Yes	LSTM + GRU
Bot Taxonomy	No	No	Binary only	Binary only	4 categories
Non-Intrusive	No	Yes	Yes	Yes	Yes (passive)
Anomaly Detection	No	Partial	Yes	Yes	Isolation Forest + OC-SVM
Open Source	N/A	No	Partial	Partial	Yes (Flask + sklearn)
Detection Accuracy	~60%	~70%	~82%	~88%	~97% (clean data)

2. BACKGROUND AND MOTIVATION

2.1 The Global Bot Threat Landscape

The threat posed by automated agents is not merely a technical nuisance but an economic and societal challenge with measurable consequences. The Imperva 2023 Bad Bot Report estimated that bad bot traffic increased by 5.1% year-over-year to represent 30.2% of all website traffic. Financial services, e-commerce, and government portals report the highest bot attack volumes, with credential stuffing attacks growing by 45% in 2022 alone. In India, the rapid expansion of digital payment systems, social commerce, and government e-service portals has created a particularly high-risk environment, where bot attacks on UPI-based platforms have resulted in documented financial fraud cases totaling thousands of crores annually.

The economic burden of bot attacks is substantial. A single credential stuffing campaign targeting a major e-commerce platform can test millions of stolen credentials within hours, with successful account takeovers yielding fraudulent transactions, loyalty point theft, and sensitive data exfiltration. Downstream costs including fraud remediation, customer support, infrastructure overload, and reputational damage typically exceed direct financial losses by a factor of 3–5x according to industry analyses.

2.2 Limitations of Current Detection Tools

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) has served as the dominant bot-prevention mechanism for over two decades. However, its effectiveness has degraded dramatically as adversarial machine learning has advanced. Modern CAPTCHA-solving services leverage convolutional neural networks achieving over 90% accuracy on standard image CAPTCHAs. Audio CAPTCHA can be defeated by off-the-shelf speech recognition. ReCAPTCHA v3's risk-scoring model, while innovative, produces high false-positive rates that penalize legitimate users on VPNs, Tor networks, or less common browsers. Most critically, CAPTCHA is inherently disruptive: each challenge event interrupts user flow, increases task completion

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

time, and reduces conversion rates — measurable user experience costs that motivate the passive detection approach proposed in this paper.

Rule-based systems attempt to identify bots through predefined thresholds on behavioral metrics such as request frequency, IP reputation, and session duration. While computationally inexpensive, these systems suffer from two fundamental weaknesses. First, they are reactive: rules must be manually updated as bot operators learn the threshold values and adapt their tools to stay below detection limits. Second, they generate high false-positive rates when legitimate power users, API clients, or automated testing tools exhibit bot-like behavior patterns that happen to cross rule thresholds.

Isolated machine learning models applied to single behavioral features — such as mouse-only analysis or keystroke-only detection — improve upon rule-based approaches but remain vulnerable to adversarial bots specifically engineered to mimic the single monitored signal. The proposed system addresses this multi-feature gap through simultaneous analysis of eight behavioral dimensions.

3. LITERATURE REVIEW

This review synthesizes research across five domains directly relevant to the proposed system: behavioral biometric authentication, bot detection using interaction dynamics, anomaly detection algorithms, sequential pattern modeling with recurrent networks, and passive real-time classification systems. The review is structured to demonstrate how each body of prior work motivates specific design decisions in the proposed architecture.

3.1 Mouse Dynamics for Bot Detection

Afanaseva and Lozhnikov [2022] demonstrated that mouse movement patterns — including speed, direction changes, and movement variability — serve as reliable discriminators between human and automated users. Their ML-based classification system achieved high accuracy but relied exclusively on mouse features, leaving keyboard and drag behavior entirely unanalyzed. This single-modality limitation directly motivates the proposed system's eight-feature multi-modal approach. The finding that human movement is inherently irregular and unpredictable while bot movement is structured and consistent forms the theoretical foundation for the feature engineering strategy employed in this work.

3.2 Keystroke Dynamics and Behavioral Biometrics

Acien et al. [2024] conducted a systematic comparison of keystroke dynamics and mouse trajectory analysis for user authentication and bot detection, finding that combining both modalities substantially improves accuracy over either alone. Their results confirm that typing rhythm — characterized by key dwell time (duration a key is held) and flight time (interval between consecutive keys) — constitutes a strong behavioral biometric that is difficult for automated systems to replicate convincingly. This finding directly motivates the inclusion of typing_speed and keys features in the proposed system's feature set alongside mouse metrics.

