Exploring Temporal GNN Embeddings for Darknet Traffic Analysis

Exploring Temporal GNN Embeddings for Darknet Traffic Analysis

Luca Gioacchini
Politecnico di Torino
Turin, Italy
luca.gioacchini@polito.it

Andrea Cavallo
Politecnico di Torino
Turin, Italy
andrea.cavallo@polito.it

Marco Mellia
Politecnico di Torino
Turin, Italy
marco.mellia@polito.it

Luca Vassio
Politecnico di Torino
Turin, Italy
luca.vassio@polito.it

ABSTRACT

Network Traffic Analysis (NTA) serves as a foundational tool for characterizing network entities and uncovering suspicious traffic patterns, thereby enhancing our understanding of network operations and security. As successfully done in other domains, due to the scarcity of labelled data, Deep Learning (DL)-based solutions for NTA have started adopting a 2-stage approach; (i) a self-supervised upstream task generates compact and information-rich representations (embeddings) of network data without the need for a ground truth; (ii) the embeddings serve as input to specialized models for downstream tasks (supervised or unsupervised) – e.g. traffic classification or anomaly detection. Since graphs are intuitive representations of network traffic, in this work, we explore the potential of temporal Graph Neural Networks (tGNNs) in generating intermediate embeddings in a self-supervised fashion. We assess the quality of such embeddings by solving a host classification problem in a darknet traffic scenario. We evaluate static and temporal GNNs over a month-long period of traffic traces. We find that the inclusion of node features and temporal aspects in the model, together with an incremental training approach, allows for an accurate description of host activity dynamics and enables the creation of 2-stage NTA pipelines.

CCS CONCEPTS

• **Networks** → **Network monitoring**; • **Security and privacy** → **Network security**.

KEYWORDS

Darknets; Artificial Intelligence; Graph Neural Networks; Cybersecurity; Network Monitoring; Embeddings

ACM Reference Format:
Luca Gioacchini, Andrea Cavallo, Marco Mellia, and Luca Vassio. 2023. Exploring Temporal GNN Embeddings for Darknet Traffic Analysis. In *Proceedings of the 2nd Graph Neural Networking Workshop 2023 (GNNet '23), December 8, 2023, Paris, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3630049.3630175

GNNet '23, December 8, 2023, Paris, France

ACM ISBN 979-8-4007-0448-2/23/12.
https://doi.org/10.1145/3630049.3630175

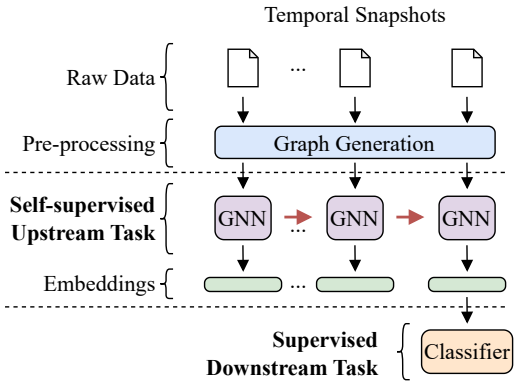


Figure 1: 2-stage DL pipeline for classification tasks.

1 INTRODUCTION

In the context of Network Traffic Analysis (NTA), Deep Learning (DL) techniques have emerged as powerful tools to address traffic classification problems [2, 18], anomaly detection [22] and exploratory traffic analysis [7, 13, 20], among others. Thanks to the availability of large data collections and user-friendly frameworks, such techniques are becoming fundamental for characterizing and comprehending the actions of network entities and unveiling significant patterns. However, differently from other domains, NTA applications typically present limited availability of labelled datasets and fast and complex dynamic data with evolving structures which complicate the training of DL models.

For this reason, recent DL-based solutions employ a 2-stage pipeline [8, 9, 15]. In the first stage, a *self-supervised upstream task* generates compact representations of input data in a latent space (i.e. embeddings) without the need for ground truth. Then, in the second stage, specialized machine learning or DL models operate on these embeddings to solve specific problems, referred to as *downstream tasks* (e.g. classification, clustering, anomaly detection). This approach is motivated by the expectation that embeddings that reflect the available relations and interactions between data points contain valuable and representative information that can be

leveraged to address several downstream problems, even though the embeddings are not specifically tailored for the final task.

