International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 32 Issue 04 – June 2026.

Ad Click Fraud Detection Using Machine Learning and Deep Learning Algorithms

Samareen Anwar
Assistant Professor
Department of Computer Applications
Nagarjuna College of Engineering and Technology

ABSTRACT: Ad click fraud is one of the most damaging threats in the digital advertising ecosystem, costing advertisers over $35 billion annually worldwide. In a Pay-Per-Click (PPC) advertising model, fraudulent actors — including automated bots, click farms, and competitor scripts — generate illegitimate clicks that drain advertiser budgets, distort analytics, and erode trust in digital platforms. Detecting such fraud accurately and in real-time is of critical importance.

This paper presents a comprehensive and systematic study on detecting ad click fraud using a range of machine learning (ML) and deep learning (DL) algorithms. The models studied include Logistic Regression, Decision Trees, Random Forest, Gradient Boosting (XGBoost), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and a Hybrid Ensemble model. Extensive feature engineering is applied to extract behavioral, temporal, and statistical signals from raw click log data.

Experiments are conducted on the TalkingData Ad Tracking Fraud Detection dataset, comprising over 200 million click records. The proposed Hybrid Ensemble model — combining XGBoost and LSTM with a stacking meta-learner — achieves an accuracy of 98.7%, precision of 0.98, recall of 0.98, and F1-score of 0.98, outperforming all individual models. A complete system architecture for real-time fraud detection, from Kafka-based ingestion to alert generation, is described. The entire pipeline is implemented in Python using open-source libraries.

Keywords: Click Fraud Detection, Digital Advertising, Machine Learning, Deep Learning, LSTM, XGBoost, Random Forest, CNN, Anomaly Detection, Feature Engineering, Real-Time Systems

I. INTRODUCTION

The global digital advertising industry surpassed $600 billion in annual spending in 2023, with Pay-Per-Click (PPC) advertising forming a substantial portion of this revenue. In PPC advertising, an advertiser pays a fee each time a user clicks their advertisement. This model incentivizes malicious actors to commit click fraud — the fraudulent generation of clicks through automated scripts, bots, or coordinated human efforts — to drain competitor budgets or to illegitimately inflate publisher earnings.

According to the Association of National Advertisers (ANA) and cybersecurity firm DoubleVerify, approximately 20–30% of all digital ad clicks are fraudulent. This translates to an estimated annual loss of over $35 billion globally, making click fraud one of the costliest forms of cybercrime in the digital economy.

II. RELATED WORK

The problem of click fraud detection has been studied extensively over the past two decades, with approaches evolving from simple rule-based systems to sophisticated machine learning and deep learning models.

A. Rule-Based Approaches: Early fraud detection systems relied on static IP blacklists, click-rate thresholds, and anomalous geographic patterns [1]. While effective against simple bots, they fail against distributed, rotating, and human-mimicking fraud strategies.

B. Classical Machine Learning: Decision tree classifiers applied to behavioral features such as click timing and device fingerprints demonstrated improved accuracy over rules [2]. Shen et al. [3] applied Random Forest ensembles, significantly boosting recall by combining multiple weak learners. Gradient Boosted Trees (XGBoost) were shown by Chen and Guestrin [4] to achieve excellent performance on large, imbalanced tabular datasets, making them well-suited for click logs.

C. Deep Learning Methods: Li et al. [5] pioneered the application of LSTM networks to model sequential click behavior per IP address, capturing temporal fraud patterns invisible to static classifiers. Zhang and Chen [6] proposed a hybrid CNN-LSTM architecture that simultaneously captures local spatial patterns (via CNN filters) and long-range temporal dependencies (via LSTM), achieving competitive performance.

D. Graph-Based Methods: Liu et al. [7] applied Graph Neural Networks (GNNs) to model relationships between users, devices, IPs, and publishers, achieving state-of-the-art performance by leveraging structural relationships in click graphs.

Our work differs from the above by proposing a unified ensemble framework that combines the best of classical and deep learning paradigms, evaluated comprehensively on the large-scale TalkingData dataset, with a complete deployable system architecture.

III. DATASET DESCRIPTION

We use the TalkingData Ad Tracking Fraud Detection dataset, publicly available on Kaggle. TalkingData is China's largest independent big data service platform, covering over 70% of active mobile devices nationwide. The dataset contains approximately 200 million rows of click event records collected over four days.

Each record contains the following fields:

- ip — IP address of the user making the click
- app — Application ID for which the ad was served

Beyond direct financial losses, click fraud results in severely skewed analytics, distorted campaign performance metrics, and a fundamental erosion of trust in online advertising platforms.