3.3 Passive and Non-Intrusive Detection

Mohitkar et al. [2023] introduced a passive CAPTCHA system that analyzes behavioral data silently without presenting explicit user challenges, demonstrating that non-intrusive detection is both technically feasible and superior in usability to traditional verification approaches. Their system used One-Class SVM for anomaly detection, which the proposed system adopts as one component of a hybrid detection ensemble. The comparative usability advantage of passive systems over interactive challenges provides the motivating rationale for the proposed system's background-operation design philosophy.

3.4 Anomaly Detection Algorithms

Liu et al. [2008]'s Isolation Forest algorithm, subsequently refined and evaluated across multiple application domains, provides the primary unsupervised anomaly detection backbone for the proposed system. The algorithm's core insight — that anomalous data points require fewer random partitions to isolate than normal points, enabling efficient anomaly scoring in O(n log n) time — makes it well-suited to the high-throughput, low-latency requirements of real-time bot detection. The One-Class SVM, applied in the proposed system as a complementary anomaly detector, learns a hypersphere boundary around normal (human) behavior in feature space, classifying any data point outside this boundary as anomalous.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

3.5 Sequential Pattern Analysis with LSTM and GRU

Hochreiter and Schmidhuber's Long Short-Term Memory (LSTM) architecture [1997], subsequently refined by Cho et al.'s Gated Recurrent Unit (GRU) [2014], provides the sequential modeling foundation for analyzing temporal interaction patterns. In bot detection, sequential models are critical for capturing the temporal dynamics of user behavior — how actions unfold over time — rather than analyzing static snapshots of aggregate feature values. Sayyad et al. [2023] demonstrated that LSTM-based analysis of mouse trajectory time series substantially outperforms static feature analysis for detecting advanced bots engineered to produce realistic aggregate statistics. The proposed system integrates both LSTM and GRU in an ensemble with tree-based classifiers to leverage complementary pattern recognition capabilities.

3.6 Multi-Modal Fusion Approaches

Pozzana and Ferrara [2024] studied the behavioral dynamics of bots and humans on social media, finding that bots exhibit predictable temporal patterns and lower behavioral entropy compared to humans across multiple modalities simultaneously. This insight supports the proposed system's multi-modal feature fusion strategy, which combines mouse, keyboard, and drag interaction signals into a unified feature vector. The entropy metric included in the proposed system's feature extraction directly operationalizes this finding, measuring the unpredictability of interaction sequences as a discriminative feature.

Table 2. Literature Survey Summary — Contributions and Limitations Addressed

Ref.	Authors	Year	Technique	Key Contribution	Limitation Addressed
[1]	Afanaseva & Lozhnikov	2022	Mouse Dynamics ML	Mouse speed-based bot detection	Binary labels only; no keyboard features
[2]	Kolomeets et al.	2022	Social Bot Identification	Human vs bot perception study	Manual inspection; unreliable at scale
[3]	Amshavalli et al.	2023	Adaptive ML Auth	Behavioral authentication over time	No real-time sliding window analysis
[4]	Mohitkar et al.	2023	Passive CAPTCHA AI	Non-intrusive bot detection	One-Class SVM; no sequential models
[5]	Sayyad et al.	2023	Multi-biometric Fusion	Mouse + keystroke combined features	No LSTM/GRU temporal analysis
[6]	von Ahn et al.	2023	Lightweight Networks	Efficient real-time mouse analysis	No grammar/deep sequential models
[7]	Acien et al.	2024	Keystroke vs Mouse	Comparative biometric analysis	No ensemble; no anomaly detection
[8]	Ferrara et al.	2024	ML Literature Review	Comprehensive survey of bot detection	Review only; no working system
[9]	Pozzana & Ferrara	2024	Behavioral Dynamics	Temporal behavior measurement	No classification pipeline
[10]	Acien et al.	2024	Synthetic Trajectories	Realistic mouse simulation	Synthetic only; no multi-class taxonomy

4. SYSTEM ANALYSIS

4.1 Existing System Analysis

4.1.1 CAPTCHA-Based Verification

Traditional CAPTCHA systems require users to solve challenges before accessing services. CAPTCHA presents distorted text, image grids, or audio puzzles. While initially effective, these systems suffer three systemic constraints that severely limit their continued utility: (1) user experience disruption, as each challenge event interrupts normal interaction flow and has been shown to

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

reduce conversion rates by 3–10% in e-commerce contexts; (2) adversarial vulnerability, as CNN-based CAPTCHA solvers now achieve over 90% accuracy on standard challenges at commercial scale; and (3) accessibility barriers, as challenges routinely fail for users with visual or auditory impairments.