Several techniques have been adopted to generate informative embeddings, i.e. the first stage of the pipeline. Many works adopt traditional feature engineering approaches and process the resulting datasets through (sparse) Autoencoders [12, 13, 16] or traditional Convolutional Neural Networks [1, 19]. Other works identify analogies between sequences of packets (or flows) and words in text documents. Hence, they generate embeddings borrowing techniques from the Natural Language Processing (NLP) field [6, 8, 20]. In the latter case, the adoption of techniques belonging to domains different from network traffic, like NLP, often requires an additional level of abstraction leading to hard-to-interpret solutions for networking experts. In fact, modelling network traffic as a graph is more intuitive and straightforward and some works started relying on Graph Neural Networks (GNNs) to capture the complex relation patterns of the network traffic. Indeed, GNNs are neural network architectures that directly operate on graphs and capture not only entity-specific information, but also connectivity patterns. Therefore, they represent a powerful tool for modelling network behaviour. Specifically, end-to-end GNN-based architectures are used to classify network packets [3, 23], and GNN-based autoencoders are applied to traffic flows classification [10, 11].

In this paper, we aim to generate robust embeddings (stage 1) that represent the activity of hosts sending traffic to darknet addresses. We propose the usage of *temporal GNNs* (tGNNs) as embedding generators to capture the complex spatial and temporal patterns found in network traffic. We design a downstream host classification task (stage 2) to evaluate the goodness of the embeddings, as sketched in Figure 1.

Darknets are sensors that observe traffic received by networks that are announced on the Internet but host neither production services nor client hosts. Specifically, they are commonly used to monitor incoming and potentially malicious attacks, since any packet reaching a darknet address is unsolicited. We model darknet traffic as a bipartite graph in which sender nodes (identified by their IP addresses) are connected to nodes representing the destination TCP ports. We process the graph to generate host embeddings using three different GNNs: a static Graph Convolutional Network (GCN) [14] adapted to dynamic scenarios through incremental training (*i-GCN*), a temporal GNN (*GCN-GRU* [24]) and the same temporal GNN trained incrementally (*i-GCN-GRU*). Additionally, to exploit the full potential of GNNs, we enrich the graph with node feature information related to the amount and type of traffic generated (received) by each host (port).

We evaluate the generated embeddings through a downstream node classification task where we label senders according to the available ground truth. We find that (i) node features are essential to map hosts belonging to the same class in the same region of the latent space (< 0.50 of average F1-Score without node features); (ii) the temporal GNN (GCN-GRU) better extracts the dynamics of host activities (0.75 of average F1-Score); (iii) the incrementally-trained temporal GNN (i-GCN-GRU) better follows the fast-changing behaviours of hosts improving the classification performance up to 0.80 of average F1-Score.

We show that the modelling of network traffic as a graph and the adoption of tGNNs extract meaningful host activity patterns and generate robust host representations, for which we envision several applications to supervised and unsupervised tasks (e.g. clustering), which can significantly advance the understanding and analysis of network behaviour.

2 HOST EMBEDDINGS WITH GNNS

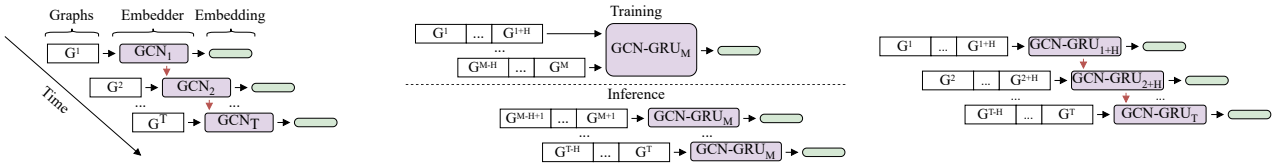
We define $\mathcal{V}_A^t, \mathcal{V}_B^t$ as two disjoint sets of nodes active in snapshot t and $\mathcal{E}_{A,B}^t$ as the set of edges that in snapshot t link nodes in $\mathcal{V}_A^t$ with nodes in $\mathcal{V}_B^t$. We define a dynamic bipartite graph $\mathcal{G} = \{\mathcal{G}^t\}_{t=1}^T$ as a sequence of T static bipartite graphs $\mathcal{G}^t = (\mathcal{V}_A^t, \mathcal{V}_B^t, \mathcal{E}_{A,B}^t)$. More specifically, an edge $\epsilon = (v_A, v_B, w) \in \mathcal{E}_{A,B}^t$ indicates that there exists a connection between nodes $v_A \in \mathcal{V}_A^t$ and $v_B \in \mathcal{V}_B^t$ with weight w. Each node $v \in \mathcal{V}_A^t \cup \mathcal{V}_B^t$ can be associated with a feature vector $\mathbf{x}_v^t \in \mathbb{R}^F$, where F is the number of features. In our case, $\mathcal{V}_A^t$ contains the external hosts sending packets to the darknet. An edge $\epsilon = (v_A, v_B, w)$ indicates that, in snapshot t, v_A sent w packets toward the destination port v_B.

