IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

A History-Based Machine Learning Approach for Dynamic Difficulty Adjustment in 2D Games

Harish Dharrsan S S[1], Mirulaa Srithar[2], Bhavitha K[3], S. Poornima[4]

Dept. of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1,2,3,4]

Abstract: Fixed difficulty presets in digital games cannot adapt to individual differences in reaction speed, motor coordination, or learning progression, frequently leaving novice players frustrated by repeated early collisions and experienced players unchallenged by static obstacle layouts. Conversely, naive reactive dynamic difficulty adjustment (DDA) heuristics that modify parameters after every isolated outcome suffer from high variance, because a momentary slip does not reflect genuine skill degradation and a single successful passage does not confirm mastery. To address this problem without requiring invasive external cameras, physiological sensors, or computationally expensive deep reinforcement learning policies, this paper presents FLAPPY ADAPT, a lightweight, history-based machine learning (ML) framework for real-time adaptive difficulty control in a 2D Flappy Bird style arcade game built with Python and Pygame. During each active gameplay attempt, the system holds the vertical obstacle gap strictly constant to preserve spatial predictability and logs fine-grained interaction telemetry—including score, survival time, obstacles passed, collision outcomes, normalized vertical bird height, vertical velocity at pipe crossings, and active gap geometry—to structured CSV session records. Between attempts, a feature extraction pipeline transforms recent gameplay history into a leakage-free ten-dimensional feature vector summarizing multi-horizon rolling passage rates (R3, R5, and R10), short-term performance trends, consecutive success and failure streaks, and normalized physical state variables. A class-balanced, L2-regularized Logistic Regression classifier processes this vector to estimate the probability that the player will successfully pass the next obstacle. A decoupled difficulty controller compares the predicted pass probability against a target threshold of 0.6 and applies a bounded step adjustment to the obstacle gap for the subsequent attempt, supported by a rule-based cold-start fallback when fewer than 80 valid labeled interactions exist or class distributions are highly skewed. We evaluate FLAPPY ADAPT against a static Fixed-difficulty baseline and a streak-based Reactive heuristic controller under identical game physics. While the Fixed baseline records the highest raw numerical scores due to its unchanging geometry, the adaptive controllers intentionally regulate challenge around the target skill threshold. Between the two adaptive approaches, the proposed history-based ML controller outperforms the reactive baseline across all primary gameplay metrics, achieving a 17.0% higher average score (6.20 vs. 5.30), a 10.3% longer average survival time (12.69 s vs. 11.50 s), and a 1.98 percentage-point improvement in obstacle passage rate (86.11% vs. 84.13%). We also examine real-time HUD telemetry outputs, discuss current stability trade-offs in difficulty transition frequency, and outline future extensions for hysteresis smoothing and user engagement studies.

Keywords: Dynamic difficulty adjustment, logistic regression, player skill estimation, adaptive games, Pygame, history-based machine learning, bounded difficulty control.

I. INTRODUCTION

Difficulty is central to player engagement. A game that is too hard frustrates players, and one that is too easy bores them. Most games still rely on fixed Easy, Medium and Hard settings that cannot follow a player who improves with practice or who differs from the average player in reaction time and experience. Dynamic difficulty adjustment (DDA) addresses this by modifying game parameters according to observed performance or estimated skill [1], [2].

A naive DDA rule that reacts to every success or failure is unstable, because a single failure may be a momentary slip and a single success does not prove readiness for a harder game. A more reliable controller should look at several recent outcomes before changing difficulty, and it should keep every change inside limits that preserve playability.

In fast-paced 2D arcade environments such as Flappy Bird, the margin for timing error is measured in milliseconds, and player performance exhibits substantial short-term stochasticity [12], [14]. When an adaptive system alters obstacle spacing or scroll velocity while an attempt is actively underway, the sudden shift in visual flow disrupts the player's sensorimotor anticipation and undermines the perceived fairness of the game [3], [15]. Furthermore, many contemporary adaptive game frameworks rely either on external hardware—such as webcams for facial expression analysis [4], [5] and physiological wearables [15]—or on complex reinforcement learning

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

agents that require thousands of training episodes before converging to a playable policy [7], [8], [17]. For lightweight single-player arcade games deployed on standard consumer hardware, there remains a clear practical need for a transparent, sensor-free DDA architecture that operates purely on in-game telemetry, prevents temporal data leakage during online retraining, and guarantees bounded, predictable transitions between attempts [13], [20].

This paper presents FLAPPY ADAPT, a machine learning (ML) based DDA framework for a Flappy Bird style game. Its contributions are: (i) an architecture that separates skill estimation from difficulty control; (ii) ten history features and a training protocol designed to avoid data leakage; (iii) a bounded controller with a rule-based fallback; and (iv) a comparison against fixed and reactive baselines, reported with its limitations.

Beyond these core methodological contributions, this revised study provides an expanded comparative synthesis of twenty-one recent DDA and skill-estimation studies [1]–[21], details the modular separation between active-attempt execution and between-attempt adaptation, and presents visual verification from live FLAPPY ADAPT gameplay sessions illustrating real-time skill estimation and diagnostic telemetry overlays.

