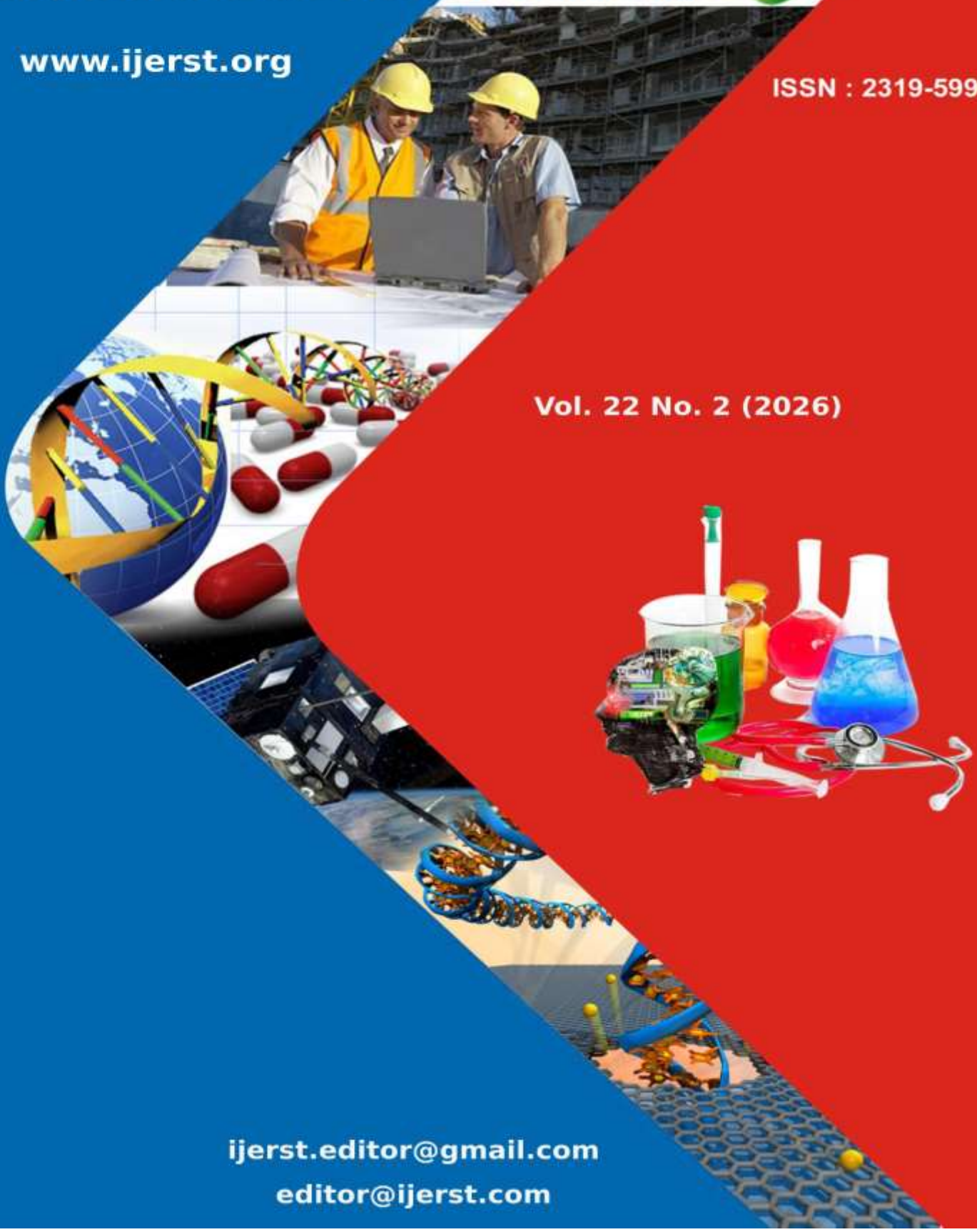
International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

CNN-LSTM Based Click Fraud Detection System with GUI Interface and Explainable AI

