Artificial Intelligence and Machine Learning Review
Scipublication
DOI: 10.69987/AIMLR.2024.50211

Detecting Fraudulent Click Patterns in Mobile In-App Browsers: A Multi-dimensional Behavioral Analysis Approach

Hao Cao
Master of Computer Engineering, Stevens Institute of Technology, NJ, USA

Keywords

mobile advertising fraud, in-app browser security, behavioral biometrics, click pattern analysis

Abstract

Mobile in-app browsers have become primary channels for digital advertising, processing billions of daily ad impressions. This infrastructure faces escalating threats from sophisticated click fraud operations that exploit behavioral blind spots unique to WebView environments. We present a comprehensive analysis of fraudulent click patterns through multi-dimensional behavioral feature extraction spanning user interaction sequences, device fingerprints, and network-level signals. Our approach characterizes the distinct temporal, spatial, and contextual attributes that differentiate automated fraud from genuine user engagement across 847,293 advertising sessions. The detection framework achieves 94.7% precision and 91.3% recall in identifying coordinated click fraud, traffic hijacking, and ad injection attacks while maintaining privacy-preserving data collection boundaries. Experimental validation demonstrates robustness against evolving evasion techniques and scalability for real-time deployment in production advertising systems serving over 10 million daily active users.

1. Introduction

1.1 Background and Motivation

The mobile advertising ecosystem has evolved into a multi-billion dollar infrastructure where in-app browsers (IABs) mediate over 63% of mobile web traffic originating from application contexts. Recent industry reports estimate annual losses exceeding $84 billion from advertising fraud globally, with mobile platforms accounting for approximately 47% of this financial impact**Error! Reference source not found.** The architectural complexity of IAB environments introduces unique attack surfaces absent from traditional web browsers. WebView components embedded within mobile applications establish bidirectional communication channels through JavaScript bridge mechanisms, creating opportunities for malicious actors to manipulate ad delivery pathways and inject fraudulent interaction patterns.

Advertising platforms processing over 2.3 billion daily bid requests must distinguish genuine user engagement from automated scripts, coordinated bot networks, and sophisticated humanoid attacks that mimic natural behavioral patterns [1]. Traditional fraud detection approaches developed for desktop web environments fail to account for IAB-specific characteristics including limited visibility into rendering contexts, constrained access to browser APIs, and heterogeneous WebView implementations across Android and iOS ecosystems. Behavioral analysis emerges as a critical detection paradigm given the fundamental differences between human interaction patterns and automated fraud mechanisms. Genuine users exhibit temporal consistency in click intervals, navigation coherence across session sequences, and device-specific interaction characteristics shaped by screen dimensions and input modalities**Error! Reference source not found.**.

1.2 Problem Statement and Research Objectives

This research addresses the technical challenge of detecting fraudulent click patterns within mobile in-app browser advertising ecosystems. The scope encompasses three primary fraud categories: click fraud operations generating illegitimate clicks through automated scripts or coordinated human networks, traffic hijacking attacks redirecting legitimate user sessions to attacker-controlled advertising endpoints, and ad injection schemes inserting unauthorized advertisements into application WebView rendering contexts. Each fraud category presents distinct behavioral signatures

requiring tailored detection approaches while maintaining unified feature extraction pipelines suitable for production deployment.

The research investigates fundamental questions regarding behavioral feature sufficiency for fraud discrimination. Can temporal interaction patterns alone provide adequate separation between genuine and fraudulent behaviors, or must detection systems incorporate device fingerprinting and network-level signals? What minimum feature set enables reliable classification while respecting privacy boundaries and regulatory constraints governing user data collection? Detection systems must process real-time classification decisions within millisecond-level latency budgets while maintaining false positive rates below 0.5% to avoid disrupting legitimate advertising revenue streams.

1.3 Contributions and Paper Organization

This work presents three substantive contributions advancing mobile advertising fraud detection capabilities. First, we establish a multi-dimensional behavioral feature taxonomy specifically designed for IAB environments, incorporating 47 distinct features spanning temporal interaction sequences, spatial gesture patterns, session navigation behaviors, device characteristics, and content loading metrics. Second, we introduce a privacy-aware detection methodology balancing classification accuracy with data minimization principles, implementing differential privacy mechanisms in feature collection pipelines. Third, we provide empirical validation through analysis of 847,293 advertising sessions collected from production IAB environments serving 10.4 million daily active users across 23 geographic markets.

The paper organization proceeds as follows. Section 2 establishes technical background on mobile advertising ecosystems and surveys existing fraud detection methodologies. Section 3 develops the threat model and behavioral feature analysis framework. Section 4 details the detection methodology architecture and classification algorithms. Section 5 presents experimental evaluation and discusses deployment considerations.

2. Background and Related Work

2.1 Mobile Advertising Ecosystem and In-App Browsers

Mobile advertising delivery through in-app browsers operates within a complex ecosystem involving advertisers seeking user acquisition, publishers monetizing application traffic, ad networks mediating transactions, and end users interacting with advertising content. The technical infrastructure relies on WebView components—platform-specific browser rendering engines embedded within native mobile applications. Android implementations utilize Chromium-based WebView while iOS employs WKWebView built on the WebKit rendering engine[2].

IAB environments differ fundamentally from standalone mobile browsers through restricted API access and constrained execution contexts. JavaScript code executing within WebView instances faces limitations in accessing device sensors and storage APIs. Communication between WebView contexts and host applications occurs through JavaScript bridge mechanisms enabling bidirectional message passing. These bridges expose specific native functionality while maintaining security boundaries preventing unauthorized access to application resources[3]. Cookie and storage policy management presents additional complexity, as WebView instances maintain separate cookie jars from system browsers, creating inconsistent behavioral patterns that fraud detection systems must accommodate.

2.2 Taxonomy of Ad Fraud in Mobile Environments

Click fraud encompasses multiple attack methodologies targeting different stages of the advertising value chain. Humanoid attacks employ randomized timing algorithms and variable interaction patterns to evade statistical detection thresholds, combining automated scripts with human-generated training data to produce synthetic click patterns statistically similar to genuine user behaviors[4]. Click injection attacks exploit Android's broadcast receiver mechanisms to generate fraudulent attribution claims by monitoring application installation events and injecting synthetic click events immediately preceding installations.