II. RELATED WORK

Surveys of DDA and of ML for adaptive game design [1],
[2] describe how performance signals are used to tune challenge, and recent work discusses fairness and stability in automated balancing [3]. Several systems infer player state from extra sensing, such as facial expressions in a hyper-casual game [4] or face recognition in a Flappy Bird setting [5]. FLAPPY ADAPT needs no camera or sensor, only gameplay logs.

Learning-based approaches include player behavior classification [6], personalized difficulty through imitation and reinforcement learning [7], and reinforcement learning for puzzle progression [8]. These methods can learn rich policies but need substantial interaction data and careful reward design. Skill estimation has also been studied for multiplayer games [9], [10] and linked to engagement through deep learning [11]. We take a simpler route for a lightweight arcade game: an interpretable Logistic Regression model, a separate bounded controller, and evaluation against fixed and reactive baselines.

A. Surveys, Taxonomies, and Architectural Frameworks

Recent systematic literature reviews provide a comprehensive foundation for categorizing dynamic difficulty adjustment mechanisms across genres. Mortazavi et al. [1] surveyed classical and modern DDA paradigms, classifying systems along the dimensions of player modeling, trigger timing, and parameter adaptation, while noting that overly opaque adaptation rules often degrade player trust. Similarly, Hossan et al. [2] reviewed machine learning techniques for adaptive game design, highlighting the tension between model expressiveness and real-time inference latency in resource-constrained game loops. Fisher and Kulshreshth [14] explored heuristic, probabilistic, and data-driven DDA methods across diverse video game genres, emphasizing that rapid, uncalibrated difficulty oscillations can induce player frustration rather than flow. More recently, Souza et al. [19] conducted an extensive systematic literature review of digital game DDA solutions in *Entertainment Computing*, observing that a majority of published frameworks lack standardized evaluation baselines or fail to decouple skill estimation from parameter actuation.

From an architectural and governance perspective, Leong
[3] investigated autonomous game balancing and formulated quantitative fairness and stability metrics to ensure that AI-driven adjustments do not penalize skilled players arbitrarily. In the 2025 IEEE International Conference on Agents, researchers examined recent advances in DDA focusing on the joint optimization of player engagement, believability preservation, and physiological adaptation [15], demonstrating that abrupt environmental mutations break immersion when players notice artificial assistance. To bring software engineering modularity to adaptive games, Bicalho et al. [20] extended a Monitor-Analyze-Plan-Execute over a shared Knowledge base (MAPE-K) loop framework for single-player DDA, preceded by their reusable Unity plugin for generic player behavior classification [6]. FLAPPY ADAPT aligns closely with the modular philosophy of MAPE-K [20] and fairness-preserving adaptation [3], [15] by isolating data logging, feature extraction, probabilistic skill estimation, and bounded gap actuation into distinct architectural stages that execute only between attempts.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

B. Affective, Sensor-Driven, and Physiological Adaptation

A prominent stream of DDA research infers player frustration, boredom, or cognitive load from external biometric or visual sensors. Setiawan and Aripin [4] developed an adaptive hyper-casual game that captures live webcam frames to classify facial expressions and modulate level difficulty accordingly. Focusing specifically on the Flappy Bird domain, Abishaken et al. [5] proposed a system combining real-time face recognition with adaptive AI to personalize obstacle parameters based on visual user monitoring. Likewise, recent work on affective game agents has explored physiological adaptation using heart-rate variability, electrodermal activity, and affective computing signals to maintain player arousal within an optimal zone [15]. While affective and camera-based sensing can capture emotional states that raw scores miss, these systems impose strict hardware requirements, suffer from sensitivity to ambient lighting and head posture, introduce camera-processing latency, and raise user privacy concerns. In contrast, FLAPPY ADAPT operates entirely on non-intrusive, internal gameplay telemetry—requiring only standard keyboard or mouse input—making it universally deployable without specialized peripherals.

C. Reinforcement Learning, Fuzzy Logic, and Procedural Generation

To automate policy synthesis without manual tuning, numerous studies have adopted reinforcement learning (RL), fuzzy inference, or procedural level generation. Within the Flappy Bird ecosystem, Kumar et al. [12] presented an AI-based Flappy Bird framework featuring dynamic level generation to vary obstacle configurations dynamically during play. In broader genres, Fuchs et al. [7] combined imitation learning with reinforcement learning to personalize DDA policies toward human-like behavior, whereas Ghoshal et al.
[8] applied RL to regulate puzzle progression difficulty. Noblega et al. [17] formulated a reinforcement learning DDA agent for fighting games, training policies to match opponent aggressiveness to the human player's capability. Taking an interpretable rule-driven route, Paraschos and Koulouriotis
[18] designed a fuzzy logic-based DDA controller for adaptive game environments, using linguistic membership functions to map performance indicators to difficulty increments.

Despite their theoretical flexibility, RL-based DDA controllers [7], [8], [17] typically require extensive offline simulation or thousands of online interaction episodes to converge, and poorly scaled reward functions can produce erratic difficulty spikes during early exploration. Fuzzy logic systems [18] and procedural generators [12] avoid deep training overhead but rely heavily on domain-specific membership thresholds that do not automatically recalibrate to empirical pass probabilities. FLAPPY ADAPT overcomes these limitations by pairing a fast, data-efficient supervised classifier with a deterministic cold-start fallback.