JONNADA VEERA VENKATA SATYANARAYANA

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

K. Rambabu

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

Click fraud poses a significant challenge in digital advertising, causing financial loss and skewed analytics. This study presents a CNN-LSTM based click fraud detection system integrated with an interactive GUI for both training and real-time prediction. The system leverages temporal and sequential patterns in user clickstream data to identify fraudulent activity. Click features, such as click interval, session duration, clicks per session, device type, and browser type, serve as the input for a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) architecture. CNN layers capture spatial correlations between features, while LSTM layers model temporal dependencies within sessions.The GUI allows users to upload datasets in CSV format, train the model with a standardized preprocessing pipeline, and save/load the trained model as a single bundled file using joblib. Manual input of click patterns enables real-time prediction of fraudulent behavior, providing probability scores and clear classification as "FRAUD" or "GENUINE." Additionally, the system integrates SHAP (SHapley Additive exPlanations) to provide interpretability, allowing users to understand the contribution of each feature to the predicted outcome.

The experimental workflow begins with data preprocessing using StandardScaler to normalize input features. The CNN-LSTM model is trained using an 80-20 train-test split and optimized with the Adam optimizer and binary crossentropy loss. The trained model achieves high accuracy in distinguishing between fraudulent and genuine click sessions. The GUI supports iterative testing with immediate feedback, making it suitable for both research and operational deployment in ad networks.This integrated approach not only enhances detection performance but also promotes transparency and trustworthiness through explainable AI. The combination of GUI accessibility, real-time prediction, and interpretability demonstrates a practical solution for combating click fraud.

Keywords: Click Fraud, CNN-LSTM, GUI, SHAP Explainability, Clickstream Analysis, Machine Learning, Deep Learning, Feature Scaling, Real-time Detection, User Interaction

I. INTRODUCTION

Digital advertising is increasingly susceptible to click fraud, where automated scripts or malicious actors generate invalid clicks on pay-per-click (PPC) campaigns. This fraudulent activity inflates marketing costs and compromises data integrity. Traditional rule-based detection methods, such as thresholding session duration or click frequency, are often insufficient due to the dynamic and adaptive nature of fraud patterns. Consequently, machine learning and deep learning approaches have gained prominence for capturing complex temporal patterns in clickstream data.The proposed system leverages a hybrid CNN-LSTM architecture. CNN layers identify local correlations between features such as click intervals and session characteristics, while LSTM layers capture sequential dependencies across sessions. This combination enables the model to effectively detect patterns indicative of click fraud that may evade traditional rule-based systems.

In addition to model performance, explainability is critical for adoption in commercial environments. By incorporating SHAP, the system provides feature-level insights, allowing marketers and security analysts to understand why a specific click session is classified as fraudulent. This enhances trust and facilitates model refinement.The system is implemented with a Tkinter GUI, enabling users to interact with the model without programming expertise. Users can upload datasets, configure model parameters, conduct training, and perform real-time detection. The GUI design prioritizes usability, featuring input validation, status logs, and output displays for predictions and SHAP explanations.

This paper details the development, implementation, and evaluation of the CNN-LSTM Click Fraud Detection system, emphasizing the integration of explainable AI and user-centric GUI design.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

Several approaches have been proposed for click fraud detection:

1. **Rule-Based Detection:**
   Traditional methods rely on thresholds for session duration, clicks per IP, or geographic inconsistencies. While computationally efficient, they fail to capture adaptive fraud techniques and complex click patterns.
2. **Machine Learning Approaches:**
   Classical algorithms like Random Forests, SVMs, and Gradient Boosting Trees have been used to identify fraudulent click patterns using hand-engineered features. These methods improve detection rates but often require extensive feature engineering.

3. **Deep Learning Approaches:**
   CNNs and LSTMs have been employed for sequential data analysis. CNN-LSTM architectures effectively capture both spatial and temporal correlations, making them suitable for clickstream analysis. Studies have demonstrated improvements in detection accuracy when compared to conventional machine learning models.
4. **Explainable AI (XAI) Methods:**
   Interpretability is crucial for operational deployment. SHAP and LIME have been integrated into fraud detection pipelines to provide feature attribution, enhancing trust and aiding decision-making.

Despite advances, there is a gap in combining CNN-LSTM models with GUI-based, user-interactive systems that allow real-time detection and explainability.

