AP2Vec: An Unsupervised Approach for BGP Hijacking Detection

Tal Shapira, *Graduate Student Member, IEEE*, and Yuval Shavitt, *Senior Member, IEEE*

***Abstract*—BGP hijack attacks deflect traffic between endpoints through the attacker network, leading to man-in-the-middle attacks. Thus its detection is an important security challenge. In this paper, we introduce a novel approach for BGP hijacking detection that is based on the observation that during a hijack attack, the functional roles of ASNs along the route change. To identify a functional change, we build on previous work that embeds ASNs to vectors based on BGP routing announcements and embed each IP address prefix(AP) to a vector representing its latent characteristics, we call it AP2Vec. Then, we compare the embedding of a new route with the AP embedding that is based on the old routes to identify large differences. We compare our unsupervised approach to several other new and previous approaches and show that it strikes the best balance between a high detection rate of hijack events and a low number of flagged events. In particular, for a two-hour route collection with 10-90,000 route changes, our algorithm typically flags 1-11 suspected events(0.01-0.05% FP). Our algorithm also detected most of the previously published hijack events.**

***Index Terms*—Internet security, BGP, IP hijack detection, deep learning, AP embedding.**

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 autonomous system is assigned a globally unique number, also called an Autonomous System Number(ASN), and advertises one or more IP address prefixes(APs). Interdomain Routing between ASes is determined by the Border Gateway Protocol(BGP). BGP is a path-vector routing protocol that uses routing update messages to propagate routing changes. Each routing update lists the entire AS path to reach an IP address prefix(AP) and carries one or more BGP attributes. BGP allows each AS to choose its own policy on accepting routes(according to its import policy), selecting the best routes, and announcing routes(according to its export policy). An AS routing policy is usually determined by the commercial contractual relationship with other administrative domains.

Like many other basic Internet services, BGP was not designed to be secured, and as a result, attacks on BGP were designed to divert traffic to the attacker network. The attacker can then send the traffic to its original destination- forming a man-in-the-middle attack, or it can use the hijacked IP space for various nefarious activities such as sending spam or setting impersonation sites. In recent years, there have been many reports of BGP hijack attacks [1], [2], e.g., more than 40% of the network operators reported that their organization had been a victim of a hijack attack in the past [1]. Attacks are attributed both to governments and criminal organizations.

To hijack an AP, a BGP speaker can announce this AP as its own. If the owner with a larger AP announces this address space, then due to the longest prefix matching(LPM) rule, all the traffic to the hijacked AP will be diverted to the attacker network. If the owner announces the same AP(most likely a/24), only traffic from a portion of the origins will be diverted. This type of hijack attack can be detected quite easily by tracking changes in the origin AS of APs [3].

To avoid easy detection, one can hijack traffic by pretending to be on an attractive path towards the destination. This can be done by pretending to be a direct upstream provider of the hijacked AS or by announcing a route to peers(through exchange points) using the preference for peers over providers. This type of hijack attack is harder to detect, and it is the main motivation for our work.

It is important to note that routes may be deflected in unreasonable ways, also due to human error, not necessarily due to malicious acts. Even such benign deflections expose traffic to MITM attacks, e.g., by traversing networks, which are involved in espionage and may lurk for interesting data. It is almost impossible to know the cause of a deflection without knowing the intent behind people's actions. Thus, we follow previous work [4], and throughout this paper, we will also use the term BGP hijacking in the context of misconfiguration.

To tackle IP hijack attacks, two types of solutions have been proposed: reactive mechanisms that improve or amend the current routing protocols in order to make hijack attacks harder, or detection solutions that alert operators when a hijack occurs. The most notable reactive mechanisms to deal with the attack are RPKI [5] and BGPsec [6]. However, most operators have not deployed them and are reluctant to deploy them due to technical and financial costs [7], [8].

IP Hijack detection is a hard task, and previous work fails to provide solutions that are practical on a global scale. A number of papers [9], [10] suggested to alert when new AS links appear in a route, namely links that were not seen in the past. Gao [10] examined several heuristics and claimed as a best-case to have 0.02% false alarm per day for 2006 Internet, which is equivalent to over 6000 false alarms when analyzing a single RouteViews OIX file(with over 30M routes). The Artemis system [9] claims that for 28% of the ASes, there will be more than two false alarms per 3 days, namely in the current Internet, we will have about 15,000 false alarms per day. ARTEMIS also requires access to local monitors and other local ground truth, which may not be trivial. In general, ARTEMIS is designed to use by a single operator locally,

while we are looking at a global detection system. Other detection solutions rely on an increase in RTT [11](which produces too many events); or identifying large bursts of BGP announcements [12], which can detect only large scale events.