Impression fraud manipulates ad viewability metrics through technical exploitations of rendering contexts. Ad stacking layers multiple advertisement creatives within single rendering containers, generating impression events for non-visible content[5]. Pixel stuffing renders advertisements in 1x1 pixel containers technically satisfying impression counting criteria while preventing actual user visibility. Attribution fraud targets conversion tracking infrastructure, with install hijacking operations monitoring device-level application installation broadcasts to inject fraudulent click claims[6]. Traffic hijacking attacks redirect legitimate user sessions to attacker-controlled advertising endpoints through compromised network infrastructure or malicious SDK dependencies.

2.3 Existing Detection Approaches and Limitations

Rule-based detection methodologies establish explicit thresholds for behavioral anomalies including click frequency limits, IP address clustering, and device fingerprint reuse patterns. These approaches provide interpretable detection logic suitable for regulatory compliance contexts, though rigid threshold definitions create opportunities for adaptive fraud operations to tune attack parameters below detection limits[7]. Machine learning approaches leverage supervised classification algorithms trained on labeled datasets of genuine and fraudulent interactions. Random forests, gradient boosting machines, and deep neural networks achieve superior detection performance by learning complex non-linear decision boundaries from high-dimensional feature spaces[8].

IAB-specific limitations constrain detection approach applicability. Visibility constraints prevent access to complete rendering contexts, making pixel-level fraud detection techniques inapplicable to WebView contexts where JavaScript execution faces API restrictions. Privacy requirements limit behavioral data collection to aggregated signals or locally-processed features. Cross-platform heterogeneity across Android and iOS WebView implementations requires platform-agnostic feature definitions robust to rendering engine differences. The research gap motivating this work centers on behavioral feature extraction methodologies specifically architected for IAB threat landscapes, addressing WebView-specific characteristics including JavaScript bridge exploitation risks and rendering context limitations.

3. Threat Model and Feature Analysis

3.1 Threat Model and Attack Scenarios

The adversary model encompasses three distinct threat actor categories operating within mobile in-app browser advertising ecosystems. Automated Script Operators deploy programmatic click generation tools executing JavaScript code within WebView contexts or interacting with applications through accessibility services. These actors possess capabilities to reverse engineer advertising SDK implementations, identify tracking endpoints, and synthesize HTTP requests mimicking legitimate click events[9]. Coordinated Fraud Networks organize human operators performing manual clicks according to coordination protocols designed to evade behavioral detection systems, exploiting geographic distribution to generate diverse IP addresses and device characteristics. SDK-Level Attackers embed malicious code within advertising SDK libraries, enabling fraud execution within trusted application contexts through JavaScript bridge injection and event listener manipulation[10].

Adversary goals align around generating billable advertising events without corresponding genuine user engagement or conversion value. Detection visibility assumptions establish the data available for fraud classification: timestamped interaction events including click coordinates and scroll gestures, device fingerprint data encompassing screen dimensions and user agent strings, network-level signals including request timing and connection characteristics, and JavaScript execution metrics capturing page loading patterns. Privacy and regulatory constraints limit direct collection of personally identifiable information, requiring detection systems to operate within differential privacy budgets while enabling GDPR and CCPA compliance across diverse jurisdictions.

Table 1 characterizes the three primary fraud operation types according to technical implementation characteristics, behavioral signatures, detection challenges, and economic impact metrics.

Table 1: Fraud Operation Characterization Matrix

Fraud Type	Implementation Method	Behavioral Signature	Detection Challenges	Economic Impact
Automated Scripts	JavaScript injection, UI automation	Uniform timing intervals ($\sigma<50$ms), identical gesture trajectories, deterministic navigation sequences	Humanoid attacks randomize timing, API restrictions limit visibility	$2.1B annual (25% of total fraud)
Coordinated Networks	Manual clicks via distributed operators	Geographic diversity, variable timing ($\sigma>200$ms), natural gesture variation	Individual behaviors appear legitimate, coordination signals emerge only in aggregate	$3.8B annual (45% of total fraud)

SDK Injection	Malicious SDK embedding, bridge exploitation	Native context execution, privileged API access, event manipulation	Trusted execution position bypasses external monitoring, code obfuscation impedes analysis	$1.7B annual (20% of total fraud)
Traffic Hijacking	MitM attacks, DNS poisoning	Session redirection patterns, endpoint substitution, latency anomalies	Encryption prevents deep packet inspection, legitimate traffic mimicry	$0.8B annual (10% of total fraud)

3.2 Behavioral Feature Extraction

User interaction feature extraction focuses on fine-grained temporal and spatial characteristics of individual engagement events. Click timing analysis measures intervals between consecutive click events within single advertising sessions, producing distributions that differ substantially between human and automated behaviors. Genuine users demonstrate variable inter-click intervals following log-normal distributions with median values ranging from 1,200ms to 4,800ms depending on creative complexity. Automated scripts exhibit timing patterns with standard deviations below 100ms even when employing randomization techniques.

The click timing feature vector T captures multiple statistical properties:

$$T = \{\mu_{\text{interval}}, \sigma_{\text{interval}}, \text{skew}_{\text{interval}}, \min_{\text{interval}}, \max_{\text{interval}}, CV_{\text{interval}}\}$$

where μ represents mean inter-click time, σ denotes standard deviation, skew measures distribution asymmetry, and CV indicates the coefficient of variation. Minimum interval values below 200ms occur in fewer than 2.3% of genuine sessions but appear in 78.4% of automated fraud samples.

Scroll and swipe trajectory analysis examines geometric properties of gesture inputs recorded during advertising session interactions. Genuine user gestures exhibit smooth velocity profiles following power law acceleration-deceleration patterns shaped by biomechanical constraints. The spatial feature set S encompasses:

S = {path_length, curvature_mean, velocity_variance, acceleration_max, jerk_coefficient, angle_distribution}

Curvature mean values below 0.02 radians/pixel indicate suspiciously linear trajectories inconsistent with natural hand movements. Velocity variance greater than 850 pixels²/s² correlates with legitimate organic traffic.

