

BGP2Vec: Unveiling the Latent Characteristics of Autonomous Systems

Tal Shapira^{ID}, *Graduate Student Member, IEEE*, and Yuval Shavitt^{ID}, *Senior Member, IEEE*

Abstract—BGP announcements hold latent information about the Internet Autonomous Systems (ASes) and their functional position within the Internet eco-system. This information can aid us in understanding the Internet structure and also in solving many practical problems. In this paper, we present *BGP2Vec*, a novel approach to revealing the latent characteristics of ASes using neural-network-based embedding. We show that our embedding indeed captures important characteristics of ASes, and then show how the embedding can be used to solve two problems: ASN business-type classification and AS Type of Relationships (ToRs) inference. ToRs inference has been heavily studied in the past two decades and is important for studying Internet routing and identifying IP hijack attacks. We use the *BGP2Vec* vectors as an input to artificial neural networks and achieve excellent results: an accuracy of 95.8% for ToR classification and an accuracy of 79.2% for AS classification.

Index Terms—Internet, BGP, routing, ASN embedding, AS classification, AS relationships, deep learning.

I. INTRODUCTION

THE INTERNET consists of thousands of Autonomous Systems (ASes), each AS operated by an administrative domain such as an Internet Service Provider (ISP), a business enterprise, or a University. Each AS is assigned a globally unique number, the Autonomous System Number (ASN), and advertises one or more IP address prefixes (APs) using the Border Gateway Protocol (BGP). BGP routing updates messages list the entire AS path to reach an address prefix (AP). BGP allows each AS to choose which routes to accept (import policy), how to select the best routes, and which routes to announce (export policy).

The commercial agreements between two connected ASes are broadly classified into three types of relationship (ToR) [1]: 1) Provider-to-customer (P2C) - the customer AS pays the provider AS for transit traffic from and to the rest of the Internet, 2) Peer-to-peer (P2P) - two ASes freely exchange traffic between themselves and their customers, but do not

Manuscript received 26 November 2021; revised 6 April 2022; accepted 11 April 2022. Date of publication 22 April 2022; date of current version 31 January 2023. This research was funded in part by a grant on cyber research from the Israeli PMO and the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University. The associate editor coordinating the review of this article and approving it for publication was J. Schönwälder. (Corresponding author: Tal Shapira.)

Tal Shapira was with the School of Electrical Engineering, Tel-Aviv University, Tel Aviv 69978, Israel. He is now with the School of Computer Science, Reichman University, Herzliya 1096, Israel (e-mail: talshapira@gmail.com).

Yuval Shavitt is with the School of Electrical Engineering, Tel-Aviv University, Tel Aviv 69978, Israel (e-mail: shavitt@eng.tau.ac.il).

Digital Object Identifier 10.1109/TNSM.2022.3169638

exchange traffic from or to their providers or other peers, and 3) Siblings (S2S) - two ASes that belong to the same administrative domain. Gao [1] defined concatenation rules for AS links in a route that model the way ASes usually configure their BGP; it is called Valley-Free (VF) since once a route descend from a provider to a customer, it cannot ascend again. An interesting observation from the VF model is that connectivity does not imply reachability, and the shortest path in the (undirected) AS graph may not be usable due to the BGP VF constraints.

ToR information allows us to infer the possible routes selected by BGP, e.g., in case of a link failure [2]. It can also be used to identify malicious fiddling with the routing system, known as IP hijack attack [3], [4]. However, ToR information is mostly not public, and thus there is a long line of research to infer it [1], [5], [6], [7], [8]. Most of these solutions are heuristic algorithms based on publicly available BGP announcement databases [9], [10]. An inherent problem in these algorithms is their use of heuristics, causing unbounded errors that are spread over all inferred relationships.

In this paper, we introduce a novel approach for AS embedding using deep learning and demonstrate its strength by applying it to solve both AS classification (i.e., the business type of each AS), and ToR classification. Our work contribution has merit from both network science and Internet networking aspect points. We show that AS level routes, just like sentences in a natural language, can reveal the latent functionality of an AS in the Internet. We then show how by using embedding, we can use this knowledge for important networking problems: ToR classification and AS classification. We made our code publicly available.¹

Over the past few years, advances in deep learning [11] have driven tremendous progress in many fields; one of them is Natural Language Processing (NLP). We build on the excellent results achieved for NLP tasks (Word2Vec [12]), where word adjacency in sentences is used to map words to low dimensional space. Instead, we use adjacency of ASNs in BGP announcements to embed ASes in a low (we selected 32) dimensional space and attach to each AS a vector of its coordinates in this space. Based on the ASN embedding, we apply artificial neural networks for both AS and ToR classification problems.

We show that the *BGP2Vec* representations exhibit linear structure, and specifically, we can characterize an ASN by its closest ASNs. Interestingly, ASNs are grouped based on multiple characteristics such as their role in BGP, e.g., tier (tier-1, tier-2,

¹<https://github.com/talshapira/BGP2Vec>

etc.), type of AS (content providers, academic institutions); and geographic location. Specifically, we show that ASNs in a country tend to have high similarity among them, and so do sibling ASNs. Tier-1 providers also form close clusters.

Our approach achieved excellent results: we classify AS ToRs with an accuracy of 95.2% and 95.8% using two AS Relationships datasets and classify AS business types with an accuracy of 79.2%. As far as we know, we are the first to solve this problem using deep learning methods. We should also mention that the embedding also allows us to detect IP hijacked routes [13], which is outside the scope of this paper.

Our initial results [14] of this work described *BGP2Vec* and an initial evaluation of its usage for ToR classification. Here, we focus on the embedding itself and thoroughly evaluated its characteristics. In addition, we also evaluated AS classification, and added many experiments including comparison with other embedding methods, and usage of additional dataset.

The rest of the paper continues as follows. After describing related work in Section II, we describe the dataset in Section III. In Section IV we describe the problem and give the motivation for our approach, in Section V we introduce some preliminaries on Neural Network Embedding and in Section VI we describe our method for ASN embedding, i.e., *BGP2Vec*, and the architectures of our neural networks. In Section VII we explore the latent characteristics of our embedding vectors, in Section VIII we present our quantitative experiments and their results with comparison to previous results, while in Section IX we compare the *BGP2Vec* embedding with other common embedding methods. Finally, the last section concludes the paper.

II. RELATED WORK

A. AS Classification

During the last two decades, there have been many works that focused on characterizing the AS-level topology, while only some of them aimed to classify the ASes themselves. Most of these works classified ASes according to their business type. Dimitropoulos *et al.* [15] constructed a set of 6 AS attributes based on data collected from IRR [10] and RouteViews [9], such as the organization description record, the number of inferred customers, providers, peers, etc. They constructed 6 classes; ‘Large ISPs’, ‘Small ISPs’, ‘Customer ASes’, ‘Universities’, ‘IXPs’ and ‘NICs’. Then they applied an AdaBoost-based algorithm and classified 95.3% of the ASes with an expected accuracy of 78.1%. Following the last work, Dhamdhere and Dovrolis [16] proposed a simple classification scheme based on four business types; ‘Enterprise Customers’, ‘Small Transit Providers’, ‘Large Transit Providers’ and ‘Content/Access/Hosting Providers’. They applied decision trees using the average number of peers and customers each AS had in the previous decade and achieved an accuracy of 78%. CAIDA [17] used a decision tree based on seven features and achieved a Positive Predictive Value (PPV) of 70% for 3 classes: ‘Transit/Access’, ‘Enterprise’, and ‘Content’.

B. ToR Inference

Many works focused on solving the problem of inferring ToRs; most of them proposed heuristic algorithms based on

extracting information from BGP announcements or generating AS level Routes from traceroutes. We focus here on works which are based on BGP route information and do not consider works that leveraged other information to improve ToR inference, such as usage of BGP communities or IXP route servers [18].

Gao [1] was the first to study the AS relationships inference problem. She presented heuristic algorithms that infer AS ToRs from BGP routing announcements based on the fact that a provider’s AS graph degree is usually larger than its customers and that peers have about the same degree. The algorithm locally identifies the top provider for each path and classifies edges (ToRs) following the VF nature of routing paths.

Subramanian *et al.* [5] introduced the ToR maximization problem, which is to label all the edges in an undirected AS graph to maximize the number of VF paths in a set of BGP routes. Their algorithm exploits the structure of partial views of the AS graph, as seen from different locations. For each location, it calculates the rank of each AS using a reverse-pruning algorithm and infer the ToR between two ASes by comparing their vectors of ranks; if the ranks are similar, the algorithm classifies the link as P2P, otherwise as C2P.

Xia and Gao [19] used the BGP Community Attribute, the AS-SET object, and the routing policies in the IRR Databases to infer AS relationships. However, their approach only obtains partial AS relationships (14% of total AS pairs on Oct. 2003). They showed that both GAO [1] and SARK [5] inferred poorly P2P relationships.

Battista *et al.* [20] proved that the ToR optimization problem [1], [5] is NP-complete and reduced the problem to the ToR-D problem that allows a small number of invalid paths. They reduced the ToR-D problem to 2SAT and introduced a heuristic algorithm for determining the ToRs. Cohen and Raz [6] defined the Acyclic Type of Relationship (AToR) problem that attempts both to minimize the number of invalid paths and keep the directed graph acyclic. They introduced a heuristic algorithm to solve the K-AToR problem.

Dimitropoulos *et al.* [7] used the IRR [10] to infer S2S relationships and then introduced a more realistic problem formulation that accepts that AS paths do not always exhibit a hierarchical pattern to infer P2C and P2P relationships. Their algorithm introduced a metric called reachability, sorted all ASes by their reachability, and grouped ASes with the same value into levels. They correctly inferred 96.5% C2P, 82.8% P2P, and 90.3% S2S relationships.

Weinsberg *et al.* [8] were motivated to reduce the usage of heuristics. They proposed a near-deterministic algorithm for solving the ToR inference problem (ND-ToR) that uses the Internet’s core (a sub-graph of top-level ASes), which was constructed in three different ways: the Greedy Max Clique (GMC) core [21], the k -Core which is based on the k -shell decomposition [22], and the CAIDA Peers Core (CP) which is the largest connected component of a P2P graph (provided by CAIDA [23])- that contains some of the largest tier-1 ASes. They inferred the rest of the links using a three-phase algorithm based on the VF rule and the k -shell index [22] of the adjacent ASes. Their algorithm succeeded to infer over 95% of

TABLE I
NUMBER OF SAMPLES IN OUR LABELED DATASETS

Labeled ASes			CAIDA AS ToRs serial-2			Labeled AS ToRs			
Transit/Access	Enterprise	Content	P2P	P2C	C2P	P2P	P2C	C2P	S2S
8,095	1,270	1,686	608,486	118,405	118,405	10,230	89,459	13,297	392

approximately 58,000 ToRs based on AS-level paths collected from RouteViews [9] and DIMES [24].