4.1.2 Rule-Based Behavioral Filters

Rule-based systems monitor basic interaction parameters such as request frequency, IP address behavior, and session timing against predefined thresholds. These systems are computationally inexpensive but suffer from three fundamental weaknesses. First, they are manually updated and reactive, requiring ongoing maintenance as bot operators learn threshold values. Second, they generate high false-positive rates when legitimate users exhibit unusual but authentic behavior. Third, they lack the sophistication to distinguish between the four bot categories (Basic, Advanced, Human-like) that modern threat landscapes require.

4.1.3 Isolated Machine Learning Approaches

Single-modality ML systems applying algorithms to mouse-only or keystroke-only data improve upon rule-based approaches but remain vulnerable to adversarial bots engineered specifically to mimic the single monitored signal. No existing commercial tool simultaneously addresses multi-modal behavioral analysis, unsupervised anomaly detection, sequential temporal modeling, and four-category classification within a unified real-time pipeline.

4.2 Proposed System Design Philosophy

The proposed Live Detection of AI-Agents Through Interaction Pattern Analysis system is designed around five foundational principles that collectively distinguish it from all benchmarked alternatives:

- Real-time analysis with sub-250ms end-to-end latency, ensuring feedback arrives within the sliding window analysis cycle before user interaction data expires from the working buffer.
- Multi-dimensional assessment covering eight behavioral features across mouse, keyboard, and drag interaction modalities — the three primary channels through which human-bot behavioral differences manifest.
- Four-category diagnostic classification (Human, Basic Bot, Advanced Bot, Human-like Bot) rather than binary human/bot labels, enabling security operators to calibrate response actions to threat sophistication.
- Confidence-scored output on a 0–100% scale enabling probabilistic decision-making and threshold adjustment for different security postures (high-security vs. user-experience-first deployments).
- Open, extensible architecture using an open-source Python/Flask/scikit-learn stack deployable by institutions of any size without proprietary licensing constraints.

5. SYSTEM ARCHITECTURE AND DESIGN

5.1 Three-Tier Architectural Overview

The system follows a classic three-tier web application architecture with an internal composition analogous to a microservice pipeline, where each analytical concern is handled by a dedicated, independently testable and upgradeable module. The three tiers — Presentation, Application, and Data Storage — communicate exclusively through well-defined contracts: HTTP/JSON between Presentation and Application tiers, and file I/O between the Application tier and persistent data stores. This strict separation of concerns ensures that any individual tier can be replaced (e.g., swapping the CSV data store for a relational database) without structural disruption to the overall system.

Table 3. Three-Tier Architecture — Components and Communication Contracts

Tier	Layer	Key Components	Interface	Scalability Strategy
Tier 1	Presentation	Browser HTML/JS, Mouse Tracker, Keystroke Logger, Scroll Monitor, Result Dashboard	WebSocket / HTTP POST JSON	CDN-hosted static assets; client-side rendering
Tier 2	Application	Flask Server, Feature Extractor, Isolation Forest, One-Class SVM, LSTM/GRU,	Python async / REST JSON	Gunicorn WSGI; horizontal scaling via Docker

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

		Random Forest Classifier		
Tier 3	Data Storage	CSV Dataset Store, Predictions Log, Session History	File I/O / pandas DataFrame	SQLite upgrade path; cloud DB for enterprise

5.2 Module Architecture

Within the Application Layer (Tier 2), the system is decomposed into five functionally independent modules communicating through well-typed Python data structures: the Data Collection Module, the Feature Extraction Module, the Anomaly Detection Module, the Sequential Analysis Module, and the Ensemble Classification Module. This decomposition follows the Single Responsibility Principle: each module has exactly one analytical responsibility and exposes a clean interface allowing independent unit testing, replacement, and scaling.

5.3 Data Flow Architecture

The data flow follows a sequential-then-parallel pattern designed to minimize end-to-end latency. Data collection (Module 1) and feature extraction (Module 2) execute sequentially since feature computation requires collected raw data. Once the feature vector is available, anomaly detection (Module 3) and sequential analysis (Module 4) execute in parallel using Python's threading model, since they are independent operations on the same input features. Ensemble classification (Module 5) executes after both parallel modules complete, combining their outputs. This parallel execution reduces end-to-end latency by approximately 30–38% compared to a fully sequential pipeline.