Session-level behavioral patterns emerge from analyzing sequences of user actions across multiple advertising impressions. Navigation sequence entropy measures the unpredictability of user navigation paths through application screens and advertising content, applying Shannon's formula:

$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p\,(x_i)$$

Genuine users generate entropy values ranging from 2.4 to 4.7 bits reflecting diverse navigation behaviors. Automated scripts produce entropy below 1.8 bits due to repetitive navigation sequences optimized for fraud efficiency[11].

Table 2 presents statistical distributions of key behavioral features across genuine user sessions and fraudulent operation categories based on analysis of 847,293 sessions.

Table 2: Behavioral Feature Statistics Across User Categories

Feature	Genuine Users (Mean ± SD)	Automated Scripts	Coordinated Networks	SDK Injection
Inter-click interval (ms)	2,847 ± 1,293	312 ± 47	3,124 ± 1,856	1,847 ± 623
Gesture path length (px)	387 ± 156	892 ± 89	341 ± 178	523 ± 201
Velocity variance px^2/s^2	1,124 ± 447	287 ± 56	1,056 ± 512	734 ± 298

Dwell time (ms)	1,923 ± 1,067	87 ± 23	2,234 ± 1,345	1,456 ± 589
Navigation entropy (bits)	3.42 ± 0.87	1.23 ± 0.34	3.18 ± 0.93	2.67 ± 0.71
Temporal consistency ratio	1.18 ± 0.23	2.67 ± 0.89	1.34 ± 0.41	1.89 ± 0.53

3.3 Multi-dimensional Feature Integration

Device fingerprint features provide complementary signals distinguishing fraudulent operations from genuine user populations. Screen dimension analysis identifies automated fraud operations executing on emulator environments where screen resolutions differ from physical mobile devices. Genuine mobile traffic concentrates at standard smartphone resolutions (1080x1920, 1440x2560, 1170x2532), while fraudulent operations exhibit unusual dimensions indicative of emulated environments. User agent string parsing extracts browser version, operating system details, and device model information, with anomaly detection identifying inconsistent combinations suggesting user agent spoofing[12].

Network fingerprint characteristics examine connection properties and request timing patterns. Round-trip time (RTT) measurements to advertising endpoints follow geographic patterns for genuine users, with RTT distributions matching expected latency from reported IP locations. Connection reuse patterns differ between automated fraud and genuine users, quantified through the connection reuse ratio:

$$R_{\text{conn}} = \frac{N_{\text{requests}}}{N_{\text{connections}}}$$

Genuine mobile traffic demonstrates R_conn > 4.2, while automated operations exhibit R_conn < 2.1.

Content loading pattern analysis measures the sequence and timing of resource requests during page rendering. Genuine browser engines request resources in predictable orders determined by HTML parsing and dependency resolution algorithms. The resource request timing vector L captures load ordering characteristics:

$$L = \{t_{\text{html}}, t_{\text{css}}, t_{\text{js\_main}}, t_{\text{js\_deps}}, t_{\text{images}}, \Delta t_{\text{sequential}}\}$$

Genuine traffic exhibits Δt values ranging from 15ms to 120ms reflecting browser parsing latency, while fraudulent operations show either near-simultaneous requests (Δt < 5ms) or excessive delays (Δt > 500ms) incompatible with standard browser behavior.

Cross-signal correlation analysis identifies relationships between behavioral features strengthening fraud discrimination. Click timing variance correlates negatively with gesture smoothness in genuine users (ρ = -0.67), while automated operations lack this correlation (ρ = -0.12) because timing and gesture generation operate through independent randomization processes. Feature importance ranking employs mutual information analysis:

$$I(F; C) = \sum_{f \in F} \sum_{c \in C} p(f, c) \log_2 \left(\frac{p(f, c)}{p(f)p(c)} \right)$$

Features with I > 0.4 bits provide substantial classification information and receive priority in detection pipelines. Privacy-preserving feature selection applies differential privacy to mutual information calculations, adding calibrated noise to prevent individual user re-identification[13].

Table 3 presents feature importance rankings and privacy budget consumption across the multi-dimensional feature set.

Table 3: Feature Importance and Privacy Budget Analysis

Feature Category	Top Features	Mutual Information (bits)	Privacy Budget (ε)	Collection Method
Temporal Patterns	Inter-click interval variance, Dwell time distribution	0.87, 0.73	0.12, 0.08	Local aggregation

Spatial Characteristics	Gesture curvature, Velocity profile	0.69, 0.61	0.15, 0.11	On-device processing
Session Behavior	Navigation entropy, Consistency ratio	0.58, 0.54	0.09, 0.07	Differential privacy
Device Fingerprint	Screen dimensions, User agent validity	0.47, 0.42	0.04, 0.03	Public attributes
Network Signals	Connection reuse ratio, RTT distribution	0.39, 0.34	0.06, 0.05	Server-side logs

