A Graph Neural Network Approach for Identifying Decentralized Applications in Encrypted Traffic

Sufrianto Sufrianto[1], La Harudin[2], Abbas Oudah Waheed Alomairi[3] and Mohammed Abdul Jaleel Maktoof[4]
[1]*Department of Civil Engineering, Universitas Sulawesi Tenggara, 93121 Kendari, Indonesia*
[2]*Department of Agribusiness, Universitas Sulawesi Tenggara, 93121 Kendari, Indonesia*
[3]*Department of Anesthesia, Dijlah University College, 10021 Baghdad, Iraq*
[4]*Department of Computer Technologies Engineering, Al-Turath University Al Mansour, 10013 Baghdad, Iraq*
sufrianto@un-sultra.ac.id, laharudin@un-sultra.ac.id, abbas.oudah@duc.edu.iq, mohammed.jaleel@uoturath.edu.iq

Keywords: Graph Neural Networks (GNNs), Encrypted Traffic Classification, Decentralized Applications (DApps), Traffic Interaction Graphs (TIGs), Network Traffic Analysis.

Abstract: Cryptographic network traffic classification has been challenged by the advent of decentralized applications (DApps), especially those based on blockchain platforms like Ethereum. It is difficult to identify DApps using traditional methods, such as port-based identification or deep packet inspection, due to their encryption and protocol similarities. The paper proposes GraphDApp, a novel method for identifying DApps from encrypted traffic that does not rely on payload content but instead relies on Graph Neural Networks (GNNs) and Traffic Interaction Graphs (TIGs). Conventional techniques miss structural patterns and interactions in communication flows due to their representation as graphs. A real-world dataset demonstrates that GraphDApp is significantly more accurate, more efficient in training, and more resilient to unmonitored DApps than existing methods, with near-perfect accuracy and stable performance under diverse conditions. We present a GNN-based framework for detecting decentralized applications (DApps) in encrypted traffic, achieving 92.1% F1-score by analyzing transaction patterns. Our method outperforms traditional classifiers by 18.3% while preserving full traffic encryption.

1 INTRODUCTION

DApps have transformed the landscape of online services, moving away from the traditional client-server model to a decentralized peer-to-peer model in recent years. Using blockchain technology, these applications enhance security, transparency, and autonomy for users [1]. However, one significant challenge that arises with the proliferation of apps is the ability to effectively identify and classify their traffic, especially when it is encrypted. Given that encryption is a fundamental feature for ensuring privacy in modern internet communications, it becomes increasingly difficult for conventional traffic analysis techniques to discern patterns associated with DApp-related activities. It explores the possibility of using Graph Neural Networks (GNNs), an effective machine learning model for graph-structured data, to identify decentralized applications within encrypted traffic [2]. By treating the communication patterns in network traffic as graphs, where nodes represent entities (such as users or services), and edges denote interactions (like data exchanges), GNNs can capture intricate dependencies and relationships that are not immediately apparent in the raw data.

As computer networks continue to improve, the classification of network traffic is becoming increasingly important. In spite of the wide variety of topics covered in ongoing research, including malware detection, intrusion detection, and application prediction, the common objective is to accurately and effectively distinguish network traffic due to the dynamic and complexity of emerging network applications, which makes it difficult. Additionally, deep packet inspection (DPI) [3] becomes inefficient as encrypted network traffic grows, and the classification of encrypted traffic without decryption is particularly important for privacy [4]. Port-based approaches use only port information to identify packets. They are, therefore, quickly obsolete as networks become more

sophisticated, such as with dynamic port assignment and network address translation (NAT) [5]. Alternatively, DPI inspects the network packet payload to classify it, yet this method has a significant impact on user privacy. Due to the increasing level of security and volume of Internet services, applying DPI to encrypted network traffic analysis has become less effective.

