

Benchmarking Graph Neural Networks for Internet Routing Data

Dimitrios P. Giakatos

dgiakatos@csd.auth.gr

Aristotle University of Thessaloniki
Greece

Pavlos Sermpezis

sermpezis@csd.auth.gr

Aristotle University of Thessaloniki
Greece

Sofia Kostoglou

sofikost@csd.auth.gr

Aristotle University of Thessaloniki
Greece

Athena Vakali

avakali@csd.auth.gr

Aristotle University of Thessaloniki
Greece

ABSTRACT

The Internet is composed of networks, called Autonomous Systems (or, ASes), interconnected to each other, thus forming a large graph. While both the AS-graph is known and there is a multitude of data available for the ASes (i.e., node attributes), the research on applying graph machine learning (ML) methods on Internet data has not attracted a lot of attention. In this work, we provide a benchmarking framework aiming to facilitate research on Internet data using graph-ML and graph neural network (GNN) methods. Specifically, we compile a dataset with heterogeneous node/AS attributes by collecting data from multiple online sources, and preprocessing them so that they can be easily used as input in GNN architectures. Then, we create a framework/pipeline for applying GNNs on the compiled data. For a set of tasks, we perform a benchmarking of different GNN models (as well as, non-GNN ML models) to test their efficiency; our results can serve as a common baseline for future research and provide initial insights for the application of GNNs on Internet data.

1 INTRODUCTION

The Internet is a network of networks, which are called Autonomous Systems (or, ASes). Today there exist more than 100k ASes originating IP prefixes in the Internet routing table, which are connected to each other through private or public peering links. Representing the ASes as nodes and their interconnections as edges, results in a large and sparse (density $< 0.01\%$) graph.

Since ASes and their interconnections play a significant role for network operations, Internet policies, routing optimization, etc., there has been many efforts to characterize these networks. Hence, there exist rich datasets with information about ASes (open datasets [4, 5, 17, 29], self-declared databases [21], data from custom measurements, etc.).

These datasets with AS attributes have been used by several works employing (traditional) ML methodologies for various applications [9, 10, 12, 16, 25, 31]. One would expect that with the advent of Graph Neural Networks (GNNs) many works would exploit the known AS-graph structure along the AS attributes to devise GNN-based methodologies for problems related to Internet routing and operations. However, there only exist a few efforts generating graph embeddings [27, 28], and, in fact, they are not based on GNNs (but on methods from the natural language processing field) and they

do not take into account the node attributes (but only the graph structure).

While there can be many reasons behind this lack of GNN-based works for Internet routing data (and it is out of our scope to investigate them), a main challenge for applying GNNs on Internet data is that significant expertise is needed in both domains: namely, a researcher needs (i) rich Internet data and (ii) a good understanding of advanced deep learning techniques and graph theory concepts. On one hand, it may be straightforward for Internet researchers to access sources of Internet data (which are typically well known within this community), but it may be a more tedious task for researchers of other domains (e.g., more focused to GNNs) to compile a rich dataset that would be needed by a GNN architecture. On the other hand, while there are widely used and well documented libraries (pytorch geometric [3], dgl [2], etc.) that have made access to GNNs easy, there are many intricacies in the application of GNNs to Internet data (e.g., imbalanced data, heavy tailed distributions, etc.), which render their efficient application a non-trivial task for an Internet-focused researcher.

Motivated by the aforementioned observation, in this paper we aim to facilitate research with GNNs on Internet data through the following contributions:

- **Dataset:** We compile a *rich dataset* of Internet data that can be used as input to GNN models (Section 2). Specifically, we collect from multiple online sources a set of 19 AS attributes, including both numerical and categorical variables. We then preprocess the data and transform them to a format that is readily available to be used as input to GNNs (e.g., all values normalized in $[0,1]$). The compiled dataset not only offers easy access to researchers, but it also serves as a *benchmark dataset*. The lack of benchmark datasets, has been identified as a key barrier that challenge ML research in networking applications [8]. Having a common dataset, on which different ML approaches are applied and compared (e.g., similarly to the ImageNet [11] and CIFAR-10 [19] datasets in computer vision), can further boost GNN research on Internet data.
- **GNN benchmarking & initial insights:** We test several GNN, graph-ML, and (non-graph) ML models on the compiled dataset, for several downstream tasks (Section 3). Our goal is not to propose a specific GNN architecture, and thus we refrain from extensive model optimization. Hence, we use a basic architecture and hyperparameter tuning for all models, and we produce initial results which can serve as a point of reference (e.g., baselines) for