Luckie *et al.* [25] introduced the AS-Rank algorithm for inferring C2P and P2P links using BGP data. Their work relies on three assumptions: 1) there is a clique of large transit providers at the top of the hierarchy, 2) most customers enter into a transit agreement to be globally reachable, and 3) cycles of C2P links should not exist for routing to converge. Based on these assumptions, they introduced a new algorithm for inferring the customer cone of an AS, which is the set of ASes that the AS can reach using P2C links, and achieved a state of the art results.

In order to overcome the inference barriers for hard cases, such as non-VF routing, limited visibility, and non-conventional peering practices, Jin *et al.* [26] identified key interconnection features and developed a probabilistic algorithm (ProbLink), and showed that their algorithm achieved an error rate that is better than AS-Rank over their validation set. However, they use additional information, such as sibling relationships, BGP communities, and IXP information.

Recent work by Feng *et al.* [27] analyzed the factors that contribute to the uncertainty of ToR inference and presented an uncertainty-aware AS relationship inference algorithm that reflects the uncertainty in the inference result. Their approach relied on two common principles: the existence of a clique of well-known Tier-1 ASes interconnected by P2P links and the VF principle.

As mentioned in [7], [8], [19], [25], [27], [28], the existing heuristic algorithms rely on assumptions such as the presence of VF paths, the existence of a peering clique of ASes at the top of the hierarchy, and more. **This highly motivated our work for introducing a deep learning based approach that relies only on the characteristic of the data (BGP routes).** As far as we know, this is the first time deep learning is used for this problem. As we will show, our deep learning method produced significantly better results than previous rule-based and heuristic algorithms.

III. THE DATASETS

In our experiments, we use data that was collected in October-November 2018. We use three types of datasets. RouteViews's **BGP announcements** (RV) [9] contains BGP path announcements collected from 19 route collectors. The dataset consists of approximately 3,600,000 BGP paths, 62,525 AS vertices, and approximately 113,400 undirected links. We use this unlabeled dataset for the first stage of our approach (i.e., ASN embedding).

Labeled AS classes dataset contain self-reported business types for 11,051 ASes from PeeringDB [29]. The types are ‘Cable/DSL/ISP’, ‘NSP’, ‘Content’, ‘Education/Research’,

‘Enterprise’ and ‘Non-profit’. Following the CAIDA AS Classification Dataset [17], we combined the ‘Cable/DSL/ISP’ and ‘NSP’ classes into a single class ‘Transit/Access’ with 8,095 ASes, and combined the ‘Enterprise’ and ‘Education/Research’ classes into a single class ‘Enterprise’ with 1,270 ASes, and use the ‘Content’ class with 1,686 ASes.

We used three **AS Relationships Datasets** from two sources. CAIDA [30] provides two datasets: serial-1, which contains 343,952 P2P pairs and 118,405 P2C/C2P pairs, that were inferred from BGP paths using AS-Rank [25], and serial-2, which contains additional 264,534 P2P pairs that were inferred from BGP communities attributes using the method described in [18]. The total number of samples is displayed in Table I. We use this dataset as labels for training our neural network and various supervised learning algorithms based on ASN embedding and as a benchmark for comparison with previous works. Note that the CAIDA dataset does not contain siblings.

For some of the experiments, we also used a proprietary ToR dataset with close to 600,000 ToRs from BGProtect (www.BGProtect.com), generated by VF algorithm which is based on Weinsberg *et al.* [8] with manual corrections.² The ones that matched our experimental data contain 10,230 P2P pairs, 89,459 P2C pairs, 13,297 C2P pairs, and 392 S2S pairs. These experiments were done in addition to the ones with the public CAIDA dataset to test performance on a dataset that is more accurate since it is used continuously, and thus, errors are manually cleaned once revealed.

IV. MOTIVATION AND PROBLEM STATEMENT

We consider the problem of classifying ASes and ASes relationships of the Internet AS-level graph into one or more categories. More formally, let \mathcal{P} be a collection of AS-level paths (e.g., BGP announcements), we can then generate the AS-level graph $G = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} are the ASNs of the Internet, and \mathcal{E} be its edges, $\mathcal{E} \subseteq (V \times V)$. Given a partially labeled edges set E_L with $Y \in \mathbb{R}^{|E_L| \times |\mathcal{Y}|}$, or partially labeled ASNs set \mathcal{V}_L with $Y \in \mathbb{R}^{|\mathcal{V}_L| \times |\mathcal{Y}|}$, such that \mathcal{Y} is the set of labels, our goal is to classify the rest of the edges $\overline{E_L}$ or ASNs $\overline{\mathcal{V}_L}$ correspondingly. We propose an unsupervised method that learns distributed representations of ASNs that capture the graph structure independent of the labels' distribution. Therefore, we utilize significant information about the dependence of ASNs to achieve great performance. Moreover, the same representation can be used for different classification problems concerning the Internet graph.

²Manual corrections (changing or adding ToRs) are done by analysts at the BGProtect SOC when handling alerts (of route deflection attacks) that are based on valley-free violation.

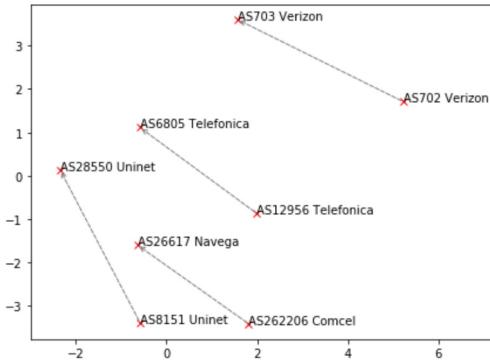


Fig. 1. Two-dimensional PCA projection of 32-dimensional BGP2Vec's vectors of selected siblings that are close to the Verizon's sibling {AS702, AS703}. The figure illustrates the model's ability to automatically organize concepts and learn the relationships between them implicitly, as during the training, we did not provide any labels for siblings.

Perozzi *et al.* [31], motivated their usage of NLP models for social graph representation learning by their similarity, as both word distribution in a language and social graphs, have power-law degree distribution. For the Internet AS-level graph, Faloutsos *et al.* [32] were the first to show that the inter-AS topology exhibits a power-law (scale-free) node degree distribution, which was later shown to be true for extended connectivity data by Shavitt and Shir [24].

Natural language models are trained using large amounts of unstructured text data, which in our case are equivalent to a collection of BGP announcements (an AS-level path is equivalent to a sentence). The word representations, which are computed using neural networks, capture many linguistic and latent characteristic [33], and also exhibit linear structure that makes precise analogical reasoning possible. For example: $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman}$ resembles the closest vector to the word \overrightarrow{Queen} [33]. We seek to obtain the same behavior. For example we expect the vectors to capture siblings relationship: $\overrightarrow{AS702} - \overrightarrow{AS703} + \overrightarrow{AS6805}$ should resembles the closest vector to $\overrightarrow{AS12956}$. Figure 1 presents the closest difference vectors to the vector $\overrightarrow{AS702} - \overrightarrow{AS703}$, and shows that indeed they represent other sibling relationships.

The relationships between ASes are known as the type of relationships (ToRs), and methods for obtaining them were studied extensively in the past two decades. *BGP2Vec* is an ideal tool for this task since it allows us to represent a ToR by the difference between the vectors on its two ASNs. Once we encounter an ASN-pair with an unknown ToR, we can use the embedding to deduct it by its closest ToRs in space.

Our goal is to learn a latent representation for ASNs, which is defined by the embedding function $\Phi : v \in \mathcal{V} \mapsto \mathbb{R}^{|\mathcal{V}| \times d}$, where d is small number of latent dimensions. This mapping Φ represents the latent representation associated with each ASN in the graph. We use these low-dimensional representations as inputs to machine learning algorithms. In Section VII we explore the obtained embedding vectors and show that these vectors capture latent characteristics of ASes.

In [34], Grover and Leskovec explored several types of binary operators for learning edge features including *Average*, *Hadamard Weighted-L1*, and *Weighted-L2*. However, all of

these proposed operators are symmetric. In the case of ToRs, the P2C ToR is opposed to the C2P ToR; therefore, we have to suggest asymmetric operators. In the rest of the paper, unless mentioned otherwise, our primary operator to obtain ToR vectors out of the ASN vectors is the *concatenate* operator, resulting in a ToR vector with twice the size of the original BGP2Vec. The second operator is the *subtraction* operator, which preserved the size of the original vector and is simply defined by:

$$[\Phi(u) - \Phi(v)]_i = \Phi_i(u) - \Phi_i(v), \quad (1)$$

where the definition corresponds to the i th component of ToR's vector.

V. PRELIMINARIES

This section introduces some preliminaries on Neural Network Embedding, which is the main ingredient of our approach. Applications of neural networks have expanded significantly in recent years [11]. One of them is embedding discrete values to continuous N-dimensional vectors. This technique is broadly used in Natural Language Processing (NLP), also known as word embedding [12], [33] or Neural Language Modeling, helps machine learning algorithms to achieve better performance by grouping similar words. Embedding is important for input to a neural network, as it is trained to work on vectors of real numbers. Moreover, the method produces vectors such that similar words have close vectors, where similarity is defined in terms of syntax and semantic.

Word2Vec [12] is one of the most popular unsupervised data-driven techniques to learn word embedding. This technique uses a shallow neural network fed with large amounts of unstructured text data. Word2Vec has two important benefits; first, its representations exhibit linear structure that makes precise analogical reasoning possible. The second benefit is that in contrast to one-hot encoding which map each word to a vector with a size equal to the number of unique words in the corpus, and therefore makes training any machine learning model on this representation infeasible, Word2Vec map each word to a vector of size d , with d significantly smaller relative to language size.

There are two kinds of Word2Vec models, which have the opposite training objective: Skip Gram and Continuous Bag Of Words (CBOW). The skip-gram model takes a word as input and seeks to predict the surrounding words in a sentence or a document. In contrast, the CBOW model takes each word's context as the input and tries to predict the word corresponding to the context.