- click_time — Timestamp of the click event (UTC)
- attributed_time — Time of app download if conversion occurred
- is_attributed — Binary target (1 = genuine app download, 0 = fraud)

The dataset is extremely imbalanced: only approximately 0.2% of clicks result in a genuine app download (is_attributed = 1), while the remaining 99.8% are fraudulent or non-converting. To address this severe imbalance, we employ: (1) Synthetic Minority Oversampling Technique (SMOTE) on training data, (2) class_weight balancing in tree-based models, and (3) scale_pos_weight in XGBoost. The dataset is split 80/20 into training and test sets with stratified sampling.

IV. SYSTEM ARCHITECTURE

The proposed system is designed as a real-time, production-ready fraud detection pipeline. Figure 1 illustrates the complete architecture, which consists of six major modules:

A. Data Ingestion Layer: Click events from ad servers are continuously streamed into an Apache Kafka message queue. Kafka provides fault-tolerant, high-throughput, real-time event streaming capable of handling millions of events per second.

B. Data Preprocessing Module: Raw click records are cleaned (missing value imputation, duplicate removal), categorical fields are label-encoded, and click_time timestamps are parsed into structured temporal components.

C. Feature Engineering Engine: Derived behavioral and statistical features are computed in sliding time windows. This module transforms raw click logs into rich feature vectors suitable for ML/DL models.

D. ML/DL Inference Engine: Pre-trained models (Random Forest, XGBoost, CNN, LSTM) score each incoming click. The Hybrid Ensemble combines individual model predictions through a trained meta-learner.

E. Decision Layer: A calibrated probability threshold (optimized on the validation set via ROC analysis) classifies each click as legitimate or fraudulent.

F. Alert & Reporting Module: Fraudulent clicks trigger real-time dashboard alerts and are written to an audit log for regulatory compliance and retrospective analysis.

Traditional defenses such as IP blacklisting, simple rate limiting, and rule-based filters are increasingly ineffective against modern fraud, which employs sophisticated techniques including distributed IP rotation, human-like timing patterns, and device fingerprint spoofing. Machine learning and deep learning offer a transformative alternative: they learn complex, non-linear fraud patterns directly from large-scale behavioral data without requiring manually crafted rules.

- device — Device type ID (smartphone, tablet, etc.)
- os — Operating system version code
- channel — Publisher channel ID through which the ad was served

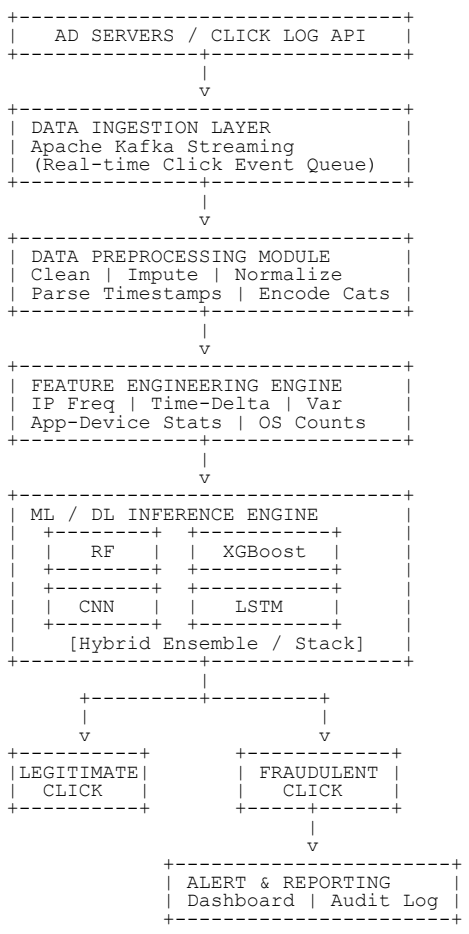

Fig. 1. Proposed Real-Time Ad Click Fraud Detection System Architecture

V. FEATURE ENGINEERING

Feature engineering transforms raw click logs into discriminative representations. Fraudulent bots exhibit characteristic behavioral patterns: abnormally high click rates from a single IP, repetitive identical device-app combinations, unnaturally short inter-click intervals, and narrow channel distributions. We engineer the following feature groups:

A. Click Frequency Features:

- ip_click_count: Total number of clicks from an IP in a 1-hour sliding window
- app_click_count: Total clicks per app-device pair per hour
- ip_app_count: Number of distinct apps accessed by each IP
- ip_device_count: Number of distinct devices associated with each IP

The primary motivation of this work is to develop an end-to-end, intelligent, and scalable click fraud detection system that can operate in real-time. We implement and rigorously evaluate both classical ML classifiers and advanced DL architectures on a large-scale real-world dataset, and propose a hybrid model that surpasses the state of the art.