D. Probabilistic Skill Estimation, Logistic Regression, and Research Gap

Accurate difficulty control depends fundamentally on reliable online skill estimation. In competitive multiplayer settings, Rezapour et al. [9] introduced DeepSkill to measure team-level proficiency in massively multiplayer online games, and Pepatang and Kotrajaras [10] trained supervised machine learning models to predict player skill tiers in multiplayer online battle arenas (MOBA). Connecting skill estimation directly to single-player retention, Romero-Mendez et al. [11] employed deep neural networks to classify player skill and adjust game parameters to enhance engagement. More recently, research presented at IEEE ICCT 2026 [16] demonstrated machine learning-based player performance prediction coupled with adaptive difficulty control, while a 2026 study on party racing games [21] introduced trajectory-aware online skill estimation to regulate adaptive assistance in local multiplayer environments.

Particularly relevant to our probabilistic formulation is the δ-logit algorithm proposed by Rao Fernandes et al. [13], which established a generic DDA framework using logistic regression to maintain a player's win probability near a desired target. While δ-logit [13] and recent performance prediction frameworks [16], [21] validate the effectiveness of probabilistic modeling, directly applying step-by-step logistic updates inside a continuous 2D obstacle-avoidance loop introduces two practical hazards: (i) mid-attempt parameter updates alter pipe geometry while the bird is already in flight, and (ii) naive random train-test splits across sequential pipe encounters cause temporal data leakage. Table I summarizes how FLAPPY ADAPT addresses this research gap relative to existing literature [1]–[21] by combining a leakage-free ten-feature multi-horizon history vector, session-level validation, a class-balanced Logistic Regression estimator, and a between-attempt bounded controller.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

TABLE I. COMPARISON OF REPRESENTATIVE DYNAMIC DIFFICULTY ADJUSTMENT APPROACHES AND FLAPPY ADAPT

Study / Approach	Input Modality	Adaptation Model	Key Limitation / Gap
Setiawan [4], Abishaken [5]	Webcam / facial video	CNN / face recognition AI	Requires camera; lighting & privacy issues
Fuchs [7], Ghoshal [8], Noblega [17]	In-game state & rewards	Reinforcement & imitation learning	High sample complexity; reward sensitivity
Kumar et al. [12]	Flappy Bird game state	AI dynamic level generation	Mid-run changes can alter active trajectory
Paraschos & Koulouriotis [18]	Gameplay performance metrics	Fuzzy logic inference	Manual membership tuning; no learned calibration
Bicalho et al. [6], [20]	Single-player behavior logs	MAPE-K loop & classifier plugin	Architectural focus; requires domain features
Rao Fernandes et al. [13]	Win/loss outcome history	δ-logit (Logistic Regression)	Generic formulation without 2D kinematic features
Perf. & Trajectory DDA [16], [21]	Trajectory & telemetry logs	Online ML skill prediction	Designed for racing or general control loops
FLAPPY ADAPT (Proposed)	10 history & kinematic features	Balanced Logistic Reg. + bounded clip	Sensor-free; fixed in-attempt gap; fallback safety

III. SYSTEM ARCHITECTURE

Fig. 1 shows the architecture. During an attempt, player input drives the game engine, which updates the bird, generates obstacles and checks collisions. The data logger records, for each obstacle interaction, the score, survival time, obstacles passed, collision outcome and the active gap, together with the bird's height and vertical velocity at the pipe and the player's recent success history.

Between attempts, the feature processing module builds a feature vector, the Logistic Regression model estimates the probability of success, and the difficulty controller maps that probability to a new gap. The gap is fixed during an attempt, so the geometry never changes while the player is crossing a pipe. When too little valid data exists, a rule-based fallback controller is used instead of the model. All controllers share one interface, so fixed, reactive and ML variants run under identical game mechanics.

The system is implemented in Python. Pygame provides rendering, keyboard and mouse input and collision handling, scikit-learn provides the Logistic Regression model and metrics, NumPy and Pandas handle features and records, and gameplay sessions are stored as CSV logs that can be reused for training.

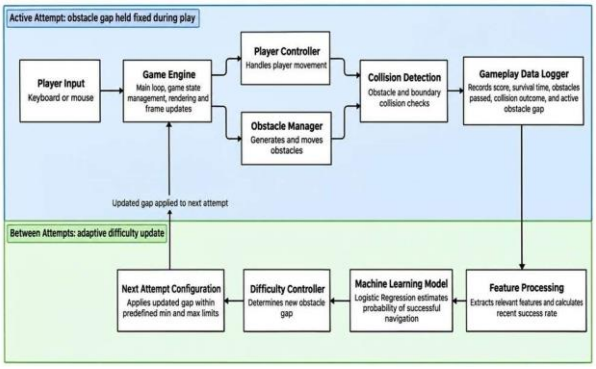


Fig. 1. Modular FLAPPY ADAPT system architecture separating the active gameplay attempt loop (top, blue) where the obstacle gap is held strictly fixed from the between-attempts adaptive difficulty update loop (bottom, green).