Table 4. Data Flow — Level-1 Process Decomposition

P#	Process Name	Input	Output	Data Store	Parallelism
P1	Capture Interactions	User actions (mouse, keyboard, scroll)	Raw event stream	DS1: Session Buffer	Continuous
P2	Feature Extraction	Raw event stream from DS1	8-dimensional feature vector	DS2: Feature Store	Sequential after P1
P3	Anomaly Detection	Feature vector from DS2	Anomaly scores	DS2: Session	Parallel with P4
P4	Sequential Analysis (LSTM/GRU)	Temporal feature sequences	Sequence classification score	DS2: Session	Parallel with P3
P5	Ensemble Classification	Scores from P3 + P4	User type + confidence %	DS3: Predictions Log	Sequential after P3+P4
P6	Render Results	Classification result	Dashboard display	(none)	Client-side

6. DETAILED MODULE IMPLEMENTATION

6.1 Data Collection Module

The Data Collection Module is the entry point of the analysis pipeline, responsible for capturing and structuring all incoming user interaction events in real time. The module operates passively through JavaScript event listeners attached to four primary interaction channels: mousemove events (capturing X/Y coordinates at 60Hz sampling rate), click events (capturing button identity and timestamp), keydown/keyup events (capturing key identity and dwell duration), and drag events (capturing drag start/end coordinates and distance). All events are tagged with high-resolution timestamps using the Performance API's performance.now() method (sub-millisecond resolution), enabling accurate computation of timing-dependent features such as typing speed and click rate.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The module implements a sliding window mechanism that aggregates events into fixed-length analysis windows of 5 seconds with 1-second stride, providing continuous real-time analysis without requiring session-level data accumulation. This design ensures that the system responds to behavioral changes within seconds, enabling detection of bots that begin with human-like behavior to evade initial classification before switching to automated operation.

6.2 Feature Extraction Module

The Feature Extraction Module transforms the raw event streams from each analysis window into an 8-dimensional feature vector. Features are organized into three behavioral categories that collectively characterize the multi-modal interaction signature of each user.

Mouse features capture spatial and temporal characteristics of pointer behavior. mouse_moves counts the total cursor displacement events within the window, providing a gross measure of interaction activity. mouse_speed computes the mean movement velocity in pixels-per-second, calculated as the ratio of total Euclidean path length to window duration, capturing the characteristic speed distribution that differentiates human micro-movements from bot linear interpolations.

Keyboard features capture typing dynamics. keys counts total keystrokes within the window, providing a baseline activity measure. typing_speed computes characters-per-second, distinguishing the highly variable human typing rhythm (typically 3–8 characters/second with natural acceleration and deceleration) from the uniform high-speed input characteristic of scripted keyboard automation.

Click and drag features capture interaction complexity. clicks and click_rate jointly characterize the clicking behavior — the former providing absolute count and the latter normalizing by time to detect unnaturally high-frequency clicking. drags and drag_distance capture the presence and magnitude of drag-and-drop interactions, a behavioral signal rarely replicated by simple bot scripts.

Table 5. Dataset Feature Definitions — Behavioral Significance and Classification Impact

S.No	Feature Name	Data Type	Class Differentiation	Clinical/Security Relevance
1	mouse_moves	Integer	Humans: varied high; Bots: low/structured	Primary indicator of natural vs automated navigation
2	mouse_speed	Float	Humans: moderate varied; Bots: extreme/constant	Bot scripts exhibit unrealistic speed profiles
3	clicks	Integer	Humans: irregular patterns; Bots: fixed rates	Click cadence reveals automation signature
4	click_rate	Float	Bots: very high consistent rates	Clicks-per-second is a key bot indicator
5	keys	Integer	Humans: natural variation; Bots: consistent high	Keystroke count reveals scripted input
6	typing_speed	Float	Humans: variable; Bots: unrealistically uniform	Typing rhythm is a strong behavioral biometric
7	drags	Integer	Humans: occasional; Bots: none or zero	Drag behavior is rarely replicated by scripts
8	drag_distance	Float	Humans: variable; Bots: fixed or zero	Drag trajectory complexity differentiates users
9	label	Categorical	Human, Basic Bot, Advanced Bot, Human-like Bot	Target variable for supervised classification

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

6.3 Anomaly Detection Module

The Anomaly Detection Module employs two complementary unsupervised learning algorithms to identify behavioral patterns that deviate from the learned distribution of normal (human) interaction data. This dual-algorithm design provides both improved detection coverage and cross-validation of anomaly scores.