GNNs [21] are deep learning models that exploit the structural information of a graph to generate meaningful representations for its nodes. We train the GNNs with a self-supervised task, i.e. link prediction. Specifically, at each snapshot t, the GNNs generate node embeddings and estimate the likelihood of the connections $\mathcal{E}_{A,B}^t$. We employ a set of non-existing edges as negative examples for training purposes. This is framed as a binary classification task, i.e. classify each edge as existing or non-existing.

Here we provide details on the three GNNs we use in our experiments. All of them are based on the Graph Convolutional Network (GCN) [14] with L layers. At each layer $l \in [2, L]$, a GCN receives in input the representations at the previous layer z_v^{l-1} for each node $v \in \mathcal{V}_A \cup \mathcal{V}_B$ and it generates the updated z_v^l. Specifically, it first transforms the node representations through a multiplication with a learnable weight matrix W^l and then produces the output representation for a node u as the weighted sum of the representations of itself and its neighbours, i.e. $z_u^l = \sum_{v \in \mathcal{N}'(u)} \hat{w_{uv}} W^l z_v^{l-1}$, where $\mathcal{N}'(u)$ is the set containing node u and its neighbours and $\hat{w_{uv}}$ is the weight of the edge between u, v normalized such that the sum of the weights of the edges of node u equals 1. At the first layer, the input node representations are their features, i.e. $z_v^1 = x_v$. The last output layer of the GNN $z_u^L \in \mathbb{R}^E$ is the embedding of node u, where E is the embedding size. We omit time indexes t for simplicity.

i-GCN: Adapting static GCN to dynamic scenarios. Since the GCN is a *static* model, we adapt it to the time-evolving scenario of darknet traffic through incremental training, as illustrated in Figure 2a. Specifically, we train a GCN on the first graph snapshot $\mathcal{G}^{t=1}$ (obtaining GCN_1) and produce the embeddings for the active nodes. Then, for each subsequent snapshot $t \in [2, T]$, we fine-tune the pre-trained model GCN_{t-1} on the graph snapshot $\mathcal{G}^t$ and we obtain GCN_t. The resulting embeddings, thus, include past and latest information.

GCN-GRU: Temporal GNN. To model the graph structure and the dynamics of a temporal network, GCN-GRU [24] (a common tGNN) couples GNNs with recurrent neural networks, a



(a) Incremental GCN (i-GCN) (b) Traditional GCN-GRU (c) Incremental GCN-GRU (i-GCN-GRU)

Figure 2: Different training strategies for Graph Neural Networks models.

Table 1: Considered node features.

Host Node Features		Port Node Features	
#Contacted darknet ports		#Hosts contacting a port	
Stats(#Packets per darknet port)		Stats(#Packets per source host contacting a port)	
#Contacted darknet hosts		0-valued dummy feature	
Stats(#Packets per darknet host)		0-valued dummy feature	
Packets per source host	Stats(Size) Stats(TTL) Stats(MSS) Stats(WIN) Stats(TS)	Packets per darknet port	Stats(Size) Stats(TTL) Stats(MSS) Stats(WIN) Stats(TS)

Table 2: Dataset and Ground Truth overview.