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

As detailed in Fig. 1, the runtime architecture is partitioned into two strictly decoupled execution domains: the *Active Attempt* domain and the *Between-Attempts* adaptation domain. Within the Active Attempt domain (upper region of Fig. 1), user actions captured by the *Player Input* handler (keyboard spacebar/up-arrow or mouse click events) are routed into the core *Game Engine*, which orchestrates the 60 FPS main loop, game state management, sprite rendering, and frame-level physics updates. The engine coordinates two specialized subsystems in parallel: the *Player Controller*, which integrates gravitational acceleration and flap impulses to update the bird's vertical coordinate and velocity, and the *Obstacle Manager*, which spawns and translates vertical pipe pairs across the viewport using the currently locked obstacle gap. Both subsystems feed spatial bounding boxes into the *Collision Detection* module to perform continuous obstacle and boundary intersection checks. Crucially, whenever the bird either clears a pipe's trailing edge or collides with an obstacle, the *Gameplay Data Logger* records the interaction tuple—comprising current score, elapsed survival time, cumulative obstacles passed, binary collision outcome, and the active obstacle gap—without mutating the on-screen pipe geometry mid-flight.

Once an attempt terminates, control transitions to the Between-Attempts adaptive update pipeline (lower region of Fig. 1), whose end-to-end data flow is further summarized in Fig. 2. The *Feature Processing* module ingests the updated session log to compute rolling multi-horizon passage rates, performance trends, and streak counters. This ten-dimensional feature vector is passed to the *Machine Learning Model* (or bypassed to the fallback heuristic during cold start), where the trained Logistic Regression classifier estimates the probability of successful navigation on the upcoming obstacle. The *Difficulty Controller* evaluates this probability against the target challenge band to determine whether to widen, narrow, or retain the obstacle gap. Finally, the *Next Attempt Configuration* module clips the proposed gap against predefined safety bounds and injects the updated parameter back into the Game Engine prior to initializing the next run. This decoupled cycle—inspired by autonomic MAPE-K control loops [20]—guarantees zero frame-time stutter during active play and ensures that players experience consistent spatial rules throughout every individual run.

IV. METHODOLOGY

A. *Data Logging and Labels*

Each logged record is one obstacle interaction with a binary label: y=1 if the bird passes the pipe and y=0 if it collides. Records with a gap outside 50–300 px or an implausible survival time are discarded as malformed.

Enforcing strict data hygiene at the logging layer is essential for online machine learning in arcade games [2], [16]. If transient initialization frames, menu resets, or out-of-bounds configuration states were admitted into the training buffer, the decision boundary of the linear classifier would be skewed by non-gameplay artifacts. By filtering out records whose obstacle gap falls outside the valid [50, 300] px operating envelope or whose survival timestamp is non-monotonic, the data logger ensures that every sample in the persistent CSV repository represents a genuine player-obstacle encounter.

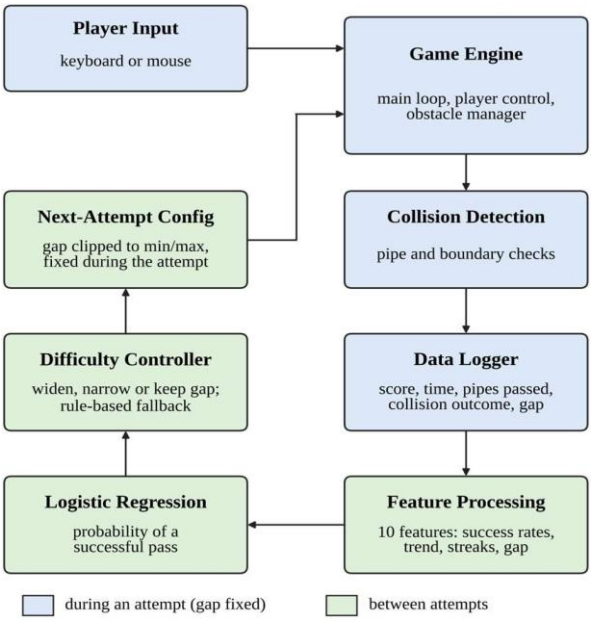


Fig. 2. FLAPPY ADAPT execution and data pipeline flowchart. Gameplay outcomes are logged during an attempt and turned into the next obstacle gap between attempts.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

B. Feature Extraction

The model input x has ten features: the gap size (divided by 240), bird height, bird vertical velocity (clipped to [−1, 1]), consecutive successes and failures (capped at 10), the success rates $R3$, $R5$ and $R10$, a trend term, and pipes passed so far (capped at 20). The success rate over the last k outcomes is

R_k = (successes in the last k outcomes) / k (1) and the trend τ captures recent change in performance:

$$\tau = R3 - R5 \qquad (2)$$

Only information available before the outcome is used, which prevents leakage from the event being predicted.