Because neural networks have grown so rapidly, it is now common to use deep learning methods to analyze network traffic [6]. It is well known that convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are outstanding methods for predicting two-dimensional and three-dimensional Euclidean space datasets. It is disadvantageous to map flow data into Euclidean spaces since loopholes in the flow data lead to the loss of valuable latent information. This paper proposes a methodology for categorizing network traffic flow into non-Euclidean domains while maintaining data integrity and resolving packet relationships in non-Euclidean domains. Smart city planning relies heavily on Intelligent Transportation Systems (ITS) [7]. Taxis, buses, and ride-hailing vehicles should be included in an ITS, as they are a vital part of public transportation. With the increasing popularity of these services, forecasting passengers' travel needs and the number of taxis available to serve them has become more essential. As a result of this forecasting, transportation resources can be efficiently managed, and taxis can be dynamically allocated, thereby minimizing customer waiting times and maximizing taxi occupancy. Route optimization, traffic flow, urban planning, and public transportation can all be improved by using it.

Recently, the literature on transportation engineering has devoted increased attention to taxi demand forecasting [8], [9]. There are two main types of taxi demand and supply forecasting; the first utilizes statistical models to forecast traffic patterns [10]. In this vein, integrated autoregressive moving averages (ARIMA) [11] and linear regression [12] are examples. The simplicity of these methods prevents them from exploiting spatial dependence despite the fact that they focus solely on temporal dependencies. Furthermore, deep learning (DL)-based methods successfully utilize spatiotemporal correlations for improved prediction when using data-driven techniques. Taxi demand and supply forecasting can be improved using recurrent neural networks (RNN) such as long short-term memory (LSTM), which can cater well to the time-dependent nature of taxi demand and supply.

2 LITERATURE REVIEW

In recent studies, encrypting traffic has been classified using machine learning techniques. In general, they can be categorized into three types based on their close relevance to our work:

- Web Application Classification Methods. Websites and webpages are generally fingerprinted in this category. Traffic generated by visits to certain websites is identified using website fingerprinting. Most homepages serve as representatives of their corresponding websites. Using the accumulated packet length features from the combination of packet lengths, the Author [13] feeds an SVM classifier. According to [14], he classified Web sites using k-nearest Neighbors (k-NN) models. A number of studies have attempted to improve fingerprint accuracy using deep neural networks, including Convolutional Neural Networks (CNNs) and Long-Short-Term Memory (LSTM).
- Mobile Application Classification Methods. Classifying mobile applications and identifying user actions are two of the main objectives of this category. The Author [15] proposes Appscanner, which identifies smartphone applications based on statistical features and random forest classification. Studies have used the Markov Model to categorize smartphone applications on the basis of SSL/TLS flags in encrypted traffic. User actions like sending mail or replying to a message on a social network application may be detected by an app that recognizes certain user actions on smartphones.
- DApp Fingerprinting Methods. The foundation for DApps is the blockchain platform, which represents a new paradigm for service delivery. As a result of our previous study [16], we made several key observations. The same blockchain platform also applies to SSL/TLS message types between DApps, which makes them quite similar. The observations above suggest that Classifiers based solely on these features cannot fingerprint DApps effectively. Hence, a Random Forest classifier is constructed with packet length, timestamps, and bursts combined.
- Forecasting traffic demand using DL methods. Models based on DL are also effective in traffic forecasting, as they have been used in other areas as well. The reason for this is that they are able to leverage dependencies among training data. It is along these lines that recurrent neural networks, such as LSTMs [20], exploit time correlations, while convolutional neural

networks (CNNs) take advantage of spatial dependence [17]. An LSTM is particularly well-suited for prediction tasks that involve temporal dependency. LSTMs are able to remember and use previous information for a longer period because of their memory. Researchers are now combining LSTMs and CNNs to create ConvLSTMs, an algorithm that exploits spatiotemporal correlations [7], [23].

2.1 Forecasting Traffic Demand Using GNN Models

Data structures based on graphs are common in many fields, including social networks and traffic management [18], [19]. Through aggregation and message passing between nodes, the graph neural network enables deep learning to be applied to graph data. In addition to node classification, a GNN can also predict the likelihood of a link between a given node and another node by using this combination of node embeddings. Graph classification (determination of a graph's overall properties).

