International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 30, No. 10, October, 2025, pp. 2571-2581
Journal homepage: http://iieta.org/journals/isi

Darknet Traffic and Application Classification Using Heterogeneous Graph Neural Network

Abdelhak Etchiali[1,2*], Wafaa Ferhi[3,4], Mohammed M'hamedi[2,5], Mourad Hadjila[3,4], Mohammed Merzoug[1,2], Mohammed Hicham Hachemi[4,6]

[1] Department of Computer Science, Faculty of Science, University of Tlemcen, Tlemcen 13000, Algeria
[2] LRIT Laboratory, Faculty of Science, University of Tlemcen, Tlemcen 13000, Algeria
[3] Department of Telecommunication, Faculty of Technology, University of Tlemcen, Tlemcen 13000, Algeria
[4] STIC Laboratory, Faculty of Technology, University of Tlemcen, Tlemcen 13000, Algeria
[5] Department of Second Cycle, Ecole Supérieure en Sciences Appliquées de Tlemcen ESSAT, Tlemcen 13000, Algeria
[6] Department of Electronics, Faculty of Electrical Engineering, University of Science and Technology of Oran-Mohamed Boudiaf, Oran 31000, Algeria

Corresponding Author Email: abdelhak.etchiali@univ-tlemcen.dz

https://doi.org/10.18280/isi.301005

Received: 14 June 2025
Revised: 16 September 2025
Accepted: 24 September 2025
Available online: 31 October 2025

Keywords:
Graph Neural Network, Heterogeneous Graph Neural Network, Darknet, VPN, TOR, classification, network traffic analysis, CIC-Darknet

ABSTRACT

The proliferation of Virtual Private Networks (VPNs) and The Onion Router (TOR) has both benefits and drawbacks for individuals and organisations. These technologies offer enhanced privacy and security online, but can also facilitate illegal or harmful behaviour by masking users' identities. Therefore, it is crucial to develop reliable methods for identifying and monitoring VPN and TOR traffic to mitigate potential risks and ensure online safety. In this paper, we propose a Darknet Heterogeneous Graph Neural Network (DHGNN) model to address the challenge of detecting traffic and applications in the Darknet. Our approach utilizes the CIC-Darknet2020 dataset, a large collection of openly available network traffic data, to train and evaluate our DHGNN classifier. The dataset is systematically explored to identify the most informative features and preprocessed into a clean tabular format. This tabular data is then converted into a graph structure suitable for the DHGNN classifier. Experimental results show that the proposed model achieves 99.80% accuracy in traffic classification and 98.80% accuracy in application classification, outperforming existing methods in Darknet classification. This approach demonstrates the effectiveness of integrating feature-driven preprocessing with graph-based neural network modeling for robust and accurate classification.

1. INTRODUCTION

The internet is a global system that links millions of computers and digital devices together, enabling communication and information exchange across the world. A key service on the internet is the World Wide Web, which is a vast collection of interconnected pages containing text, images, and videos, accessed online. The Web relies on key components: web pages (documents written in a special code), hyperlinks (which enable navigation between pages), and web servers (computers that host and deliver these pages). In contrast, the Deep Web is a hidden part of the internet that standard search engines like Google cannot access, containing private data such as bank records, medical history, and government files not intended for public viewing [1].

In contrast, the Dark Web constitutes a part of the internet accessible solely through specialized software like the Tor browser, characterized by its anonymity. Infamous for its association with illicit activities, it serves as a marketplace for illegal transactions, including drugs, weaponry, and stolen data, and facilitates the exchange of prohibited information and materials [2]. The proliferation of encrypted and anonymized network traffic, largely driven by the widespread use of privacy-enhancing technologies (PETs) like Tor and VPNs, has created a formidable challenge for modern cybersecurity operations [3, 4]. While these technologies serve legitimate privacy needs, their infrastructure also forms the backbone of Darknets, which are increasingly exploited for illicit activities such as coordinated hacking campaigns, espionage, and the deployment of advanced persistent threats (APTs). Consequently, the accurate and efficient classification of Darknet traffic has become a critical imperative for enabling proactive cyber threat intelligence and strengthening national and organizational security postures.

The expansion of the Internet has transformed global communication, enabling individuals to interact across borders with an Internet connection. The Tor network is a technology that provides users with privacy and anonymity online. It was developed by a team of mathematicians and computer experts based at the Naval Research Laboratory (NRL) in response to concerns about privacy and anonymity during the early stages of the Internet's development [5].

The Tor network uses a technique called onion routing to route traffic through multiple servers and encrypts it, making it nearly impossible for anyone to trace the user's online activities [3, 6]. Despite its importance, the task of Darknet traffic classification is fraught with inherent difficulties. The very nature of anonymity tools is to obfuscate traffic patterns and payload content, rendering traditional deep packet inspection (DPI) methods ineffective. While machine learning (ML) approaches that rely on flow-based statistical features have shown more promise, they often exhibit critical limitations [7, 8]. These methods typically treat traffic flows as independent, isolated instances, failing to capture the complex relational structures and temporal dependencies that exist between communication nodes in a network [9]. This inability to model the underlying graph-like nature of network data, where connections between IP addresses, sessions, and protocols contain vital discriminative information, results in suboptimal accuracy, limited generalizability, and poor scalability to the vast volumes of traffic generated in real-world Darknet environments [10].

To bridge this gap, we propose a Darknet Heterogeneous Graph Neural Network (DHGNN) framework. The core innovation of our approach is the reformulation of the traffic classification problem from a tabular learning task to a graph learning task. By constructing a heterogeneous graph that represents various network entities (e.g., hosts, packets, services) and the rich relationships between them, our DHGNN model can directly learn from the topological structure of the network traffic. This allows the model to discern subtle, relational fingerprints of different applications and services that are invisible to conventional classifiers [11, 12]. The proposed method offers significant advantages, including enhanced classification accuracy by leveraging multi-relational data, inherent scalability for large-scale network analysis, and a more nuanced understanding of network behavior.

To validate our approach, we utilize the comprehensive CIC-Darknet2020 dataset [13], a benchmark containing labeled traffic from Tor, VPN, and non-VPN applications. Our methodology involves a rigorous feature analysis to inform the graph construction process, followed by a data preprocessing pipeline that transforms raw tabular traffic data into a structured heterogeneous graph. The experimental results demonstrate that our DHGNN classifier achieves superior performance, outperforming previous state-of-the-art traffic classifiers and confirming the efficacy of graph-based learning for tackling the complexities of Darknet traffic analysis.

The rest of this paper is organized as follows: Section II provides a summary of the relevant literature and prior research in the field. Section III covers the basics of Heterogeneous Graph Neural Networks, such as graph data construction, heterogeneous message passing, and model architecture. Section IV presents the proposed methodology, encompassing the description of the CIC-Darknet2020 dataset, data exploration and preprocessing, as well as the model design. Section V provides the experimental results for both traffic and application classification scenarios. Following this, Section VI is dedicated to the conclusion and suggestions for future research directions.

2. RELATED WORK

Recent research shows that many scientists now prefer to use Graph Neural Networks (GNNs) instead of traditional machine learning methods. GNNs are especially good at understanding relationships between data points when the data can be represented as a graph. This strength is very useful in areas such as social network analysis, protein–protein interaction modelling, and anomaly detection—tasks where older methods often struggle to capture the connections between features.