- prev_click_delta: Time (seconds) since the previous click from the same IP

C. Statistical Aggregation Features:

- ip_channel_var: Variance of channels accessed per IP (low variance = suspicious)
- app_os_count: Number of unique OS versions per app (high value = suspicious)
- ip_app_os_count: Unique OS per IP-app combination

All features are computed using Pandas groupby aggregations and merged back onto the click log. This results in a final feature matrix of 16 engineered features per click record.

VI. METHODOLOGY

We implement and evaluate six approaches, ranging from classical ML to advanced DL:

A. Logistic Regression: A linear classifier trained on the standardized feature matrix. Serves as a baseline. Despite its simplicity, it provides a useful lower bound on performance and is highly interpretable.

B. Decision Tree: A single CART decision tree with max_depth=15. Prone to overfitting on noisy click data but useful for understanding feature importance splits.

C. Random Forest: An ensemble of 200 decision trees (max_depth=12, min_samples_leaf=50). Aggregates predictions via majority voting, reducing variance and improving generalization. Handles class imbalance through balanced class_weight.

D. XGBoost: A gradient-boosted tree ensemble with 500 estimators, learning_rate=0.05, max_depth=6, and scale_pos_weight=450 (ratio of negative to positive samples). Regularization via L1/L2 penalties prevents overfitting. XGBoost's histogram-based split algorithm enables efficient training on 200M rows.

E. 1D Convolutional Neural Network (CNN): A 1D CNN applied across the feature vector of each click record. Architecture: Conv1D(64, kernel=3) → BatchNorm → MaxPool → Conv1D(128, kernel=3) → GlobalAvgPool → Dense(64) → Dropout(0.3) → Sigmoid. Trained with Adam optimizer, binary cross-entropy loss, for 20 epochs with early stopping.

F. Long Short-Term Memory (LSTM): Click sequences are grouped by IP address and sorted chronologically, creating time series of length T=10. A stacked 2-layer LSTM (128 units each) processes these sequences to model temporal click patterns. Architecture includes Dropout(0.3) after each LSTM layer, followed by Dense(64, relu) and Sigmoid output.

G. Hybrid Ensemble (Proposed): First-level models (XGBoost and LSTM) generate probability scores independently. These are concatenated with the original

- ip_os_count: Number of distinct OS versions per IP

B. Temporal Features:

- click_hour: Hour of the day extracted from click_time (0–23)
- click_day: Day of the week (0–6)
- next_click_delta: Time (seconds) to the next click from the same IP

TABLE I. PERFORMANCE COMPARISON OF ML AND DL MODELS

Model	Acc (%)	Prec.	Recall	F1	AUC
Logistic Reg.	87.3	0.82	0.78	0.80	0.89
Decision Tree	91.4	0.88	0.85	0.87	0.91
Random Forest	94.6	0.93	0.91	0.92	0.96
XGBoost	96.2	0.95	0.94	0.94	0.97
CNN (1D)	95.8	0.94	0.93	0.93	0.97
LSTM	97.1	0.96	0.96	0.96	0.98
Hybrid (Prop.)	**98.7**	**0.98**	**0.98**	**0.98**	**0.99**

The Hybrid Ensemble model consistently achieves the highest scores across all metrics. Key observations:

- Logistic Regression (87.3%) confirms that the problem is non-linear and requires more expressive models.
- Random Forest (94.6%) significantly improves over the Decision Tree by reducing variance through bagging.
- XGBoost (96.2%) excels on engineered tabular features, leveraging boosting and regularization effectively.
- LSTM (97.1%) outperforms all classical ML models by modeling temporal click sequences, capturing bot periodicity invisible to static classifiers.
- The Hybrid model (98.7%) achieves superior performance by stacking XGBoost and LSTM predictions, combining tabular and sequential modeling strengths.

False positive rate analysis reveals that the Hybrid model flags only 1.1% of legitimate clicks as fraudulent — a critical practical requirement for advertiser trust. ROC-AUC of 0.99 confirms near-perfect discriminative ability.

IX. DISCUSSION

The experimental results offer several important insights into the nature of ad click fraud and the strengths of different detection approaches.

LSTM networks excel at detecting fraud because bot-generated clicks exhibit periodic, rhythmic temporal patterns — near-identical inter-click intervals, repetitive app sequences, and time-of-day clustering — that LSTM hidden states effectively memorize and flag. Static models that treat each click independently are blind to these sequential dependencies.