	Total			Daily (Avg.)		
	Hosts	Ports	Packets	Hosts	Ports	Packets
Mirai-like	16 147	2 094	1 982 205	2 158	95	63 942
Brute-forcers	976	9 791	10 191 173	246	1 020	3 28 747
Spammers	1 014	49 783	4 891 353	252	7 048	157 785
Shadowserver	289	42	218 443	287	38	7 046
Driftnet	252	9 246	564 854	252	2 247	18 221
Internetcensus	271	252	213 909	224	234	6 900
Censys	329	65 069	3 400 900	245	12 455	109 706
Rapid7	344	139	60 469	283	25	6 046
Onyphe	115	186	39 030	97	158	1 393
Netsystems	45	199	226 559	43	191	8 391
Shodan	36	1 232	320 861	31	1 007	10 350
Exploiters	430	33	148 210	27	8	4 780
Securitytrails	18	207	107 826	17	15	3 478
Intrinsec	12	8	9 403	9	6	303
Unknown	39 828	65 535	39 654 818	3 594	49 461	1 279 188
Total	60 106	65 535	62 030 013	7 566	53 638	2 000 968

popular tool to deal with time-evolving data. Specifically, it employs Gated Recurrent Units (GRUs) [5], which rely on a memory to keep past information and merge it with new incoming data. More in detail, the GCN-GRU applies a GCN to H subsequent graph snapshots independently and forwards their outputs through a GRU to model temporal behaviours. Formally, $Z^t = \text{GRU}\left(\text{GCN}(\mathcal{G}^{t-H}), \ldots, \text{GCN}(\mathcal{G}^{t-1}), \text{GCN}(\mathcal{G}^t)\right)$, where $Z^t \in \mathbb{R}^{|\mathcal{V}_A^t \cup \mathcal{V}_B^t| \times E}$ is the matrix containing the embeddings for all the nodes active in snapshot t.

In Figure 2b we overview the training (top) and inference (bottom) phases to produce node embeddings. We use the first M snapshots to train the GCN-GRU (GCN-GRU$_M$). Notice that in this way we train the model with $M - H$ sequences of temporal graphs of length H. Then, we freeze it and use GCN-GRU$_M$ to compute embeddings for nodes active in each snapshot $t \in [M+1, T]$.

i-GCN-GRU: Incremental extension of tGNN. Finally, we extend the incremental training approach to the tGNN to better follow the fast-changing dynamics of host activity patterns and produce more robust embeddings. In Figure 2c we overview the incremental training approach. At each new time snapshot t, we generate the new GCN-GRU$_t$ by updating GCN-GRU$_{t-1}$. Notice that we update both the GCN and GRU layers. Differently from i-GCN, we produce embeddings by feeding the model with the current graph snapshot and H past graphs to exploit the GRU memory.

3 DARKNET TRAFFIC

Darknets are sets of IP addresses announced on the Internet but without hosting any services. Thus, all received traffic is unsolicited. They collect large-scale Internet scans and represent a valuable source of information for cybersecurity. In this work, we collect data from a /24 darknet in our university campus network for 31 days (2021-12-01 to 2021-12-31). We focus on TCP traffic, which accounts for 93.7% of traffic, and remove hosts that send less than 5 packets per day [8]. We observe 60 106 remaining hosts sending more than 62 million packets in a month.

3.1 Darknet traffic as a bipartite graph

We consider each day of our collection as a snapshot. According to definitions in Section 2, let $\mathcal{V}_A^t$ be the set of hosts targeting darknets at snapshot t and let $\mathcal{V}_B^t$ be the set of the 2 500 most contacted darknet ports at snapshot t plus one additional node representing all the remaining ports [8].

We obtain the dynamic bipartite graph $\{\mathcal{G}^t\}_{t=1}^{T=31}$ which has 6 392 average nodes per snapshot and 49 198 average edges per snapshot.

We compute a set of features for both host nodes and ports. Thus, each node is associated with a feature vector $\mathbf{x}_v^t$ of size 37, which summarizes the traffic intensity and type as detailed in Table 1. The function Stats$(\cdot)$ extracts the sum, minimum, maximum, average and standard deviation of the provided entity.

3.2 Ground Truth

To perform classification as downstream task, we generate a ground truth for hosts considering four data sources: (i) ground truth available from [8] (ii) presence of fingerprints of Mirai-like malwares observed in received packets [4]; (iii) information from a public

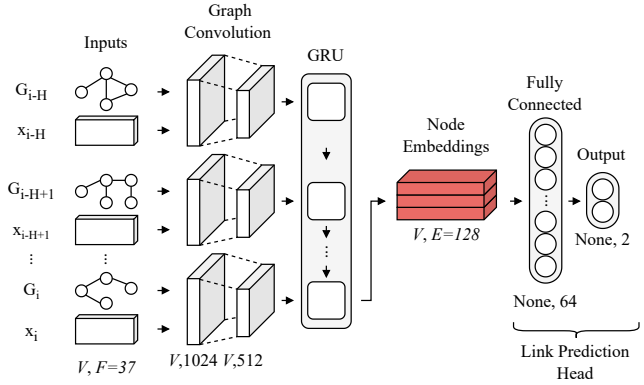


Figure 3: Adopted GCN-GRU architecture.