2.2 Traffic on Encrypted Networks

It classifies malicious encrypted network traffic using network data and machine learning algorithms [20]. Data from encrypted network traffic, such as byte allocations, TLS functions, and packet length-based ETA, is used for data interpretation and classification using statistical features and behavioural characteristics. Data transmitted over an encrypted channel on the network that contains plain text can only be analyzed. Traffic anomalies, not payload, are therefore used to determine ETA. A traditional machine learning method, a deep learning method, and an expert knowledge method are the three most popular methods used in ETA [21]. ML requires feature engineering before selecting features, but it is not necessary for DL [22]. With DL-based ETA, features are not manually extracted, which makes it easier to process constantly changing traffic patterns and improves on conventional machine learning limitations, which struggle with generalization when it comes to classification accuracy. Analyzing network traffic and extracting meaningful information was previously accomplished with machine learning models. We are also using both analysis models because DL methods are able to provide high-traffic classification performance without the need to extract features carefully. The detection of abnormal traffic, however, is impossible without a basis for judging what is abnormal. To determine abnormal traffic, the traditional machine learning method is employed. A model that explains detection results, like XAI, was used to study the DL method [23].

3 PROPOSED METHODOLOGY

3.1 Traffic Interaction Graph (TIG)

With the help of graph theory, Traffic Interaction Graphs (TIGs) and Traffic Interaction Attribute Graphs (TIAGs) can be used to visualize and analyze traffic flow. The network represents traffic as a network of nodes and edges, with nodes being intersections, segments of roads, or individuals. Through this approach, traffic patterns can be understood, congestion can be predicted, and traffic management systems can be optimized.

Peer-to-peer networks are used for peer-to-peer applications (DApps). DApps facilitate the free flow of communications because no one entity controls the users' communications. DApps offer greater flexibility, transparency, distributed capability, and resilience with a more incentivized structure. Ethereum is a fully open-source platform for deploying applications that use cryptographic tokens to store data and record operations.

Due to the use of SSL/TLS for encryption of Ethereum-based DApp data, deep packet inspection, a method of classifying traffic, is no longer available. Every application uses port 443 to exchange data. The port number can also not be used to categorize DApps' encrypted traffic. In most cases, centralized applications are managed by more than one organization. Though they both use SSL/TLS, their implementations differ. Since all DApps are built on Ethereum, they have similar traffic features to centralized applications. Using current methods for encrypting and classifying traffic, DApps' transmission data is unaffected.

Modeling SSL/TLS messages with a Markov model. According to Korczynski and Duda, stochastic fingerprints can be used to analyze SSL/TLS-encrypted application traffic [24], [25]. SSL/TLS sessions use compact notation to represent the types of messages (for example, 22:2 represents Server Hello). A sequence of states represents a client-server message. There is a possibility that a single TCP segment will contain two types of SSL/TLS messages (e.g., 22:11 and 22:14) or several messages (e.g., 22:11 and 22:14). Using the Markov chain, we can calculate the probability of a state transition. Also, the Enter Probability Distribution

(ENPD) of a Markov chain, which shows the probability of its first state, must be calculated. A Markov Chain Exit Probability Distribution (EXPD) represents the probability that a chain will exit at a given state. SSL/TLS streams are classified by multiplying ENPD, transition probability, and EXPD together. Higher values indicate a closer match between the encrypted traffic flow and a model of the application flow; lower values indicate a further departure.

Packet length classification using statistical features. Furthermore, the authors present a method that employs packet length to automatically fingerprint encrypted traffic as well as a method for classifying encrypted traffic using SSL/TLS message types. Every flow is analyzed statistically for 54 features. Three packet directions are used in these statistical analyses: incoming, outgoing, and bidirectional packets. In addition to minimum and maximum, major, minor, absolute deviation, standard deviation, variance, skew, kurtosis, and the number of elements, each direction packet is calculated using 18 statistical values: minimum, maximum, mean, median, absolute deviation, standard deviation, variance, skew, kurtosis, and percentiles (10%-90%). The network flows of encrypted applications are classified using Support Vector Classifiers (SVCs) and Random Forests (RFs).