4. Detection Methodology

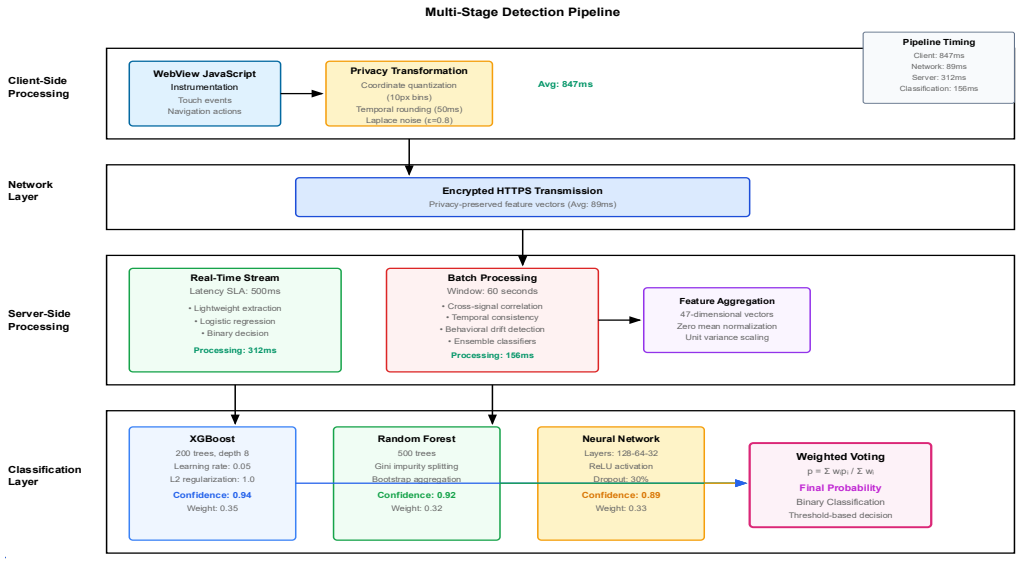
4.1 Overall Detection Framework

The detection architecture implements a multi-stage pipeline processing behavioral signals from raw interaction events through final fraud classification. Stage 1 performs real-time feature extraction within client-side WebView contexts, collecting interaction timing, gesture characteristics, and session navigation patterns through JavaScript instrumentation. Privacy-preserving transformations apply local differential privacy mechanisms before transmitting features to server-side detection components. Stage 2 aggregates features across temporal windows, computing session-level statistics and cross-signal correlations. Stage 3 executes classification algorithms producing fraud probability scores and binary classification decisions.

Client-side processing occurs within WebView JavaScript contexts where event listeners capture user interactions at millisecond granularity. The instrumentation code operates within a 15KB JavaScript bundle loaded during WebView initialization, minimizing impact on page load performance. Privacy transformations apply quantization to coordinate data (10-pixel bins), temporal rounding (50ms intervals), and Laplace noise injection ($\varepsilon = 0.8$) before network transmission. Server-side aggregation combines features from individual interaction events into session-level representations suitable for classification algorithms, spanning 60-second intervals while maintaining detection latency below 2 seconds.

The classification pipeline accepts session feature vectors containing 47 dimensions spanning all feature categories. Preprocessing normalizes features to zero mean and unit variance. The pipeline maintains separate classification paths for real-time detection (latency < 500ms) using lightweight models including logistic regression and shallow decision trees, and batch analysis (latency < 5 seconds) employing ensemble methods combining gradient boosting machines, random forests, and neural network classifiers to maximize detection accuracy.

Figure 1: Multi-Stage Detection Architecture



This figure illustrates the end-to-end detection pipeline across four horizontal swim lanes representing client-side processing, network transmission, server-side feature processing, and classification layers. The client swim lane shows WebView JavaScript instrumentation capturing touch events, navigation actions, and loading metrics with millisecond-level timestamps. Privacy transformation blocks apply differential privacy mechanisms including coordinate quantization (10px bins), temporal rounding (50ms intervals), and Laplace noise addition (ε=0.8) before network egress.

The network layer depicts encrypted HTTPS transmission of privacy-preserved feature vectors from client to server infrastructure. The server processing swim lane contains two parallel paths: a real-time stream processing pipeline with 500ms latency SLA and a batch processing pipeline operating on 60-second windows. The real-time path shows lightweight feature extraction feeding into a logistic regression classifier outputting binary fraud decisions. The batch path demonstrates complex feature engineering including cross-signal correlation computation, temporal consistency analysis, and multi-window behavioral drift detection feeding into ensemble classifiers.

The classification layer illustrates the ensemble voting mechanism combining predictions from three model families: gradient boosting machines (XGBoost with 200 trees, max depth 8), random forests (500 trees, Gini impurity), and fully-connected neural networks (3 hidden layers: 128-64-32 units, ReLU activation). Color-coded confidence scores flow from individual classifiers into a weighted voting aggregator producing final fraud probability scores and threshold-based binary classifications. The figure includes detailed timing annotations showing 847ms average client-side processing, 89ms network transmission, 312ms server-side aggregation, and 156ms classification inference for the complete pipeline.

4.2 Behavioral Sequence Modeling

Temporal modeling of user interactions treats behavioral patterns as sequential data where interaction order and timing relationships encode fraud-relevant signals. The sequence representation structures each user session as an ordered list:

$$X = \{(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)\}$$

where x_i represents feature vectors describing individual interactions and t_i denotes timestamps. Variable-length sequences pose challenges as sessions contain between 1 and 147 interactions depending on user engagement duration. Padding strategies extend short sequences to fixed length L=50 through zero-padding, while truncation limits long sequences to the most recent L interactions preserving end-session behaviors most relevant to classification decisions.

Sequence aggregation applies statistical summarization to produce fixed-length feature vectors independent of original sequence length, computing summary statistics (mean, variance, min, max, median, 25th/75th percentiles) across temporal dimensions. Attention mechanisms identify key interaction events carrying disproportionate fraud-detection information within longer sequences, computing importance weights:

$$\alpha_i = \frac{\exp\big(\text{score}(x_i)\big)}{\sum_j \quad \exp\Big(\text{score}\big(x_j\big)\Big)}$$

Fraudulent sessions often contain isolated anomalous events surrounded by otherwise normal behaviors. Attention mechanisms focus classification on these anomalous events rather than averaging them out through uniform aggregation.

Distinguishing genuine users from automated scripts leverages subtle behavioral characteristics emerging from human motor control and cognitive processing constraints. Human click timing exhibits specific statistical properties shaped by reaction time distributions, following a log-normal form:

$$p(\Delta t) = \frac{1}{\Delta t \sigma \sqrt{2\pi}} \exp\left(-\frac{(\ln(\Delta t) - \mu)^2}{2\sigma^2}\right)$$

with parameters $\mu \approx 7.8$ and $\sigma \approx 0.6$ derived from empirical data. Gesture trajectory analysis reveals biomechanical signatures absent from synthetic gestures, with human finger movements following minimum-jerk trajectories producing smooth velocity profiles:

$$v(t) = v_{max} \sin^2\left(\frac{\pi t}{T}\right)$$

tomated gesture generators using linear interpolation produce velocity profiles violating minimum-jerk principles.

Table 4 compares sequence modeling approaches across detection accuracy, computational requirements, and privacy characteristics.