The Isolation Forest algorithm operates by constructing an ensemble of 100 isolation trees, each built by recursively partitioning the feature space using random feature selection and random split values. The key insight is that anomalous data points — those with unusual feature combinations characteristic of bot behavior — are isolated in fewer partitioning steps than normal points, yielding shorter average path lengths from root to leaf. The anomaly score for each observation is derived from the normalized average path length across all trees in the ensemble.

The One-Class Support Vector Machine complements the Isolation Forest by learning a hypersphere boundary around the training distribution of human behavioral features using a radial basis function (RBF) kernel. Any feature vector falling outside this boundary receives an anomalous classification. The OC-SVM is particularly effective at detecting bot categories that produce feature combinations outside the convex hull of human behavioral variation, whereas the Isolation Forest excels at detecting outliers in high-density regions where the OC-SVM boundary may be overfit.

6.4 Sequential Analysis Module (LSTM and GRU)

The Sequential Analysis Module addresses a fundamental limitation of static feature analysis: it captures the temporal structure of interaction sequences rather than merely their aggregate statistics within a single window. Human users exhibit characteristic temporal patterns in their interaction sequences — for example, mouse movements that gradually accelerate when approaching a target button and decelerate on arrival, or typing patterns that show consistent inter-key timing signatures specific to individual users. Bot scripts, by contrast, produce temporally uniform sequences with constant inter-event timing regardless of context.

The LSTM model processes sequences of 10 consecutive feature vectors (representing 10 seconds of interaction history at the 1-second stride) through two stacked LSTM layers with 64 hidden units each, followed by a dense classification layer with softmax output across the four user categories. The GRU model employs an equivalent architecture with GRU units substituted for LSTM cells, providing approximately 35% faster inference at a modest accuracy trade-off. Both models are trained on the synthetic dataset described in Section 7.1.

6.5 Ensemble Classification and Scoring

The Ensemble Classification Module aggregates the outputs of all preceding modules into a final user category prediction and confidence score using a weighted voting mechanism. The Random Forest classifier receives the 8-dimensional feature vector and produces a four-class probability distribution. The Isolation Forest and OC-SVM each produce binary anomaly flags with associated confidence scores. The LSTM and GRU each produce four-class probability distributions over the temporal sequence.

The final prediction is computed as a weighted average of the five component scores, with weights determined through a grid-search calibration study on a held-out validation set: Random Forest 35%, LSTM 25%, GRU 20%, Isolation Forest 10%, OC-SVM 10%. This weighting reflects the empirical finding that the Random Forest and LSTM components contribute the highest individual accuracy on the evaluation corpus, while the anomaly detection components provide particularly strong differentiation for previously unseen bot variants not well-represented in the training data.

6.6 REST API Layer

The REST API Layer, implemented using Flask 2.3+ with Gunicorn as the production WSGI server, orchestrates all modules and exposes the system's capabilities as documented HTTP endpoints. Flask's lightweight routing framework combined with Python's threading model enables the server to process multiple concurrent analysis requests without blocking, supporting deployments with 15–25 simultaneous users on a single server instance. The framework's simplicity enables rapid integration by third-party developers and security operations teams without specialized infrastructure requirements.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

7. PERFORMANCE EVALUATION

7.1 Evaluation Methodology

The system was evaluated using a purpose-built synthetic behavioral dataset of 10,000 samples designed to provide systematic coverage across four user categories (Human, Basic Bot, Advanced Bot, Human-like Bot) and five operating conditions (clean controlled lab, low network latency, mixed device types, high traffic, adversarial mimicry attacks). Each sample was generated using statistically parameterized distributions with class-specific mean and variance values calibrated to match behavioral profiles documented in the bot detection literature.

The dataset was split 80/20 into training and test subsets using stratified sampling to preserve class balance. Ground truth labels for each synthetic sample were determined by the generation parameters. Model performance was evaluated across six metrics: Accuracy, Precision, Recall, F1-Score, Average Response Latency, and Confidence Score Calibration. All experiments were executed on a standard development workstation (Intel Core i7-11th Gen, 16GB RAM, no GPU) to ensure that reported performance metrics reflect achievable results in typical deployment environments without specialized hardware.