We will focus on describing the skip-gram model, which is the one we use in *BGP2Vec*. In the skip-gram model, we analyze each sentence in the corpus, and for each word, we predict words within a certain range before and after the current word (as shown in Fig. 3 for AS-path). As a result, the model learns to characterize a word by its context, i.e., neighboring words. The model itself consists of a neural network with a single hidden layer. The input layer is of size $|\mathcal{V}|$, equal to the number of unique words in the corpus (or vocabulary), such that each word is represented by a unique one-hot encoding vector. The hidden layer is a fully-connected layer of size equals

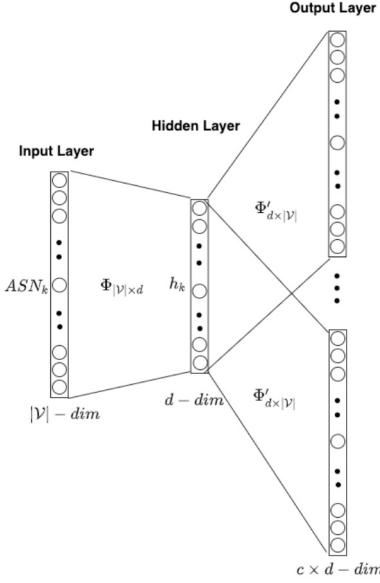


Fig. 2. Our *BGP2Vec* Skip-Gram architecture (with $|\mathcal{V}| = 62,525$, $d = 32$, and $c = 4$).

to the embedding size (d in Fig. 2). Within a *fully-connected layer*, neurons between two adjacent layers are fully pairwise connected, while neurons within the same layer share no connections. Each artificial neuron summarizes the dot product of its inputs and its weights and adds a bias term. The hidden layer weight matrix is of size $|\mathcal{V}| \times d$, such that each word in the corpus corresponds to a d -features vector; these are the word vectors which are learned by the model. Therefore, the output of the hidden layer is the embedding for the input word. The output layer is the *softmax layer*, which is a generalization of the logistic function that normalized a vector of arbitrary real values to a probability distribution vector of real values in the range $[0, 1]$ that add up to 1. In Word2Vec, the softmax layer is of size $|\mathcal{V}|$ (the number of unique words in the corpus) for each desired output.

The training procedure is done by feeding the network with the word pairs; the input is a one-hot vector representing the input word, and the training outputs, which are also one-hot vectors representing the output words (the context words). Then applying gradient descent learning [35] (also known as back-propagation) to adjust the weights of the network in order to maximize the log probability of any context word given the input word.

VI. METHODS

In this section, we describe our method for classification. We first use a set of AS Paths that is fed into a Skip Gram model, a method we term *BGP2Vec*, to embed the ASNs into a large dimensional latent space. This embedding captures functional and other latent characteristics of the ASNs, which we further explore in the next section. Then, for each task (AS classification or ToR classification), we activate an Artificial Neural Network (ANN) that receives the vectors from the previous stage. In this section, we will introduce in detail both stages.

Algorithm 1: $\text{BGP2Vec}(\mathcal{P}, w, d, \alpha)$

```

inputs :  $\mathcal{P}$ : AS paths list
           $w$ : window size
           $d$ : embedding size
           $\alpha$ : learning rate
output: matrix of vertex representations  $\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$ 
1 begin
2   Initialization: Generate vocabulary  $\mathcal{V}$  from  $\mathcal{P}$ ,
3   Sample  $\Phi$  from  $\mathbb{R}^{|\mathcal{V}| \times d}$ 
4   for each  $path \in \mathcal{P}$  do
5     | SkipGram( $\Phi, path, w, \alpha$ )
6   endfor
7   return  $\Phi$ 
8 end

```

Algorithm 2: $\text{SkipGram}(\Phi, path, w, \alpha)$

```

1 for each  $v_t \in path$  do
2   for each  $v_j \in path[t - w : t + w]$  do
3     |  $J(\Phi) = -\log \Pr(v_j | \Phi(v_t))$ 
4     |  $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$ 
5   endfor
6 endfor

```

A. ASN Embedding With *BGP2Vec*

As mentioned in Section I, in the first stage we produce a d -dimensional continuous vector representation for each ASN. As in the training process of word embedding in NLP, i.e., Word2Vec [12], we train our network over a large corpus of AS paths \mathcal{P} (described in Section III), which are equivalent to the sentences in NLP tasks. We apply a similar Skip-Gram model as introduced in [12], such that for each ASN in a certain AS-path we predict ASNs within a certain range before and after the current ASN (as shown in Fig. 3). As a result, the model learns to characterize an ASN by its context, i.e., neighboring ASNs.

As presented in Algorithm 1 we intend to learn d dimensional embeddings of ASNs from paths \mathcal{P} in the dataset. We begin by building a vocabulary of all the ASNs in the dataset (line 2), and randomly initializing the embeddings for all ASNs in the vocabulary ($\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$) (line 3). Then for each path we apply the Skip Gram model (Algorithm 2). Given an AS-level path $\{v_1, v_2, v_3, \dots, v_T\}$ and a context window of size w , for each target AS v_t the Skip-Gram model attempts to predict the words that appear in its context $(v_{t-w}, \dots, v_{t+w})$. More formally the objective of the Skip-Gram model is to learn the latent representation $\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$ which maximizes the average log co-occurrence probability:

$$\frac{1}{T} \sum_{t=1}^T \sum_{-w \leq j \leq w, j \neq 0} \log \Pr(v_{t+j} | v_t; \Phi). \quad (2)$$

As presented in [12], the basic neural-network based Skip-Gram formulation defines the conditional probability $\Pr(v_{t+j} | v_t)$ using the *softmax* function:

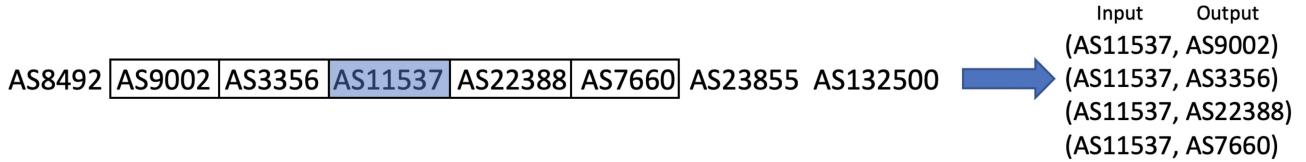


Fig. 3. An example for generating input/output ASNs for the training process of *BGP2Vec* using a window size of 2.

$$\Pr(v_{t+j}|v_t) = \frac{\exp(\Phi_{v_t}^T \Phi'_{v_{t+j}})}{\sum_{v \in \mathcal{V}} \exp(\Phi_{v_t}^T \Phi'_v)} \quad (3)$$

where Φ_v and Φ'_v are the input and output vectors of AS v , as presented in Figure 2. However, this formulation is impractical because the computation cost of the conditional probability is proportional to $|\mathcal{V}|$, which in our case is very large ($10^5 - 10^6$ terms). Therefore we use the negative-sampling approach, as introduced in [12], as a more efficient way of deriving ASN embeddings, by replacing the log-likelihood term in the Skip-Gram objective with the term:

$$\log \sigma(\Phi'_{v_{t+j}}^\top \Phi_{v_t}) + \sum_{i=1}^k \mathbb{E}_{v_i \sim P_n(v)} [\log \sigma(-\Phi'_{v_i}^\top \Phi_{v_t})], \quad (4)$$

where σ is the *Sigmoid* function, which is defined by $\sigma(x) = 1/(1 + \exp(-x))$. Negative sampling selects k ASNs that are not in the context at random noise distribution $P_n(w)$ instead of considering all ASNs in the graph. In our experiments we used the same noise distribution as described in [12].

Our shallow two-layers neural network contains an input layer of size 62,525 (the number of distinct ASes in our dataset, denoted by $|\mathcal{V}|$), one fully connected (FC) hidden layer whose size is the embedding size d (using a grid search method, we found that an embedding of size 32 achieves best results, see Section VIII-B), and an output layer whose size, denoted by $c = 2w$. The hidden layer weight matrix Φ is of size $|\mathcal{V}| \times d$, such that each ASN in the corpus corresponds to a d -features vector; these are the ASN-vectors that are learned by the model. The output layer is of size $|\mathcal{V}|$ for each desired output.

We choose to apply a window of size 2 (see Figure 3), which is the maximum distance between the input ASN and a predicted ASN (the output) within an AS-level path, which results in an output layer with a maximum size equals to $4 \times 62,525$. In order to improve the representation, we use 5 negative samples [12] for each target ASN. We build and run our network using the *Gensim* [36] library. The training procedure is done by feeding the network with the ASN pairs; the input is a one-hot vector representing the input ASN and the training outputs, which are also one-hot vectors representing the output ASNs (the context ASNs). Then applying gradient descent learning [35] (also known as back-propagation) to adjust the weights of the network, in order to maximize the log probability of any context word given the input word based on Equations (2), (4). We repeat this process for 3 epochs.

TABLE II
THE ARCHITECTURE OF OUR NEURAL NETWORKS BASED ON *BGP2Vec*
WITH EMBEDDING SIZE OF 32; FOR ToR CLASSIFICATION AND AS
CLASSIFICATION

ToR Classification		AS classification	
Layer Type	Input/Output Size	Layer Type	Input/Output Size
Embeddings:	Input: 2, 1 Output: 2, 32	Embeddings:	Input: 1 Output: 32
Conv1D:	Output: 2, 32	Fully Connected:	Output: 100
MaxPool:	Output: 2, 16	Softmax:	Output: 3
Conv1D:	Output: 32, 16		
MaxPool:	Output: 16, 16		
Fully Connected:	Output: 100		
Softmax:	Output: 3		

B. ANNs Architectures

For both AS and ToRs classification problems, we choose to use simple ANNs (see Table II). The architectures are presented in Table II.

The ToR classification's architecture comprises seven layers, not counting the input. A sequence of ASNs is fed into the first layer of the network, which is an embedding layer. Each ASN is embedded into a 32-dimensional vector based on the first stage. The next layer is a 1-dimensional convolutional layer [37] (labeled as ConV1D) followed by ReLU [38] activation function with 32 filters of length 3 and a total number of 3,104 trainable parameters (3072 weights and 32 bias parameters). The next layer is a max-pooling layer with 32 feature maps of size 2, where each unit in each feature map outputs the maximum value of 2 neurons in the corresponding feature map in ConV1D. The next layer is a second ConV1 layer (with 224 trainable parameters) followed by a second max-pooling layer. The next layer is a fully-connected layer with 100 neurons and a ReLU activation function (with 25700 trainable parameters). Finally, our output layer is the softmax layer with 3 outputs, one for each class. The last layer contains 303 trainable parameters. For the AS classification problem, we use a simple ANN with a single hidden layer. This network contains a total number of 3,603 trainable parameters.