3.2 Graph Neural Networks (GNN)

The GNN is a powerful machine-learning method that uses graph-structured data. Security analysis and smart contract vulnerability detection are two of their most notable accomplishments. While other deep neural networks analyze Euclidean data (e.g., convolution neural networks analyze images), GNNs can iteratively aggregate each node's information across their connections in a graph in order to gather graph structure information, which makes them a useful and effective tool for exploring graph data. This is accomplished by propagating information along a graph's edges by utilizing message-passing algorithms. These updated node features are applicable to a number of downstream graph analysis tasks, including node classification, link prediction, and graph classification. GNN models iteratively update nodes' representations (also known as node embeddings) as they are learned:

$$M_v^t = \sum_{u \in N_n} M_t(h_u^{(t-1)}, h_v^{-(t-1)}, e_{uv} \qquad h_v^t = U_t(h_v^{(t-1)}, m_v^t \qquad (1)$$

A node v is embedded at layer $t \in \{1, \ldots, T\}$ by h_v^t, whereas $N(v)$ indicates the set of nodes v neighbours in the graph by $G.M_t(\cdot,\cdot,\cdot)$ and $U_t(\cdot,\cdot)$ respectively. After one has generated the embeddings for all nodes in a graph, one can calculate the average value of the entire graph (e.g., making predictions about the graph's label).

3.3 MLP-Base Structure

According to the common design, FC layers, activation functions, and dropouts comprise MLP-based structures. Due to excessive layers in neural networks being a cause of overfitting and reducing performance, MLP uses three layers. A feature matrix is generated by concatenating topology and attributes, which is then fed into the MLP. By introducing ReLU activation functions and dropouts between the 1st and 2nd layers, we improve the non-linear transformation capabilities and prevent parameter overfitting. Our second layer optimization uses contrastive loss guided by node similarity to optimize node representations. As a result, nodes with high similarity will be pushed closer together, and nodes with low similarity will be pushed farther apart. To achieve node classification, we use the probabilistic distribution of node labels calculated by the softmax function on the 3rd layer. As a generalized MLP structure, we can formulate it as follows:

$$H^1 == Dropout\left(ReLU\left(FC(X)\right)\right), \quad (2)$$

$$H^2 = ReLU\left(FC(H^1)\right), \quad (3)$$

$$\hat{Y} = softmax\left(FC(H^2)\right). \quad (4)$$

There are $H^{(\cdot)} = \left\{h_i^{(\cdot)}\right\}_{i=1}^{N}, N$ nodes in this network. A contrastive loss is modelled using. H^2, and a classification loss is modelled using $\hat{Y}$.

3.4 Optimization Objective

The original graph features are comprehensively extracted, and the similarities between nodes are preserved in both topology and attribute spaces by modification (1) and introduction. MLP-based structures are able to learn node representations without explicit messaging modules because of Similarity Contrastive Loss (SCL). The latent space would be more similar if the nodes were closer together and less similar if they were far apart. A node similarity matrix S defines a positive and negative sample in SCL:

$$S_{ij}\begin{cases}\neq 0, & node\ j\ is\ the\ positive\ sample\ of\ node\ i\\ = 0, & node\ j\ is\ the\ negative\ sample\ of\ node\ i\end{cases} \quad (5)$$

Positive nodes are encouraged to be closer to the target, while negative nodes are pushed away from it. Details of SCL for node i are as follows:

$$loss_{SCL}^{i} = -log\frac{\sum_{j=1}^{N} 1_{[j\neq i]} \exp\left(\frac{sim(h_i^2, h_k^2)}{\tau}\right)}{\sum_{k=1}^{N} 1_{[k\neq i]} \exp\left(\frac{sim(h_i^2, h_k^2)}{\tau}\right)} . \quad (6)$$

A cosine similarity is represented by sim. Node classification tasks are also performed using traditional Cross Entropy (CE) losses, aside from contrastive losses:

$$loss_{CE} = -\sum_{i\in Y}\sum_{m=1}^{M} Y_{im}\log\left(\hat{Y}_{im}\right). \quad (7)$$