The study by Kisanga et al. [14] introduces a new approach to network security by building activity and event networks with GNNs. This method aims to better handle both large-scale attacks and long-term threats. The authors tested their approach on two datasets, TOR-nonTOR and a DDoS dataset, and achieved accuracy scores of 78% and 88%.

In another study, Purnama et al. [15] explore how GNNs can improve modern Intrusion Detection Systems (IDS). They use E-GraphSAGE, a variant of the GraphSAGE algorithm, to efficiently gather and use information from neighbouring nodes in graph-structured data. This approach helps capture the complex patterns that appear in network traffic, which is essential for correctly identifying and classifying malicious activity. A related investigation [16] applies GNNs to malware detection and classification. Using the CIC-AndMal2017 dataset, which contains a wide range of Android malware samples, the researchers evaluate how well GNN models can detect and classify different malware types. They test the models using accuracy, precision, recall, and ROC curves. Their results show that GNNs are a strong option for improving malware detection.

An interesting survey [17] reviews how graph representation learning can be used for network- and host-based intrusion detection. It provides an overview of different GNN models, explains how these models are built, and discusses examples from published research. The survey also looks at how robust these methods are against adversarial attacks. In the end, the authors summarise the strengths and weaknesses of GNN-based intrusion detection and suggest directions for future research.

Another survey [18] examines graph adversarial attacks and defence methods. It highlights that although GNNs work very well in many tasks, they can be vulnerable to such attacks. The study reviews key algorithms, comparing their strategies, benefits, and drawbacks, and offers benchmark results that show the ongoing challenge of making GNNs more robust while addressing their security weaknesses.

In the study by Habibi Lashkari et al. [13], a CNN-based model was proposed for binary traffic classification, where Tor/VPN traffic was labelled as "Darknet" and non-Tor/non-VPN traffic as "clearnet." The model reached 94% accuracy in distinguishing between benign and Darknet traffic. It also achieved 86% accuracy in identifying the type of application generating the traffic, classifying them into categories such as video streaming, audio streaming, VoIP, browsing, chat, email, file transfer, and P2P downloads. Similarly, Sarkar et al. [19] used a Deep Neural Network (DNN) to differentiate between Tor and non-Tor traffic, using the UNB-CIC (ISCX Tor-2016) dataset [20]. They designed two models: a three-layer DNN-A and a five-layer DNN-B. DNN-A achieved 98.81% accuracy, while DNN-B performed even better with 99.89%.

The study [21] focused on analysing traffic types using a standard dataset and several machine learning classifiers. They grouped the data into benign and Darknet classes and also considered four traffic types for multi-class classification (Tor/non-Tor or VPN/non-VPN). Their results showed that the Random Forest classifier performed the best, achieving an F1 score of 98.61% in the multi-class setting.

Demertzis et al. [22] extended the application types into 11 subcategories and employed the WANN classifier architecture (Weighted Agnostic Neural Networks). Unlike regular Artificial Neural Networks (ANNs), WANNs neither adjust the weights of neurons; instead, they modify their neural network architecture incrementally. The WANN methodology first evaluates candidate structures based on performance and design complexity, and then uses the best-performing architecture to construct new layers. Their leading WANN classifier achieved an accuracy rate of 92.68% in application-layer classification.

The research detailed in the study by Sarwar et al. [23] focused on classifying traffic and application types using Convolutional Neural Networks (CNNs), as well as two alternative approaches based on Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU). Initially, they selected 20 features based on three methods: Decision Trees (DT), Principal Component Analysis (PCA), and Extreme Gradient Boosting (XGBoost), followed by the implementation of hybrid architectures such as CNN/LSTM and CNN/GRU. In this framework, the CNN layer was responsible for feature extraction, while LSTM and GRU were employed for time-series prediction based on these input features. Notably, CNN/LSTM combined with XGBoost for feature selection demonstrated the highest F1-scores, achieving 96% accuracy in traffic type classification and 89% accuracy in application type classification.

Marim et al. [24] studied and classified real Darknet traffic using the CIC-Darknet2020 dataset. They used feature extraction and applied an n-gram technique to group potential subnets. They also evaluated the importance of the most relevant features using Recursive Feature Elimination. Their results show that simple models such as Decision Trees and Random Forests can achieve more than 99% accuracy in classifying traffic. Overall, their method provides up to a 13% improvement compared with current state-of-the-art approaches.

Alimoradi et al. [25] developed a new decision support system for detecting Tor and VPN traffic. Their system classifies Darknet data into four categories: Tor, non-Tor, VPN, and non-VPN. The model uses a DNN with 79 neurons in the input layer and 6 neurons in the hidden layers to capture the complex nonlinear patterns found in real darknet traffic. They tested their approach on the DIDarknet benchmark dataset, achieving an accuracy of 96%. Impressively, these strong results were obtained without any preprocessing steps such as feature extraction, data balancing, or data cleaning. This highlights the ability of their model to handle raw Darknet traffic effectively.

Zhu et al. [26] introduced the Darknet Traffic Graph (DTG), an interactive graph that visualizes connections between source and destination nodes (servers and clients) in Darknet traffic. Based on DTG, they combined Graph Neural Networks with an attention mechanism to create Darknet Graph Neural Networks (DGNNs). This model makes effective use of both benign and Darknet traffic characteristics. When evaluated on the CIC-Darknet2020 dataset, DGNNs achieved excellent accuracy scores: 98.52% for traffic classification and 99.06% for application classification, outperforming other existing classifiers.

Even with the state-of-the-art use of deep learning models such as CNNs, RNNs, and Transformers for traffic classification, there is still a serious problem. These models ignore the intricate graph relationships between hosts and services, treating traffic as discrete tabular or sequential data [11]. As a result, they are unaware of the critical topological fingerprints of applications. In Darknet environments such as Tor and VPNs, where these structural patterns are the main observable feature, this limitation significantly reduces their efficacy in categorizing encrypted traffic.

Table 1 is included to explicitly contrast tabular and graph-based approaches, thereby emphasizing the distinctive strengths of DHGNN [25].

Table 1. Tabular vs. Graph-based methods: Emphasizing DHGNN strengths

Study/Method	Core Approach	Data Representation	Key Limitation/Gap	DHGNN Improvement
Statistical Methods (SVM, Naive Bayes)	Feature Engineering	Tabular/Vector	Ignores inherent relationships, sensitive to feature selection.	Leverages all relational data directly via the graph structure.
Deep Packet Inspection (DPI)	Signature Matching	Payload Data	Fails with encryption (Tor, VPN), violates privacy.	Operates on metadata and topology, inherently robust against encryption.
State-of-the-Art ML/DL Classifiers (CNN, RNN)	Automated Feature Extraction	Tabular/Sequential	Cannot capture inter-entity relationships (host-packet-service); misses topological fingerprints.	Reformulates to Graph Learning to capture subtle, multi-relational fingerprints.
Proposed Method (DHGNN)	Heterogeneous Graph NN	Heterogeneous Graph	N/A	Enhanced accuracy through topological structure learning; inherent scalability, nuanced understanding of Darknet behavior.