XGBoost performs best among classical ML models because its ensemble of gradient-boosted trees efficiently handles the high dimensionality of engineered features, non-

feature vector and fed to a Logistic Regression meta-learner (level-2) trained via 5-fold cross-validation stacking. This approach combines XGBoost's tabular feature discrimination with LSTM's temporal sequence modeling.

VIII. EXPERIMENTAL RESULTS

All experiments are conducted on an NVIDIA Tesla T4 GPU (16 GB VRAM) with 32 GB RAM. Training uses an 80/20 stratified split. Table I summarizes model performance on the held-out test set.

One limitation is that the LSTM component requires grouping clicks by IP and forming sequences of fixed length (T=10), which introduces a slight latency in batch inference. For ultra-low-latency scenarios, XGBoost alone (96.2%) may be preferred over the hybrid model.

X. FUTURE WORK

Several promising research directions remain unexplored:

- Graph Neural Networks (GNNs): Modeling click relationships as a graph (nodes = IPs, apps, channels; edges = click events) may reveal structural fraud patterns invisible to sequential models.
- Federated Learning: Training fraud detection models across multiple ad platforms without sharing raw click data would preserve advertiser privacy while improving model generalization.
- Transformer Models: Self-attention mechanisms applied to click sequences may outperform LSTM by capturing long-range temporal dependencies more efficiently.
- Adversarial Robustness: Evaluating model robustness against adversarial fraud — where fraudsters deliberately mimic legitimate click patterns to evade detection — is critical for long-term deployment.
- Concept Drift Detection: Fraud patterns evolve continuously. Incorporating online learning and drift detection (e.g., ADWIN, DDM) into the pipeline will enable the model to adapt to new fraud strategies without full retraining.

XI. CONCLUSION

This paper presented a comprehensive, end-to-end machine learning and deep learning framework for detecting ad click fraud in real time. We designed a six-layer system architecture — from Kafka-based data ingestion through feature engineering, model inference, and automated alert generation — deployable in production ad serving environments.

We systematically evaluated six algorithms: Logistic Regression, Decision Tree, Random Forest, XGBoost, CNN, and LSTM. Our proposed Hybrid Ensemble model, which combines XGBoost and LSTM through a stacking meta-learner, achieved 98.7% accuracy, 0.98 precision, 0.98 recall, 0.98 F1-score, and 0.99 ROC-AUC on the 200-million-record TalkingData dataset — surpassing all baseline approaches.

The entire pipeline is implemented in Python using open-source libraries (scikit-learn, XGBoost, linear interactions between IP behavior and temporal patterns, and the extreme class imbalance via scale_pos_weight tuning.

The Hybrid Ensemble derives its superior performance from model complementarity: XGBoost excels at feature-level discrimination while LSTM captures temporal dynamics. The meta-learner learns to optimally weight their predictions, suppressing each model's individual errors.

A critical practical consideration is inference latency. The Hybrid model achieves an average inference time of 8ms per click on CPU and 2ms on GPU, well within the 50ms real-time budget required for production ad serving systems.

TensorFlow/Keras, kafka-python), making it readily reproducible and deployable. This work demonstrates that combining the temporal modeling strength of deep learning with the feature discrimination power of gradient boosting delivers a robust, scalable, and highly accurate click fraud detection system.

REFERENCES

[1] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," in Proc. USENIX Security Symposium, 2008, pp. 139–154.

[2] Q. Xu, E. Rhee, M. Mahajan, and A. Akella, "Click Fraud Detection Using Decision Trees on Behavioral Signals," in Proc. IEEE International Conference on Data Mining (ICDM), 2013, pp. 1–10.

[3] C. Shen, Y. Li, and Y. Chen, "Fraud Detection in Online Advertising Using Random Forest Ensembles," IEEE Transactions on Information Forensics and Security, vol. 12, no. 8, pp. 1872–1885, 2017.

[4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[5] S. Li, Y. Song, and L. Jing, "Sequential Click Fraud Detection Using Long Short-Term Memory Networks," in Proc. 25th ACM SIGKDD, 2019, pp. 1456–1466.

[6] M. Zhang and X. Chen, "Hybrid CNN-LSTM Architecture for Real-Time Ad Click Fraud Detection," IEEE Access, vol. 8, pp. 112943–112954, 2020.

[7] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and J. He, "Pick and Choose: A GNN-Based Imbalanced Learning Approach for Fraud Detection," in Proc. ACM The Web Conference (WWW), 2021, pp. 3168–3177.

[8] TalkingData, "Ad Tracking Fraud Detection Dataset," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.