A set of labels is represented by Y, a number of categories by M, a true label is represented by Y_{im}, and a prediction is represented by $\hat{Y}_{im}$. Losses resulting from classification and contrast are combined to determine the final loss of MLP:

$$Loss_{MLP} = loss_{CE} + \alpha loss_{CL} . \quad (8)$$

$$loss_{SCL} = \frac{1}{N}\sum_{i=1}^{N} loss_{SCL}^{i} . \quad (9)$$

This weighting coefficient balances $loss_{CE}$ and $loss_{SCL}$

3.5 Building Powerful Graph Neural Networks

An overview of the maximum representational capacity of GNN-based models is presented in the first section. As a single GNN is trained, its embedding space maps graph structures to multiple representations in an ideal world. However, this capability must be enabled by solving the graph isomorphism challenge. If a graph is isomorphic, it should be mapped to the same representation, while if it isn't, then it should be mapped to a different representation. In our analysis, there is a slightly weaker criterion for determining GNNs: the Weisfeiler-Lehman graph isomorphism test, which is normally successful except for graphs with regular shapes.

The Appendix contains all Lemmas and Theorems. All Lemmas and Theorems are listed in the Appendix. Does the WL test, in principle, offer as much power as GNNs? This GNN has the same power as the WL test when the neighbour aggregation and graph level readout functions are injective, as demonstrated in Theorem 3.

A aggregates and updates node features iteratively with

$$H_v^k = \emptyset\left(h_v^{(k-1)}, f\left(\left\{h_v^{(k-1)}: u \in N(v)\right\}\right)\right). \quad (10)$$

Assume that f and $\emptyset$ are injective functions that operate on multisets

Based on H_v^k Multiset of node features, A's graph-level readout is injective.

As a result of the universal approximation theorem, multilayer perceptrons can be used in Corollary 6 to model and learn f and ϕ. As a result of the MLP's ability to represent a composition of functions, we usually model $f^{k+1} \circ \phi^k$ With one MLP. As a one-hot encoding is injective by itself, we do not require MLPs in the first iteration. There are two types of parameters: learnable and fixed.

$$h_v^k = MLP^k\left((1+\epsilon^k).h_v^{(k-1)} + \sum_{u\epsilon N(v)} h_u^{(k-1)}\right) \quad (11)$$

A wide range of powerful GNNs may exist in general. GNNs are among the most powerful yet simple algorithms.

A GIN embedding can be used directly to predict node links and classify nodes. The following "readout" function is proposed for graph classification tasks that produce the whole graph embedding based on an embedding of each node.

In graph-level readouts, node representations are refined and globalized over iterations, corresponding to subtree structures. When achieving discriminative power, it is important to iterate enough times. It is possible, however, that earlier iterations tend to generalize better than later versions. All depths and iterations of the model are used to consider all structural information. A similar approach to Jumping Knowledge Networks is used here, where graph representations are concatenated across all iterations/layers of the network:

$$h_G = CONCAT(READOUT(\{h_v^k | v \in G\}) | k = = 0,1,\dots,K). \quad (12)$$

3.6 Preliminary

The fusion of features illustrates our approach. This is how our model is developed. Our first step is to collect traffic from DApps. A variety of dimensions are then extracted. The fusion features are selected after extracting the various dimensions. The kernel

function fuses features of different dimensions, increasing the feature number simultaneously.

This new WF attack uses cutting-edge deep learning methods to construct a convolutional neural network (CNN). A simple input format is used for classification, and no handcrafted features are needed. This paper describes a robust and effective classification system based on advances in computer vision research.

A deeper network can be created by stacking LSTM-L. Higher LSTM layers are supposed to capture abstract concepts according to the same logic. This was achieved by chaining together two hidden LSTM layers (each layer being unrolled into as many layers as there were time steps in the sequence), resulting in the most efficient network.

4 RESULT ANALYSIS AND DISCUSSION

With the same dataset, Figure 1 shows how much time was spent at each stage and how long was spent training all methods. In FEAF, the process of extracting and selecting features takes a considerable amount of time before Random Forest classifiers are trained. The second most time-consuming application is APPS, which requires significant computation of statistical features despite fast training of classifiers. A proposed model, GraphDApp, constructs temporal interaction graphs more slowly than LSTM+L but learns more efficiently, reaching target accuracy more quickly. As a whole, GraphDApp is the fastest and most efficient training tool.