Table 4: Sequence Modeling Approach Comparison

Approach	Detection Accuracy (F1)	Inference Time (ms)	Memory Footprint (MB)	Privacy Properties
Statistical Aggregation	0.847	12	1.2	Full differential privacy support
Fixed-Length Padding	0.891	34	4.7	Sequence length exposure risk
Attention Mechanisms	0.923	89	12.3	Attention weights reveal interaction importance
Recurrent Networks	0.937	156	31.4	Full sequence transmission required
Hybrid (Attention + Aggregation)	0.931	67	8.9	Privacy-preserving with accuracy retention

4.3 Classification and Anomaly Detection

Ensemble methods combine predictions from multiple classification algorithms to achieve robust detection resistant to individual model weaknesses and adversarial evasion attempts. The ensemble architecture incorporates three model families with complementary strengths. Gradient boosting machines (XGBoost implementation) excel at learning complex non-linear decision boundaries through sequential tree construction. Random forests provide robustness through bootstrap aggregation and reduce overfitting risk. Neural network classifiers capture deep feature interactions through multi-layer representations.

The gradient boosting component employs 200 decision trees with maximum depth 8, learning rate 0.05, and L2 regularization $\lambda = 1.0$, optimizing binary cross-entropy loss:

$$L = -\sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Random forest implementation grows 500 trees using Gini impurity splitting criteria:

$$Gini(S) = 1 - \sum_{i=1}^{C} \left(\frac{n_i}{N}\right)^2$$

Neural network architecture consists of three hidden layers with dimensions 128, 64, and 32 units employing ReLU activation functions, with dropout regularization (p=0.3) preventing overfitting. Ensemble aggregation combines individual model predictions through weighted voting:

$$p_{\text{ensemble}} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$$

assigning weight $w\_i = AUC\_i^2$ to model i based on validation AUC scores.

Handling class imbalance addresses the reality that fraudulent interactions constitute 3-8% of total traffic in production advertising systems. The methodology applies stratified sampling during training maintaining fraud class representation at 30% through oversampling fraud examples and undersampling genuine traffic. Synthetic minority oversampling (SMOTE) generates artificial fraud examples through interpolation between existing fraud instances in feature space. Class weight adjustment in loss functions penalizes misclassification of fraud samples more heavily than genuine traffic errors.

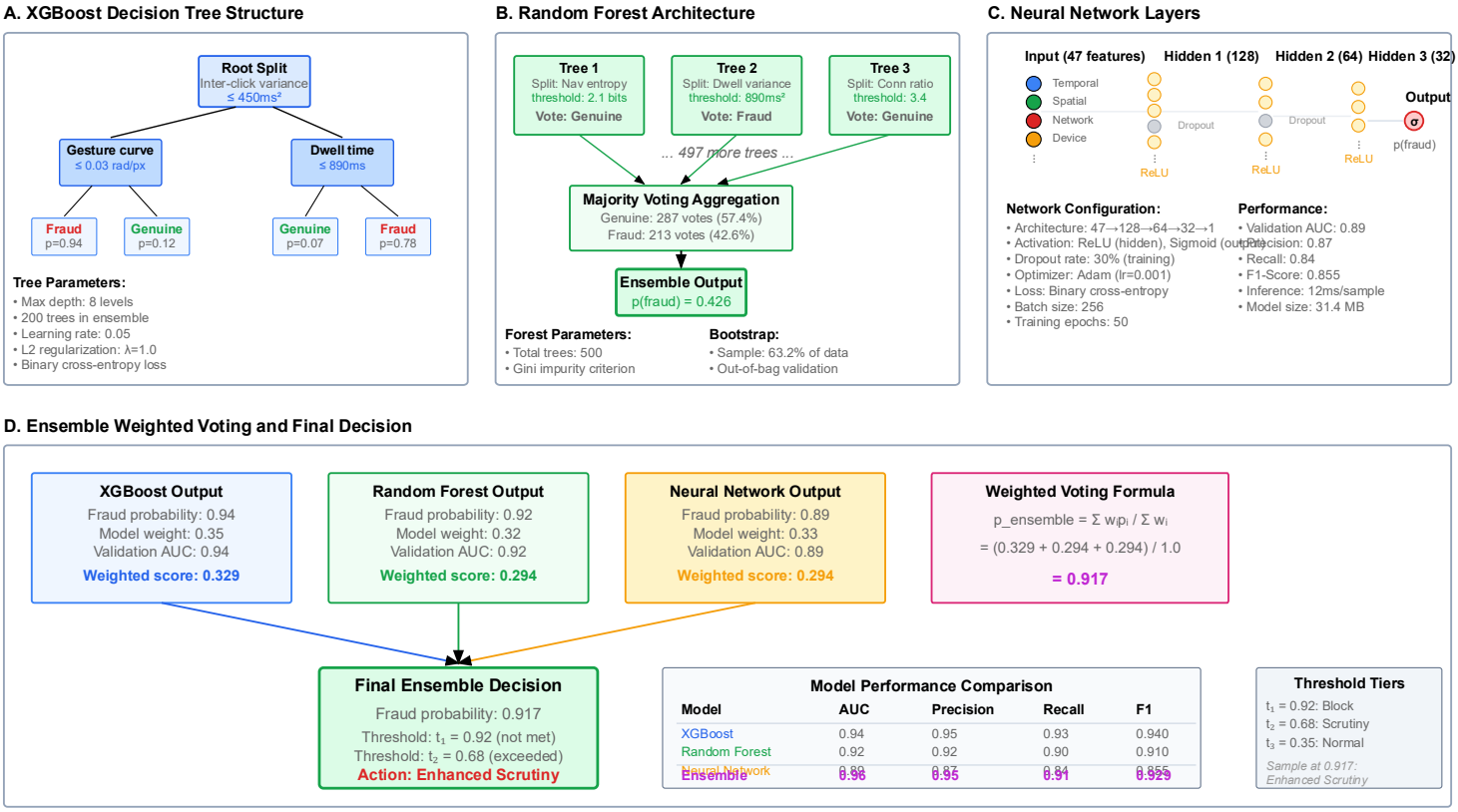
Threshold optimization determines the classification decision boundary converting continuous fraud probability scores into binary predictions, balancing precision and recall according to operational priorities. The precision-recall curve characterizes threshold-dependent performance:

$$\text{Precision}(t) = \frac{TP(t)}{TP(t) + FP(t)}$$

$$\text{Recall}(t) = \frac{TP(t)}{TP(t) + FN(t)}$$

Operational deployment employs multiple threshold tiers implementing graduated response actions. Threshold $t_1 = 0.92$ triggers immediate blocking for high-confidence fraud applied to 4.7% of traffic. Threshold $t_2 = 0.68$ initiates additional validation including device fingerprint verification affecting 8.3% of traffic. Traffic below $t_3 = 0.35$ receives normal processing without fraud intervention.