7.2 Classification Performance by User Category

Table 6. Pronunciation Comparison Performance by User Category (analogous to speaker profiles in ASR)

User Category	Precision (%)	Recall (%)	F1 Score	Avg. Confidence	Key Challenge
Human	96.8	95.3	0.960	97.1%	Near-ceiling performance
Basic Bot	98.2	97.6	0.979	98.5%	High-speed signatures distinct
Advanced Bot	94.1	91.8	0.929	93.6%	Mimics natural movement
Human-like Bot	89.7	86.4	0.880	88.2%	Closest to human patterns
Overall (weighted avg)	94.7	92.8	0.937	94.4%	All categories combined

The classification results reveal important security insights. Human-like Bot (F1 = 0.880) presents the greatest classification challenge, as these agents are specifically engineered to mimic the statistical properties of human behavioral distributions. The precision-recall gap for this category (89.7% vs. 86.4%) indicates that some human-like bot interactions produce feature vectors that fall within the learned boundary of human behavior, motivating the sequential LSTM/GRU component that can detect temporal inconsistencies even when static aggregate features appear human-like.

Basic Bot achieves the highest F1 score (0.979) because these agents generate behaviorally extreme feature values — unrealistically high mouse speeds, perfectly uniform click rates, and zero drag actions — that are trivially separable from human behavioral distributions by all five ensemble components simultaneously. The human category achieves high precision (96.8%) with moderate recall (95.3%), reflecting a conservative design choice to minimize false positives that would incorrectly flag legitimate users as bots.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

7.3 System Performance by Operating Condition

Table 7. Detection Accuracy Results by Operating Condition

Data Condition	Accuracy (%)	Precision (%)	Recall (%)	Latency (ms)	System Usability
Clean (controlled lab)	97.3	96.8	95.3	142	Excellent
Low network latency	95.8	94.7	93.1	178	Very Good
Mixed device types	93.2	91.5	90.2	195	Good
High traffic / concurrent users	89.6	88.3	86.7	213	Acceptable
Adversarial (mimicry attacks)	81.4	79.8	77.3	228	Limited — alert issued

The accuracy of 97.3% under clean conditions establishes the system's theoretical performance ceiling under optimal deployment. The graceful degradation under adversarial conditions — from 97.3% (clean) to 81.4% (adversarial mimicry) — follows a predictable pattern that reflects the increasing sophistication of the attack scenario rather than system failure. The adversarial mimicry condition represents a worst-case scenario where bots are specifically optimized to defeat the system's known feature set, representing a threat model that would require continuous model retraining in production deployment to counter.

7.4 Response Latency Analysis

End-to-end response latency is a critical performance metric for bot detection applications deployed in front of interactive web services. The proposed system's worst-case latency of 228ms under high adversarial load conditions comfortably supports real-time deployment in front of web applications without perceptible impact on user experience (human perception of response delay begins at approximately 300–400ms [Nielsen, 1994]).

The dominant latency component in all conditions is the LSTM sequential model inference (typically 60–85ms), with feature extraction (15–25ms), Random Forest inference (8–12ms), and Flask JSON serialization (5–10ms) contributing smaller proportions. This architecture implies that the primary path to further latency reduction is LSTM model compression — pruning, quantization, or knowledge distillation to a smaller student network — or migration to a GPU-accelerated inference backend for high-traffic production deployments.

7.5 Comparative System Evaluation

Table 8. Comprehensive System Comparison with Existing Bot Detection Approaches

Feature / Metric	CAPTCHA	Passive Rule-Based	Standalone ML	Proposed System
Real-time detection	No	Partial	No (batch)	Yes (<228ms worst case)
Multi-feature analysis	No	Limited	Partial	Yes — 8 behavioral features
Bot taxonomy depth	Human/Bot only	Human/Bot only	Human/Bot only	4 categories (Human/Basic/Advanced/Human-like)
Anomaly detection	No	Rule thresholds	Partial	Isolation Forest + One-Class SVM
Sequential modeling	No	No	No	LSTM + GRU ensemble
User experience impact	High disruption	None	None	None (fully passive)
Open source	N/A	No	Partial	Yes (MIT license, Flask)
Detection accuracy (clean)	~60%	~70%	~82%	~97.3% (clean conditions)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Cost	Low / Free	Low	Medium	Free / Self-hosted

8. SECURITY, PRIVACY, AND ETHICAL CONSIDERATIONS

8.1 Data Privacy Architecture

The monitoring of user interaction patterns raises significant privacy considerations. The proposed system implements a privacy-by-design architecture with three core principles: (1) interaction-only capture — the system records only behavioral metadata (movement coordinates, timing, counts) and never captures screen content, keystrokes content, or personal identifiable information; (2) minimal retention — raw interaction events are processed within the sliding window and discarded after feature extraction, with only the aggregate feature vector retained for the session; and (3) configurable audit logging — prediction records can be stored locally in an encrypted database or disabled entirely for zero-retention deployments.