repository of acknowledged scanners[1], i.e. non-hostile hosts performing scanning activities or providing services like search engines; (iv) expert labels (brute-forcer, spammer and exploiter) based on activities the same host performs on a honeypot. The resulting ground truth labels 34% of the hosts, responsible for 36% of the total traffic, into 14 strongly unbalanced classes. We mark all the remaining hosts as *Unknown*. We report details for each of the classes in our dataset in Table 2. These statistics suggest distinct behaviours characterizing different classes. Non-hostile groups (e.g. Shadowserver, Rapid7) likely engage in (i) vertical scans, targeting a limited set of ports with under 300 000 daily packets, possibly running routine cybersecurity scans, and (ii) horizontal scans (e.g. Shodan, Driftnet), covering large port ranges with less than 600 000 monthly packets, probably surveying TCP port usage. In contrast, malicious classes (e.g. Mirai-like, Brute-forcers, Spammers) appear to conduct massive scans, sending millions of monthly packets.

4 VALIDATING THE EMBEDDINGS

Given our assumption that good embeddings can serve any kind of specialized model for any kind of task, we solve the supervised classification problem of identifying the 14 classes described in Section 3.2.

We rely on a k-Nearest-Neighbour (kNN) classifier, which assigns the most frequent label among the k nearest neighbours in the embedding space. Thus, the closer the embeddings of hosts engaged in similar activities, the higher the classification performance.

We use the first 20 days of our traffic to train the models and test the classification performance on the subsequent 11 days. Since the *Unknown* class includes nodes whose characteristics we cannot verify, we do not report classification metrics for them, but we consider samples belonging to this class when computing the embeddings neighbourhood.

4.1 Experimental settings

After a parameter tuning stage, for which we omit the details for the sake of brevity, we design the model architecture as follows: in all models, the graph convolution modules have three layers (L=3) of 37 (input features), 1024 and 512 neurons respectively; for GCN and i-GCN, we add a fully connected layer at the end with size $E = 128$; for GCN-GRU variants (overviewed in Figure 3), the GRU outputs embeddings sized $E = 128$. The prediction head of all models consists of a hidden layer with 64 neurons and an output layer for the pretext task with 2 neurons (*existing* or *non-existing* edge) on which we apply the Negative Log Likelihood loss function. Note that for inference we do not use the link prediction head.

We train the GCN and GCN-GRU for 50 epochs with an early stopping condition of 3 iterations as patience[2]. For incremental trainings, we train and update the weights for 1 epoch on each snapshot if node features are present, and for 5 epochs otherwise.

Unless otherwise specified, we set the GRU history $H = 5$ and $k = 3$ for the kNN classifier.

We develop all the models using the Python library PyTorch and run the experiments on a Tesla V100-PCIE-16GB. We hope our results and methodological insights can inspire the application of temporal GNN to the analysis of other network traffic traces too. For that, we release our source code and the dataset used in the paper.[3].

4.2 NLP embeddings as baseline

As baseline, we compare GNN-based embeddings with our previous approach i-DarkVec [8], which relies on Word2Vec [17] to create the intermediate embeddings for hosts. We assume that the reader is familiar with Word2Vec and provide a brief overview of i-DarkVec. In a nutshell, at each snapshot, we group packets addressed to the same darknet TCP port and extract the sequence of senders (i.e. the source hosts generating them) as they reach each port. Analogously to NLP, hosts represent "words", whereas ports represent "sentences". We feed the generated corpus as input to Word2Vec. i-DarkVec produces contextual host embeddings such that hosts co-occurring in time when targeting similar ports appear close in the latent space. The kNN classifier on these embeddings achieves 0.77 ± 0.02 of average F1-Score over the 11 testing days.

5 EXPERIMENTAL RESULTS

In Table 3 we report the per-class average F1-Score and standard deviation over the 11 test snapshots. Classes are ordered by decreasing support size. The support size counts host appearances for each class over the 11 test snapshots (possibly with repetitions).