The training of the neural networks is by optimizing the *categorical cross entropy* [39] cost function, which is a measure of the difference between the softmax layer output and a one-hot encoding vector of the same size, representing the correct label of the sample. For the optimization process, we use the *Adam* [40] optimizer, which is an extension to the stochastic gradient descent algorithm. We use the default

hyper-parameters as provided in Kingma and Ba [40] and set our batch size to 64.

We build and run our networks using the *Keras* [41] library with *Tensorflow* [42] as its back-end. We use 80% of the samples as a training set and 20% of the samples as a test set. We split each dataset such that the ratio between the quantities of the classes remains the same in both the training set and the test set, while there is no ToR in the training set in which its inverse appears in the test set. We run our network for 40 epochs of the training set for both AS and ToR classification. We save the result which achieves the best accuracy during the training process. During the test time, our network classifies a ToR in an average time of 0.1 milliseconds.

VII. QUALITATIVE EXPLORATION OF ASN EMBEDDING

In this section, we explore the *BGP2Vec* embedding in order to show that ASNs tend to cluster according to similar characteristics. The following sections will show how we can utilize this behavior to solve real networking problems: ToR classifications and ASN classification. We also delay the discussion on hyper-parameter optimization to these sections.

It is important to mention that such embedding is not perfect, and we should expect a small percentage of the ASNs to be placed not where expected. This can be due to multiple reasons: insufficient data on ASN, conflicting constraints, or simply statistical errors. We will point out these problems in the data we present. Nevertheless, given the errors, our results show that our embedding produces excellent results.

A. Similarity and Ranking Metrics

1) *Similarity*: We define similarity between two *BGP2Vec* vectors $\Phi_v, \Phi_u \in \mathbb{R}^d$ of the corresponding ASes $v, u \in \mathcal{V}$ as the *cosine similarity*:

$$\text{sim}(\Phi_v, \Phi_u) = \frac{\Phi_v^T \Phi_u}{\|\Phi_v\|_2 \cdot \|\Phi_u\|_2}, \quad (5)$$

and find the nearest neighbor of *BGP2Vec* vector $\Phi_v \in \mathbb{R}^d$ using:

$$\text{NearestNeighbor}(v) = \arg \max_{u \in \mathcal{V}} \text{sim}(\Phi_v, \Phi_u). \quad (6)$$

2) *Average Cosine Similarity Correlation*: We calculate the *average cosine similarity correlation* of a specific group of ASNs by taking the average of all the cosine similarities of all the pairs in this group.

B. Visualization of the ASN Embedding

First, we explore the strong characteristics of the embedding vectors, as can be inferred by looking at the entire AS graph. Figure 4 presents a 2-dimensional UMAP projection [43] of all the ASNs which have transit degrees [44] greater than 0, i.e., we disregard stub ASNs. The transit degree of a node is the number of neighbors it connects to others in at least one path. Each ASN is represented by a dot, where the size of each dot grows logarithmically with its transit degree, and its color is determined by the region of ownership.

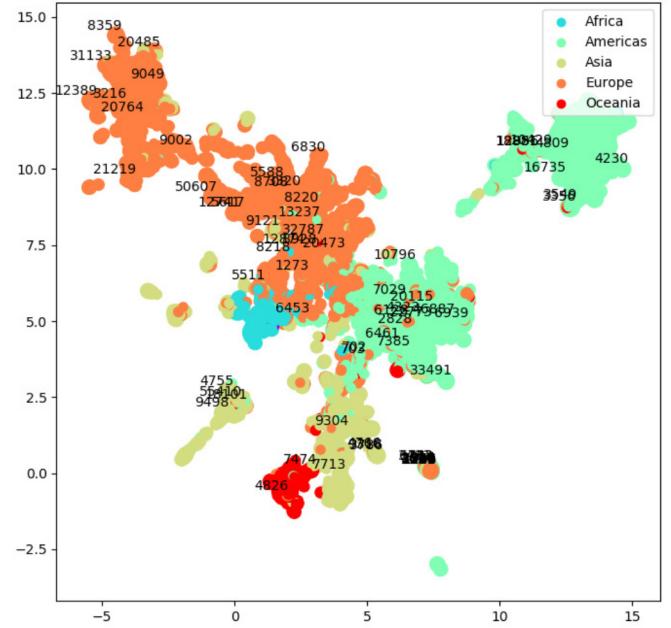


Fig. 4. Two-dimensional UMAP projection of 32-dimensional BGP2Vec's vectors of the entire AS graph.

As can be observed, most of the Tier-1 providers are located in the upper-right cluster, which also contains many North-American ASNs. The ASNs at the upper-left corner of the graph are mostly Russian, while Central/Western European ASNs are located at the center of the graph. At the bottom of the graph, we can find Australian ASNs. Between Oceania and Europe, we can find ASNs from Eastern-Asia and South/Central-America (mostly Brazil), while Africa is connected to Central Europe. A large separated Asian cluster is located near the bottom-left corner, and it is comprised entirely of Indian ASNs.

There is also a single separated cluster at the bottom of the graph, which is located far away from the rest of the graph. The cluster is composed of networks of the US DoD (Department of Defence) with other American ASNs associated with military and governmental organizations.

C. Exploration of Nearest ASN Neighbors

To test the quality of the embedding we examine the 5 nearest neighbors of selected ASNs, and tabulate the results in Table III. We selected different kinds of ASes in order to demonstrate different latent characteristics of ASes. For each explored AS, we find its 5 nearest neighbors using the cosine similarity; and for each neighbor, tabulate general properties: AS owner, cosine similarity with the specified AS, AS degree based on our RV dataset, distance from Tier-1 (we calculate the distance based on our RV dataset), AS PeeringDB class, AS country, and ToR with the specified AS. Besides, we assigned a custom ranking grade for each neighbor according to its proximity rank (5 for the nearest, 1 for the fifth nearest) if it is similar to the specified AS.

The most similar vector to AS3356 (Level3, Tier-1 provider) is the vector of its sibling AS3549, with a high cosine

TABLE III
EXPLORATION OF THE 5 NEAREST NEIGHBORS FOR DIFFERENT ASes

Neigh- bor	ASN	Owner	Cosine Similarity	Degree	Hops from Tier-1	AS Class	Country	ToR	Ranking Grade
-	3356	Level3	1	5035	0	NSP	USA	-	-
1st	3549	Level3	0.897	2334	0	NSP	USA	S2S	5
2nd	1299	Telia	0.850	1747	0	NSP	Sweden	P2P	4
3rd	701	Verizon	0.849	1216	0	NSP	USA	P2P	3
4th	286	KPN	0.844	267	0	NSP	Netherlands	P2P	2
5th	6071	Unisys	0.843	4	1	-	USA	P2C	0
-	7018	AT&T	1	2418	0	NSP	USA	-	-
1st	701	Verizon	0.960	1216	0	NSP	USA	P2P	5
2nd	1299	Telia	0.835	1747	0	NSP	Sweden	P2P	4
3rd	577	Bell	0.832	314	1	NSP	Canada	C2P	0
4th	57866	Fusix	0.821	54	1	NSP	Netherlands	None	0
5th	1239	Sprint	0.820	378	0	NSP	USA	P2P	1
-	378	IUCC	1	4	2	Research	Israel	-	-
1st	3268	CyNet	0.929	3	2	Research	Cyprus	None	5
2nd	1930	FCT	0.927	10	2	Research	Portugal	None	4
3rd	40981	U. of Montenegro	0.925	2	2	Research	Montenegro	None	3
4th	57961	RASH	0.911	3	2	Research	Albania	None	2
5th	199354	ASREN	0.901	4	2	Research	Jordan	None	1
-	812	Rogers	1	159	1	Cable/DSL/ISP	Canada	-	-
1st	5769	Videotron Telecom	0.810	58	1	Cable/DSL/ISP	Canada	None	5
2nd	40788	Start Communications	0.796	10	1	Cable/DSL/ISP	Canada	None	4
3rd	15290	Allstream	0.736	148	1	Cable/DSL/ISP	Canada	None	3
4th	18634	Fusix	0.729	1	3	Cable/DSL/ISP	Canada	None	2
5th	30048	Convergia	0.728	4	1	Cable/DSL/ISP	Canada	None	1
-	15169	Google	1	54	1	Content	USA	-	-
1st	36385	Google	0.759	4	1	Content	USA	S2S	5
2nd	63293	Facebook	0.755	17	1	Content	USA	None	4
3rd	16591	Google	0.743	11	1	Cable/DSL/ISP	Canada	S2S	3
4th	133982	EXCITEL	0.739	6	2	Cable/DSL/ISP	India	None	0
5th	38726	VTC Digicom	0.736	16	1	NSP	Vietnam	None	0

TABLE IV
TIER-1 (TOP-5) RANKING GRADE FOR PRIMARY TIER-1 ASes (FOR ASes WITH RANKING GRADES GREATER THAN 0)

Tier-1 ASN	2914	3356	1299	6762	3257	209	7018	701	6453	3549	3491	1239
Ranking Grade	15	14	14	12	12	11	10	9	8	6	6	3

similarity of 0.897. The next 3 nearest neighbors are other tier-1 providers, while the 5th nearest neighbor, Unisys (AS6071), does not seem to be related to AS3356 and thus counted as zero; therefore, the total ranking grade is 14. Another tier-1 provider, AT&T (AS7018, not shown in table), also has 3 tier-1 providers among its 5 nearest neighbors (Verizon, Telia, and Sprint); Bell Canada (AS577), a large tier-2 provider; and Fusix, an NSP.

All the 5 nearest neighbors of IUCC (The Israeli academic network, AS378) are also Educational/Research ASes from Europe and the Middle East with small degree. Note that none of these ASes are connected directly. Also, note the high cosine similarity score. AS812 (Rogers Communications) is the largest Canadian telecom provider. All its 5 nearest neighbors are Canadian ‘Cable/DSL/ISP’ providers, two of them Videotron and Allstream are large national providers. For the last example, we chose AS15169 (Google). Its first and third nearest neighbors (NN) are also owned by Google. The second NN is a Facebook ASN. Note that all the cosine scores for Google are below 0.76.

Generally, for the above (and many other) examples, the embedding seems to capture functionality similarity between

ASes. In some cases, there is no sufficient learning data, and ASes get poor embedding, e.g., in the case of AS6071.