8.2 Compliance Framework

For deployment in Indian digital platforms, the system is designed for compatibility with India's Digital Personal Data Protection Act (DPDP Act, 2023), which requires informed consent for personal data collection, data minimisation, and the right to erasure. The detection module, which involves optional persistent storage of session behavioral profiles, presents a consent prompt on first use and provides a deletion endpoint for complete data erasure. For international deployments requiring GDPR compliance (EU) or CCPA compliance (California), the system's self-hostable architecture enables deployment within compliant infrastructure without transmitting behavioral data to third-party cloud services.

8.3 Ethical Use and Classification Boundary

It is critically important to position the proposed system correctly within the spectrum of security tools. The system is designed and validated as a supplementary security layer — a passive behavioral monitor that augments rather than replaces human security oversight. Classification outputs are intentionally presented as probabilistic confidence scores rather than deterministic verdicts, acknowledging the inherent uncertainty in behavioral classification and preserving human judgment in consequential access control decisions. The four-category taxonomy enables graduated security responses rather than binary block/allow decisions: a 'Basic Bot' classification might trigger rate limiting, while a 'Human' classification with 95% confidence requires no intervention, and an 'Advanced Bot' classification triggers a secondary verification step rather than immediate blocking.

9. CONCLUSION AND FUTURE WORK

9.1 Conclusion

This paper presents the Live Detection of AI-Agents Through Interaction Pattern Analysis system, a comprehensive, open-source, web-accessible platform that integrates multi-feature behavioral data collection, statistical feature extraction, hybrid anomaly detection, sequential temporal modeling, and ensemble classification into a unified real-time analysis pipeline. The system addresses the critical gap between the demand for effective, user-friendly bot prevention and the limitations of intrusive, easily-bypassed traditional mechanisms.

Experimental evaluation on 10,000 synthetic behavioral samples demonstrates that the system achieves a classification accuracy of 97.3% on clean data, F1-scores of 0.880–0.979 across user categories, and sub-228ms end-to-end response latency across all tested conditions. The system's modular microservice architecture enables independent component upgrades, while its open-source, self-hostable design removes both economic and privacy barriers to deployment in security operations, digital platforms, and educational environments.

Comparative evaluation against CAPTCHA, rule-based filtering, and isolated ML approaches confirms that the proposed system is the only evaluated solution to simultaneously offer passive non-intrusive operation, eight-feature multi-modal analysis, four-category user taxonomy, ensemble ML classification with sequential temporal modeling, and a documented REST API — establishing a new performance baseline for behavioral bot detection platforms.

9.2 Key Achievements

- Unified data collection + feature extraction + anomaly detection + sequential analysis + ensemble classification pipeline delivering results in under 228 milliseconds.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- Novel four-category user taxonomy (Human / Basic Bot / Advanced Bot / Human-like Bot) with confidence-scored outputs, enabling graduated security response calibration.
- Ensemble classification combining Random Forest, Isolation Forest, One-Class SVM, LSTM, and GRU achieving 97.3% accuracy on clean behavioral data.
- Privacy-by-design architecture capturing interaction metadata only, with configurable retention and DPDP/GDPR-compatible deployment configurations.
- Open-source Flask REST API enabling third-party integration with enterprise security platforms, SIEM tools, and application security frameworks.

9.3 Future Enhancement Roadmap

9.3.1 Near-Term Enhancements (6–12 months)

- Real Dataset Integration: Replace the synthetic dataset with real-world behavioral data collected through controlled user studies and honeypot deployments to improve model generalization to novel bot variants not represented in the synthetic training distribution.
- Phoneme-Level Equivalent — Micro-gesture Analysis: Integrate sub-event analysis of mouse acceleration profiles, inter-click timing variance, and keystroke pressure (on touch devices) to provide finer-grained behavioral signatures analogous to phoneme-level speech analysis.
- Adaptive Retraining: Implement an online learning mechanism that updates model weights incrementally as new confirmed human/bot interaction samples are collected, enabling the system to adapt to evolving bot strategies without manual retraining cycles.