3. HETEROGENEOUS GRAPH NEURAL NETWORK

Heterogeneous Graph Neural Networks (HGNNs) are designed to process graph-structured data consisting of multiple node and edge types, allowing the model to capture complex relational semantics across diverse entities [27]. The presence of such heterogeneous node and edge categories is what distinguishes heterogeneous graphs from their homogeneous counterparts [28].

The HeteroGNN algorithm effectively captures the complex and rich relationships present in such data [29], making it useful for various tasks like node-level classification, link prediction, and graph-level classification across various domains, including social networks, knowledge graphs, and malware detection [30].

3.1 Graph data construction

This phase is foundational, involving the translation of raw network traffic data into a format that a GNN can process.

Consider a heterogeneous graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges [31].

•The node set V is categorized into two subsets, $V = V_h \cup V_f$, with V_h being the host nodes and V_f the flow nodes.

•Host nodes have feature $X_h \in$, and flow nodes have feature matrix $X_f \in R^{(|V_f| \times d_f)}$, where d_h and d_f represent the feature dimensions for host and flow nodes, respectively. These features serve as the input to the HGNN [29].

•Identify the relationships or connections between nodes that will be represented as edges. We consider that the edge set E is divided into two different types of flow and host $E = E_h \cup E_f$, where E_{hf} are edges from host to flow nodes, and E_{fh} are edges from flow to host nodes. This distinction is crucial for heterogeneous graphs where different types of nodes and edges may encode distinct kinds of information and relationships.

3.2 Heterogeneous message passing

The core of the HGNN algorithm is the heterogeneous message passing mechanism, which updates node embeddings by aggregating information from neighboring nodes across different types [31, 32]. This can be formalized as follows for a two-node-type system using Heterogeneous Graph Convolution (HConv).

For E_{hf} edges, the update rule for flow node features is:

$$H_f^{(l+1)} = \sigma\left(W_{hf}^{(l)} \cdot AGG\left(\left\{h_h^{(l)} \mid \forall\left(v_h, v_f\right) \in E_{hf}\right\}\right)\right) \tag{1}$$

For E_{fh} edges, the update rule for host node features is:

$$H_h^{(l+1)} = \sigma\left(W_{fh}^{(l)} \cdot AGG\left(\left\{h_f^{(l)} \mid \forall\left(v_h, v_f\right) \in E_{fh}\right\}\right)\right) \tag{2}$$

Here, $H_f^{(l+1)}$ and $H_h^{(l+1)}$ denote the embeddings for flow and host nodes at layer l.

$W_{hf}^{(l)}$ and $W_{fh}^{(l)}$ are the learnable weights for each edge type, σ represents a non-linear activation function, and AGG denotes an aggregation function, and their role in combining information from a node's neighborhood:

(1) Mean Aggregation:

$$AGG_{mean} = mean\left(H\left[v_i\right], \left(\forall\left(v_i, v_j\right) \in E\right)\right) \tag{3}$$

(2) Sum Aggregation:

$$AGG_{sum} = sum\left(H\left[v_i\right], \left(\forall\left(v_i, v_j\right) \in E\right)\right) \tag{4}$$

3.3 Model architecture

The model consists of L HConv layers, where the node embeddings are iteratively updated [31, 33]. Initially:

$$H_h^{(0)} = X_h \; and \; H_f^{(0)} = X_f \tag{5}$$

After L layers, the embeddings for flow nodes, $H_f^{(L)}$, are used in a linear prediction layer:

$$Y = W_{out} . H_f^{(L)} + b_{out} \tag{6}$$

$Y \in \mathbb{R}^{|V_f| \times C}$ represents the logits for C classes for each flow node, with W_{out} and b_{out} being learnable parameters.

The multi-layer structure with L HConv layers allows for progressively complex feature extraction and representation learning. The transition from initial node features to final embeddings capable of supporting classification tasks illustrates the model's capability to transform raw data into actionable insights. Finally, the model is optimized via supervised learning, minimizing the cross-entropy loss between the predicted scores Y and the true labels Y_{True}:

$$L = -\sum_i \sum_C Y_{True,\,ic} \cdot \log\left(Y_{ic}\right) \tag{7}$$

Here, I index over flow nodes, and c over classes.

The parameters (W_{out}, $W_{fh}^{(l)}$, $W_{hf}^{(l)}$, and b_{out}) are optimized to minimize L using gradient-based optimization techniques such as Adam, Adamax, AdaGrad, and RMSProp.

The model consists of 7 HeteroGNN layers with a hidden dimension of 64. Host node features Xh and flow node features Xf are embedded into 32 and 11 dimensions, respectively. Each HeteroGNN layer uses SAGEConv with mean aggregation and LeakyReLU activation. The final flow node embeddings are passed through a linear output layer with softmax activation for classification.

4. METHODOLOGY

4.1 Dataset description

Released in 2020 by the Canadian Institute for Cybersecurity [34], the CIC-Darknet2020 dataset aggregates network traffic from a range of simulation environments, capturing both normal background patterns and malicious behaviors. Its purpose is to provide researchers and practitioners with a comprehensive collection of data on network traffic analysis to support the advancement, assessment, and experimentation of cybersecurity methods. The dataset is structured into two levels. The first level employs a dual-layered strategy to create benign and Darknet network traffic.

The second tier encompasses various traffic scenarios such as Browsing, P2P, Audio-Stream, Transfer, Chat, Email, Video-Stream, and VoIP within the Darknet traffic. To ensure representativeness, the dataset combines previous datasets like ISCXTor-2016 and ISCXVPN-2016. The VPN and Tor traffic is to be merged into the relevant Darknet categories [19].

4.2 Data exploration and preprocessing

•There are a few interesting features we can use for our DHGNN model:

The timestamp we can process to extract information about the day of the week and the time of day. In general, network traffic is seasonal, and connections that occur at night or on unusual days are suspicious.

•Processing IP addresses like 192.168.200.4 can be challenging due to their non-numeric nature and adherence to complex rules. One approach could involve categorizing them into a few groups based on knowledge of our local network configuration. Alternatively, a widely used and more adaptable solution involves converting them into binary representation, where ’192’ would be represented as ’11000000’.

•Flow Duration, the number of packets, and the number of bytes are features that usually display heavy-tailed distributions. Therefore, they will require special processing if that is the case.

The CIC-Darknet2020 dataset contains a diverse set of feature types, including numerical attributes (e.g., flow duration), categorical variables such as Label or Label 1, which serve as the target class and additional data elements like timestamps and IP addresses.

To prepare these heterogeneous features for downstream analysis and model training, we apply a series of representation strategies informed by domain knowledge.

•First, temporal information embedded in the timestamp is used to extract the week of data collection. This feature is subsequently one-hot encoded, and the resulting indicator variables are renamed to enhance interpretability.

•Second, we derive the time of day from each timestamp and apply min–max normalization to scale this value between 0 and 1, ensuring consistency with other continuous inputs.

•Both source and destination IP addresses are transformed using binary encoding. Instead of encoding all 32 bits of each IPv4 address, we retain only the least significant 16 bits, as these carry the most relevant variability for our context. The most significant 16 bits are typically constant commonly representing the internal network prefix 192.168 and therefore contribute limited discriminative value.

•We establish a train/validation/test split using ratios of 80/10/10.