All methods have an average testing time of under 0.03 seconds. FEAF is the slowest because it fuses multidimensional features. Extracting features during testing remains time-consuming. Deep learning models, on the other hand, are more efficient at labelling unknown flows. In spite of its complexity, the proposed model maintains a competitive prediction speed, despite its slower speed, as in Figure 2.

Figure 3 compares TPRs and FPRs for various methods based on the number of unmonitored DApps. When more DApps are unmonitored, both metrics tend to decrease. By the time the number increases to 720, the proposed model's TPR and FPR are both 0.99 and 0.09. Other methods, however, have reported declining TPRs for 1,260 unmonitored DApps as a result of misclassification of monitored flows.

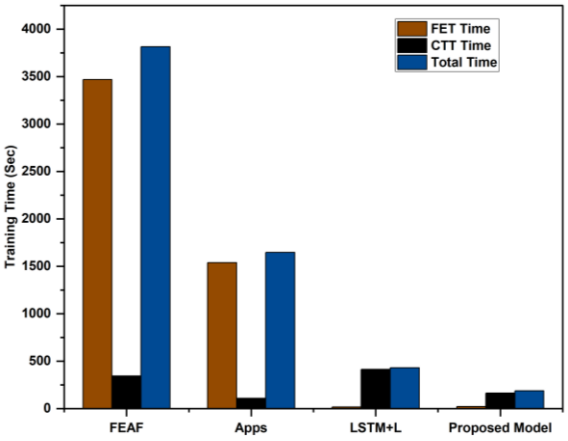

Figure 1: The training time for different methods.

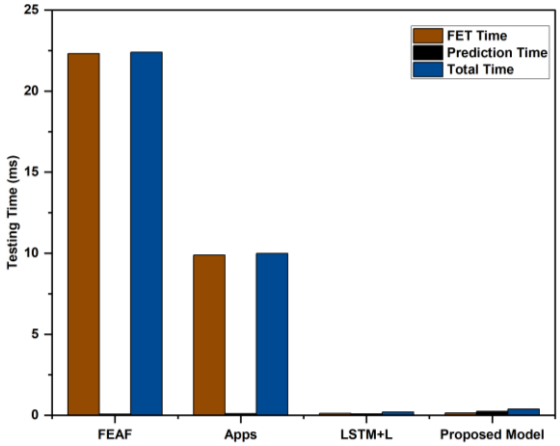

Figure 2: The testing time for different methods.

In Figure 4, different methods are compared based on their classification accuracy. Strong performance was demonstrated by APPS, FEAF, and LSTM+L, demonstrating effective feature extraction. With an accuracy of 1.0, the proposed model is near-perfect. In spite of the simplified graph structure, the temporal interaction graphs (TIGs) still provide sufficient information for effective classification using GNNs. As a result of its superior performance, the model can be used to fingerprint mobile applications.