D. Exploration of Tier-1 ASes

Table IV display the ranking grades for all primary Tier-1 ASes which achieved scores greater than 0. We define Tier-1 ASes according to the clique at the top of the hierarchy of the CAIDA relationships dataset [30]. The Tier-1 ranking grades are calculated based on the 5 NNs for each AS; points are assigned if the neighbor is either a tier-1 provider or a sibling. A few Tier-1 ASes do not have even a single Tier-1 AS among their 5 NNs. To understand this, Figure 5 displays the Tier-1 AS embedding after dimension reduction using PCA. We apply a DB-SCAN algorithm (with a minimum of 2 samples in a cluster) to cluster the Tier-1 ASes. The size of each point in Figure 5 represents the AS degree, and the color corresponds to its cluster (black for no cluster). All primary Tier-1 ASes (including all the top 9 ASNs in the CAIDA AS ranking [23]) which have Tier-1 ranking grade greater than 0 belong to the ‘Green’ cluster, which consists of the highest-degree ASes. The ‘Red’ cluster consists of the European and

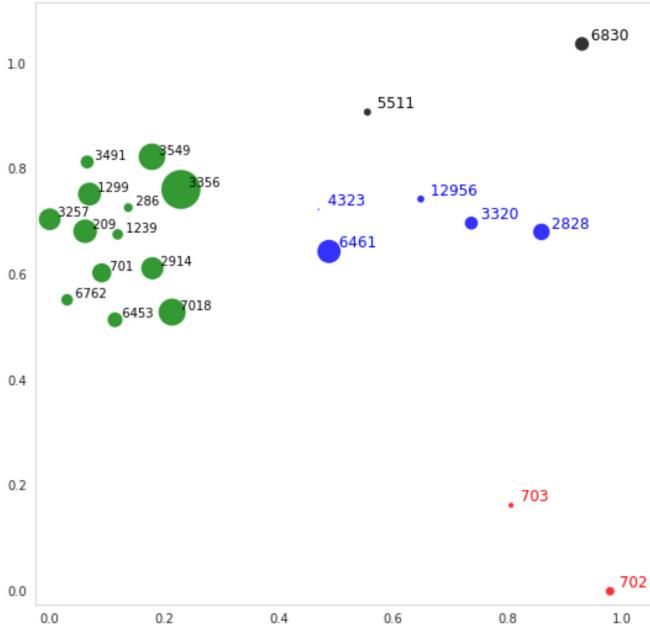


Fig. 5. PCA visualization for Tier-1 ASes based on *BGP2Vec* embeddings. Each point represents an AS with a size relative to its degree, and with a color corresponds to the cluster achieved by applying a DB-SCAN (Black points do not belong to any cluster).

Asian Verizon ASes, AS702, and AS703; these ASNs do not behave like classical tier-1 ANS since they operate in one continent. The ASN that is the farthest from any other tier-1 ASN is Liberty Global (AS6830), ranked 24th on the CAIDA list.

The average vector of a group of ASes is calculated by averaging each vector coordinate. The similarity between the average vector of all the Tier-1 ASes vectors and the average vector of all ASes is 0.721. On the other hand, the similarity of the average vector of all the stub ASes (ASes with transit-degree of 0) with the average vector is equal to 0.998.

E. Exploration of Countries

For each country, we create its vector by taking the average of all BGP2Vec's vectors that belong to the corresponding country. We use [23] to obtain the owner-country of each ASN. For some ASNs, the country affiliation is problematic since an ASN may operate in multiple countries, or an ASN may operate in a different country than its registration country. However, these mistakes are averaged out since the majority of ASNs are purely local.

As shown in Figure 6, the country vectors reveal the connection between countries. There are four major clusters in the graph: the upper one which belongs mostly to African and Middle-East countries, the left cluster which belongs mostly to Oceania and Eastern-Asia, the right cluster which belongs mostly to North-America, Europe, and Western Asia, and the bottom cluster which belongs mostly to South-America.

Countries in the same cluster have high cosine-similarity. For example, Germany, Holland, Belgium, Switzerland, Austria, Hungary, Poland, and Czechia are very close to each other, with an average cosine-similarity (between the country vectors) of 0.901 with a standard deviation of 0.033.

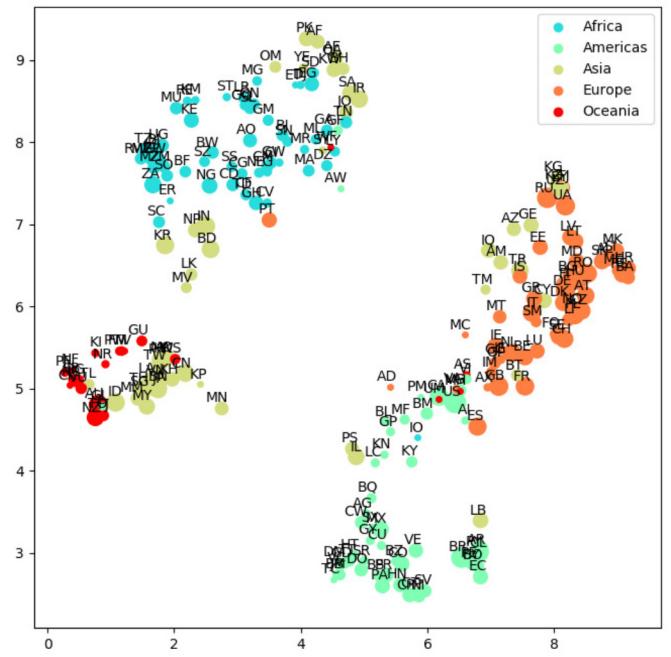


Fig. 6. Two-dimensional UMAP projection of average-country vectors. The dot size corresponds to the number of ASNs in a country.

Interestingly, Israel and Lebanon are located far away from all the other Middle-East countries, which are located near African countries. This can be explained by the different upstream providers that ASes are using in these nations: in the majority of the Middle East NTT (AS2914) and Orange (AS5511) are responsible for a significant portion of the transit market, while in Israel and Lebanon, their share is significantly smaller; Telecom Italia (AS6762) is a significant provider in Lebanon and the rest of the Middle East, but not as strong in Israel. This information was extracted from obtaining the first Tier-1/1.5 provider in the BGP route announcements towards destinations in these nations; we used RoutViews data from October 2018.

F. Exploration of Organizations

We use the list of organizations in the CAIDA ASRank dataset [23] to explore different organizations. The US Department of Defence has an average cosine similarity correlation, which is extremely high, 0.880. Moreover, among its nearest neighbors, we can find many other military and governmental organizations such as the US Air-Force, the USAISC headquarters, the Navy Network Information Center, and Microsoft's AS12076, which serves the US government.

We generated an average vector for each organization by taking the average of all of its ASNs vectors. Among the top-5 nearest Oracle neighbors, we find average vectors of IBM, Yelp, and Disney. Among the top-5 nearest neighbors of AT&T, we find L3 Technologies, Sony, and AT&T Japan.

G. Exploration of IXPs

We use [29] to collect all the ASNs that belong to each IXP of the 38 IXPs of Equinix and then create an average vector

TABLE V
A COMPARISON OF THE MEAN AVERAGE COSINE SIMILARITY CORRELATION FOR DIFFERENT CHARACTERISTICS

Characteristic	Number of Categories	mACS	Std. of ACS
Distance From Tier-1	7	0.660	0.167
Countries	147	0.750	0.079
Equinix IXPs	38	0.540	0.128
Organizational Ownership	91	0.800	0.101

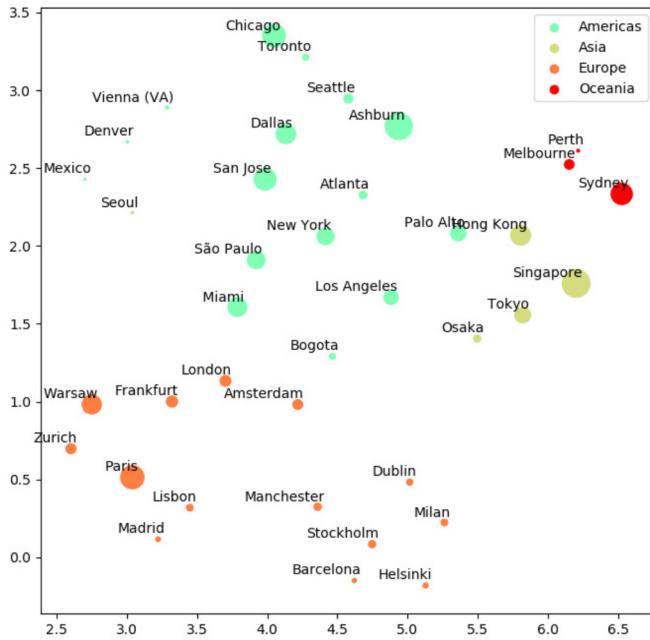


Fig. 7. Two-dimensional UMAP projection of average-IXP vectors of Equinix. The dot size corresponds to the number of participating ASNs.

for each IXP, as displayed in Figure 7. Clearly, the IXPs are clustered by continent.

H. Summary

Table V summarizes the effect of several characteristics over the average cosine similarity. For each characteristic, we calculate the average cosine similarity correlations (denoted by ACS) of all of its categories (for example, a specific country), and then calculate the mean ACS. For this evaluation, We evaluate only organizations with more than 9 ASNs each. Note that the average cosine similarity between all ASNs in the dataset is 0.551.

Overall, the *BGP2Vec* embedding preserve various (latent) characteristics: distance from Tier-1, business type of AS, ToR, geographical similarity, etc.

VIII. QUANTITATIVE EXPERIMENTS AND RESULTS

In this section, we report our experimental results for ToR classification and AS classification. We present a comparison of our ToR classification to best known previous results. Moreover, we will show that our method had found many mistakes in our dataset, emphasizing the strength of our results.