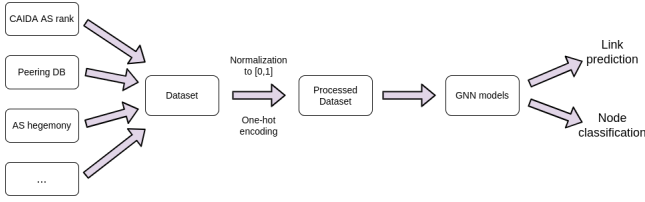


Figure 1: Overall methodology pipeline.

future research. Our experimental results (Section 4) provide initial insights about the efficiency of GNNs on Internet data related tasks (e.g., the role of graph structure and/or node attributes for different tasks), and reveal several challenges.

- **Open data and code:** We make publicly available the compiled dataset and our code (using a popular GNN library [2, 33])¹ in [1].

2 DATASET

In this section, we present the data sources (Section 2.1) and the preprocessing (Section 2.2) we applied on the data to generate the compiled dataset. The overall methodology is depicted in Fig. 1.

2.1 Data sources

Each network or Autonomous System (AS) can be characterized by a multitude of features, such as, location, connectivity, traffic levels, etc.. We collect data from multiple online (public) data sources to compile a dataset, which contains multiple information for each AS.

The first three data sources are widely used by Internet researchers and operators for multiple purposes:

- CAIDA AS-rank [4]: various information about ASes, such as, location, network size, topology, etc.
- CAIDA AS-relationship [5]: a list of AS links (i.e., edges), which are used to build the AS-graph.
- PeeringDB [7, 21]: online database, where network operators register information about the connectivity, network types, traffic, etc., of their networks

We also use the following sources that provide data related to the routing properties of ASes and their business types:

- AS hegemony [23]
- Country-level Transit Influence (CTI) [6]
- ASDB [29]

From the above sources, we collect the most relevant attributes per AS, resulting to a dataset of 19 attributes/features (see Table 1 for the detailed list). For ease of analysis, in the online repository [1] we also provide a visual exploratory data analysis with the detailed distributions of all attributes.

2.2 Data preprocessing

The collected data are highly heterogeneous, including both numerical and categorical attributes. Moreover, numerical attributes take values in different ranges, and some of them span ranges several orders of magnitude larger than others (see Table 1). Since, it is

well known that non-homogeneous data values can impact the performance of deep learning models, we need to preprocess the data. In the following we describe the transformation we apply to each type of attributes to generate a dataset with normalized attributes taking values in the interval $[0,1]$.

Categorical features. For every categorical feature, one-hot encoding is applied. In the one-hot encoding technique, a new feature is created for every value of the categorical feature. For example, the "Location-continent" feature contains 6 values (Africa, Asia, Europe, N. America, S. America, Oceania), which means that after the one-hot encoding 6 new numerical columns are created; hence, an AS located in Europe will have a value of 1 in the respective new feature for Europe, and a value of 0 in the other 5 new features that correspond to the other continents.

Numerical features. As it can be seen in Table 1, some numerical attributes take values in very large ranges (e.g., the customer cone of ASes spans from 1 to more than 48k ASNs). Also, for many of these attributes the values for different ASes are not distributed uniformly, but they have a heavy tail distribution (e.g., almost 95% of ASes have a customer cone of 1 ASN). To alleviate this large heterogeneity and variability of the numerical features, we perform the following transformations.

- First, for every numerical feature, except for the *AS hegemony* and the *CTI top* features that only take values less than 1, we apply a logarithmic transformation to decrease their variability, as follows: $x \rightarrow \log(x + 1)$.
- Then, we normalize all numerical feature according to the Min-Max scaling method: $x \rightarrow \frac{x - \min(x)}{\max(x) - \min(x)}$. As a result, all the resulting values are in the range of $[0, 1]$.