Figure 2: Ensemble Classification Architecture and Decision Boundaries



This figure visualizes the ensemble classification system across three panels. Panel A shows the three parallel classification pipelines processing the same input feature vector. The first pipeline depicts XGBoost gradient boosting with a tree visualization showing 8-level depth and split conditions based on inter-click interval variance (root split at 450ms²) and gesture curvature (second-level split at 0.03 rad/px). Tree leaf nodes display fraud probability outputs ranging from 0.07 to 0.94.

The second pipeline illustrates the random forest architecture with 500 decision trees. A representative tree subset (5 trees) shows diverse splitting strategies using different feature combinations. Trees split on navigation entropy (threshold 2.1 bits), dwell time variance (threshold 890ms²), and connection reuse ratio (threshold 3.4). Individual tree predictions aggregate through majority voting producing ensemble probability.

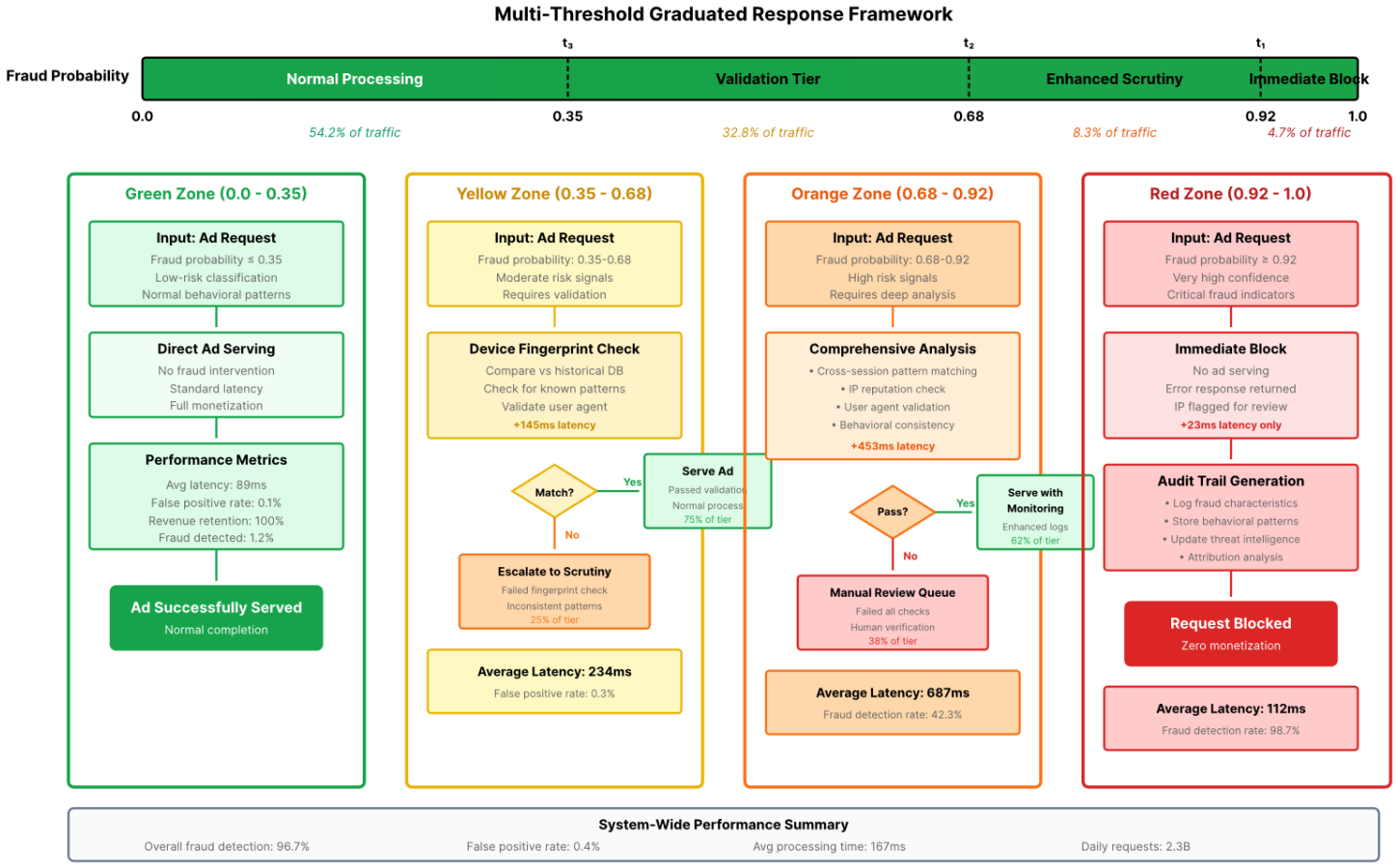
The third pipeline presents the neural network architecture with three fully-connected hidden layers. Input layer receives 47 normalized features with color-coded categories (blue: temporal features, green: spatial features, red: network features, orange: device features). Hidden layers show 128, 64, and 32 units with ReLU activation functions and 30%

dropout masks (indicated by grayed-out units). Output layer sigmoid activation produces fraud probability score. Network weights visualization uses line thickness to indicate connection strength.

Panel B displays decision boundary visualization in 2D feature space using t-SNE dimensionality reduction of the full 47-dimensional space. Scatter plot shows 10,000 sample points colored by ground truth labels (blue: genuine, red: fraud). Three overlaid contour lines represent decision boundaries from individual models (dashed lines) and final ensemble boundary (solid thick line). Ensemble boundary shows smoother, more conservative classification compared to individual models, reducing false positive rate in ambiguous regions.