•Finally, we need to address the scaling of three features: flow duration, the number of packets, and the number of bytes. We use **PowerTransformer()** from scikit-learn to modify their distributions.

Table 2 presents the node features utilized in the DHGNN model.

Table 2. DHGNN model node features

Feature Type	Feature Description
Host	Source IP (ipsrc_1 to ipsrc_16)
Host	Destination IP (ipdst_1 to ipdst_16)
Flow	Daytime
Flow	Day of the Week (Monday to Friday)
Flow	Flow Duration
Flow	Flow Packets/s
Flow	Flow Bytes/s
Flow	FIN Flag Count
Flow	SYN Flag Count
Flow	RST Flag Count
Flow	Protocol

4.3 Design model

The design of our model is depicted in Figure 1. The model consists of 7 HeteroGNN layers with a hidden dimension of 64. Host and flow node features are passed through embedding layers, then processed via SAGEConv-based aggregation (mean) and LeakyReLU activation. Separate projection matrices (Whf, Wfh) transform aggregated features. Final flow node embeddings are passed through a linear layer and softmax for classification.

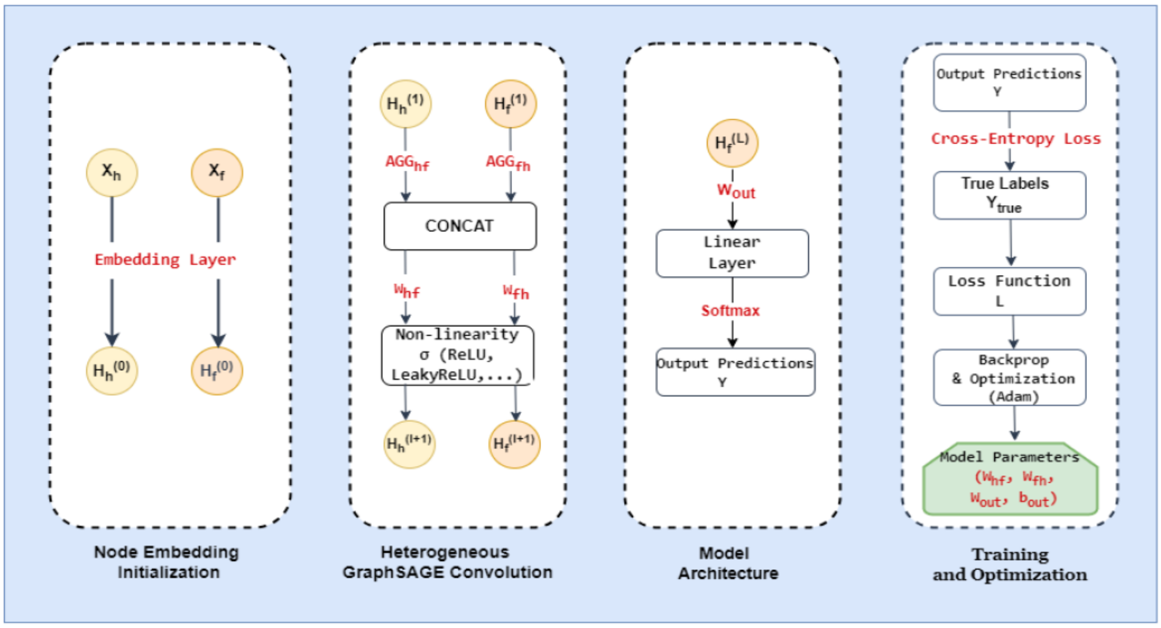


Figure 1. Model design process of the proposed DHGNN

The input features for host nodes X_h and flow nodes X_f are passed through an embedding layer to obtain initial node embeddings $H_h^{(0)}$ and $H_f^{(0)}$. For host nodes, X_h includes network-specific features such as source and destination IP address bits. For flow nodes, X_f incorporates temporal and statistical features including daytime, weekday, flow duration, total bytes and packets transmitted, TCP flag counts, and protocol information. These embedded representations serve as the foundation for subsequent graph neural network processing.

At each HeteroGNN layer l, the node embeddings from the previous layer ($H_h^{(l)}$ and $H_f^{(l)}$) are processed through distinct aggregation functions (AGG_{hf} and AGG_{fh}). The AGG_{hf} function aggregates features from neighboring flow nodes to update host node embeddings, while AGG_{fh} performs the complementary operation by aggregating host node features to update flow node embeddings. These aggregation functions are implemented as SAGEConv (GraphSAGE) layers, which compute a weighted combination of neighboring node features processed by a Nonlinear activation function. This architecture enables effective information propagation between the two node types while maintaining their distinct feature spaces.

The aggregated embeddings from AGG_{hf} and AGG_{fh} are concatenated to form a joint representation. This combined embedding is subsequently transformed through two separate learnable projection matrices, W_{hf} and W_{fh}, which independently process the host-to-flow and flow-to-host aggregated features, respectively. These methods allows the model to learn different patterns from each direction of the relationship. At the same time, it carefully preserves the important network structure and connections that were identified during the earlier combining step.

The projected embeddings are then passed through a non-linear activation function σ (typically ReLU or LeakyReLU) to introduce non-linearity into the transformations. This produces the updated node embeddings $H_h^{(l+1)}$ and $H_f^{(l+1)}$ that will serve as input to the next HeteroGNN layer. After processing through all seven HeteroGNN layers, the final flow node embeddings $H_f^{(L)}$ undergo a linear projection via the output weight matrix W_{out} to generate the model's predictions. The complete layer-wise transformation ensures both local neighborhood information and global graph structure are effectively captured in the final representations.

The projected embeddings are transformed through a final linear layer to produce the output logits. These logits are then normalized using a softmax activation function $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ to obtain the model's final predictions **Y**. This yields a probability distribution over the K target classes, where each element $y_i \in \boldsymbol{Y}$ represents the predicted probability for class i, enabling probabilistic interpretation of the model's outputs. The model's output predictions, called Y are evaluated against the ground truth labels Y_{true}. We use a special formula to measure how wrong the predictions are; this formula is called the cross-entropy loss function or L. The formula is $L = -\sum_{i=1}^{K} y_{true,i} \log(y_i)$. Here, K is the number of classes we are trying to predict. This loss L calculates the difference between the predicted probability distribution and the correct class distribution, with the computed loss value L serving as the optimization objective during model training. The cross-entropy formulation is particularly suitable for classification tasks as it penalizes confident incorrect predictions more heavily while remaining computationally efficient.

The computed loss L is backpropagated through the entire network to obtain gradients with respect to all trainable parameters, including the weight matrices (W_{hf}, W_{fh}, W_{out}) and their corresponding bias terms. These gradients are then used by an optimization algorithm (typically Adam) to update the model parameters through gradient descent. This entire process is repeated for **100 epochs**. One epoch means the model has seen all the training data once. With each epoch, the model's parameters are slowly refined to reduce the classification error. This repeated training helps the model learn better and more discriminative features, which improves its ability to make correct predictions.

Therefore, the theoretical AGG is practically instantiated using SAGEConv with mean aggregation for aggregating neighbor information within each relation type in the heterogeneous graph.