Graph preprocessing. The AS graph contains a large number of leaf nodes (i.e., edge networks with a single upstream). These nodes are of limited interest in the ML downstream tasks we consider (see Section 3.2), namely, for (i) link prediction: they only have a single link, and (ii) node classification: the characteristics/classes we consider can be easily inferred for edge networks. Moreover, taking them into account would lead to a graph structure that is more challenging to be captured by a GNN or graph-ML model. Hence, we preprocess the graph and remove all nodes with degree equal to one (and repeat two more times this process); the resulting graph has around 46K nodes and 434K edges.

3 GNN BENCHMARKING METHODOLOGY

To benchmark GNNs on the compiled dataset, we use a set of GNN, graph-ML, and traditional ML models (Section 3.1), and design the downstream tasks on which the efficiency of the models will be tested (Section 3.2).

3.1 Models

GNN models: We consider three widely used GNN models.

GraphSAGE [15] learns a function (neural network) that generates embeddings for a node by sampling and aggregating node features from its local neighborhood. The embeddings capture both the local graph structure of a node and the feature distribution of its neighborhood.

¹As well as, all the experimental results of the paper, for reproducibility purposes.

Table 1: Summary of AS attributes/features in the compiled dataset.

Feature	Description	Data type	Source
RIR region	Regional Internet registry	Categorical (6 categories)	[4]
Customer cone (ASNs)	Number of ASNs in the <i>customer cone</i>	Numerical $\in [1, 48790]$	[4]
Customer cone (prefixes)	Number of IP prefixes in the <i>customer cone</i>	Numerical $\in [0, 737792]$	[4]
Customer cone (addresses)	Number of IP addresses in the <i>customer cone</i>	Numerical $\in [0, 2090939967]$	[4]
#Neighbors	Total number of neighbors (in # of ASNs)	Numerical $\in [0, 9547]$	[4]
#Customers	Total number of customers (in # of ASNs)	Numerical $\in [0, 6505]$	[4]
#Peers	Total number of peers (in # of ASNs)	Numerical $\in [0, 7516]$	[4]
#Providers	Total number of providers (in # of ASNs)	Numerical $\in [0, 133]$	[4]
Location-continent	Registered location of the headquarters of the ASN	Categorical (6 categories)	[4]
Traffic ratio (PDB)	Type of traffic ratio (e.g., inbound, outbound, balanced)	Categorical (6 categories)	[7, 21]
Scope (PDB)	Regional scope of the AS (e.g., regional, global, Europe)	Categorical (10 categories)	[7, 21]
Network type (PDB)	Network type (e.g., ISP, content provider, enterprise)	Categorical (11 categories)	[7, 21]
Peering policy (PDB)	Peering policy (e.g., open, selective, restrictive)	Categorical (4 categories)	[7, 21]
#IXPs (PDB)	Number of IXPs where the AS is present	Numerical $\in [0, 288]$	[7, 21]
#facilities (PDB)	Number of interconnection facilities where the AS is present	Numerical $\in [0, 768]$	[7, 21]
AS hegemony	Metric measuring avg. fraction of routing paths crossing an AS	Numerical $\in [0, 0.2]$	[17]
CTI top	"Country-level Transit Influence" metric of an AS	Numerical $\in [0, 0.95]$	[6]
CTI origin	Percentage of addresses initiated by an AS in a country	Numerical $\in [0, 97.39]$	[6]
ASDB	Industry type of the organization of the AS	Categorical (17 categories)	[29]

GCN [18] (Graph Convolutional Networks) is the graph analogy to CNNs for images. It uses a spectral convolution of the graph, which leverages the Laplacian matrix in a trainable function that aggregates features from neighboring nodes.

GAT [32] (Graph Attention Networks), similarly to the above models, aggregates node features from a node’s neighborhood. However, it can learn different weights for different nodes in a neighborhood, thus capturing the different levels of importance that the neighbors of a node may have.

For each model, we build a basic architecture that comprises two GNN layers (with a 32-dimensional output), followed by an MLP layer. We refrain for extensive tuning, and consider fully connected layers (no dropout), a learning rate of 0.01 and a few hundreds epochs per model.

Graph embedding models (non-GNNs): The goal of graph embeddings is to map the nodes of a graph to an embedding space (i.e., a vector) of lower dimensions. In our experiments, we use two methods that generate node embeddings based on the graph structure, but *without taking into account the node attributes*.