Firstly, we focus on the macro average F1-Score. We observe that (i) tGNN without node features fails to capture meaningful dynamics from darknet traffic (average F1-Score < 0.50 for all the models); (ii) associating graph nodes with features empowers the tGNN to generate more informative embeddings. The average F1-Score improves by up to ≈ 0.39.

Focusing on single classes, we observe that (i) since darknet implies a low level of interaction with senders, the classes deduced from honeypot (i.e. Brute-forcer, Spammer, and Exploiter) exhibit different patterns resulting in weak classification performance (F1-Score < 0.65); (ii) for the other classes, all GNNs achieve an average per-class F1-Score > 0.80.

Comparing standard versus incremental models, we observe that (i) a static model (GCN) with standard training over more epochs

[1] https://gitlab.com/mcollins_at_isi/acknowledged_scanners

[2] Number of epochs with no improvement after which we stop the training.

[3] https://github.com/SmartData-Polito/gnn-for-darknet.git

Table 3: Average F1-Score and standard deviation for the 3-Nearest-Neighbors classifier applied on the host embeddings generated by different models. The best results for each task are in bold, and results within the standard deviation interval of the best are in blue.

		(t)GNN without Node Features				(t)GNN with Node Features			
	Support	**GCN**	**i-GCN**	**GCN-GRU**	**i-GCN-GRU**	**GCN**	**i-GCN**	**GCN-GRU**	**i-GCN-GRU**
Mirai-like	22 878	0.63±0.03	0.76±0.02	0.69±0.03	0.72±0.02	**0.98±0.00**	**0.98±0.00**	0.86±0.04	**0.98±0.00**
Brute-forcer	3 530	0.10±0.04	0.53±0.03	0.53±0.03	0.52±0.01	0.62±0.03	0.61±0.03	0.61±0.03	**0.63±0.03**
Spammer	3 187	0.14±0.04	0.23±0.04	0.29±0.04	0.19±0.07	**0.47±0.06**	0.46±0.06	0.42±0.08	**0.47±0.08**
Shadowserver	3 149	0.14±0.13	0.49±0.15	0.64±0.07	0.49±0.16	0.91±0.01	0.90±0.02	0.95±0.01	**0.96±0.01**
Driftnet	2 772	0.77±0.09	0.73±0.06	0.87±0.05	0.90±0.02	0.88±0.04	0.85±0.05	**0.97±0.02**	**0.97±0.02**
Internetcensus	2 335	0.26±0.12	0.33±0.08	0.68±0.05	0.47±0.13	0.64±0.12	0.58±0.14	0.90±0.04	**0.92±0.02**
Censys	1 974	0.53±0.06	0.60±0.05	0.59±0.05	0.60±0.03	0.89±0.03	0.88±0.03	0.91±0.02	**0.92±0.01**
Rapid7	1 255	0.91±0.05	0.76±0.11	0.91±0.02	0.83±0.10	0.84±0.04	0.84±0.02	0.69±0.47	**0.97±0.04**
Onyphe	709	0.06±0.05	0.05±0.04	0.11±0.06	0.05±0.05	**0.93±0.03**	**0.93±0.02**	0.92±0.01	0.91±0.05
Netsystems	439	0.02±0.03	0.09±0.08	0.32±0.05	0.06±0.05	**0.97±0.03**	**0.97±0.02**	0.80±0.09	0.95±0.04
Shodan	341	0.12±0.16	0.14±0.16	0.23±0.10	0.10±0.14	**0.81±0.05**	0.79±0.06	0.76±0.05	0.79±0.06
Exploiter	254	0.00±0.00	0.07±0.20	0.00±0.00	0.06±0.16	0.07±0.20	0.07±0.20	0.00±0.00	**0.10±0.21**
Securitytrails	180	0.12±0.25	0.86±0.07	0.87±0.04	0.67±0.04	**0.98±0.04**	0.96±0.05	0.97±0.04	**0.98±0.03**
Intrinsec	47	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.35±0.24	0.24±0.17	**0.75±0.12**	0.73±0.13
Macro avg.	43 050	0.26±0.05	0.41±0.04	0.49±0.04	0.41±0.04	0.74±0.02	0.73±0.03	0.75±0.03	**0.80±0.02**