5. EXPERIMENT RESULTS

In the DHGNN classifier, we'll set up four or eight layers of SAGEConv with LeakyRELU for each node type (four layers in the case of traffic classification and eight layers in the case of application classification). Then, a linear layer will produce a four or eight-dimensional vector, with each dimension representing a class. Additionally, we'll train this model in a supervised manner utilizing cross-entropy loss and the Adam optimizer. Next, we specify the heterogeneous GNN, incorporating three parameters: the count of hidden dimensions, the count of output dimensions, and the number of layers.

Hyperparameters optimization for GNN classifiers is crucial, given that it directly affects the model's classification accuracy. Our experiments, detailed in Table 3, thoroughly explore key parameters, assessing accuracy within defined ranges. We find that a network depth of seven layers, trained over 100 epochs with the Adam optimizer and the LeakyReLU activation function, is the ideal configuration.

In the following subsections, we will present results for both traffic classification and application classification scenarios.

Table 3. DHGNN model hyperparameters and exploration range

Hyperparameter	Value Interval	Optimal Value	Description
Device	-	Dynamic	Computed based on CUDA availability; 'cuda' or 'cpu'.
Number of Layers	[3, 4, 5, 6, 7]	7	The number of layers in the model indicates the depth of the network.
Dimension of Hidden Layer	64	64	Fixed dimensionality of the hidden layers in the model.
Output Dimension	8 and 4	8 and 4	The dimensionality of the output layer is fixed at 8 and 4 for this model.
Activation Function	[Tanh, Relu, LeakyRelu]	LeakyRelu	Activation functions are considered, with LeakyReLU chosen for the optimal model.
Learning Rate (LR)	[0.001, 0.002, 0.003, 0.004, 0.005, 0.006]	0.004	Range of learning rates explored, with 0.004 selected as the optimal rate.
Optimizer	[Adam, AdamW, RMSprop, Adamax]	Adam	Different optimization algorithms were considered, with Adam chosen for the optimal model.
Loss Function	Cross Entropy	Cross Entropy	The loss function used for training was not varied in this experimentation.
Total Epochs	[50, 100, 150, 200]	100	Total epochs explored with 101 chosen for the optimal training duration.

5.1 Traffic classification (case of 4 classes)

Our focus was primarily on the initial layer of the CIC-Darknet2020 dataset, which included two primary classes, VPN/Tor Darknet traffic and Non-VPN/Non-Tor benign traffic. The four classes are represented by the pie chart shown in Figure 2, where the counts of each value are as follows: “Non-Tor”: 93309, “Non-VPN”: 23861, “VPN”: 22919, and “Tor”: 1392 occurrences. We assess the performance of the proposed approach using Precision, Recall, F1-score, and Accuracy evaluation metrics.

In traffic classification, the accuracy of our DHGNN model is 99.80%, indicating a very high level of correct predictions. A plot of training loss and validation loss while training the model is shown in Figure 3.

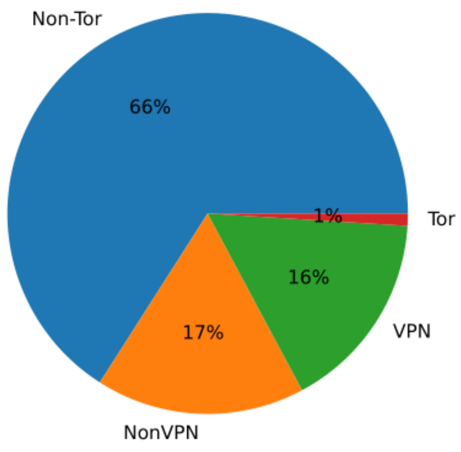


Figure 2. Proportion of each class in the CIC-Darknet2020 dataset – Traffic case

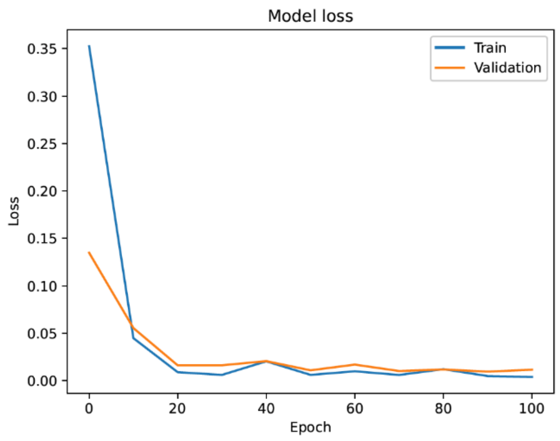


Figure 3. Model’s loss as a function of an epoch during the training process – Traffic case

A plot of the proportion of misclassified samples is given in Figure 4. If we compare this pie chart to the original proportions in the dataset, we see that the model performs better for the majority classes. This is not surprising since minority classes are harder to learn (fewer samples), and not detecting them is less penalizing (with 93309 Non-Tor flows versus 1392 Tor). NonVPN and VPN detection could be improved with techniques such as oversampling and introducing class weights during training.

An analysis of the confusion matrix presented in Figure 5 indicates that the classifier exhibits exceptionally high discriminatory power. The model's performance is characterized by misclassification rates that remain predominantly below 0.5% across all four defined anonymity categories. Despite this robust performance, two specific confusion patterns are discernible. The most significant occurs between Tor and NonVPN traffic, where approximately 10 Tor samples (0.45%) were misclassified. This is likely attributable to overlapping statistical patterns when Tor circuits encapsulate traffic resembling normal encrypted sessions, a challenge compounded by the relatively lower number of Tor samples (n = 121). A second, minor pattern involves a handful of Non-Tor samples being confused with NonVPN and VPN classes (≤ 4 samples each, < 0.2%), potentially stemming from similar feature distributions like packet lengths and timing in various encrypted streams. Overall, the model performs remarkably well, but the Tor vs. NonVPN distinction remains challenging, primarily due to their similar traffic patterns and the imbalance between classes.

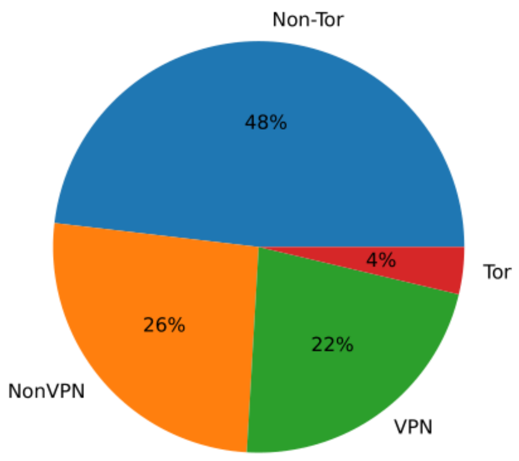
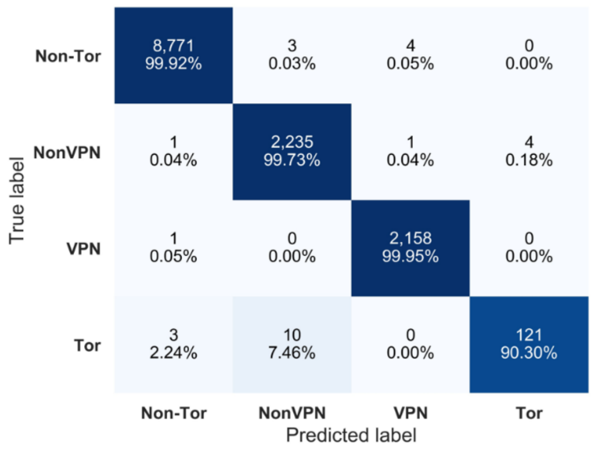