Node2vec [14] is based on the popular word2vec [20] method used in Natural Language Processing to represent words in a text with vectors. Node2vec generates a collection of random walks on the graph, starting each time from different nodes. Those random walks are lists of nodes, which act similarly to sentences in a text (where the nodes are the words). Those lists are used as input to train a skip-gram model (shallow neural network) to predict the probability of a certain word/node to be present in a sentence when an input word/node is present.

Bgp2vec [28] is a method designed specifically to generate AS embeddings. It is also based on word2vec, but rather than performing random walks, it uses BGP announcements collected by route

collectors [24], and in particular the AS-paths in them as the lists of node sequences. The model is trained over a large corpus of AS paths, and learns to characterize an AS by its context, i.e., its neighboring ASes.

In both models we generate embeddings of size 16. In the node2vec model we use 20 random walks of a walk length equal to 4, which is around the average AS path length in the Internet.²

Traditional ML model: Finally, we use a Random Forest (RF) model with 100 trees as a baseline, which takes as input the node features but *neglects the graph structure*.

3.2 Learning tasks

We selected two (sets of) tasks for the benchmarking:

Link prediction between the nodes of the graph, i.e., inference of AS-AS links. It is well-known that our view of the AS-graph is incomplete, and there is evidence that many links (in particular, peer-to-peer links at Internet eXchange Points, or IXPs) cannot be seen by the public measurements from which we construct the known AS-graph. Therefore, being able to predict links could be important for several Internet routing use cases.

Node classification: We predict attributes of ASes that are declared in the PeeringDB database [21]. Network operators voluntarily register the information of their ASes in PeeringDB, which leads to incomplete knowledge; in our dataset, only around 25% of the ASes have information about the PeeringDB attributes. Nevertheless, knowing the PeeringDB attributes can be helpful for a number

²The other detailed parameters for the models are

- *node2vec*: $p = 1, q = 1, workers = 4, window_size = 5, epochs = 1, lr = 0.05, min_count = 1$.
- *bgp2vec*: $negative = 5, epochs = 3, window = 2, shuffle = False$

of operational/policy/economic reasons, and has been identified as a need by the network operators community.

Link prediction and node classification tasks have the same pipeline: We get the node embeddings from GNN models (GraphSAGE, GCN, GAT) or the node2vec/bgp2vec models, and feed them to an MLP network to get the predictions³. We use a binary cross entropy as a loss function for the link prediction task and the cross entropy loss (due to the many classes) for the node classification tasks⁴.

Finally, since (i) the AS-graph is very sparse and (ii) for some categorical features there are classes with very few samples, we balance the train datasets by selecting equal number of existing/non-existing links and class samples. In the case of classes with very few samples (less than 500), we group these classes together, in order to get enough samples to train the GNN models. The final classes for each categorical features are given in Table 2.

Table 2: Classes and number of samples in each class for the categorical features used in the node classification tasks.

Feature	Classes	nb. of samples
Traffic ratio (PDB)	Balanced	1546
	Heavy Inbound	319
	Heavy Outbound	202
	Mostly Inbound	1403
	Mostly Outbound	517
	Not Disclosed	1769
Scope (PDB)	Asia Pacific	530
	Europe	869
	Global	600
	North America	388
	Not Disclosed	1526
	Regional	1515
	Other	380
Network type (PDB)	Cable/DSL/ISP	1910
	Content	649
	Enterprise	348
	NSP	998
	Not Disclosed	1315
	Other	579
Peering policy (PDB)	Open	4379
	Selective	1018
	Other	231

4 RESULTS

4.1 Link prediction

Table 3 presents the link prediction results, reported as the average AUC score (over 10 different models, with the same hyperparameters, and random weight initialization). The largest the AUC score

³In the case of the random forest, we directly get the predictions.

⁴Since labels are not known for all nodes, for the classification task, while we consider all nodes in the generation of the embeddings (GNN), we train/test the MLP by using only the values for the nodes with labels (i.e., by masking the nodes without labels).