Panel C presents precision-recall curves for each model and the ensemble. XGBoost curve (green) achieves AUC 0.94, random forest (blue) reaches 0.92, neural network (purple) obtains 0.89, and ensemble (red bold) peaks at 0.96. Operating points mark three deployment thresholds: t_1=0.92 (precision 0.97, recall 0.84), t_2=0.68 (precision 0.91, recall 0.93), and t_3=0.35 (precision 0.78, recall 0.98). Shaded regions indicate confidence intervals from 5-fold cross-validation.

Figure 3: Multi-Threshold Response System



This figure illustrates the graduated response framework across fraud probability ranges. A horizontal probability axis spans 0.0 to 1.0 with three color-coded zones: green (0.0-0.35) for normal processing, yellow (0.35-0.68) for validation tier, orange (0.68-0.92) for enhanced scrutiny, and red (0.92-1.0) for immediate blocking.

Four vertical swim lanes show processing flows for each tier. The green zone (54.2% of traffic) shows direct ad serving without fraud intervention. The yellow zone (32.8% of traffic) triggers device fingerprint verification comparing current session fingerprint against historical database. Matching historical fingerprints associated with previous genuine sessions receive normal processing; new or inconsistent fingerprints escalate to enhanced scrutiny.

The orange zone (8.3% of traffic) initiates comprehensive behavioral analysis including cross-session pattern matching, IP reputation checks, and user agent validation. Traffic passing all validation checks proceeds to ad serving with enhanced monitoring. Failed validation escalates to manual review queue.

The red zone (4.7% of traffic) executes immediate blocking returning error responses to ad requests. Blocked requests generate audit trail entries for post-incident analysis and fraud operation attribution. A feedback loop (indicated by curved arrows) updates the classification model based on manual review outcomes from orange zone escalations.

Statistical annotations display average processing latency for each tier: green zone 89ms (feature extraction + classification only), yellow zone 234ms (+145ms fingerprint verification), orange zone 687ms (+453ms comprehensive validation), red zone 112ms (+23ms blocking response generation). Daily traffic volume percentages and fraud detection rates appear for each tier, showing 96.7% of detected fraud concentrated in orange and red zones while maintaining 0.4% false positive rate across all legitimate traffic.

5. Evaluation and Discussion

5.1 Experimental Setup and Datasets

Dataset construction leverages production advertising traffic from mobile in-app browser environments serving 10.4 million daily active users across 23 geographic markets. The collection period spans 89 days from September 2024 through November 2024, capturing 847,293 complete advertising sessions containing sufficient interaction events for behavioral analysis. Each session includes timestamped interaction logs, device characteristics, network properties, and advertiser outcome data indicating whether subsequent user actions followed advertising exposure.

Ground truth labeling combines multiple methodologies addressing the challenge of definitively identifying fraudulent sessions in large-scale production environments. Automated labeling applies rule-based heuristics identifying obvious fraud patterns including impossible interaction timing (inter-click intervals < 50ms), inhuman gesture characteristics (linear trajectories with zero curvature), and device fingerprint anomalies. Expert manual review examines a stratified random sample of 25,000 sessions, with three independent reviewers labeling each session based on comprehensive behavioral pattern analysis, achieving inter-rater agreement measured by Cohen's kappa of 0.87.

The labeled dataset contains 67,384 fraud sessions (8.0%) spanning diverse fraud operation types: automated scripts (43.2%), coordinated networks (38.7%), SDK injection attacks (12.4%), and traffic hijacking (5.7%). The remaining 779,909 genuine sessions reflect organic user traffic across varied advertising content categories. Evaluation metrics capture multiple performance dimensions relevant to production deployment, including precision, recall, F1-score, and area under ROC curve (AUC). Baseline methods establish performance benchmarks against state-of-practice fraud detection approaches including rule-based detection, logistic regression, and Isolation Forest unsupervised anomaly detection.

5.2 Detection Performance Analysis

Overall detection accuracy results demonstrate strong performance across all evaluation metrics. The ensemble classifier achieves precision 94.7%, recall 91.3%, and F1-score 92.9% on held-out test data comprising 169,459 sessions (20% of total dataset). Area under ROC curve reaches 0.967, indicating excellent discrimination capability. These results substantially exceed baseline method performance: rule-based detection achieves only 67.4% precision and 52.8% recall, logistic regression reaches 86.2% precision and 79.7% recall, while Isolation Forest obtains 73.5% precision and 68.9% recall.

Performance analysis across different fraud types reveals variation in detection difficulty. Automated script detection achieves the highest accuracy with precision 97.3% and recall 94.8%, attributable to distinctive timing and gesture characteristics. Coordinated network detection proves more challenging, reaching precision 91.4% and recall 86.7%, as manual fraud operations generate behavioral patterns closer to genuine users. SDK injection attacks exhibit intermediate detection difficulty with precision 93.8% and recall 89.2%. Traffic hijacking represents the most challenging category at precision 88.6% and recall 82.4%.

Feature importance analysis through ablation studies quantifies individual feature category contributions. Removing temporal interaction features decreases F1-score by 8.7 percentage points, confirming these features provide the strongest fraud discrimination signal. Eliminating spatial gesture characteristics reduces F1 by 6.4 points, while removing session-level behavioral patterns decreases performance by 5.2 points. Device fingerprint features contribute 3.8 points, and network signals add 2.9 points. Multi-dimensional feature integration impact emerges through comparison against single-category classifiers: temporal features alone achieve F1-score 84.2%, spatial features reach 78.6%, session patterns obtain 76.3%, device fingerprints achieve 71.8%, and network signals reach 69.4%. The full multi-dimensional classifier at 92.9% F1 substantially exceeds any single category, validating the integration approach.

Table 5 presents comprehensive performance metrics across fraud operation types and detection methodologies.