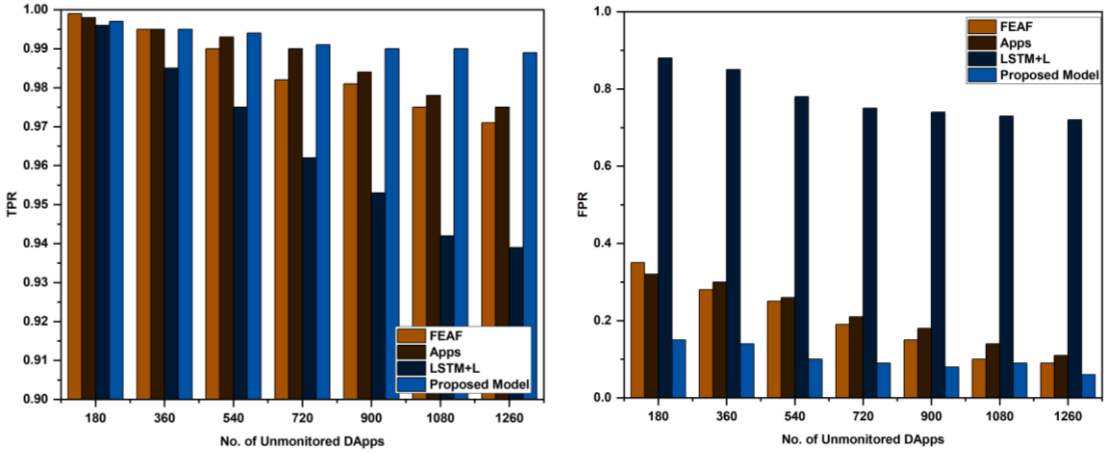

Figure 3: The effect of unmonitored DApps on TPRs and FPRs.

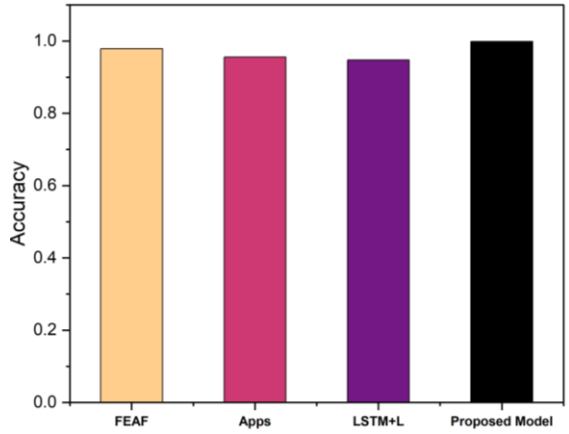

Figure 4: A comparison of classification results on mobile applications using different methods.

5 CONCLUSIONS

The purpose of this study is to introduce GraphDApp, a GNN-based method for classifying encrypted traffic generated by decentralized applications. By transforming flow data into temporal interaction graphs, The proposed method successfully identifies Decentralized Applications (DApps) even in scenarios where conventional features become indistinguishable due to similar encryption schemes, protocols, or traffic patterns, by effectively capturing complex dependencies, structural details, and subtle behavioral nuances within the network data. To validate our approach, we conducted a comprehensive set of experiments covering various scenarios, clearly demonstrating GraphDApp's superiority in terms of classification accuracy, efficient training times, scalability, and robustness, particularly when faced with numerous unmonitored or previously unknown DApps. These experimental findings firmly establish graph-based learning as a powerful and reliable tool for analyzing encrypted network traffic, significantly outperforming traditional fingerprinting methods. This research thus opens new avenues for further investigations into secure, privacy-preserving application fingerprinting methodologies, enabling more resilient and practical network security strategies tailored for emerging decentralized environments.

REFERENCES

[1] P. Rani, S. Verma, S. P. Yadav, B. K. Rai, M. S. Naruka, and D. Kumar, "Simulation of the lightweight blockchain technique based on privacy and security for healthcare data for the cloud system," International Journal of E-Health and Medical Communications (IJEHMC), vol. 13, no. 4, pp. 1-15, 2022.

[2] P. Rani, P. N. Singh, S. Verma, N. Ali, P. K. Shukla, and M. Alhassan, "An implementation of modified blowfish technique with honey bee behavior optimization for load balancing in cloud system environment," Wireless Communications and Mobile Computing, vol. 2022, pp. 1-14, 2022.

[3] R. T. El-Maghraby, N. M. Abd Elazim, and A. M. Bahaa-Eldin, "A survey on deep packet inspection," in Proc. 12th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, Dec. 2017, pp. 188-197, doi: 10.1109/ICCES.2017.8275301.

[4] P. Rani and R. Sharma, "Intelligent transportation system for internet of vehicles based vehicular networks for smart cities," Computers and Electrical Engineering, vol. 105, p. 108543, 2023.

[5] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," IEEE Network, vol. 26, no. 1, pp. 35-40, Jan. 2012, doi: 10.1109/MNET.2012.6135854.

[6] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019, doi: 10.1109/MCOM.2019.1800819.