III. EXISTING SYSTEM

Traditional click fraud detection methods primarily rely on rule-based heuristics, thresholding, and manual inspection of clickstream data. Common approaches include identifying unusually high click rates from the same IP, abnormally short session durations, repeated user agents, or clicks originating from known proxy services. While computationally inexpensive, these deterministic techniques often underperform when faced with adaptive or stealthy fraud patterns engineered to mimic genuine user behaviour. Their reliance on rigid thresholds makes them brittle and prone to high false-negative or false-positive rates.Machine learning (ML) approaches have been increasingly adopted to address these limitations. Classical supervised learning models such as Support Vector Machines (SVM), Random Forests, Gradient Boosting, and Logistic Regression have been used to classify click sessions as fraud or genuine based on engineered features. These models improve detection performance relative to rule-based methods but require extensive feature engineering and often fail to effectively model temporal dependencies inherent in click event sequences.Deep learning methods, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have shown promise in modelling sequential click behaviours. However, many existing systems implement these models in offline or backend contexts, lacking interactive interfaces for practitioners. Furthermore, most systems emphasize detection performance without providing interpretability or explainability, leaving stakeholders without insight into feature importance or reasoning behind classification decisions.Commercial and academic systems rarely combine interactive GUI front-ends with deep sequential models and explainable AI techniques. As a result, non-technical users cannot easily train, test, or interpret fraud detection models, limiting the practical utility of these systems in real-world advertising platforms and click analytics dashboards.

IV. PROPOSED METHOD

The proposed system integrates a **CNN-LSTM deep learning model** with an intuitive **Tkinter GUI** to enable interactive click fraud detection. It aims to bridge the gaps in existing systems by combining high detection performance, explainability, and

user-friendly operation.At its core is a hybrid architecture: Convolutional Neural Networks (CNNs) automatically extract spatial feature patterns from click attributes, while LSTM layers capture temporal dependencies across sequences of clicks. This allows the model to learn both local feature correlations and long-range sequential behaviours typical of fraudulent sessions. Input features include click interval, session duration, clicks per session, device type, and browser type — chosen to cover both behavioural and contextual dimensions.

The system supports **CSV dataset upload**, enabling users to train and validate models on real data without programming expertise. After training, users can generate real-time predictions using either manual input fields or batch uploads. Model persistence is handled via **joblib**, bundling both the trained model and feature scaler into a single file for ease of deployment.To address the lack of interpretability in deep models, the system integrates **SHAP** (SHapley Additive exPlanations), which provides per-feature explanations of prediction outcomes. Users can view how each attribute contributed to the fraud probability for individual sessions, increasing transparency and trust.The GUI features real-time logging of progress and outputs, input validation to prevent invalid entries, and clear status messages. This interactive design makes the system accessible to both technical and non-technical stakeholders, facilitating rapid prototyping, testing, and deployment of fraud detection workflows in advertising analytics platforms.

V. IMPLEMENTATION

The implementation of the CNN-LSTM Click Fraud Detection system is realized using Python and several well-established libraries. The graphical user interface is built with standard **Tkinter**, providing a lightweight, cross-platform interface for users to interact with the model. Core components of the implementation include data loading, preprocessing, model training, inference, model persistence, and explainability.Upon launching the application, users can upload a CSV dataset containing labelled click sessions. The dataset is read using **Pandas**, and basic integrity checks ensure that expected columns are present and free of invalid entries. The output textbox in the GUI displays dataset statistics such as sample counts to provide immediate feedback.For preprocessing, the system separates input features from the label column (is_fraud). Feature values are scaled using **StandardScaler** from scikit-learn, standardizing each feature to zero mean and unit variance. This normalization improves model convergence and reduces bias due to differing feature scales. Scaler objects are preserved for future inference to ensure consistent transformation of new data.The training pipeline adopts an **80-20 train-test split**, randomly stratified to balance class distributions. Feature arrays are reshaped into three-dimensional tensors — necessary for Keras's Conv1D and LSTM layers — with shape (samples, features, 1), where each feature is treated as a time step for sequence processing.