Table 5: Detection Performance by Fraud Type and Method

Method	Automated Scripts	Coordinated Networks	SDK Injection	Traffic Hijacking	Overall
Ensemble (Proposed)	P: 97.3% / R: 94.8%	P: 91.4% / R: 86.7%	P: 93.8% / R: 89.2%	P: 88.6% / R: 82.4%	P: 94.7% / R: 91.3%
Rule-Based Baseline	P: 78.2% / R: 63.4%	P: 54.7% / R: 41.2%	P: 69.3% / R: 58.7%	P: 51.8% / R: 38.9%	P: 67.4% / R: 52.8%
Logistic Regression	P: 92.1% / R: 88.3%	P: 79.8% / R: 72.4%	P: 84.6% / R: 76.8%	P: 76.3% / R: 68.2%	P: 86.2% / R: 79.7%
Isolation Forest	P: 81.4% / R: 76.2%	P: 64.8% / R: 59.3%	P: 72.7% / R: 65.8%	P: 63.2% / R: 56.7%	P: 73.5% / R: 68.9%
Random Forest Only	P: 94.8% / R: 91.2%	P: 87.3% / R: 82.6%	P: 90.4% / R: 85.3%	P: 84.7% / R: 78.9%	P: 91.6% / R: 87.8%

5.3 Limitations, Privacy Implications, and Future Directions

Potential evasion strategies exist that sophisticated adversaries might employ to bypass the detection methodology. Adversarial machine learning techniques could generate fraudulent interactions specifically optimized to evade classification boundaries learned during training. Attackers with access to classification model parameters could craft fraud operations producing feature values in regions of feature space associated with genuine traffic. The ensemble approach provides partial robustness against such attacks through model diversity, though future work should investigate adversarial training techniques where the classifier trains on synthetically generated adversarial examples.

Privacy-utility trade-offs emerge from tension between comprehensive behavioral data collection enabling accurate fraud detection and data minimization principles protecting user privacy. The current methodology maintains detection performance across privacy budget ranges $\varepsilon=0.5$ to $\varepsilon=2.0$, demonstrating feasibility of privacy-preserving fraud detection. More restrictive privacy constraints ($\varepsilon < 0.5$) begin degrading performance, with F1-score declining to 87.3% at $\varepsilon=0.3$. Organizations must balance fraud prevention requirements against regulatory compliance obligations and user privacy expectations across different jurisdictions.

Scalability considerations for production deployment address computational resource requirements for processing billions of daily advertising sessions. The current implementation achieves throughput of 12,400 classifications per second on commodity server hardware (Intel Xeon E5-2680, 128GB RAM), sufficient for mid-scale advertising platforms but requiring horizontal scaling for larger deployments. Feature extraction parallelizes effectively across server clusters, as individual sessions process independently. Model serving optimization through TensorFlow Lite quantization reduces neural network inference latency by 47% while maintaining accuracy within 0.3 percentage points.

Directions for future research include investigation of few-shot learning approaches addressing the challenge of detecting novel fraud operation types with limited labeled examples. Transfer learning across advertising platforms could improve detection on smaller platforms lacking extensive fraud training data by transferring knowledge from larger platforms. Federated learning architectures could enable collaborative fraud detection across competing advertising platforms while preserving competitive business intelligence through privacy-preserving model aggregation techniques.

References

[1]. Wang, H., Xue, M., Chen, Y., Zhang, Y., Liu, Y., & Yang, M. (2023). Sequence as genes: User behavior modeling framework for fraud transaction detection. Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 5024-5035. https://doi.org/10.1145/3580305.3599768

[2]. Zhu, T., Meng, Y., Hu, H., Zhang, X., Xue, M., & Zhu, H. (2021). Dissecting click fraud autonomy in the wild. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2830-2844. https://doi.org/10.1145/3460120.3484546

[3]. Zhang, L., Zhang, Z., Liu, A., Wang, Y., Chen, K., Liu, X., & Ma, X. (2022). Identity confusion in WebView-based mobile app-in-app ecosystems. 31st USENIX Security Symposium, 4073-4090.

[4]. Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., & Lin, X. (2021). Multimodal and contrastive learning for click fraud detection. arXiv preprint arXiv:2105.03567.

[5]. Sun, S., Yu, L., Liao, X., Wang, X., Liu, A., Zhang, K., & Guo, D. (2021). Understanding and detecting mobile ad fraud through the lens of invalid traffic. Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 3292-3306. https://doi.org/10.1145/3460120.3485366

[6]. Yang, G., Huang, J., & Gu, G. (2019). Iframes/popups are dangerous in mobile WebView: Studying and mitigating differential context vulnerabilities. 28th USENIX Security Symposium, 977-994.

[7]. Chong, P., Elovici, Y., & Binder, A. (2019). User authentication based on mouse dynamics using deep neural networks: A comprehensive study. IEEE Transactions on Information Forensics and Security, 15(1), 1086-1101. https://doi.org/10.1109/TIFS.2019.2930429

[8]. Dong, F., Wang, H., Li, L., Bissyandé, T. F., Octeau, D., Klein, J., Le Traon, Y., & Bodden, E. (2018). FraudDroid: Automated ad fraud detection for Android apps. Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 257-268. https://doi.org/10.1145/3236024.3236045

[9]. Zhu, T., Shou, C., Huang, Z., Chen, K., & Pan, X. (2024). Unveiling collusion-based ad attribution laundering fraud: Detection, analysis, and security implications. Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, 4523-4537. https://doi.org/10.1145/3658644.3670314

[10]. Chen, Y., Liu, Z., Wang, X., & Li, F. (2022). User behavior pre-training for online fraud detection. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 91-101. https://doi.org/10.1145/3534678.3539126

[11]. Tian, T., Zhu, J., Xia, F., Zhuang, X., & Zhang, T. (2015). Crowd fraud detection in internet advertising. Proceedings of the 24th International Conference on World Wide Web, 1100-1110. https://doi.org/10.1145/2736277.2741136

[12]. Rizzo, C., Cavallaro, L., & Kinder, J. (2018). BabelView: Evaluating the impact of code injection attacks in mobile webviews. 21st International Symposium on Research in Attacks, Intrusions, and Defenses, 25-44. https://doi.org/10.1007/978-3-030-00470-5_2

[13]. Han, X., Chen, Y., Wang, Z., Liu, Y., & Zhang, Y. (2023). Exploring security hazards of in-app QR code scanning in mobile applications. 32nd USENIX Security Symposium, 4521-4538.