TABLE VI
A COMPARISON OF THE ToR CLASSIFICATION ACCURACY AND RECALLS

Algorithm	Accuracy	P2P Rc.	C2P Rc.	P2C Rc.
CAIDA AS Relationships Dataset (serial-1)				
GAO	38.4%	1.0%	99.1%	85.7%
SARK	81.9%	16.1%	95.4%	95.2%
NDToR CP-CORE	89.0%	99.9%	70.8%	71.2%
NDToR Kmax-CORE	84.6%	12.3%	99.5%	99.4%
RUAN	78.6%	2.3%	97.8%	97.7%
<i>BGP2Vec</i> - ANN	94.2%	89.0%	93.1%	98.5%
<i>BGP2Vec</i> - LR	81.6%	93.7%	51.1%	49.0%
<i>BGP2Vec</i> - SVM	76.5%	89.1%	58.6%	57.7%
<i>BGP2Vec</i> - KNN	92.6%	94.0%	90.0%	90.1%
<i>BGP2Vec</i> - D-KNN	92.1%	94.0%	89.6%	89.9%
CAIDA AS Relationships Dataset (serial-2)				
<i>BGP2Vec</i> - ANN	95.2%	98.0%	88.4%	87.6%
Labeled AS ToRs Dataset				
<i>BGP2Vec</i>	95.8%	82.1%	89.9%	98.2%
CAIDA	90.2%	90.0%	87.8%	90.6%
GAO	90.1%	10.5%	94.3%	98.6%
SARK	87.9%	19.9%	88.7%	95.5%
NDToR CP-CORE	65.7%	91.4%	61.0%	63.5%
NDToR Kmax-CORE	91.1%	17.9%	95.0%	98.9%
RUAN	90.2%	13.5%	93.4%	98.5%

A. Evaluation Criteria

We use the **accuracy** as a primary criterion to evaluate our model performance, which is defined as the proportion of examples for which the model produces the correct output of all predictions made. A formal definition of the accuracy for multiclass classification is

$$\text{Accuracy} = \frac{\sum_{i \in \text{classes}} TP_i}{\sum_{i \in \text{classes}} (TP_i + FP_i)},$$

where TP_i and FP_i are the true positive and the false positive of the class i , respectively. Furthermore, we also present the **recall** achieved for each class, which is defined by $Rc = \frac{TP}{TP+FN}$ (where FN is the false negative).

B. Hyper-Parameter Optimization

For the choice of hyper-parameters, we mainly followed the recommendations described in [45] and focused on analyzing the effects of different choices of the embedding size and the window size. Figure 8 shows the results of a grid search over different ASN embedding sizes (d) and window sizes in order to optimize these parameters to achieve the best accuracy for both ToRs classification and AS classification tasks. We performed a grid search for window sizes of 1, 2, and 3; and for embedding sizes in powers of 2 ranging from 2 to 512. In each experiment, we used the same architecture as depicted in Section VI, with only one modification, which is the output of the embedding layer.

Figure 8a shows that embedding sizes greater than 8, and different windows sizes give similar results for the ToRs classification problem. For the AS classification problem (Figure 8b), an embedding size of 256 with a window size of 3 achieve the best result; however, the difference in the accuracy is small above size 32: less than 1%. Since increasing the embedding size increases dramatically the number of parameters in the neural network, we select an embedding size of 32 and a window of 2.

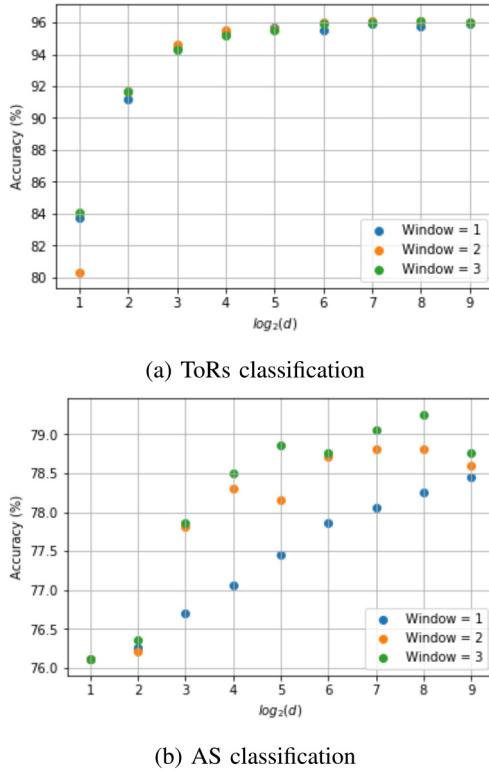


Fig. 8. Grid search results for; (a) ToRs classification accuracy and (b) AS classification accuracy, both as a function of the embedding size (d) and the window size of the *BGP2Vec*.

TABLE VII
A COMPARISON OF THE AS CLASSIFICATION
RESULTS OVER A SUBSET OF THE TEST SET

Algorithm	Accuracy	Transit/Access Rc.	Enterprise Rc.	Content Rc.
<i>BGP2Vec</i>	67.2%	99.0%	34.7%	7.1%
CAIDA	53.6%	92.2%	8.0%	14.3%

C. AS Classification Results

After training our ANN over the training set, we evaluate our method over a test set consists of 20% of the PeeringDB dataset, and achieve an accuracy of **79.2%**, and recalls of 98.0%, 23.8%, 17.0% for ‘Transit/Access,’ ‘Enterprise’ and ‘Content,’ respectively, better than All previous results.

We compare (see Table VII) our method to the CAIDA AS Classification Dataset [17], by creating a sub-set of the test set which contains only ASes that were classified by CAIDA. Our method achieves an accuracy of 67.2% over the subset, while CAIDA achieves an accuracy of only 53.6%. However, both methods perform poorly in classifying enterprises and content providers.

D. ToR Classification Results

We first follow the approach described in [12] for examining relationships between objects: the relationship between two ASNs is defined by taking the difference between the two ASN vectors, and the result is added to another ASN. In our example, which is presented in Figure 1, we choose the Verizon’s siblings {AS702, AS703}; add the subtraction of their vectors to the vectors of AS6805, AS26617, and AS28550; and

successfully obtained their siblings. Furthermore, among the 10-nearest neighbors of Verizon’s siblings, 4 ToRs are also siblings that belong to Verizon; AS2830, AS1321, AS6256, and AS12234.

The top part of Table VI compares *BGP2Vec* results to other previously suggested algorithms based on our CAIDA AS Relationship serial-1 dataset, which contains AS relationships inferred from BGP using the method described in [25]. In our comparison, we focused only on methods that are based on AS-level paths, thus do not include the PTE [19] algorithm. Shavitt *et al.* [8] observed a problem when executing the heuristic phases for inferring P2P (and S2S) relationships of the AToR [6] and BPP [20] algorithms. Thus, we omit them from the table and present here only the P2C and C2P assignments of both algorithms. The AToR algorithm succeeded to accurately infer 93.7% and 98.1% of the C2P ToRs and P2C ToRs, respectively, while BPP infers only 83.7% and 68.2% of these ToRs.

We implemented the following algorithms for the comparison: 1) the first ToR inference algorithm which was proposed by Gao [1] (referred to as GAO), 2) the SARK algorithm that was presented by Subramanian *et al.* [5], 3) the near deterministic AS ToRs inference algorithm [8] (i.e., ND-ToR) using two different cores; CP-Core and kmax-Core as described in Sec II, and 4) a recent algorithm (RUAN) introduced by Ruan and Varghese [28]. We implement this algorithm based on 22 Tier-1 ASes according to the clique at the top of the hierarchy of the CAIDA relationships dataset [30].

Note, that previous work could not take advantage of the labeled training set without a significant change. Even when we tried to take advantage of the training set and constructed a new core for the ND-ToR, which consists of the training set edges with their labeled ToRs, the method achieves poor results relative to the other cores.

A comparison of the ToR classification results is presented in Table VI. Our method achieves the best performance, with an accuracy of **94.2%** (*BGP2Vec* - ANN), 5.2% higher than the second-best algorithm (NDToR CP-Core).

The middle part of Table VI shows our *BGP2Vec* algorithm results using the CAIDA AS Relationships serial-2 dataset, which adds about 300k links inferred from BGP communities using the method described in [18]. Despite the imbalanced dataset, our method succeeds in achieving even better results compared to the serial-1 dataset with an accuracy of **95.2%**. Previous works, which were described before, achieved similar results for both datasets.

The bottom part of Table VI display our *BGP2Vec* algorithm results over the proprietary AS ToRs dataset. We trained and tested our model using the same dataset and achieved an accuracy of **95.8%**, which is even better than the result we achieve using the CAIDA dataset. Our results are significantly better than the results of the CAIDA [25] for the proprietary AS ToRs dataset (90.2%), which are obtained by comparing the two datasets.

For the last experiment, we compare our algorithm with ProbLink [26], which is, as far as we know, the most recent work in the field, even though, ProbLink is not based only on AS-level paths and uses additional information, such as sibling

relationships, BGP communities, and IXP information. While the overlap between the ProbLink dataset and the CAIDA AS Relationship dataset was only 62.8%, ProbLink achieved an accuracy of 94.9% on the overlap set. We also checked *BGP2Vec* on the same overlap set and achieved an accuracy of 94.8%. The overlap between ProbLink and the proprietary AS ToRs dataset was 78.5%. Based on the overlap set *BGP2Vec* achieved an accuracy of 95.14%, while ProbLink achieved an accuracy of 95.10%. To summarize, *BGP2Vec* achieves the same accuracy as ProbLink without the extra side information.

We manually explored some misclassified test results over the CAIDA dataset. For the 30 misclassified ToRs with the highest softmax scores (all above 0.999), 12 were correct (i.e., incorrect labels in the dataset), 12 were indeed misclassifications, 3 were siblings, and 3 were unclear. Namely, of the classifiable ToRs, half of our mistakes were actually correct.

For example, the ToR [AS5009, AS6939] is labeled by CAIDA as P2P, while our network predicts it correctly as C2P. By examining AS5009 routing, one can easily infer that AS6939 is its main transit provider since almost all of AS5009 traffic is traversing through AS6936 (Hurricane Electric). Since March 2018, CAIDA corrected this ToR. Another example is the ToR [AS52614, AS267221], which is labeled by CAIDA as P2C, while our network predicts it correctly as C2P. According to AS52614's IRR AS267221 is its provider, which is also clear by the fact that many of its routes traverse AS267221. We could not find any route from AS267221 that traverses AS52614.

Finally, for the proprietary dataset, we examined 25 ToRs that do not exist in our dataset. Typically, these are ASes from the edges of the Internet, where our database probably has many missing ToRs or misclassified ToRs (ASes from Brazil and Nepal together comprised of 38% of the 25 ToR's endpoints). For this challenging group, we got 18 correct classifications (most with softmax scores above 0.95), 4 errors (only 1 with a high softmax score), and 3 siblings.

Overall, when the softmax score is sufficiently high, our classification seems to work very well. The threshold seems to be in the range [0.85, 0.9], but it requires more manual data tagging to pin the exact threshold.

An interesting finding in our small manual experiments is the large percentage of sibling ToRs. This may hints that many of our misclassifications are attributed to siblings.

