Poster: Detecting Network Anomalies from Small Traffic Samples using Graph Neural Network

Aviv Yehezkel
Cynamics
Israel
aviv@cynamics.ai

Eyal Elyashiv
Cynamics
Boston, US
eyal@cynamics.ai

***Abstract*—Detecting anomalies in computer networks is a classic, long-term research problem. While almost every kind of model architecture has been proposed, previous works usually analyzed the entire network traffic. However, such analysis implies high memory and processing overhead, and is becoming less applicable for large networks. In this poster we present a work in progress which studies a previously under-researched setting where only a small fraction of the network traffic is given. Our approach pre-processes the samples and transforms the computer network into a graph neural network (GNN). It distills node features, edge features and graph representations to learn a vector embedding for each endpoint which characterizes its normal behaviors. Then, a link predictor model is used to estimate the likelihood of network communications and detect anomalies.**

***Index Terms*—Deep-Learning; Graph Neural Network; Network Anomaly Detection; Network Security**

1. Introduction

Detecting anomalies in computer networks is a classic, long-term research problem. Almost every kind of model architecture has been studied: statistical, clustering, classification, information-theory, deep-learning, and others (see [1] for a recent comprehensive survey on network anomaly detection). Specifically, various learning-based approaches were used [2]–[6]. However, previous works were based on analyzing the entire network traffic, which is less applicable for large networks.

A much less-studied approach is the sampling approach, which is especially suitable for high-speed (gigabit or more) or high throughput networks, where only a small fraction of the packets (for example 1%) is being sampled and summarized [7].

In a recent work, we showed how a small percent of uniform sampling can be used to efficiently and accurately detect network anomalies and attacks using the concept of ”auto-encoder losses transfer learning” [8]. Our approach collected 1% network samples for each client, trained an auto-encoder neural network for each client's network and then normalized all auto-encoder losses of the different clients, providing the ability to transform loss vectors of different client networks with potentially significant varying characteristics, properties, and behaviors into a similar statistical distribution. The normalized losses can then be forwarded to a global detection model that detects and classifies threats in a generalized way that is agnostic to the specific client. We used extensive simulation study to compare the ”sampled” approach to the existing “un-sampled” state-of-the-art approach and showed its superior detection accuracy.

In this poster, we present a work in progress which intends to take our previous research one step deeper in the network - from the gateways to the endpoints. The proposed approach will transform the computer network into a graph neural network (GNN). It will distill node features, edge features and graph representations to learn a vector embedding for each endpoint which characterizes its normal behaviors. Then, a link predictor model will be used to estimate the likelihood of network communications and detect anomalies in the endpoint level. As this work is still in progress, the poster abstract will mainly present our research idea and approach, without providing evaluations and preliminary results.

In addition to the lower processing cost, a sampling-based network anomaly detection approach has many advantages, such as: (a) Data privacy: the packet's payload is not analyzed at any moment; (b) built-in robustness against sophisticated attacker utilizing ML techniques, due to the randomized nature.

The key contributions of the work being presented in the poster are:

- A novel approach for transforming a computer network to graph neural network.
- Learning a vector, characterizing the normal behaviours of each endpoint, using multiple aspects of the sampled traffic data (of the node, edge and graph) and detecting anomalies in the endpoint level in a manner that is agnostic to the traffic volume and network size.
- A new method for network anomaly detection using a small fraction of network samples based on a combination of graph neural network and link predictor model.

2. The Approach

The approach consists of three main stages (see in Figure 1):

1) Computer networks are transformed to graph neural networks based on key aspects of the network traffic, endpoints (nodes) and communications (edges).