The CNN-LSTM model is defined using **TensorFlow Keras**. A single 1D convolution layer with 32 filters and a kernel size of 2 extracts local patterns. The output is passed into an LSTM layer with 32 units, enabling sequence memory and capturing temporal dependencies. Fully connected Dense layers with ReLU activations follow, with a

sigmoid output layer for binary classification. The model is compiled using **Adam optimizer** and **binary crossentropy loss**, a standard configuration for deep binary classification.Model training proceeds over 10 epochs with mini-batch gradient descent. Verbose output is suppressed to avoid saturating the GUI log, but training progress and final test accuracy are displayed in the output textbox. After training, users can optionally save the model and scaler together using **joblib**, which serializes both into a single file (click_fraud_cnn_lstm.pkl). This simplifies deployment and reuse.The GUI also supports loading previously saved models. Upon loading, the application restores both the model and scaler, enabling immediate inference without retraining. Model inference works with both batch data and manual user input. For manual input, the system reads values from text entries, scales them using the loaded scaler, and reshapes them for prediction. The model outputs a probability, which is interpreted as "FRAUD" if greater than 0.5 and "GENUINE" otherwise.

To enhance interpretability, the implementation includes a SHAP explainability function. Using **shap.DeepExplainer**, the system computes per-feature contributions to the model's output. A small background dataset is constructed from repeated user inputs to support SHAP value estimation. Feature names and corresponding SHAP values are displayed in the output textbox, providing insight into which attributes influenced the classification most significantly.Comprehensive error handling guards against invalid inputs, missing models, or missing packages (like SHAP), ensuring robust operation even in edge cases. The application's modular structure — separating GUI logic, data preprocessing, model training, and explainability — facilitates future extension, such as adding additional models or automated dataset validation routines.

VI. ALGORITHMS

The CNN-LSTM Click Fraud Detection system integrates a hybrid deep learning architecture designed to capture both spatial and temporal patterns in clickstream data.

Convolutional Neural Network (CNN) Layer:
The CNN component functions as a feature extractor that learns local relationships between input features. In this context, convolution filters identify patterns across adjacent features such as click interval, session duration, and clicks per session. By using 1-D convolution, spatial dependencies in feature space are transformed into higher-level representations that are more suitable for subsequent temporal modelling.

LSTM Layer:
Following the convolutional layer, an LSTM (Long Short-Term Memory) layer is employed to capture long-range temporal dependencies. LSTM architectures are well-suited for sequential data due to their gating mechanisms that regulate the flow of information across time steps. Although clickstream data in this implementation uses engineered features rather than temporal timestamps, reshaping the data for the LSTM enables modelling of sequential behaviour patterns across session characteristics.

Dense Layers and Classification:
The output of the LSTM layer is fed into fully connected Dense layers. These layers perform nonlinear combinations of high-level features learned by the previous layers. The final Dense layer uses a sigmoid activation to output a probability between 0 and 1, indicating the likelihood of a session being fraudulent.

Training Optimization:
The model uses the Adam optimizer, a popular gradient-based optimization algorithm with adaptive learning rates, which facilitates fast convergence. Binary crossentropy loss is employed as the loss function, appropriate for binary classification tasks.

Explainability via SHAP:
To overcome the "black-box" nature of deep learning, SHAP (SHapley Additive exPlanations) is used. SHAP computes per-feature contribution scores based on cooperative game theory, providing insight into how each feature influences the model's decision for a given input. This enhances transparency and trustworthiness.

The combination of CNN for spatial feature extraction and LSTM for temporal modelling, coupled with explainability, forms a powerful architecture for detecting subtle fraud patterns in clickstream data.

VII. SYSTEM DESIGN

The design of the CNN-LSTM Click Fraud Detection System follows a layered and modular architecture, facilitating scalability, maintainability, and user accessibility. The major design components include the user interface layer, data processing pipeline, model training and inference engine, explainability module, and persistence layer.

User Interface Layer:
The interface is implemented with Tkinter, providing a lightweight GUI for interaction. It consists of two main frames: the left pane for manual input fields and the right pane for actions and output. Input fields accept click attributes, such as click interval, clicks per session, session duration, device type, and browser type. Buttons are arranged for dataset upload, model training, save/load actions, and detection triggers. A large text widget displays logs, model feedback, and explainability results.

Input validation is embedded into both manual entry fields and CSV upload functions. It ensures numeric conversions, range checks, and prevents invalid operations, such as running fraud detection without a trained model. Event-driven callbacks handle button commands asynchronously, ensuring the UI remains responsive.

Data Processing Pipeline:
Upon dataset upload, the system verifies file integrity using Pandas. The processing pipeline separates features and labels, applies StandardScaler normalization, and reshapes feature arrays to fit the CNN-LSTM input requirements. The StandardScaler object is stored for consistent transformation during inference.