E. Testing *BGP2Vec* Embedding Performances Using Different Machine Learning Algorithms

In order to emphasize the strength of the *BGP2Vec* embedding, we apply ‘classic’ supervised and unsupervised machine learning algorithms for the ToRs classification problem. Table VI summarizes our results using a 64-dimensions embedding-vector for each Tor by concatenating together two 32-dimensions embedding-vectors, one for each AS.

We tested two basic supervised learning algorithms, Multinomial Logistic Regression (denoted as LR) and Support Vector Machine (SVM), which achieve accuracies of 81.6% and 76.5%, respectively. Table VI shows that both methods

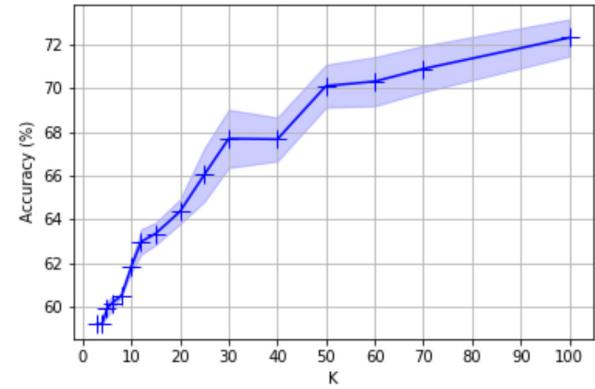


Fig. 9. K-Means accuracy over the CAIDA-s1 test-set for difference K values, the shaded area around the markers represents one standard deviation.

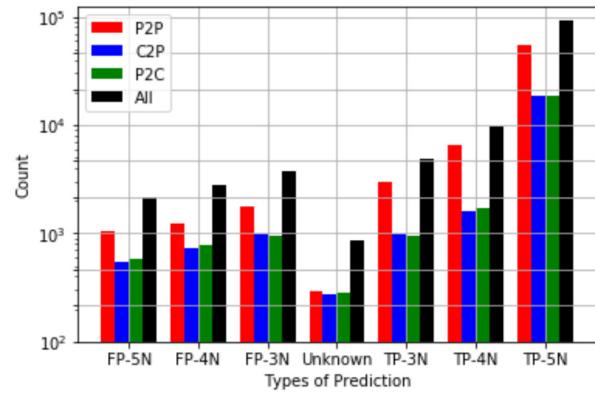


Fig. 10. Number of predictions of each type, based on D-KNN with K = 5 over the test set for; P2P, C2P, P2C and combined (All).

perform poorly for C2P/P2C ToRs, which is mainly due to the imbalanced dataset (for example, by applying weighted loss to balance the imbalances in the data, we achieve similar recalls for both P2P, C2P, and P2c ToRs).

In order to understand the strength of the ToR similarity, for each AS-pair, we first generate a distance-embedding vector by subtracting the embedding of two ASN vectors and get a 32-dimensions difference-vector. Then we apply the K-Nearest Neighbors (KNN) algorithm over the difference-vectors (denoted as D-KNN in Table VI) to determines the label of a given AS-pair according to the majority of the ToRs among the 5 nearest neighbors. The D-KNN achieves an accuracy of 92.1%. We also apply a KNN algorithm using the concatenated 64-dimension vectors and achieve an accuracy of 92.6%.

Figure 10 depicts the number of predictions of each type, based on the D-KNN algorithm with K=5 over the CAIDA's test set for; P2P, C2P, P2C, and combined (denotes as ‘All’). We consider 7 cases: TP-3N, TP-4N, TP-5N, which are True Positive predictions with 3, 4, 5 neighbors with the correct ToR prediction, correspondingly; FP-3N, FP-4N, FP-5N, which are False Positive predictions with 3, 4, 5 neighbors of same ToR, correspondingly; and unknown, which is the case where there are 2 neighbors of one class and other 2 neighbors of another class. As can be seen, 79.3% of predictions are of type TP-5N, i.e., all the ToRs of the 5 neighbors are correct, and overall

the correct predictions are above 92%, false predictions are about 7%, and in about 0.7% of the cases, there is no resolution. Namely, the *BGP2Vec* embedding difference-vectors characterize well the ToRs and achieve predictions which are better than previous algorithms. We note that comparison is a bit unfair as these predictions cannot start from scratch but are useful for labeling new ToRs for existing databases. However, what is important here is that the difference-vectors capture the ToRs.

We manually investigated the ‘FP-5N’ predictions in the CAIDA dataset by comparing them to the proprietary dataset (which we believe is more reliable since it is used daily and corrected when a label is problematic). Remember that in the ‘FP-5N’ case, all 5-neighbors had the same ToR, which is not the correct label. For a sample of 102 ‘P2P’ ToRs (as labeled by CAIDA), we find that 55 were actually predicted correctly by our model, namely CAIDA labels were wrong, 14 of them were both labeled (by CAIDA) and predicted (by our model) incorrectly. The rest was indeed ‘P2P’, and the model was wrong. However, out of a sample of ‘P2C’ and ‘C2P’ ToRs, most of the CAIDA’s labels were indeed correct.

Last, we apply the K-Means unsupervised-learning algorithm. Then we determine the class of each cluster by applying a majority vote. We test various values for K, and for each K, we run 10 experiments. For example, with K=10, we achieve an average accuracy of 61.6% with a standard deviation of 0.25% over the CAIDA-s1 test set. Previously presented algorithms that are based on a few close neighbors, such as KNN, achieve good results, much higher than the K-means, which is based on a larger cluster. In the larger cluster, other characteristics may be stronger for some clusters. This result shows that although similar ToRs are located close to each other (as can be concluded by the KNN results), they are not spatially arranged in clusters because of these additional strong characterizations of ASNs that are learned, such as the geographical location, type of ASN, etc.

In summary, our deep learning method achieves a state of the art results in a relatively short evaluation time, with an average inference time of 0.1 milliseconds (on a single CPU) for both AS classification and ToRs classification. Moreover, we show that ToRs can be inferred using simple machine learning algorithms based on *BGP2Vec* embedding.

IX. COMPARISON WITH OTHER EMBEDDING METHODS

In this section, we compare our *BGP2Vec* embedding method with the following commonly used embedding algorithms:

- *Spectral Embedding* [46]: A matrix factorization method which generates d-dimensional vector for each node from the d-smallest eigenvectors of the normalized Laplacian matrix of the undirected graph G . The graph G is generated based on the RV’s BGP announcements dataset.
- *Node2Vec* [34]: This method returns feature representations that maximize the likelihood of preserving network neighborhoods of nodes in d-dimensional feature space by using a second-order random walk approach to sample network neighborhoods for nodes and then training

a *Word2Vec* model based on the obtained walks. We evaluate several configurations of *Node2Vec* using the same hyper-parameters of the Skip-Gram algorithms as we used in our *BGP2Vec*, with only changing the ‘return’ parameter p , which controls the likelihood of immediately revisiting a node in the walk, and the ‘in-out’ parameter q , which allows the search to differentiate between “inward” and “outward” nodes. We generated 40 walks per ASN with a walk length of 6, which results in a total number of edges, which is equal to the total number of edges in our BGP announcements (RV) dataset.

- *DeepWalk* [31]: This method learns d-dimensional feature representations by simulating uniform short random walks. *DeepWalk* can also be described as a specific case of *Node2Vec* with $(p = 1, q = 1)$, i.e., the probability of the next step is always uniform.
- *SVD2Vec* [45]: A “count-based” representations method that generates d-dimensional vectors by factorizing the Positive Pointwise Mutual Information (PPMI) matrix using truncated Singular Value Decomposition (SVD). We run this method using the same window size and the number of negative samples we use for *BGP2Vec*.

We evaluate the feature representations obtained through these techniques on the two supervised learning tasks: ASN business types classification for nodes over a test set of the Peering-DB dataset and ToR classification for edges over a test set of the CAIDA AS Relationships (s-1). For each algorithm, we generate ToR representations using the *subtraction* operator and obtain 32-dimensional distance-vectors. To facilitate the comparison, we use the exact same test-sets (which contain 20% of the samples) for all of the experiments. For each algorithm, we conduct several experiments; randomly sampling different portions of the training set for each experiment. For all models, we use a one-vs-rest logistic-regression with L2 regularization implemented by *scikit-learn* [47] for the classification (note that here we use both a different edge operator and a different linear regression model than those we used for the evaluation in Table VI). We repeat this process 10 times and report the average performance in terms of accuracy scores (which in the case of a single-label classification are also equal to the Micro-F1 scores) for comparing performances in Figure 11. The trends are similar for Macro-F1 and are not shown.

Figure 11 shows that *BGP2Vec* outperforms all the other baseline algorithms while achieving an accuracy of 76.86% with only 10% of the training samples (i.e., ~1000 samples) over the ASN classification task, and 1.41% higher than the second-best (*Spectral Embedding*) with 80% of the training samples. For the ToR classification task, *BGP2Vec* achieved an accuracy of 67.67% with only 0.2% of the training samples (i.e., ~116,000 samples), 2.15% higher than the second-best (*DeepWalk*) with 80% of the training samples.

There are multiple reasons why *BGP2Vec* performs so much better than the other methods. Random walks, which are at the heart of many of these network embedding, tend to capture proximity and community structure [48], and thus not the node functionality we seek. For our case, random walks on the AS level graph may generate routes which are not feasible in the Internet due to VF violation [1].

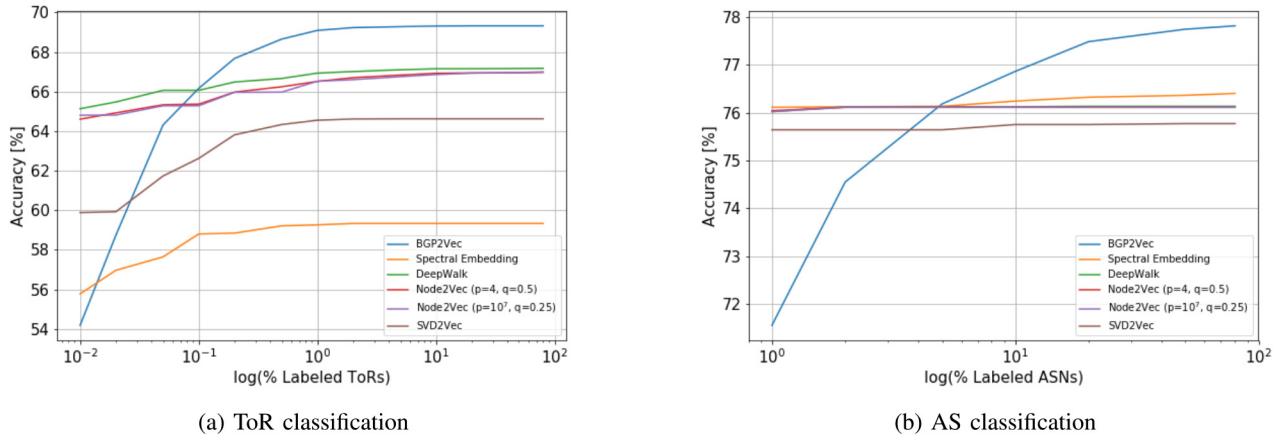


Fig. 11. A comparison of; (a) ToR classification accuracy scores and (b) AS classification accuracy scores using different embedding algorithms.