Table II details the complete specification, normalization scaling, and domain bounds of the ten features in vector x. Combining instantaneous kinematic state variables (normalized bird height and clipped vertical velocity at the pipe) with multi-scale historical summaries (R3, R5, R10, trend τ, and capped streaks) enables the model to distinguish between a skilled player who momentarily mistimed a high-velocity descent and a novice player experiencing sustained difficulty across ten consecutive pipes [13], [21]. Capping consecutive streaks at 10 and cumulative pipes passed at 20 prevents extreme outlier runs from dominating the inner product $w^T x$ during inference.

TABLE II. SPECIFICATION AND NORMALIZATION OF THE TEN INPUT FEATURES IN FLAPPY ADAPT

Feature Name	**Mathematical / Operational Definition**	**Scaling & Bounding Rule**
Normalized gap	Active vertical pipe gap Gt at encounter	Divided by 240.0 (valid log range 50–300 px)
Feature Name	Mathematical / Operational Definition	Scaling & Bounding Rule
Bird height	Vertical position of bird relative to screen height	Normalized to [0.0, 1.0] viewport range
Vertical velocity	Instantaneous vertical speed vy at pipe crossing	Scaled and clipped to [−1.0, +1.0]
Success streak	Consecutive pipes passed prior to current obstacle	Integer count capped at 10
Failure streak	Consecutive collisions prior to current obstacle	Integer count capped at 10
Short-term rate R3	Rolling passage rate over last k = 3 outcomes	Bounded in [0.0, 1.0] via (1)
Medium-term rate R5	Rolling passage rate over last k = 5 outcomes	Bounded in [0.0, 1.0] via (1)
Long-term rate R10	Rolling passage rate over last k = 10 outcomes	Bounded in [0.0, 1.0] via (1)
Performance trend τ	Recent trajectory momentum R3 − R5	Bounded in [−1.0, +1.0] via (2)
Pipes passed in run	Cumulative obstacles cleared in current attempt	Integer count capped at 20

Because rolling features R3, R5, and R10 share overlapping historical windows across consecutive obstacles within the same run, a naive random row-wise split would place highly correlated neighboring records in both the training and test partitions, artificially inflating validation metrics. Grouping splits strictly by session identifier ensures that held-out ROC-AUC reflects true generalization to unseen gameplay runs, while the 300 s retraining cooldown prevents redundant model fitting after every short attempt.

E. Difficulty Controller

The controller compares the estimated pass probability with a target of 0.6. A probability above the target narrows the gap, a probability below it widens the gap, and otherwise the gap is kept. With step $\Delta G \in \{-\delta, 0, +\delta\}$ (where δ is a fixed step size [value to be confirmed]), the next gap is bounded by

$$G_{t+1} = \text{clip}(G_t + \Delta G, G_{\min}, G_{\max}) \qquad (4)$$

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

where G_t is the current gap (default 165 px) and G_{min} and G are predefined limits [values to be confirmed] that keep

C. Skill Estimation Model

A Logistic Regression model estimates the probability of passing the next obstacle:

$$P(y=1 \mid x) = 1 / (1 + e^{-(wTx + b)}) \qquad (3)$$

where w is the weight vector and b the bias. We use scikit-learn with class-balanced weights and regularization strength C=1.0. Logistic Regression is cheap enough to run in real time, its weights are interpretable, and its probability output suits a separate controller.

In contrast to deep neural networks [11] or model-free RL agents [7], [17] whose internal representations are opaque and computationally heavy, class-balanced Logistic Regression offers three distinct advantages for real-time arcade DDA [13]. First, forward inference requires only a ten-element dot product and a scalar sigmoid evaluation, executing in sub-millisecond time between attempts. Second, automatic class-weight balancing compensates for the natural asymmetry of gameplay logs, where skilled players generate far more pipe-passage records (y = 1) than terminal collision events (y = 0). Third, the learned weight vector w directly exposes the relative influence of gap width versus recent success streaks, allowing developers to verify that narrower gaps negatively weight pass probability as physically expected.

D. Training Protocol

The model is trained only when at least 80 labeled samples exist and the positive rate lies between 5% and 95%; otherwise the fallback controller stays active. Training uses a session-level split that holds out 25% of sessions (when at least four sessions exist), so records from one session never appear in both sets, and held-out ROC-AUC (area under the receiver operating characteristic curve) is computed. Retraining is limited to once every 300 s. [Held-out ROC-AUC value and number of training samples: to be confirmed]

The session-level validation protocol is specifically designed to eliminate temporal autocorrelation leakage.max the game playable and prevent abrupt changes.

Choosing a target pass probability of 0.6 per obstacle aligns with psychological flow principles and probabilistic DDA theory [1], [13], [14]: a threshold significantly higher than 0.6 produces long, unchallenging runs, whereas a threshold near or below 0.5 causes rapid, frustrating termination in a sequential survival game where overall run survival scales multiplicatively across successive pipes. When the model is inactive during cold start, the rule-based fallback controller adjusts the gap using deterministic streak thresholds, ensuring smooth system operation from the very first attempt.