(in the interval [0,1]) the more accurate the prediction; a score of 0.5 corresponds to a (dummy) random predictor. Since, the underlying graph is very sparse, we also present the Recall ($\frac{\#TP}{\#P}$) and Precision ($\frac{\#TP}{\#TP+\#FP}$) metrics that focus on the "true positive" (TP) samples, i.e., existing links that are predicted correctly, and their fractions with respect to the number of all existing links ($\#P$) and all links predicted correctly or incorrectly ($\#TP + \#FP$).

We can see that the GraphSAGE and GCN models are very efficient in predicting links between nodes, whereas the GAT model has poor capacity. While the highest AUC score is achieved by the random forest (RF), its Recall value is very low: this indicates that while RF does not mispredict non-existing links (low false positives, FP) it is only able to predict 1/4 of the actual links. The graph-ML models (and, in particular, bgp2vec [28]) achieve also a high performance. On one hand, this shows that only the structure of the AS-graph (without node features) can help us to predict and characterize links, as already shown in [28]. However, the fact that the GCN model with very light tuning can outperform bgp2vec, indicates that taking into account node features can be promising for tasks related to link prediction and characterization (e.g., inference of peering relationships).

Table 3: Results for the link prediction task: average AUC, Recall, and Precision metrics over 10 runs per model.

Model	AUC	Recall ($\frac{\#TP}{\#P}$)	Precision ($\frac{\#TP}{\#TP+\#FP}$)
GraphSAGE	94.7%	82.7%	86.8%
GCN	95.3%	85.5%	95.9%
GAT	64.4%	24.2%	28.4%
node2vec	86.5%	82.7%	95.5%
bgp2vec	93.0%	85.5%	91.5%
Rnd. forest	96.2%	24.2%	96.3%

In Table 4 we do a deeper inspection, to understand what types of links are easier to predict. We consider the GraphSAGE model and group nodes in three categories based on the size of their neighborhood: nodes with *low* (< 10), *medium* ($\in [10, 20)$), and *large* (≥ 20) number of neighbors; around 80% of nodes belong to the first group and around 10% to each of the other groups. We can see that predicting links between nodes of high degrees is easy, whereas links between nodes with "low" number of neighbors (i.e., the majority of nodes) are more difficult to be inferred. This indicates a challenge and a need for efficient designs of GNN models that focus on the sparse parts of the AS-graph.

Table 4: Detailed link prediction results (Recall / Precision) for node pairs of different size of neighborhoods.

	Low	Medium	High
Low	31.6% / 89.8%	80.7% / 98.2%	67.2% / 90.5%
Medium		94.4% / 98.1%	94.4% / 95.4%
High			99.5% / 99.0%

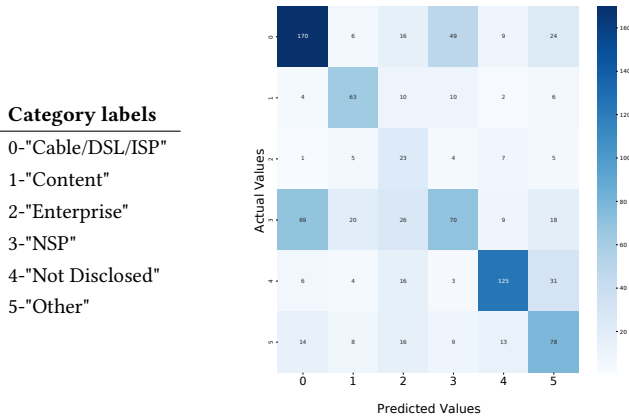


Figure 2: Heatmap of predicted vs. actual values of the GraphSAGE model for the "Network type" attribute.

4.2 Node classification

Table 5 presents the results of the node classification, where we try to infer different attributes of ASes related to the PeeringDB (see column names). We present the average Accuracy (ACC) and the F1 score metrics. We stress that for each attribute there are several categories (cf. Table 1), i.e., the problems in hand are *multi-class* classification problems.

We can see that the best performing model differs among attributes. GraphSAGE performs consistently well, while GAT has now a comparable performance to other GNN models. Nevertheless, we believe that there may be significant room for improvement in future work, e.g., through optimization of GNN architectures.

As an example, Fig. 2 depicts the detailed predictions of the GraphSAGE model for the "Network type" attribute. In all categories (rows), the number of samples (cell values) that are predicted correctly (diagonal) are higher than in any other category (columns) in the same row. The most missclassifications are between ASes that are characterized "Cable/DSL/ISP" and "NSP (Network Service Providers)", which in practice can have several common characteristics.