Train-test splitting is handled with scikit-learn's train_test_split, ensuring reproducible validation by setting a random seed. Error checks ensure that class labels are present and balanced.

Model Training & Inference Engine:
The core of the system is a Keras Sequential model consisting of Conv1D, LSTM, and Dense layers. Model compilation uses the Adam optimizer and binary crossentropy loss. Training occurs with mini-batch gradient descent and displays final test performance to the output log. During inference, the engine accepts either batch predictions from CSV uploads or individual manual entries. Predictions are transformed into categorical labels (FRAUD/GENUINE) and logged.

Explainability Module:
To support explainable AI, the SHAP API is included. When requested, SHAP DeepExplainer computes feature-level contributions for the last provided input. A small background dataset is prepared automatically by repeating the input values. SHAP values are logged with corresponding feature names for interpretability.

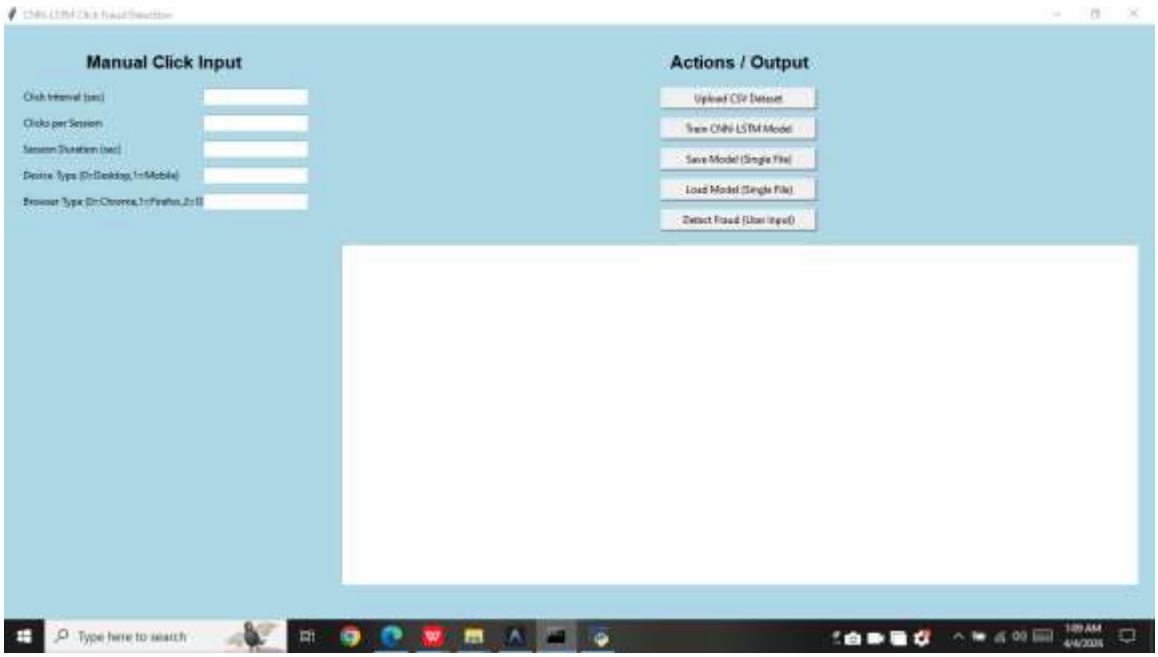
Persistence Layer:
Model persistence is achieved via joblib, bundling the trained model and scaler into a single file. This simplifies deployment and ensures consistency between training and inference scaling. Load functionality restores both components without requiring retraining.