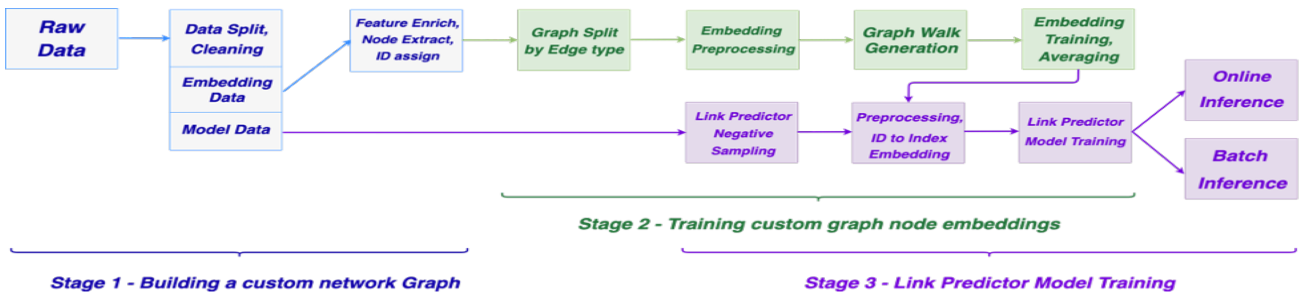

Raw Data
Data Split, Cleaning
Embedding Data
Model Data
Feature Enrich, Node Extract, ID assign
Graph Split by Edge type
Embedding Preprocessing
Graph Walk Generation
Embedding Training, Averaging
Link Predictor Negative Sampling
Preprocessing, ID to Index Embedding
Link Predictor Model Training
Online Inference
Batch Inference
Stage 2 - Training custom graph node embeddings
Stage 1 - Building a custom network Graph
Stage 3 - Link Predictor Model Training

Figure 1. An high-level overview of the proposed approach.

2) A vector embedding is learned for each graph node, characterizing its normal behaviors by fusing together different node perspectives (node features, edge features and graph representations).
3) A link-predictor model is trained to estimate a likelihood of network communications and detect anomalous links.

First, sampled IP flows[1] data is collected from main network gateways (e.g., firewalls and switches). Each record represents a meta-data summarization of communication between two endpoints (IP addresses) in the network with their flow details: source IP address, destination IP address, source port, destination port, IP protocol, creation time, number of packets in flow, flow length in bytes.

Then, the network traffic is transformed to a graph neural network with two types of nodes:

- IP entities – the flow's source and destination IPs.
- ”PPP” triplets entities – an ”artificial” node representing the communication between IP entities, composed of a combination between the flow's source port, destination port and IP protocol, i.e., (source port, IP protocol, destination port) triplet.

2.1. Building the Network Graph

In this stage, the raw sample data is used to build a network graph with different node extraction methods for the IP entities set and PPP triplets set.

The IP entities set consists of different network IP entities. Every IP is first enriched with its network category, either internal (local endpoint in the client network), external (a public IP owned by the client network) or public (public internet IP outside the client network). Both internal and external IP entities are assigned with their own node IDs.

Public IPs are processed differently: each public IP is further enriched with its hosting country and organization details, by querying an IP enrichment repository. Then, each unique tuple of (country, organization) is assigned a node ID. Thus, different public IPs that share the same hosting country and organization will be assigned the same ID and treated by the network graph as the same entity. This is done to reduce the graph dimension and improve the model generalization in inference-time for new ”unseen” IPs, by ensuring that the same logical communication will not be treated differently because of different raw IP values.

1. A flow is defined as a set of IP packets sharing a set of common properties, such as source/destination IP addresses and TCP/UDP ports.

The PPP set consists of triplets in the form of (source port, IP protocol, destination port). It is also pre-processed in order to reduce the network size and improve generalization. First, too-rare IP protocols with a total flow packet count lower than a certain threshold (e.g., 0.1% of the total count) are assigned a general IP protocol identifier, so as not to negatively affect the model learning by their significant imbalanced proportion. Then, an additional pre-processing is done on the port values. Each port value (source/destination) is assigned a category, according to the Internet Assigned Numbers Authority (IANA [9]) port range convention: common service ports have high significance and thus each will be assigned a unique identifier, but client ports have low significance for their specific value and thus will all be assigned a shared identifier.

After both protocols and ports are categorized, every unique triplet of (categorized source port, categorized IP protocol, categorized destination port) is assigned its own node ID. The IP and PPP nodes together represent our vocabulary. An additional dedicated graph node is being created for out-of-vocabulary IP entities and/or PPP triplets, mapping to a general unknown identifier.