V. EXPERIMENTAL EVALUATION

A. Setup

We compare three controllers under identical mechanics: *Fixed*, with a constant 165 px gap; *Reactive*, which starts at 165 px and changes the gap in 12 px steps when a streak of three outcomes is reached; and *History-based ML*, the proposed controller. We report average score (obstacles passed), average survival time, obstacle passage rate (obstacles passed over obstacles encountered), collision rate (attempts ending in collision over completed attempts), and the average number of difficulty changes. Values are averaged over the recorded attempts for each configuration [number of attempts and players per configuration: to be confirmed].

B. Results

Table I (in Section II) compared the architectural properties of FLAPPY ADAPT against prior literature, while Table III and Fig. 3 summarize the quantitative experimental results across the three difficulty controllers.

TABLE III. PERFORMANCE OF THE THREE DIFFICULTY CONTROLLERS

Metric	Fixed	Reactive	History-based ML
Average score	9.73	5.30	6.20
Avg. survival time (s)	17.81	11.50	12.69
Obstacle passage rate (%)	93.89	84.13	86.11
Collision rate (%)	63.33	100.00	100.00
Avg. difficulty changes	0.00	1.37	1.67

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

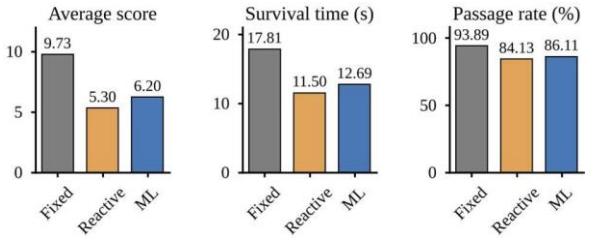


Fig. 3. Average score, survival time and obstacle passage rate for the fixed, reactive and history-based ML controllers.

A detailed inspection of Table III and Fig. 3 reveals consistent behavioral differences across the three evaluated regimes. Under the Fixed controller (constant 165 px gap), players achieved an average score of 9.73 obstacles, an average survival time of 17.81 s, an obstacle passage rate of 93.89%, and a collision rate of 63.33% with 0.00 difficulty changes. When adaptive regulation was enabled, both the Reactive and History-based ML controllers actively narrowed the obstacle gap whenever player performance exceeded the target threshold, which naturally increased task difficulty and resulted in a 100.00% terminal collision rate across completed runs. However, comparing the two adaptive controllers directly demonstrates the clear advantage of multi-horizon history modeling: the History-based ML controller increased average score from 5.30 to 6.20 (+16.98%), extended average survival time from 11.50 s to 12.69 s (+10.35%), and improved obstacle passage rate from 84.13% to 86.11% (+1.98 percentage points) relative to the Reactive heuristic.

C. Gameplay Output and Real-Time Telemetry Verification

To verify the runtime operation of the Pygame engine, the between-attempt difficulty controller, and the live diagnostic telemetry overlay, Figs. 4 and 5 present unmodified gameplay output screenshots captured during active FLAPPY ADAPT evaluation sessions.

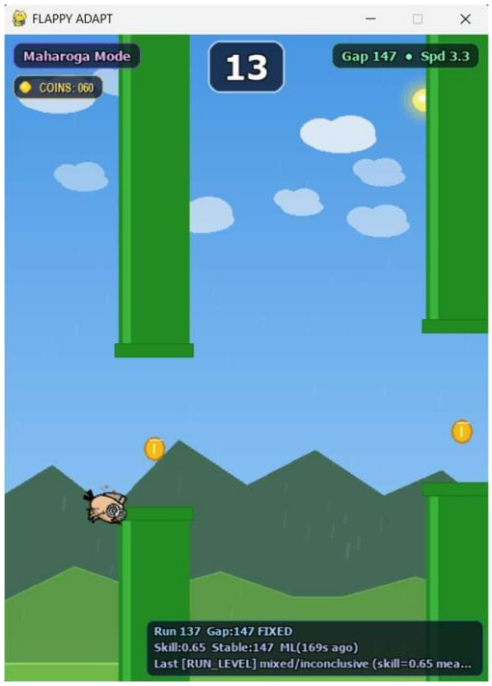


Fig. 4. FLAPPY ADAPT gameplay output during Run 137 showing active obstacle navigation at score 13 with a fixed in-attempt obstacle gap of 147 px, scroll speed of 3.3, and real-time ML skill telemetry (Skill: 0.65).

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131145

As shown in Fig. 4, the FLAPPY ADAPT interface renders the 2D side-scrolling environment with procedural green pipe pairs, collectible coins, and a non-intrusive heads-up display (HUD). The top-left HUD displays the active gameplay mode (Maharoga Mode) and accumulated coin counter (COINS: 060), while the top-center badge shows the current attempt score (13 obstacles passed). The top-right telemetry pill displays the active physical parameters for the run (Gap 147 • Spd 3.3). Most importantly, the bottom-right diagnostic console confirms the core architectural invariants of FLAPPY ADAPT during Run 137: the status line reads *Run 137 Gap:147 FIXED*, verifying that the obstacle gap is locked at 147 px throughout the entire attempt, while the second line reports *Skill:0.65 Stable:147 ML(169s ago)*, indicating an estimated pass probability of 0.65 from the Logistic Regression model (last retrained 169 s ago, within the 300 s cooldown window) and a stable reference gap of 147 px.