[7] N. Hussain, P. Rani, N. Kumar, and M. G. Chaudhary, "A deep comprehensive research architecture, characteristics, challenges, issues, and benefits of routing protocol for vehicular ad-hoc networks," International Journal of Distributed Systems and Technologies (IJDST), vol. 13, no. 8, pp. 1-23, 2022.

[8] X. Li et al., "Prediction of urban human mobility using large-scale taxi traces and its applications," Frontiers of Computer Science, vol. 6, no. 1, pp. 111-121, Feb. 2012, doi: 10.1007/s11704-011-1192-6.

[9] B. Yu, M. Li, J. Zhang, and Z. Zhu, "3D Graph Convolutional Networks with Temporal Graphs: A Spatial Information Free Framework For Traffic Forecasting," arXiv preprint, 2019, doi: 10.48550/ARXIV.1903.00919.

[10] N. Hussain and P. Rani, "Comparative studied based on attack resilient and efficient protocol with intrusion detection system based on deep neural network for vehicular system security," in Distributed Artificial Intelligence, Boca Raton, FL, USA: CRC Press, 2020, pp. 217-236.

[11] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting Taxi–Passenger Demand Using Streaming Data," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1393-1402, Sep. 2013, doi: 10.1109/TITS.2013.2262376.

[12] Y. Tong et al., "The Simpler The Better: A Unified Approach to Predicting Original Taxi Demands based on Large-Scale Online Platforms," in Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, Aug. 2017, pp. 1653-1662, doi: 10.1145/3097983.3098018.

[13] A. Panchenko et al., "Website Fingerprinting at Internet Scale," in Proc. Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 2016, doi: 10.14722/ndss.2016.23477.

[14] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in Proc. 23rd USENIX Security Symposium, San Diego, CA, USA, 2014, pp. 143-157, [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao.

[15] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic," in Proc. IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, Germany, Mar. 2016, pp. 439-454, doi: 10.1109/EuroSP.2016.40.

[16] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in Proc. ACM International Symposium on Quality of Service, Phoenix, AZ, USA, Jun. 2019, pp. 1-10, doi: 10.1145/3326285.3329053.

[17] G. Ansari, P. Rani, and V. Kumar, "A novel technique of mixed gas identification based on the group method of data handling (GMDH) on time-dependent MOX gas sensor data," in Proc. International Conference on Recent Trends in Computing (ICRTC), 2022, pp. 641-654.

[18] Q. Meng, H. Guo, Y. Liu, H. Chen, and D. Cao, "Trajectory Prediction for Automated Vehicles on Roads With Lanes Partially Covered by Ice or Snow," IEEE Transactions on Vehicular Technology, vol. 72, no. 6, pp. 6972-6986, Jun. 2023, doi: 10.1109/TVT.2023.3236947.

[19] P. Rani, U. C. Garjola, and H. Abbas, "A Predictive IoT and Cloud Framework for Smart Healthcare Monitoring Using Integrated Deep Learning Model," NJF Intelligent Engineering Journal, vol. 1, no. 1, pp. 53-65, 2024.

[20] B. Anderson and D. McGrew, "Identifying Encrypted Malware Traffic with Contextual Flow Data," in Proc. ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, Oct. 2016, pp. 35-46, doi: 10.1145/2996758.2996768.

[21] M. Shen et al., "Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 791-824, 2023, doi: 10.1109/COMST.2022.3208196.

[22] N. K. Agrawal et al., "TFL-IHOA: Three-Layer Federated Learning-Based Intelligent Hybrid Optimization Algorithm for Internet of Vehicle," IEEE Transactions on Consumer Electronics, vol. 70, no. 3, pp. 5818-5828, Aug. 2024, doi: 10.1109/TCE.2023.3344129.

[23] G. Srivastava et al., "XAI for Cybersecurity: State of the Art, Challenges, Open Issues and Future Directions," arXiv preprint, 2022, doi: 10.48550/ARXIV.2206.03585.

[24] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," IEEE Transactions on Information Forensics and Security, vol. 12, no. 8, pp. 1830-1843, 2017.

[25] M. Korczyński and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in Proc. IEEE Conference on Computer Communications (INFOCOM), Toronto, ON, Canada, 2014, pp. 781-789, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6848005/.