Finally, the graph edges are created to transform each flow of (source IP, PPP, destination IP) to two unidirectional edges: the first is between source IP node to PPP node, and the second is between the PPP node to destination IP node. The main building blocks of this stage are summarized in Figure 2.

2.2. Training Vector Embeddings

In this stage we employ a graph embedding technique to distill node features, edge (link) features (between graph nodes) and graph structure information to learn a vector embedding representation to each node. These embeddings provide a “network context” for each of the network entities that will be used in the last stage for learning a link predictor and detecting anomalies.

We will use several edges types:

- By average packet volume to account for different traffic volumes sent over the flow. We will create 10 different edge types by splitting the flow packet count to 10 equal sized bins.
- By time of day. We will create 24 different edge types by splitting the flow by its hour.

We will use several node features, creating a vector for each node consisting of: IP address subnet B ID

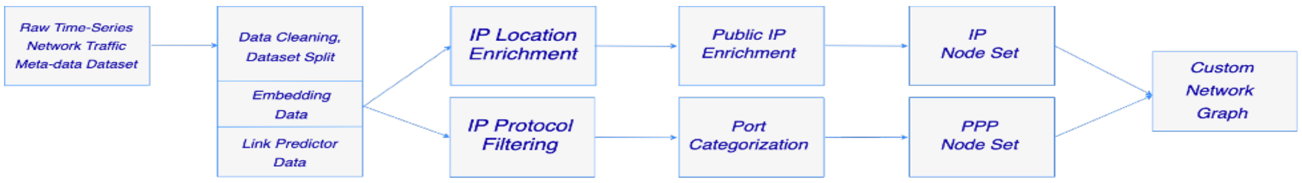


Figure 2. Overview of the main building blocks used for creating the network graph.

(255.255.X.X) , IP address subnet C ID (255.255.255.X), IP address country ID , IP address organization ID and network location category (internal, external or public).

Then, the embeddings will be learned based on the metapath2vec algorithm of graph walks [10]. Each node is sampled for random graph "walks" or paths, starting from the selected node. This process results in a 200-length vector embedding learned for each node for each one of the 36 extracted graphs (10 packet volume average graphs and 24 hourly graphs). Thus, each graph node is represented by a (36, 200) matrix. To reach a one-dimensional node representation, the 36 node vectors are averaged to a single 200-length vector. The main building blocks of this stage are summarized in Figure 3.

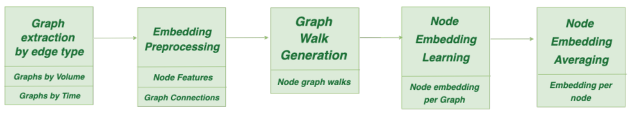


Figure 3. Overview of the main building blocks used for learning graph node embeddings.

2.3. Link Predictor Model

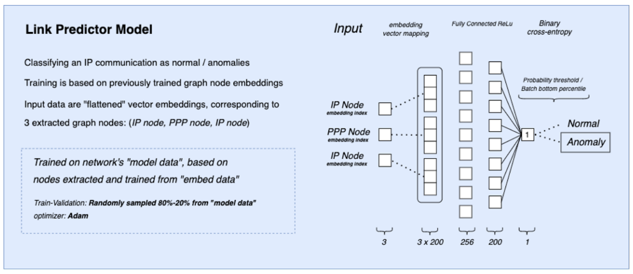
In the final stage, a link predictor model is being trained to detect anomalies. Traditionally, a link predictor model is given a pair of graph nodes and is trained to estimate the likelihood for an edge between them [11].

In our case, the link predictor model will learn to estimate a likelihood for network communications: given a triplet combination (IP node, PPP node, IP node) as input, instead of the traditional 2-node single graph connection. Positive labeled inputs are sampled from the given data, and negative labeled inputs are generated by pairing random triplets that were not part of the given data. Finally, combining the positive and negative datasets forms the input data for the link predictor model.