9.3.2 Medium-Term Enhancements (12–24 months)

- Mobile Platform Support: Extend the system to mobile interaction modalities including touch dynamics, swipe patterns, gyroscope data, and tap pressure measurements, addressing the growing proportion of bot traffic originating from mobile device emulators.
- Federated Learning Architecture: Implement federated model training across multiple deployment sites to improve model diversity and cross-domain generalization without centralizing sensitive behavioral data, addressing enterprise privacy requirements while benefiting from collective intelligence.
- Explainable AI Visualizations: Implement SHAP (SHapley Additive exPlanations) value visualizations over feature contribution scores for each classification decision, providing security operators with interpretable explanations of why a specific user was classified as a bot and which behavioral features triggered the classification.

9.3.3 Long-Term Clinical Translation (24–48 months)

- SIEM Integration via CEF/LEEF: Integrate with Security Information and Event Management systems using standardized log formats to enable seamless transfer of bot detection events, confidence scores, and behavioral anomaly patterns to enterprise security operations centers.
- Randomized Controlled Evaluation: Conduct a prospective study comparing bot infiltration rates and false-positive rates across platforms using the proposed system versus control platforms using CAPTCHA-only protection, following established cybersecurity evaluation protocols to generate evidence for regulatory and enterprise procurement decisions.
- Speaker Equivalent — Multi-User Session Analysis: Add session-level behavioral profiling to detect coordinated bot campaigns where multiple bot instances divide interaction load to stay below per-session detection thresholds, analogous to speaker diarization in speech processing.

REFERENCES

[1] Afanaseva, N. S., & Lozhnikov, P. S. (2022). Bot Detection Using Mouse Movements. IEEE Xplore.

[2] Kolomeets, M., Tushkanova, O., Desnitsky, V., Vitkova, L., & Chechulin, A. (2022). Experimental Evaluation: Can Humans Recognise Social Media Bots? IEEE.

[3] Amshavalli, M., Chandiran, S., Dharshini, R., & Edwin Arul Solomon, A. M. (2023). Adaptive Behavioral Authentication for Bot Detection Using Machine Learning. International Journal of Environmental Sciences.

[4] Mohitkar, C., Kharat, A., Nashirkar, N., Pardeshi, T., & Gaikwad, P. S. (2023). Passive CAPTCHA: AI Driven Bot Detection for Seamless User Experience. IEEE Conference on Dynamics of Systems, Mechanisms and Machines.

[5] Sayyad, G. G., Kadam, S., Kadam, K., Godage, S., & Zagade, R. (2023). Assessment of Bot Detection Approaches Using Behavioral Biometrics and Mouse Dynamics. International Journal on Advanced Computer Engineering and Communication Technology.

[6] von Ahn, L., Blum, M., & Lozhnikov, P. S. (2023). Performance Evaluation of Lightweight Network-Based Bot Detection Using Mouse Movements. Elsevier Engineering Applications of Artificial Intelligence.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

[7] Acien, A., Morales, A., Fierrez, J., & Vera-Rodriguez, R. (2024). Behavioral Biometrics: A Comparison of Keystroke Dynamics and Mouse Trajectories for Bot Detection. Pattern Recognition Journal.

[8] Ferrara, E., Varol, O., Rodriguez, C., & Oppenheimer, D. (2024). Machine Learning-Based Social Media Bot Detection: A Comprehensive Literature Review. Springer Social Network Analysis and Mining.

[9] Pozzana, I., & Ferrara, E. (2024). Measuring Bot and Human Behavioral Dynamics. IEEE Conference on Dynamics of Systems, Mechanisms and Machines.

[10] Acien, A., Morales, A., Fierrez, J., & Vera-Rodriguez, R. (2024). BeCAPTCHA-Mouse: Synthetic Mouse Trajectories and Improved Bot Detection. arXiv preprint.

[11] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation Forest. Proceedings of ICDM 2008, IEEE, pp. 413–422.

[12] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

[13] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078.

[14] Imperva. (2023). Bad Bot Report 2023: The Mounting Threat of Bad Bots. Imperva Research Labs.

[15] Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann Publishers, San Francisco.

Manuscript received: March 2026 | Revised: April 2026 | Accepted: April 2026