X. CONCLUSION

In this paper, we introduce a novel approach for numerical characterization of ASes using deep learning methods and applying it to achieve a state of the art results for two problems: AS classification and ToRs classification. Our solution consists of two stages: learning dense representations of ASNs from BGP routes (*BGP2Vec*) and applying ANN using the ASN embedding as inputs for the classification. As far as we know, we are the first to employ deep learning for this problem.

By exploring the ASN embedding, we can reveal latent characteristics of ASes. We found high cosine similarities between ASes which share the same function: Tier-1 ASes, universities, organizations, IXPs, and content providers, and also within a combination of function and geographical location.

Identifying the functional role of an ASN has additional applications. For example, recently, we published initial results [13] that show that with *BGP2Vec* we identify IP hijacking. We can detect IP hijack from benign route change since as long as the route uses similar functional ASNs along the path, the exact ASN is not essential. In a hijack attack, the ‘function’ of the ASes along the path is badly formed. We plan to study this direction further.

Sibling identification is mostly ignored by ToR classification papers since it is a hard problem. Our qualitative results suggest that *BGP2Vec* may be a good candidate for this task.

REFERENCES

- [1] L. Gao, “On inferring autonomous system relationships in the Internet,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, Dec. 2001.
- [2] D. Dolev, S. Jamin, O. Mokryny, and Y. Shavitt, “Internet resiliency to attacks and failures under BGP policy routing,” *Comput. Netw.*, vol. 50, pp. 1–14, Nov. 2006.
- [3] C. C. Demchak and Y. Shavitt, “China’s maxim—Leave no access point unexploited: The hidden story of China telecom’s BGP hijacking,” *Military Cyber Affairs*, vol. 3, no. 1, p. 7, Oct. 2018.
- [4] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, “A survey among network operators on BGP prefix hijacking,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 1, pp. 64–69, Jan. 2018.
- [5] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, “Characterizing the Internet hierarchy from multiple vantage points,” in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Jun. 2002, pp. 618–627.
- [6] R. Cohen and D. Raz, “Acyclic type of relationships between autonomous systems,” in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, 2007, pp. 1334–1342.
- [7] X. Dimitropoulos *et al.*, “As relationships: Inference and validation,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 29–40, Jan. 2007. [Online]. Available: <http://doi.acm.org/10.1145/1198255.1198259>
- [8] U. Weinsberg, Y. Shavitt, and E. Shir, “Near-deterministic inference of AS relationships,” in *Proc. ConTel*, Zagreb, Croatia, Jun. 2009, pp. 191–198.
- [9] (Univ. Oregon Adv. Netw. Technol. Center, Eugene, OR, USA). *University of Oregon Route Views Project*, 2018. [Online]. Available: <http://www.routeviews.org/>
- [10] “Internet Routing Registry.” 2018. [Online]. Available: <http://www.irr.net/>
- [11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2013, pp. 3111–3119.
- [13] T. Shapira and Y. Shavitt, “A deep learning approach for IP hijack detection based on ASN embedding,” in *Proc. Workshop Netw. Meets AI ML*, New York, NY, USA, 2020, pp. 35–41.
- [14] T. Shapira and Y. Shavitt, “Unveiling the type of relationship between autonomous systems using deep learning,” in *Proc. IEEE/IFIP NOMS Int. Workshop Anal. Netw. Service Manage. (AnNet)*, Apr. 2020, pp. 1–6.
- [15] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy, “Revealing the autonomous system taxonomy: The machine learning approach,” in *Proc. Passive Act. Netw. Meas. Workshop (PAM)*, Mar. 2006, pp. 1–10.
- [16] A. Dhamdhere and C. Dovrolis, “Ten years in the evolution of the Internet ecosystem,” in *Proc. 8th ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2008, pp. 183–196.
- [17] “The CAIDA UCSD AS Classification Dataset.” Nov. 2018. [Online]. Available: http://www.caida.org/data/as_classification.xml
- [18] V. Giotsas, S. Zhou, M. Luckie, and K. C. Claffy, “Inferring multilateral peering,” in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 247–258.
- [19] J. Xia and L. Gao, “On the evaluation of AS relationship inferences,” in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 3, Nov. 2004, pp. 1–10.
- [20] G. D. Battista, M. Patrignani, and M. Pizzonia, “Computing the types of the relationships between autonomous systems,” in *Proc. IEEE INFOCOM*, vol. 1, Mar. 2003, pp. 156–165.
- [21] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos, “A simple conceptual model for the Internet topology,” in *Proc. Global Telecommun. Conf. (GLOBECOM)*, vol. 3, 2001, pp. 1667–1671.
- [22] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, “A model of Internet topology using k-shell decomposition,” *Proc. Nat. Acad. Sci. (PNAS)*, vol. 104, no. 27, pp. 11150–11154, 2007.
- [23] “CAIDA Rank.” Jul. 2018. [Online]. Available: <http://as-rank.caida.org/>
- [24] Y. Shavitt and E. Shir, “DIMES: Let the Internet measure itself,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 71–74, Oct. 2005.

- [25] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and K. C. Claffy, "AS relationships, customer cones, and validation," in *Proc. Internet Meas. Conf.*, 2013, pp. 243–256.
- [26] Y. Jin, C. Scott, A. Dhamdhere, V. Giotsas, A. Krishnamurthy, and S. Shenker, "Stable and practical AS relationship inference with ProbLink," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Boston, MA, USA, Feb. 2019, pp. 581–598.
- [27] G. Feng, S. Seshan, and P. Steenkiste, "UNARI: An uncertainty-aware approach to AS relationships inference," in *Proc. 15th Int. Conf. Emerg. Netw. Experiments Technol.*, New York, NY, USA, 2019, pp. 272–284.
- [28] L. Ruan and J. S. Varghese, "Computing observed autonomous system relationships in the Internet," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Rep. 367, 2014.
- [29] "PeeringDB." 2018. [Online]. Available: <http://www.peeringdb.com>
- [30] "T-Cell Receptor Dataset." 2018. [Online]. Available: <http://www.caida.org/data/active/as-relationships/>
- [31] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 701–710.
- [32] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, 1999.
- [33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
- [34] A. Grover and J. Leskovec, "Node2Vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 855–864.
- [35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [36] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. Workshop New Challenges NLP Frameworks*, Valletta, Malta, May 2010, pp. 45–50.
- [37] Y. L. Cun *et al.*, "Handwritten zip code recognition with multilayer networks," in *Proc. 10th Int. Conf. Pattern Recognit.*, vol. 2, Jun. 1990, pp. 35–40.
- [38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
- [39] D. Campbell, R. A. Dunne, and N. A. Campbell, "On the pairing of the Softmax activation and cross-entropy penalty functions and the derivation of the Softmax activation function," in *Proc. 8th Aust. Conf. Neural Netw.*, Melbourne, VIC, Australia, 1997, pp. 181–185.
- [40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [41] F. Chollet *et al.*, "Keras." 2015. [Online]. Available: <https://github.com/fchollet/keras>
- [42] M. Abadi *et al.*, "TensorFlow." 2015. [Online]. Available: <https://www.tensorflow.org/>
- [43] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.
- [44] M. Huth and B. Fabian, "Inferring business relationships in the Internet backbone," *Int. J. Netw. Virtual Org.*, vol. 16, no. 4, pp. 315–345, Feb. 2016.
- [45] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Trans. Assoc. Comput. Linguist.*, vol. 3, pp. 211–225, May 2015.
- [46] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Min. Knowl. Disc.*, vol. 23, no. 3, pp. 447–478, 2011.
- [47] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
- [48] R. A. Rossi, D. Jin, S. Kim, N. K. Ahmed, D. Koutra, and J. B. Lee, "On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 5, p. 63, Aug. 2020.



Tal Shapira (Graduate Student Member, IEEE) was born in Rishon-Lezion, Israel, in 1991. He received the B.Sc. degree in physics and minor in mathematics from the Hebrew University of Jerusalem, Jerusalem, Israel, in 2012, the M.Sc. degree in electrical engineering (*magna cum laude*) from Tel-Aviv University, Tel-Aviv, Israel in 2018, and the Ph.D. degree from Tel-Aviv University, Tel-Aviv, Israel, in 2022.

From 2009 to 2012, he served as a cadet with the Talpiot elite Israeli Defense Forces Program. From 2013 to 2015, he served with Israeli Prime Minister Office as an RF and Antennas Engineer, from 2015 to 2018 as a Data Scientist and a Data Science Team Leader, and in the last two years of his military service as a Head of the Cybersecurity Research and Development Group. After graduation, he worked as the Head of Deep Learning and Computer Vision Algorithms Group in an automotive startup called Guardian Optical-Technologies. In 2020, he co-founded RecoLabs Inc., where he serves as the Chief Scientist. He has been a Postdoctoral Fellow with the School of Computer Science, Reichman University, Herzliya, Israel, since 2022. His research focuses on deep learning, computer networks, and cybersecurity.



Yuval Shavitt (Senior Member, IEEE) received the B.Sc. degree in computer engineering (*cum laude*), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion—Israel Institute of Technology, Haifa, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served with Israel Defense Forces first as a System Engineer and the last two years as a Software Engineering Team Leader. After graduation, he spent a year as a Postdoctoral Fellow with the Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA. From 1997 to 2001, he was a Member of Technical Staff with Networking Research Laboratory, Bell Laboratories and Lucent Technologies, Holmdel, NJ. Since 2000, he has been a Faculty Member with the School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel. His recent research focuses on Internet measurements, and deep learning solutions for various networking problems. He served as a TPC member for many networking conferences, and on the executive committee of INFOCOM 2000, 2002, and 2003, and a Guest Editor for JSAC and JWWW. He was a TCP Co-Chair for TMA 2011, Keynote Speaker at PAM 2012, and an Editor of Computer Networks 2003–2004. In 2014, he co-founded BGProtect LTD, a company that monitors the Internet for IP hijack attacks, where he serves as the CTO.