The link predictor architecture is a feed-forward neural network that consists of the following layers (presented in Figure 4):

- An input embedding layer, with an input size of (3, 200) accounting for input triplet of (IP,PPP,IP), each embedding of size 200.
- A flatten operation combining the 3 input node vectors.
- Two dense layers with ReLU activation.
- A final output dense layer with sigmoid activation to output a likelihood probability.

Training loss will be calculated with binary cross-entropy loss, and network optimization will be done using a


Link Predictor Model

Classifying an IP communication as normal / anomalies

Training is based on previously trained graph node embeddings

Input data are "flattened" vector embeddings, corresponding to 3 extracted graph nodes: (*IP node, PPP node, IP node*)

Trained on network's "model data", based on nodes extracted and trained from "embed data"

Train-Validation: ***Randomly sampled 80%-20% from "model data"***
optimizer: ***Adam***

Figure 4. The link predictor model architecture.

stochastic gradient descent algorithm, such as the Adam optimizer.

At the final stage, model inference will be done in real-time using a predetermined probability threshold, detecting network connections beneath the threshold as anomalies. The threshold will be learned statistically after the model training (e.g., the 99.99th probabilities' percentile).

References

[1] Gilberto Junior, Joel Rodrigues, Luiz Carvalho, Jalal Al-Muhtadi, and Mario Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 2019.

[2] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *NDSS*, 2018.

[3] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6), 2020.

[4] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. Multi-task network anomaly detection using federated learning. In *Proceedings of the 10th international symposium on information and communication technology*, 2019.

[5] Poonam Mehetrey, Behrooz Shahriari, and Melody Moh. Collaborative ensemble-learning based intrusion detection systems for clouds. In *2016 International Conference on Collaboration Technologies and Systems (CTS)*.

[6] Lianbing Deng, Daming Li, Xiang Yao, David Cox, and Haoxiang Wang. Mobile network intrusion detection for iot system based on transfer learning algorithm. *Cluster Computing*, 22(4), 2019.

[7] Baek-Young Choi and Supratik Bhattacharyya. On the accuracy and overhead of cisco sampled netflow. 2005.

[8] Aviv Yehezkel, Eyal Elyashiv, and Or Soffer. Network anomaly detection using transfer learning based on auto-encoders loss normalization. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2021.

[9] Michelle Cotton, Lars Eggert, Dr. Joseph D. Touch, Magnus Westerlund, and Stuart Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, 2011.

[10] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. *KDD*, 2017.

[11] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *NIPS*, 2018.

Cynamics

Detecting Network Anomalies from Small Traffic Samples using Graph Neural Network

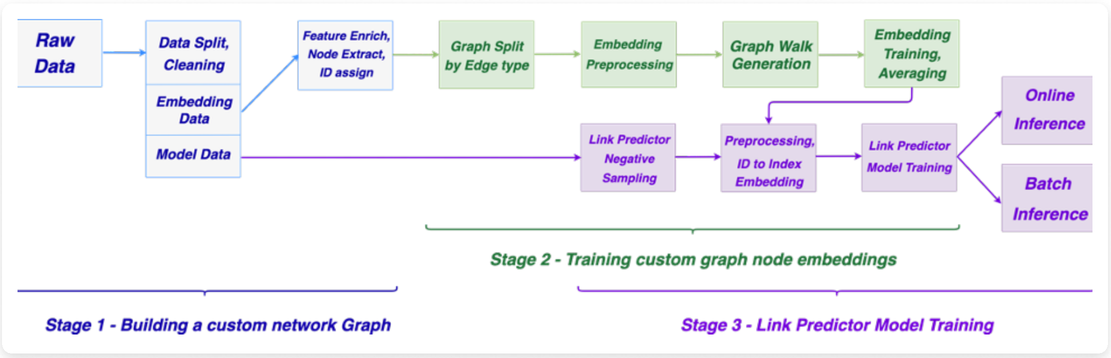
Aviv Yehezkel
Cynamics
Israel
aviv@cynamics.ai