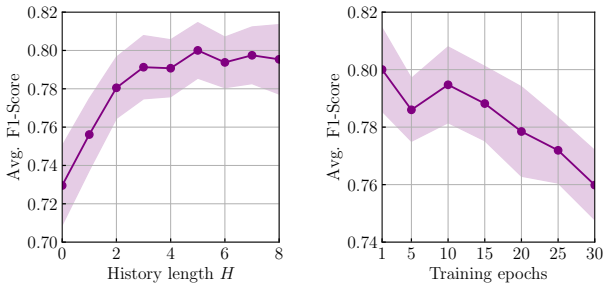
(best over 50) performs comparably to the 1-epoch-incremental training (i-GCN) resulting in 0.74 of average F1-Score versus 0.73; (ii) the temporal aspect introduced by GCN-GRU brings a slight additional improvement (0.75 of average F1-Score). However, the standard training makes the model overfit on the last data penalising classes with fast behavioural changes (i.e. Exploiter, Rapid7 and Spammer); (iii) coarsely fine-tuning a model pre-trained on the previous day for 1 epoch (i-GCN-GRU) preserves salient information in the embeddings while updating them with current snapshot dynamics. This results in a gain of up to ≈0.30 in the F1-Score of the classes penalized by the standard training and an average F1-Score of 0.80.

Comparing i-DarkVec NLP-style and tGNN embeddings, we notice that they perform comparably for downstream task results (0.77 of F1-Score versus 0.80) and training times (74.51s versus to 69.27s). The main advantage of the tGNNs approach relies on enhanced intuitiveness and flexibility: (i) Whereas i-DarkVec requires extracting sequences of hosts by destination ports to highlight coordinated behaviours, tGNNs leverage the explicit connections between hosts and ports; (ii) Enriching NLP embeddings with host features, although not explored in this work, requires additional manipulation of the embeddings (e.g. concatenation of features and embeddings), whereas the tGNNs inherently include node features by design.

5.1 Impact of parameters

Finally, we evaluate the impact of the history parameter H and the training epochs for i-GCN-GRU.

Impact of history H. In Figure 4a we evaluate the impact of the length of the temporal component of the tGNNs by reporting the average F1-Score for different values of H. Note that $H = 0$ corresponds to the i-GCN of Table 3. Generally, enhancing the memory length yields better results until a saturation point. This underscores the significance of incorporating historical data using time-aware GNNs, which effectively capture the evolution of the traffic over time.



(a) Impact of history H **(b) Impact of training epochs**

Figure 4: Impact of parameters for i-GCN-GRU.

Impact of training epochs. In Figure 4b we report the average F1-Score when training (fine-tuning) the i-GCN-GRU for an increasing number of epochs. Here the considerations of Table 3 are confirmed: when the model is trained for a large number of epochs on the current snapshot, it tends to overfit the model on the last time snapshot of data, leading to a loss of past learned information. This is reflected by a decrease in classification performance.

6 CONCLUSIONS

In this paper, we presented an initial exploration of tGNNs for darknet traffic analysis. We represented darknet traffic at packet level as a bipartite graph and generated host embeddings in a self-supervised way relying on both static and temporal GNNs. We defined node features and we experimented incremental training strategies to better follow the dynamics of the network.

Experimental results show that (i) GNNs without node features fail to extract similar behaviours among senders; (ii) When using node features, GNN embeddings are comparable with those produced by i-DarkVec, which relies on an NLP technique; (iii) coarse

fine-tuning of a tGNN model pre-trained on the previous snapshot can preserve useful past information, following the frequent changes in daily traffic.

All in all, despite the growing trend of NLP and cross-domain adaptation like i-DarkVec, this preliminary work highlights that more intuitive solutions based on GNNs can yield comparable results. Temporal GNNs are a more natural tool than NLP-style approaches for enhancing our comprehension of highly dynamic network traffic, facilitating the enhancement of embedded knowledge with richer information through node or edge features.

Future developments include the evaluation of the proposed method on additional datasets encompassing other darknet traces as well as different computer network scenarios. Furthermore, the comparison with other approaches relying on feature engineering or ML can provide further insights on the advantages and disadvantages of GNNs. A promising extension of the proposed architecture comprises a deeper investigation of both node and edge features and the application of different and more sophisticated GNN architectures. Moreover, the generated embeddings can serve as information-rich representations for several supervised or unsupervised downstream tasks, such as clustering or anomaly detection.

ACKNOWLEDGMENTS

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and the PRIN Project xInternet (eXplainable Internet - 20225CETN9 - PRIN 2022).