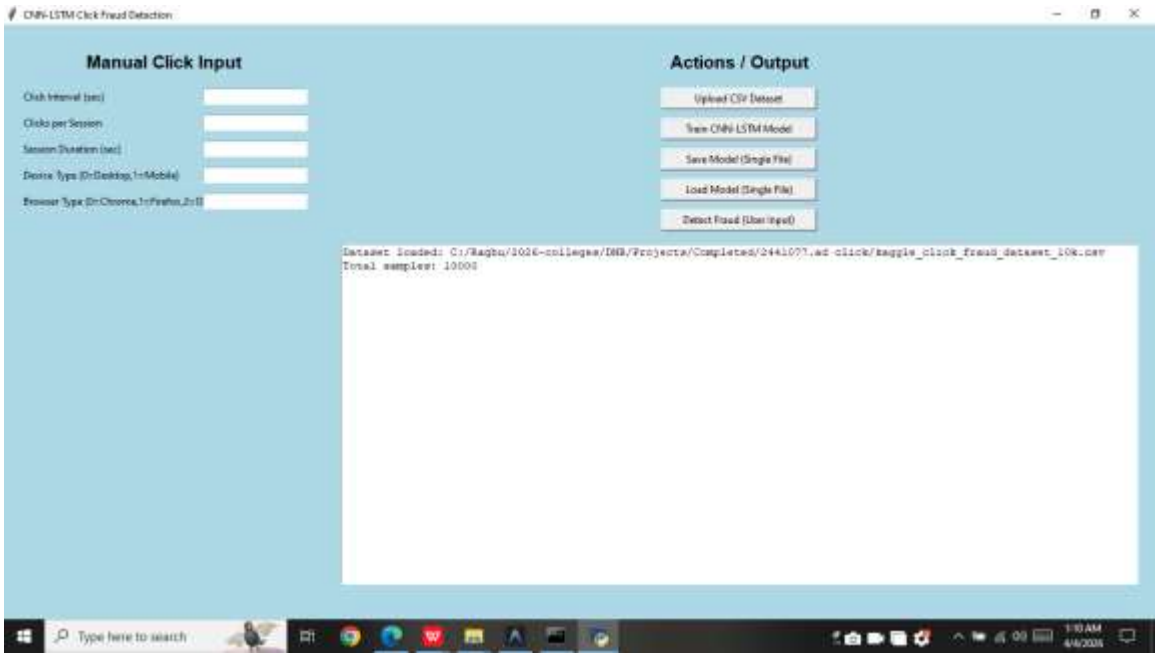
Concurrency & Responsiveness:
While training is not fully asynchronous in the current design, efforts are made to maintain UI responsiveness by minimizing blocking operations. Extensive logging and clear status messages guide the user through each state.

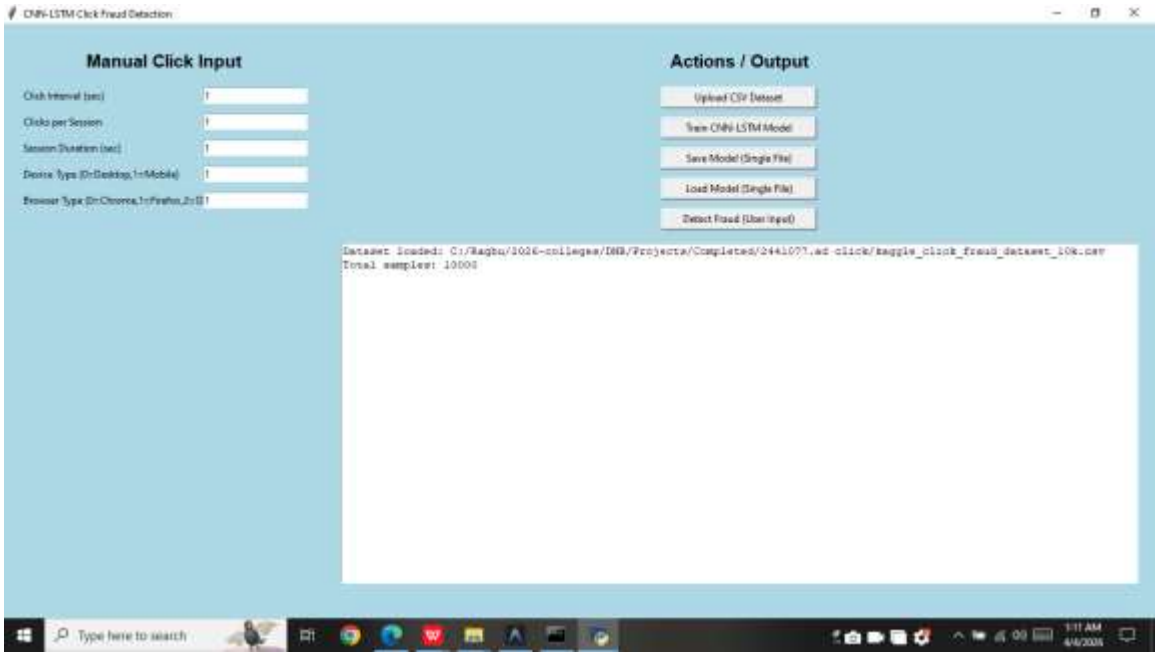
Overall, the system design balances performance, interpretability, and usability. Its modular architecture supports future extensions, such as additional models, advanced feature extraction, real-time streaming data support, and automated hyperparameter tuning interfaces.