Figure 4. Proportion of each misclassified class – Traffic case



	Non-Tor	NonVPN	VPN	Tor
Non-Tor	8,771 99.92%	3 0.03%	4 0.05%	0 0.00%
NonVPN	1 0.04%	2,235 99.73%	1 0.04%	4 0.18%
VPN	1 0.05%	0 0.00%	2,158 99.95%	0 0.00%
Tor	3 2.24%	10 7.46%	0 0.00%	121 90.30%

Figure 5. Confusion matrix for multi-class flow classification – Traffic case

Table 4 compares the performance of various graph neural network (GNN) models, including ResNet, GIN, GCN, GAT, GraphSAGE, RevGNN, DGNN, and the proposed DHGNN model, on a traffic classification task. The performance metrics reported are accuracy, precision, F1-score, and recall (TPR). The proposed DHGNN classifier achieves the highest accuracy of 0.9980, precision of 0.9898, F1-score of 0.9820, and recall of 0.9322.

Table 4. Performance evaluation of various traffic classification methods

Model	Accuracy	Precision	F1	Recall
DGNN [26]	0.9852	0.9662	0.9488	0.8322
GIN [35]	0.9358	0.8653	0.7569	0.6741
GCN [36]	0.9260	0.8464	0.7108	0.6158
GAT [37]	0.9235	0.8324	0.7021	0.6086
GraphSAGE [38]	0.9556	0.9142	0.8407	0.7791
RevGNN [39]	0.9131	0.8384	0.6348	0.5131
ResNet [40]	0.9043	0.7625	0.7120	0.6991
Our DHGNN	**0.9980**	**0.9898**	**0.9820**	**0.9748**

5.2 Application classification (case of 8 classes)

Our focus now was directed towards the second data layer of the CIC-Darknet2020 dataset, which included data samples classified as Media Streaming (Audio and Video), Live Chat, E-mail, P2P, Voice over IP, File Transfer, and Web Browsing. We employ Precision, Recall, F1-score, and Accuracy evaluation metrics to evaluate the performance of the proposed approach.

Figure 6 shows a curve representing the model's loss as a function of the number of epochs during the training process. The curve starts at a relatively high loss value, indicating that the model's initial predictions were quite inaccurate or far from the true values at the beginning of the training process. In the early stages of training, the loss curve exhibits a steep decline. This rapid decrease in loss suggests that the model is learning quickly and improving its predictions significantly with each epoch. During this phase, the model is making substantial adjustments to its internal parameters to minimize the error between its predictions and the true values. As the training progresses, the rate at which the loss decreases starts to slow down gradually. This indicates that the model is still improving, but the improvements are becoming smaller. In application classification, the accuracy of our DHGNN model is 98.80%, indicating a very high level of correct predictions.

The circular chart given in Figure 7 shows the distribution of each misclassified class in the case of application classification.

Figure 8 presents the confusion matrix for the 8-class application classification task, illustrating the model's evaluation performance by comparing predicted labels against true labels. The matrix shows generally excellent performance, though with interpretable patterns of misclassification. The most common errors are with Video-Streaming. It is sometimes mistaken for Audio-Streaming (75 samples, 2.42%) and Chat (47 samples, 2.02%). This makes sense because a video stream also contains an audio track, and its data can arrive in bursts, similar to a chat application. Similarly, Audio-Streaming shows minor confusion with VoIP, and Email traffic is moderately confused with Browsing (72 samples, 6.33%), reflecting the real-world ambiguity of HTTPS-based webmail services. A notable case is the VOIP class, which has a higher error rate. It is often misclassified as Browsing, Email, or Audio-Streaming. On the other hand, applications with very unique traffic patterns, like P2P and File-Transfer, are almost never confused with anything else. They are identified perfectly. In summary, most of the model's mistakes happen between application types that naturally have very similar network behavior. This shows a real-world challenge in classifying network traffic, not a major problem with the model itself.

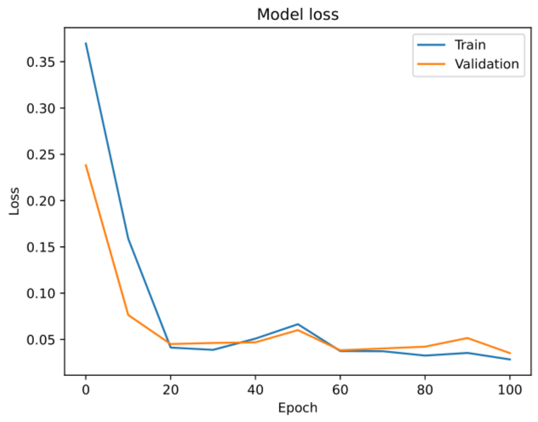


Figure 6. Model's loss as a function of an epoch during the training process – Application case

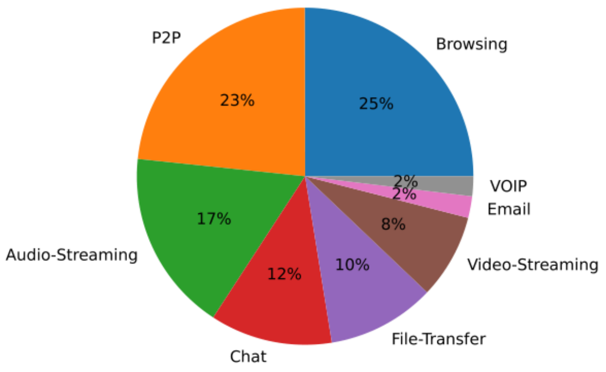


Figure 7. Proportion of each misclassified class – Application case

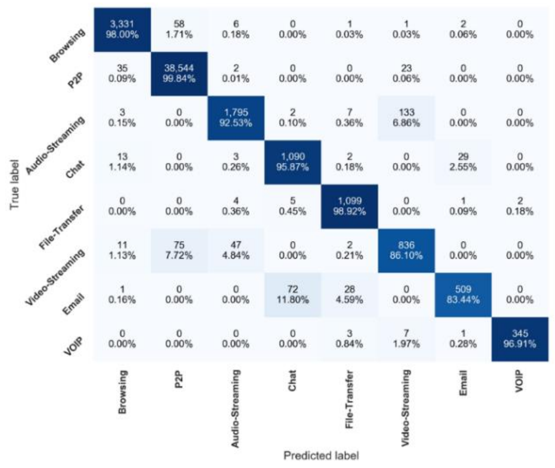


Figure 8. Confusion matrix for multi-class flow classification – Application case

Table 5 presents the recall values of the different models for classifying various applications, such as Audio-streaming, Browsing, Chat, Email, Transfer, P2P, VOIP, and Video-streaming. The proposed DHGNN classifier achieves the highest recall for most applications, including Browsing (0.9800), Chat (0.9587), Transfer (0.9892), P2P (0.9984), VOIP (0.9691), and Video-stream (0.8610). DGNN [26] performs better for Audio-Streaming and Email classification with a recall of 0.9788 and 0.9682, respectively.