REFERENCES

[1] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. 2019. MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. *Computer Networks* (2019). https://doi.org/10.1016/j.comnet.2019.106944

[2] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé. 2019. Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges. *IEEE Transactions on Network and Service Management* (2019). https://doi.org/10.1109/TNSM.2019.2899085

[3] P. Bo, F. Yongquan, R. Siyuan, W. Ye, L. Qing, and J. Yan. 2021. CGNN: Traffic Classification with Graph Neural Network. https://doi.org/10.48550/arXiv.2110.09726

[4] J. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville, and C. Margi. 2019. Improving IoT Botnet Investigation Using an Adaptive Network Layer. *Sensors* (2019). https://doi.org/10.3390/s19030727

[5] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/D14-1179

[6] D. Cohen, Y. Mirsky, Y. Elovici, R. Puzis, M. Kamp, T. Martin, and A. Shabtai. 2020. DANTE: A Framework for Mining and Monitoring Darknet Traffic. https://doi.org/10.48550/arXiv.2003.02575

[7] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z.B. Houidi, and D. Rossi. 2021. DarkVec: automatic analysis of darknet traffic with word embeddings. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. https://doi.org/10.1145/3485983.3494863

[8] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z.B. Houidi, and D. Rossi. 2023. i-DarkVec: Incremental Embeddings for Darknet Traffic Analysis. *ACM Transactions on Internet Technology* (2023). https://doi.org/10.1145/3595378

[9] Z.B. Houidi, R. Azorin, M. Gallo, A. Finamore, and D. Rossi. 2022. Towards a Systematic Multi-Modal Representation Learning for Network Data. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. https://doi.org/10.1145/3563766.3564108

[10] G. Hu, X. Xiao, M. Shen, B. Zhang, X. Yan, and Y. Liu. 2023. TCGNN: Packet-grained network traffic classification via Graph Neural Networks. *Engineering Applications of Artificial Intelligence* (2023). https://doi.org/10.1016/j.engappai.2023.106531

[11] T. Huoh, Y. Luo, P. Li, and T. Zhang. 2023. Flow-Based Encrypted Network Traffic Classification With Graph Neural Networks. *IEEE Transactions on Network and Service Management* (2023). https://doi.org/10.1109/TNSM.2022.3227500

[12] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben. 2017. Unsupervised Traffic Flow Classification Using a Neural Autoencoder. In *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. https://doi.org/10.1109/LCN.2017.57

[13] M. Kallitsis, R. Prajapati, V. Honavar, D. Wu, and J. Yen. 2022. Detecting and Interpreting Changes in Scanning Behavior in Large Network Telescopes. *IEEE Transactions on Information Forensics and Security* (2022). https://doi.org/10.1109/TIFS.2022.3211644

[14] T.N. Kipf and M. Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1609.02907

[15] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and S.Y. Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022). https://doi.org/10.1109/TKDE.2022.3172903

[16] M. Lotfollahi, R.S.H. Zade, M.J. Siavoshani, and M. Saberian. 2017. Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning. https://doi.org/10.48550/ARXIV.1709.02656

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv* (2013). https://doi.org/10.48550/arXiv.1301.3781

[18] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar. 2019. Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. *IEEE Communications Surveys & Tutorials* (2019). https://doi.org/10.1109/COMST.2018.2883147

[19] S. Rezaei and X. Liu. 2018. How to Achieve High Classification Accuracy with Just a Few Labels: A Semi-supervised Approach Using Sampled Packets. https://doi.org/10.48550/ARXIV.1812.09761

[20] M. Ring, A. Dallmann, D. Landes, and A. Hotho. 2017. IP2Vec: Learning Similarities Between IP Addresses. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. https://doi.org/10.1109/ICDMW.2017.93

[21] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* (2009). https://doi.org/10.1109/TNN.2008.2005605

[22] F. Soro, T. Favale, D. Giordano, L. Vassio, Z.B. Houidi, and I. Drago. 2021. The New Abnormal: Network Anomalies in the AI Era. *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning* (2021). https://doi.org/10.1002/9781119675525.ch11

[23] B. Sun, W. Yang, M. Yan, D. Wu, Y. Zhu, and Z. Bai. 2020. An Encrypted Traffic Classification Method Combining Graph Convolutional Network and Autoencoder. In *2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC)*. https://doi.org/10.1109/IPCCC50635.2020.9391542

[24] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li. 2020. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems* (2020). https://doi.org/10.1109/TITS.2019.2935152