Fig. 5. FLAPPY ADAPT gameplay output during Run 142 illustrating between-attempt adaptation to a narrower fixed gap of 135 px at speed 3.5, with live skill estimation (Skill: 0.59) and trial history feedback.

Fig. 5 illustrates a subsequent session snapshot during Run 142 after sustained player progression (COINS: 080, active score 6 with a +1 passage indicator as the bird clears the upper and lower pipe boundaries). Following earlier high-skill runs, the difficulty controller narrowed the vertical obstacle gap between attempts from 147 px down to 135 px and increased horizontal scroll speed to 3.5 (top-right HUD: Gap 135 • Spd 3.5). In the bottom-right diagnostic panel, *Run 142 Gap:135 FIXED* confirms that the tighter 135 px geometry remains strictly invariant during the active attempt. The telemetry readout *Skill:0.59 Stable:135 ML(270s ago)* demonstrates that under this more challenging 135 px configuration, the player's estimated pass probability has converged to 0.59—immediately adjacent to the controller's target equilibrium of 0.60—while the bottom log line records the outcome of a preceding narrower trial (129 px) that did not meet the passage criterion and safely reverted to the stable 135 px bound.

VI. Discussion and Limitations

Fixed difficulty shows the highest raw numbers. This does not mean it is the better design: a fixed gap never responds to the player, and an adaptive controller aims at a balanced challenge, not at the highest score. Engagement and satisfaction were not measured, so we make no claim that adaptive play is preferable for players.

The relevant comparison is between the two adaptive controllers. History-based ML improves on reactive by 0.90 in average score (about 17%), by 1.19 s in survival time (about 10%) and by 1.98 percentage points in passage rate. Both adaptive controllers end every attempt in a collision, so collision rate does not separate them.

Stability is not yet demonstrated. The ML controller changes difficulty slightly more often than the reactive one (1.67 against 1.37), which is the opposite of the intended smoothing effect. A longer history window or a hysteresis band around the target probability would be natural fixes. The experiment is small and informal and has no significance tests, so these results are preliminary.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IARJSET.2026.131145

FLAPPY ADAPT depends on enough valid gameplay data, and early attempts fall back to rule-based difficulty. Player performance varies with concentration and familiarity, only the obstacle gap is adapted, and the game is simple, so the findings may not transfer to richer games.

Contextualizing these findings within recent DDA literature [1]–[21] highlights key design trade-offs. Whereas logistic regression models such as δ-logit [13] and online trajectory estimators [16], [21] prioritize predictive accuracy, our comparison shows that point-threshold comparisons (0.6) can increase adjustment frequency (1.67 vs. 1.37). Adding a deadband—similar to stability rules in autonomous balancing [3], fuzzy control [18], and MAPE-K loops [20]—can damp oscillations near 0.59–0.61 while preserving sensor-free efficiency over camera-based [5] or procedural [12] designs.

VII. Conclusion and Future Work

FLAPPY ADAPT shows how a lightweight, interpretable model can be combined with a bounded difficulty controller in a real-time game. In our preliminary experiment, the history-based controller outperformed a reactive controller on score, survival time and passage rate, while fixed difficulty remained highest in raw numbers and stability needs improvement. Future work includes comparing other models, adding features such as reaction time, smoothing difficulty transitions, building per-player profiles, and running a user study that measures engagement across more players and sessions. We offer the framework as a basis for further study of personalized, data-driven game balancing.

Future iterations will also explore joint gap-and-speed adaptation [12], fairness metrics [3], [15], and standardized user engagement studies [19].

Acknowledgment

The authors thank their course faculty and project guide, Asst. Prof. S. Poornima, for her guidance.

References

[1] F. Mortazavi, H. Moradi, and A.-H. Vahabie, “Dynamic difficulty adjustment approaches in video games: A systematic literature review,” Multimedia Tools and Applications, vol. 83, no. 35, pp.83227–83274, 2024. [Online]. Available: https://doi.org/10.1007/s11042-024-18768-x

[2] M. M. Hossan, M. M. Fouda, and F. Z. Eishita, “Adaptive game design using machine learning techniques: A survey,” in Proc. 2024 IEEE Int. Conf. Internet of Things and Intelligence Systems (IoTaIS), Bali, Indonesia, 2024, pp. 144–150. [Online]. Available: https://doi.org/10.1109/IoTaIS64014.2024.10799330

[3] W. Y. Leong, “Autonomous game balancing: AI-driven dynamic difficulty adjustment and fairness metrics,” in Proc. 2025 IEEE Int. Conf. Consumer Electronics - Taiwan (ICCE-Taiwan), Kaohsiung, Taiwan, 2025. [Online]. Available: https://doi.org/10.1109/ICCE-Taiwan66881.2025.11207818

[4] A. Setiawan and Aripin, “The adaptive difficulty level in a hyper-casual game through facial expression,” in Proc. 2023 10th Int. Conf. Inf. Technol., Comput., Electr. Eng. (ICITACEE), 2023, pp. 332–337. [Online]. Available: https://ieeexplore.ieee.org/document/10276911