For a fair and direct comparison, the proposed DHGNN model is evaluated alongside the DGNN model [26] under identical experimental conditions, including dataset splits, preprocessing steps, and evaluation metrics. The results demonstrate that our model achieves superior performance. The results for other models (e.g., GCN, GAT, GraphSAGE) are provided as a general benchmark from the cited literature and were obtained under different experimental setups.

Table 5. Comparison of application classification methodologies' recall results

Model	Audio-Streaming	Browsing	Chat	Email	Transfer	P2P	VOIP	Video-Streaming
DGNN [26]	**0.9788**	0.7577	0.9458	**0.9682**	0.8804	0.9702	0.9548	0.8557
GIN [35]	0.8928	0.3397	0.8341	0.1015	0.5059	0.7724	0.4706	0.3977
GCN [36]	0.8557	0.0072	0.7995	0.0020	0.3638	0.4968	0.5561	0.0240
GAT [37]	0.8061	0.0228	0.7838	0.1009	0.0731	0.7223	0.2545	0.0217
GraphSAGE [38]	0.9044	0.2423	0.8485	0.7804	0.5364	0.8756	0.8439	0.4438
RevGNN [39]	0.7472	0.0023	0.7723	0.0179	0.0181	0.0124	0.0089	0.0042
ResNet [40]	0.8237	0.7808	0.8946	0.5553	0.6774	0.8898	0.7864	0.6104
DHGNN (Ours)	0.9253	**0.9800**	**0.9587**	0.8344	**0.9892**	**0.9984**	**0.9691**	**0.8610**

Table 6 compares the performance of different methods, including Random Forest (RF), ensemble methods (RF+KNN+DT), ResGAT, DeepImage, CNN+LSTM, and the proposed DHGNN and DGNN models, on an application classification task. The metrics reported are accuracy, F1-score, and the model used. The proposed DHGNN model achieves the highest F1-score of 0.9879 and an accuracy of 0.9880, outperforming the other methods. DGNN also performs well with an accuracy of 0.9906 and an F1-score of 0.9569.

Table 6. Performance comparison of different application classification models

Method	Accuracy	F1-Score	Model
[41]	—	0.922	RF
[42]	0.9788	0.94	RF+KNN+DT
[43]	—	0.8807	ResGAT
[19]	0.86	0.86	DeepImage
[44]	0.9222	0.92	CNN+LSTM
[45]	0.8599	0.86	RF
[26]	**0.9906**	0.9569	DGNN
Our	0.9880	**0.9879**	DHGNN

6. CONCLUSIONS

In this paper, we propose an approach for detecting and classifying Darknet traffic using a Heterogeneous Graph Neural Network (DHGNN). The classifier is evaluated using the CIC-Darknet2020 dataset, which includes four traffic types (Tor, Non-Tor, VPN, Non-VPN) and eight application categories (Audio-Stream, Browsing, Chat, E-mail, P2P, Transfer, Video-Stream, VOIP). Our evaluation tests show that our model, called DHGNN, performs better than other methods at classifying Darknet traffic. This proves it has great potential to improve network security by effectively detecting this hidden, often malicious traffic. For future work, we plan to continue in a few directions: We will test other types of Graph Neural Network (GNN) classifiers. We will investigate hybrid models that mix different GNN architectures. The goal is to combine their individual strengths to create a model that is both stronger and more flexible. We also understand that for cybersecurity, it is not enough for a model to be accurate—it must also be understandable. Therefore, we will develop methods to improve the interpretability of DHGNN. We want to make its decision-making process clear and transparent. This focus on explainability is crucial for building trust in the system, ensuring we can see how it works, and for gaining a deeper understanding of the threats it finds.

REFERENCES

[1] Chertoff, M. (2017). A public policy perspective of the Dark Web. Journal of Cyber Policy, 2(1): 26-38. https://doi.org/10.1080/23738871.2017.1298643

[2] Nazah, S., Huda, S., Abawajy, J., Hassan, M.M. (2020). Evolution of dark web threat analysis and detection: A systematic approach. IEEE Access, 8: 171796-171819. https://doi.org/10.1109/ACCESS.2020.3024198

[3] Averin, A., Samartsev, A., Sachenko, N. (2020). Review of methods for ensuring anonymity and de-anonymization in blockchain. In 2020 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS), Yaroslavl, Russia, pp. 82-87. IEEE. https://doi.org/10.1109 /itqmis51053.2020.9322974

[4] Ezra, P.J., Misra, S., Agrawal, A., Oluranti, J., Maskeliunas, R., Damasevicius, R. (2021). Secured communication using virtual private network (VPN). In Cyber Security and Digital Forensics. Lecture Notes on Data Engineering and Communications Technologies, 73: 309-319. https://doi.org/10.1007/978-981-16-3961-6_27

[5] dos Santos Horta, M.S.P. (2022). Tor K-anonymity against deep learning watermarking attacks. Master's thesis. Universidade NOVA de Lisboa (Portugal).

[6] Alharbi, A., Faizan, M., Alosaimi, W., Alyami, H., Agrawal, A., Kumar, R., Khan, R.A. (2021). Exploring the topological properties of the Tor Dark Web. IEEE Access, 9: 21746-21758. https://doi.org/10.1109/ACCESS.2021.3055532

[7] Saleem, J., Islam, R., Islam, Z. (2024). Darknet traffic analysis: A systematic literature review. IEEE Access, 12: 42423-42452. https://doi.org/10.1109/ACCESS.2024.3373769

[8] Almomani, A. (2025). Darknet traffic analysis, and classification system based on modified stacking ensemble learning algorithms. Information Systems and E-Business Management, 23: 209-240. https://doi.org/10.1007/s10257-023-00626-2

[9] Dutta, P., Mayilvaghanan, K., Sinha, P., Dukkipati, A. (2024). Deep representation learning for prediction of temporal event sets in the continuous time domain. In 15th Asian Conference on Machine Learning (ACML), Istanbul, Turkey, pp. 343-358.

[10] Ban, T., Eto, M., Guo, S., Inoue, D., Nakao, K., Huang, R. (2015). A study on association rule mining of darknet big data. In International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, pp. 1-7. https://doi.org/10.1109/IJCNN.2015.7280818

[11] Xu, S., Han, J., Liu, Y., Liu, H., Bai, Y.(2025). Few-shot traffic classification based on autoencoder and deep graph convolutional networks. Scientific Reports, 15(1): 8995. https://doi.org/10.1038/s41598-025-94240-6

[12] Maddumala, V.R., R, A. (2020). A weight based feature extraction model on multifaceted multimedia bigdata using convolutional neural network. Ingénierie des Systèmes d'Information, 25(6): 729-735. https://doi.org/10.18280/isi.250603

[13] Habibi Lashkari, A., Kaur, G., Rahali, A. (2020). Didarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning. In Proceedings of the 2020 10th International Conference on Communication and Network Security (ICCNS), New York, USA, pp. 1-13. https://doi.org/10.1145/3442520.3442521

[14] Kisanga, P., Woungang, I., Traore, I., Carvalho, G.H. (2023). Network anomaly detection using a graph neural network. In 2023 International Conference on Computing, Networking and Communications (ICNC), Hawaii, USA, pp. 61-65. https://doi.org/10.1109/ICNC57223.2023.10074111