Compared to the RF model, GNNs predict better only the "Scope" and "Network type" attributes. This indicates that not all node attributes may be strongly related to the underlying graph structure; in some cases the graph structure can help our predictions, whereas in other cases using a simpler (i.e., easier to train) model may be the best solution.

Finally, it is clear from the node2vec/bgp2vec performance that using only graph information is not enough for the node classification tasks. This further highlights the need for future research on applying GNNs (i.e., both graph structure and node attributes) for Internet routing related tasks.

5 RELATED WORK

In the last few years, there have been several attempts for combining Internet routing data with ML, with the majority of them focusing on BGP anomaly detection. Ding et al. [12] and Dai et al. [10] try to detect BGP anomalies using traditional ML models, various features, and advanced feature selection methodologies, such as minimum

redundancy maximum relevance [12] or Fisher linear analysis and Markov random fields [10]

Typically, for Internet routing tasks, features can be extracted from BGP messages [9, 13, 16, 26, 31], for example, volume, AS path features and BGP attributes [13, 16], network importance metrics [9], network observations [26], or graph-level metrics [16]. Sanchez et al. [25] went a step further to consider more robust graph features, such as node centrality, clique theory, etc. They conclude in the fact that centrality metrics are more likely to detect large-scale incidents.

The first work that proposed the use of graph embeddings on ASN level is [27, 28], which uses BGP messages and the AS-paths in them to propose the bgp2vec model. These embeddings can be used to predict node and link properties [28] or classify BGP routes as standard or hijacked [27].

Finally, some recent benchmarking efforts for GNNs in the domain of networking (but not for Internet routing data) are (i) the IGNITION [22] framework for prototyping GNNs for communication networks, which contains tools to design, train and evaluate a GNN model, and (ii) the GNNet challenge [30] for designing GNN models for predicting network performance.

6 CONCLUSION

In this paper we compiled a benchmark dataset with Internet data, preprocessed in a way that is compatible to be used by GNN architectures. The benefits from the dataset are twofold: (i) it enables researchers to focus on designing GNN architectures rather than collecting and processing data, which can be a time consuming task (or even prohibitive for non Internet experts), and (ii) it can serve as a common dataset where different works can be compared on (following the example of [30]), which is a key requirement for progressing ML research for networking [8].

Using our dataset and pipeline, we performed various experiments to test the performance of different GNN models. Our initial results, not only can be used as a baseline in future work, but also provide useful insights. We showed that capturing both node attributes and graph structure can be beneficial for link prediction and node classification tasks, however, the benefits of each factor may vary depending on the problem. Given the lack of previous works with GNNs on Internet data, and thus the lack of reported insights, we believe these findings can be a useful starting point for future research.

ACKNOWLEDGMENTS

This research is co-financed by Greece and European Union through the Operational Program Competitiveness, Entrepreneurship and Innovation under the call RESEARCH-CREATE-INNOVATE (projects T2EDK-04937 and T2EDK-03898), the European High-Performance Computing Joint Undertaking (GA No. 951732), and RIPE NCC (AI4NetMon project).

REFERENCES

- [1] 2022. Benchmarking GNNs for Internet routing data - Public repository. Available at <https://github.com/dpgiakatos/gnn-internet-data>.
- [2] 2022. Deep graph library (DGL). Available at <https://www.dgl.ai/>.
- [3] 2022. PyTorch Geometric library (PyG). Available at <https://pytorch-geometric.readthedocs.io/en/latest/>.
- [4] CAIDA. 2022. AS-rank dataset. Available at <https://asrank.caida.org/>.

Table 5: Results for the node classification tasks: average accuracy (ACC) and F1 score metrics over 10 runs per model.