Eyal Elyashiv
Cynamics
Boston, US
eyal@cynamics.ai

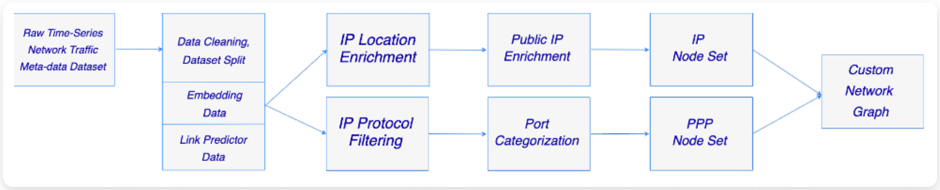
Detecting anomalies in computer networks is a classic, long-term research problem. While almost every kind of model architecture has been proposed, previous works usually analyzed the entire network traffic. However, such analysis implies high memory and processing overhead, and is becoming less applicable for large networks. In this poster we present a work in progress which studies a previously under-researched setting where only a small fraction of the network traffic is given and detects anomalies using a novel graph neural network (GNN) approach.

The approach consists of three main stages:

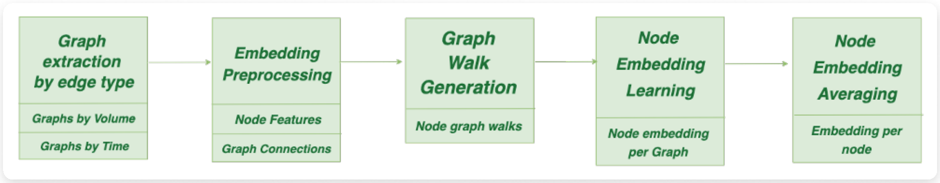
1. Computer networks are transformed to graph neural networks with two types of nodes: IP entities (flow's source and destination) and "PPP" triplets entities (an "artificial" node representing the flow's source port, destination port and IP protocol).
2. A vector embedding is learned for each graph node, characterizing its normal behaviors.
3. A link-predictor model is trained to estimate a likelihood of network communications and detect anomalous links.

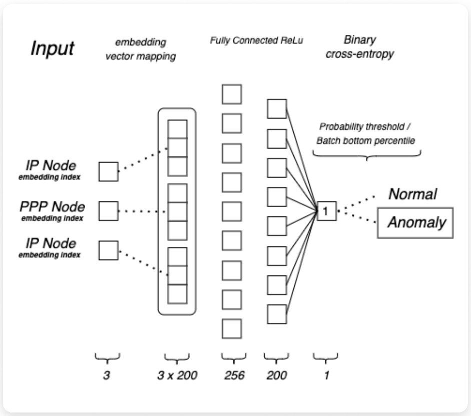

Raw Data
Data Split, Cleaning
Embedding Data
Model Data
Feature Enrich, Node Extract, ID assign
Graph Split by Edge type
Embedding Preprocessing
Graph Walk Generation
Embedding Training, Averaging
Link Predictor Negative Sampling
Preprocessing, ID to Index Embedding
Link Predictor Model Training
Online Inference
Batch Inference
Stage 2 - Training custom graph node embeddings
Stage 1 - Building a custom network Graph
Stage 3 - Link Predictor Model Training

Building the Network Graph with different node extraction methods for the IP entities set and PPP triplets set.



Training Vector Embeddings: employing a graph embedding technique to distill node features, edge (link) features (between graph nodes) and graph structure information to learn a vector embedding representation ("network context") to each node.





In the final stage, a **link predictor model** is being trained to detect anomalies by estimating a likelihood for network communications: given a triplet combination (IP node, PPP node, IP node) as input.

The key contributions of this poster:

- A novel approach for transforming a computer network to graph neural network.
- Learning a vector, characterizing the normal behaviours of each endpoint using multiple aspects of the sampled traffic data and detecting anomalies in the endpoint level in a manner that is agnostic to the traffic volume and network size.
- A new method for network anomaly detection using a small fraction of network samples based on a combination of graph neural network and link predictor model.