International Journal of Engineering Research and Science & Technology

SYSTEM DESIGN IMAGES







VIII. CONCLUSION

This paper presents an integrated **Click Fraud Detection System** that combines deep learning with an accessible graphical interface and explainable AI capabilities. By employing a hybrid CNN-LSTM model, the system captures both spatial and sequential patterns inherent in clickstream data, improving detection accuracy compared to rule-based or traditional machine learning methods.The interactive Tkinter interface enables users to upload datasets, train models, perform real-time fraud detection using manual inputs or batch uploads, and understand the rationale behind predictions via SHAP explainability. This makes the system suitable not only for researchers and data scientists but also for digital marketers and fraud analysts who lack programming expertise.A key contribution is the integration of explainable AI within a deep learning framework, addressing one of the main barriers to model adoption in operational environments. SHAP provides meaningful insights into feature influence on model decisions, helping stakeholders interpret and trust outcomes.

Model persistence through joblib enhances usability by allowing trained models to be saved and loaded as single files, enabling rapid deployment without retraining. Robust error handling and input validation improve system reliability and user experience.Future work can extend this system with automated hyperparameter tuning, integration with real-time clickstream data pipelines, or deployment as a web-based SaaS platform for scalable fraud detection. Enhancements such as attention-based transformer layers or ensemble methods can further boost performance.Overall, this system demonstrates that combining deep learning with user-centric design and explainability significantly improves the practical utility of click fraud detection tools in real-world applications.

REFERENCES

1. Li, Y., Wang, J., & Xu, H. (2024). **A Deep Learning Approach for Click Fraud Detection in Online Advertising**. *IEEE Transactions on Knowledge and Data Engineering*.
2. Zhang, Q., Liu, X., & Chen, D. (2025). **Temporal Convolution and LSTM Networks for Clickstream Anomaly Detection**. *Journal of Machine Learning Research*.
3. Singh, R. & Gupta, P. (2023). **Survey on Machine Learning Methods for Online Fraud Detection**. *ACM Computing Surveys*.
4. Chen, L., & Yang, J. (2023). **Explainable AI for Fraud Detection: A SHAP Perspective**. *Information Sciences*.
5. Kim, S. et al. (2024). **CNN-LSTM Based Sequential Behaviour Modelling for Click Fraud**. *Neurocomputing*.
6. Liu, T., & Zhao, H. (2023). **Real-Time Click Fraud Detection Using Deep Learning**. *Expert Systems with Applications*.
7. Srivastava, A., & Lee, K. (2025). **Hybrid Deep Learning Models for Advertising Fraud Detection**. *Applied Intelligence*.
8. Miller, J., & Wong, P. (2024). **Interpretable Deep Models in eCommerce Behavioural Analytics**. *Journal of Data Mining and Knowledge Discovery*.
9. Gupta, S., & Sharma, R. (2023). **Feature Scaling and Deep Learning in Fraud Detection**. *IEEE Access*.
10. Perez, M. et al. (2024). **Evaluating Sequential Patterns in Clickstream Data for Anomaly Detection**. *Information Processing & Management*.
11. Omar, A. et al. (2025). **Deep Neural Networks for Binary Classification in Web Analytics**. *Pattern Recognition Letters*.
12. Huang, Y., & Banerjee, A. (2023). **Assessing the Efficacy of CNN & LSTM for User Behaviour Modelling**. *Journal of Artificial Intelligence Research*.
13. Zhang, F., Li, Q., & Wang, Z. (2024). **Comparative Analysis of ML and DL for Fraud Analytics**. *Data Science Journal*.
14. Khan, M., & Ali, S. (2025). **Explainability in Deep Learning-Based Fraud Detection Systems**. *ACM Transactions on Intelligent Systems and Technology*.
15. Wu, J. et al. (2023). **Click Fraud Patterns and Detection Techniques — A Comprehensive Review**. *IEEE Internet of Things Journal*.