Model	Traffic ratio (PDB)		Scope (PDB)		Network type (PDB)		Peering policy (PDB)	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
GraphSAGE	44.8%	35.9%	49.2%	47.2%	54.7%	53.1%	34.9%	30.6%
GCN	38.7%	30.5%	40.1%	37.1%	46.6%	44.6%	37.0%	31.1%
GAT	38.0%	31.3%	41.8%	38.4%	49.9%	47.2%	32.2%	28.8%
node2vec	20.9%	19.4%	16.3%	15.7%	19.0%	18.6%	31.1%	27.0%
bgp2vec	14.8%	13.4%	14.8%	13.0%	19.9%	19.1%	29.4%	26.8%
Rnd. Forest	51.1%	35.8%	36.6%	33.7%	49.8%	42.9%	54.6%	34.8%

- [5] CAIDA. 2022. AS-relationships dataset. Available at <https://publicdata.caida.org/datasets/as-relationships/>.
- [6] CAIDA. 2022. Country-level Transit Influence (CTI). Available at <https://github.com/CAIDA/mapkit-cti-code>.
- [7] CAIDA. 2022. PeeringDB Dataset. Available at <https://publicdata.caida.org/datasets/peeringdb/>.
- [8] Pedro Casas. 2020. Two decades of ai4nets-ai/ml for data networks: Challenges and research directions. In *IEEE NOMS*.
- [9] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. 2019. BGP hijacking classification. *2019 Network Traffic Measurement and Analysis Conference (TMA)* (2019).
- [10] Xianbo Dai, Na Wang, and Wenjuan Wang. 2019. Application of machine learning in BGP anomaly detection. In *Journal of Physics: Conference Series*, Vol. 1176. IOP Publishing, 032015.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR conference*. 248–255.
- [12] Qingye Ding, Zhida Li, Prerna Batta, and Ljiljana Trajković. 2016. Detecting BGP anomalies using machine learning techniques. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.
- [13] Paulo Fonseca, Edjard S Mota, Ricardo Bennessy, and Alexandre Passito. 2019. Bgp dataset generation and feature extraction for anomaly detection. In *IEEE ISCC*.
- [14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proc. ACM SIGKDD*.
- [15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
- [16] Kevin Hoarau, Pierre Ugo Tournoux, and Tahiry Razafindralambo. 2021. Suitability of graph representation for bgp anomaly detection. In *Proc. IEEE LCN*.
- [17] IJ. 2022. Internet Health Report. Available at <https://ihr.ijlab.net/ihr/en-us/>.
- [18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [19] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).
- [20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).
- [21] PeerinDB. 2022. The Interconnection Database. Available at <https://www.peeringdb.com/>.
- [22] David Pujol-Perich, José Suárez-Varela, Miquel Ferriol, Shihan Xiao, Bo Wu, Albert Cabellos-Aparicio, and Pere Barlet-Ros. 2021. IGNNITION: Bridging the Gap between Graph Neural Networks and Networking Systems. *IEEE Network* 35, 6 (2021), 171–177.
- [23] Internet Health Report. 2022. AS hegemony. Available at <https://ihr.ijlab.net/ihr/hegemony/>.
- [24] RouteViews. 2022. RouteViews route collectors. Available at <http://www.routeviews.org/peers/peering-status.html>.
- [25] Odnan Ref Sanchez, Simone Ferlin, Cristel Pelsser, and Randy Bush. 2019. Comparing machine learning algorithms for BGP anomaly detection using graph features. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big Data, Machine Learning and Artificial Intelligence for Data Communication Networks*. 35–41.
- [26] Pavlos Sermpezis, Vasileios Kotronis, Konstantinos Arakadakis, and Athena Vakali. 2021. Estimating the Impact of BGP Prefix Hijacking. In *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–10.
- [27] Tal Shapira and Yuval Shavitt. 2020. A Deep Learning Approach for IP Hijack Detection Based on ASN Embedding. In *Proc. Workshop on Network Meets AI and ML*.
- [28] Tal Shapira and Yuval Shavitt. 2022. BGP2Vec: Unveiling the Latent Characteristics of Autonomous Systems. *IEEE Transactions on Network and Service Management* (2022).
- [29] Stanford. 2022. ASDB. Available at <https://asdb.stanford.edu/>.
- [30] José Suárez-Varela et al. 2021. The graph neural networking challenge: a world-wide competition for education in AI/ML for networks. *ACM SIGCOMM Computer Communication Review* 51, 3 (2021), 9–16.
- [31] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. 2019. Profiling BGP serial hijackers: capturing persistent misbehavior in the global routing table. In *Proceedings of the Internet Measurement Conference*. 420–434.
- [32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [33] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019).