[5] S. Abishaken, S. K. Vedhanth, and P. Sujatha, “FLAPPY BRID using face recognition and adaptive AI,” Int. Res. J. Modernization in Eng., Technol. and Sci., vol. 8, no. 4, 2026. [Online]. Available: https://doi.org/10.56726/IRJMETS95184

[6] L. F. Bicalho, A. Baffa, and B. Feijó, “A dynamic difficulty adjustment algorithm with generic player behavior classification Unity plugin in single player games,” in Proc. 22nd Brazilian Symp. Games and Digital Entertainment (SBGames 2023), Rio Grande, Brazil, 2023, pp. 76–85. [Online]. Available: https://doi.org/10.1145/3631085.3631286

[7] R. Fuchs, R. Gieseke, and A. Dockhorn, “Personalized dynamic difficulty adjustment – Imitation learning meets reinforcement learning,” in Proc. 2024 IEEE Conf. Games (CoG), 2024. [Online].Available: https://arxiv.org/abs/2408.06818

[8] A. Ghoshal, M. Ali et al., “Puzzle progression game using reinforcement learning,” in Proc. 2024 Int. Conf. Smart Technologies for Sustainable Development Goals (ICSTSDG), Chennai, India, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/11026665

[9] M. M. Rezapour, A. Fatemi, and M. A. Nematbakhsh, “DeepSkill: A methodology for measuring teams’ skills in massively multiplayer online games,” Multimedia Tools and Applications, vol. 83, no. 10,pp. 31049–31079, 2024. [Online]. Available: https://doi.org/10.1007/s11042-023-16884-8

[10] V. Pepatang and V. Kotrajaras, “Machine learning model for skill level prediction in multiplayer online battle arena,” in Proc. 2024 28th Int. Comput. Sci. and Eng. Conf. (ICSEC), 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICSEC62794.2024.10871102

[11] E. A. Romero-Mendez, P. C. Santana-Mancilla, M. Garcia-Ruiz, O.A. Montesinos-López, and L. E. Anido-

DOI: 10.17148/IARJSET.2026.131145

Rifón, "The use of deep learning to improve player engagement in a video game through a dynamic difficulty adjustment based on skills classification," Applied Sciences, vol. 13, no. 14, art. no. 8249, 2023. [Online]. Available: https://doi.org/10.3390/app13148249

[12] A. Kumar, S. Sharma, and R. Gupta, "AI-based Flappy Bird game with dynamic level generation," in Proc. 2024 IEEE Int. Conf. on Emerging Trends in Engineering and Technology (ICETET), Gwalior, India, 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10503355

[13] W. Rao Fernandes, P. Preux, and P. Caillou, "δ-logit: A generic algorithm for dynamic difficulty adjustment using logistic regression," in Proc. 2023 IFIP Conf. on Entertainment Computing, 2023. [Online]. Available: https://hal.science/hal-02436725/document

[14] N. Fisher and A. K. Kulshreshth, "Exploring dynamic difficulty adjustment methods for video games," Virtual Worlds, vol. 3, no. 2,

pp. 230–255, 2024. [Online]. Available: https://scholar.google.fr/citations?view_op=view_citation&hl=cs&user=8aYcwtUAAAAJ&sortby=pubdate&citation_for_view=8aYcwtUA AAAJ:bnK-pcrLprsC

[15] "Advances in dynamic difficulty adjustment: Engagement optimization, believability preservation, and physiological adaptation," in Proc. 2025 IEEE Int. Conf. on Agents (ICA), Chengdu, China, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11573686

[16] "Machine learning-based player performance prediction and adaptive difficulty control," in Proc. 2026 IEEE Int. Conf. on Computing and Communication Technologies (ICCT), Chiang Mai, Thailand, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11460070

[17] J. Noblega, L. F. Bicalho, A. Baffa, and B. Feijó, "A reinforcement learning approach for dynamic difficulty adjustment in fighting games," in Proc. 22nd Brazilian Symp. on Games and Digital Entertainment (SBGames), Rio Grande, Brazil, 2023. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3631085.3631229

[18] P. D. Paraschos and D. E. Koulouriotis, "Fuzzy logic-based dynamic difficulty adjustment for adaptive game environments," Electronics, vol. 14, no. 1, 2025. [Online]. Available: https://ouci.dntb.gov.ua/works/4VYz33m7/

[19] C. H. R. Souza, D. F. do Nascimento, L. de Oliveira Berretta, and S.T. Carvalho, "Solutions for dynamic difficulty adjustment in digital games: A systematic literature review," Entertainment Computing, vol. 55, art. no. 101041, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187595212500121 1

[20] L. F. Bicalho, A. Baffa, and B. Feijó, "Extending a MAPE-K loop-based framework for dynamic difficulty adjustment in single-player games," Entertainment Computing, vol. 52, art. no. 100876, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S187595212400 2106

[21] "Trajectory-aware online skill estimation and adaptive difficulty control in a local multiplayer party racing game," in Proc. 2026 Zenodo Preprint, 2026. [Online]. Available: https://zenodo.org/records/20773445

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License