[15] Purnama, S.R., Istiyanto, J.E., Amrizal, M.A., Handika, V., Rochman, S., Dharmawan, A. (2022). Inductive Graph Neural Network with causal sampling for IoT network intrusion detection system. In 2022 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM), Surabaya, Indonesia, pp. 241-246. https://doi.org/10.1109/CENIM56801.2022.10037304

[16] Lin, H.C., Wang, P., Lin, W.H., Lin, Y.H., Chen, J.H. (2023). Graph neural network for malware detection and classification on renewable energy management platform. In 2023 IEEE 5th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), Taiwan, pp. 164-166. https://doi.org/10.1109/ECBIOS57802.2023.10218478

[17] Bilot, T., El Madhoun, N., Al Agha, K., Zouaoui, A. (2023). Graph neural networks for intrusion detection: A survey. IEEE Access, 11: 49114-49139. https://doi.org/10.1109/ACCESS.2023.3275789

[18] Zhai, Z., Li, P., Feng, S. (2023). State of the art on adversarial attacks and defenses in graphs. Neural Computing and Applications, 35(26): 18851-18872. https://doi.org/10.1007/s00521-023-08839-9

[19] Sarkar, D., Vinod, P., Yerima, S.Y. (2020). Detection of Tor traffic using deep learning. In 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), Antalya, Turkey, pp. 1-8. https://doi.org/10.1109/AICCSA50499.2020.9316533

[20] Lashkari, A.H., Gil, G.D., Mamun, M.S.I., Ghorbani, A.A. (2017). Characterization of Tor traffic using time based features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017), Porto, Portugal, pp. 253-262. https://doi.org/10.5220/0006105602530262

[21] Iliadis, L.A., Kaifas, T. (2021). Darknet traffic classification using machine learning techniques. In 10th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, pp. 1-4. https://doi.org/10.1109/MOCAST52088.2021.9493386

[22] Demertzis, K., Tsiknas, K., Takezis, D., Skianis, C., Iliadis, L. (2021). Darknet traffic big-data analysis and network management for real-time automating of the malicious intent detection process by a weight agnostic neural networks framework. Electronics, 10(7): 781. https://doi.org/10.3390/electronics10070781

[23] Sarwar, M.B., Hanif, M.K., Talib, R., Younas, M., Sarwar, M.U. (2021). Darkdetect: Darknet traffic detection and categorization using modified convolution-long short-term memory. IEEE Access, 9: 113705-113713. https://doi.org/10.1109/ACCESS.2021.3105000

[24] Marim, M.C., Ramos, P.V.B., Vieira, A.B., Galletta, A., Villari, M., de Oliveira, R.M., Silva, E.F. (2023). Darknet traffic detection and characterization with models based on decision trees and neural networks. Intelligent Systems with Applications, 18: 200199. https://doi.org/10.1016/j.iswa.2023.200199

[25] Alimoradi, M., Zabihimayvan, M., Daliri, A., Sledzik, R., Sadeghi, R. (2022). Deep neural classification of darknet traffic. In Artificial Intelligence Research and Development, 356: 105-114. https://doi.org/10.3233/FAIA220323

[26] Zhu, Y., Tao, J., Wang, H., Yu, L., et al. (2023). DGNN: Accurate darknet application classification adopting attention graph neural network. IEEE Transactions on Network and Service Management, 21(2): 1660-1671. https://doi.org/ 10.1109/TNSM.2023.3344580

[27] Saheed, K., Henna, S. (2023). Heterogeneous graph transformer for advanced persistent threat classification in wireless networks. In IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Dresden, Germany, pp. 15-20. https://doi.org/10.1109/NFV-SDN59219.2023.10329745

[28] Mika, G.P., Bouzeghoub, A., Wegrzyn-Wolska, K., Neggaz, Y.M. (2023). HGExplainer: Explainable heterogeneous graph neural network. In 2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Venice, Italy, pp. 221-229. https://doi.org/10.1109/WI-IAT59888.2023.00035

[29] Fu, X., Zhang, J., Meng, Z., King, I. (2020). MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In Proceedings of the Web Conference 2020, Taipei, Taiwan, pp. 2331-2341. https://doi.org/10.1145/3366423.3380297

[30] Lou, X., Liu, G., Li, J. (2023). ASIAM-HGNN: Automatic selection and interpretable aggregation of meta-path instances for heterogeneous graph neural network. Computing and Informatics, 42(2): 257-279.

https://doi.org/10.31577/cai_2023_2_257

[31] Labonne, M. (2023). Hands-on Graph Neural Networks Using Python: Practical Techniques and Architectures for Building Powerful Graph and Deep Learning Apps with Pytorch. Birmingham, UK: Packt Publishing.

[32] Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M. (2018). Modeling relational data with graph convolutional networks. In the Semantic Web: 15th International Conference ESWC 2018, Heraklion, Crete, Greece, pp. 593-607. https://doi.org/10.1007/978-3-319-93417-4_38

[33] Yu, P., Fu, C., Yu, Y., Huang, C., Zhao, Z., Dong, J. (2022). Multiplex heterogeneous graph convolutional network. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 2377-2387. https://doi.org/10.1145/3534678.3539482

[34] Canadian Institute for Cybersecurity. (n.d.). Canadian Institute for Cybersecurity. University of New Brunswick. https://www.unb.ca/cic/.

[35] Xu, K., Hu, W., Leskovec, J., Jegelka, S. (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826. https://doi.org/10.48550/arXiv.1810.00826

[36] Kipf, T.N., Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907

[37] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903. https://doi.org/10.48550/arXiv.1710.10903

[38] Hamilton, W., Ying, Z., Leskovec, J. (2017). Inductive representation learning on large graphs. In 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[39] Li, G., Müller, M., Ghanem, B., Koltun, V. (2021). Training graph neural networks with 1000 layers. In International Conference on machine learning PMLR, pp. 6437-6449.

[40] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, pp. 770-778.

[41] Rust-Nguyen, N., Sharma, S., Stamp, M. (2023). Darknet traffic classification and adversarial attacks using machine learning. Computers Security, 127: 103098. https://doi.org/10.1016/j.cose.2023.103098

[42] Mohanty, H., Roudsari, A.H., Lashkari, A.H. (2022). Robust stacking ensemble model for darknet traffic classification under adversarial settings. Computers & Security, 120: 102830. https://doi.org/10.1016/j.cose.2022.102830

[43] Chang, L., Branco, P. (2021). Graph-based solutions with residuals for intrusion detection: The modified E-GraphSAGE and E-ResGAT algorithms. arXiv preprint arXiv:2111.13597. https://doi.org/10.48550/arXiv.2111.13597

[44] Lan, J., Liu, X., Li, B., Li, Y., Geng, T. (2022). DarknetSec: A novel self-attentive deep learning method for darknet traffic classification and application identification. Computers & Security, 116: 102663. https://doi.org/10.1016/j.cose.2022.102663

[45] Karagöl, H., Erdem, O., Akbas, B., Soylu, T. (2022). Darknet traffic classification with machine learning algorithms and SMOTE method. In 2022 7th International Conference on Computer Science and Engineering (UBMK), Diyarbakir, Turkey, pp. 374-378. https://doi.org/10.1109/UBMK55850.2022.9919462