We are only aware of one direction that is using deep learning for this IP hijack detection, termed BGP2Vec [13], and our work extends this approach. The main problem with BGP2Vec is that it requires a labeled data system to detect a hijack attack. Obtaining such labeling is challenging.

In this paper, we introduce a novel approach for detecting BGP hijack attacks on a global scale, which is based on the observation that during a hijack attack, the ASes along the route have different functional roles than the usual route, which is true regardless of the hijack type. We term a route with functional changes of one or more of its ASNs, as a structurally-changed route. We also observe that when non-malicious route changes occur, the functionality along the route does not change. The functional role of an AS is not well defined. While in some cases it is easy to define the function that a certain AS plays, e.g., a tier-1 provider or a cloud provider, in general, an AS can have a more complex description of its function in the routing system, e.g., a tier-2 provider in Europe. To overcome the difficulty of defining the important functional classes and classifying the ASes into these classes, we use deep-learning-based embeddings that automatically perform this task.

To capture an ASN functionality, Shapira and Shavitt [13] build on the excellent results from Natural Language Processing(NLP) analysis where tools such as Word2Vec [14] are successfully used to map words in a language to their latent functional roles. Thus, they treated the BGP paths as sentences in NLP and introduced an extension of word vectors for creating a dense vector representation of ASNs(BGP2Vec). Using the BGP2Vec embedding, they trained a recurrent neural network using a labeled dataset to classify BGP routes as standard or hijacked routes. Our goal is to remove the dependence on the labeled dataset.

We suggest two methods to use the ASN embedding in order to identify functional changes along the route. First, to partition the ASNs into several functional classes(based on the embedding) and to examine changes in the functional roles of the ASNs along the path between two time periods. In the second method, we extend the concept of BGP2Vec and introduce AP2VEC, which embeds each AP to a dense vector. Similar to Paragraph Vectors [15] in the field of NLP, we use routes as sentences and train each AP Vector to predict ASNs in the path. Unlike BGP2Vec, we do not use any labeled datasets and present an unsupervised system for BGP hijacking detection.

II. RELATED WORK

Approaches for defending against BGP hijacking can be sorted into two categories: prevention and detection. BGP prefix hijacking prevention solutions, also called proactive solutions, are based on cryptographic authentications such as RPKI and BGPsec [5], [6], [16], [17]. There are many different approaches for the detection of BGP hijacking. We divide these approaches into three main categories, based on the information they use: 1) Control-plane approaches [9], also called passive solutions. These methods analyze routing information from a distributed set of BGP monitors and route collectors to detect anomalous behavior. 2) Data plane approaches [2], [11], [18], [19] - only rely on real-time data plane information that is obtained from multiple sensors that deploy active probing (pings/traceroutes). Some of these methods are based on analyzing IP TTL (Time to Live) or an increased RTT (round-trip delay time), and others are based on analyzing changes in AS links.3) Hybrid approaches [20], [21], [22] - These approaches use both control plane and data plane information and sometimes also use external databases to perform joint analysis.

Several recent works [4], [23], [24], [25], [26] examined machine learning techniques with manually generated features to identify malicious origin ASes; mainly leveraging historical BGP data from RouteViews [27] and RIPE. Testart et al. [25] focused on identifying dominant characteristics of serial hijackers over a long-term period of 5 years (such as intermittent AS presence, short prefix origination duration, etc.). They generated dominant features, used a tree-based machine learning classifier model for identifying ASes with BGP origination patterns similar to serial hijackers, and identified about 900 ASes with suspected malicious behavior.

Cho et al. [4] classified hijack events in order to understand the nature of reported events. They introduced four categories of BGP hijack attacks: typos, prepending mistakes, origin changes, and forged AS paths. Unlike previous works, their work aimed to classify detected hijack events and not detect new hijack events. They generated five features, including AS hegemony [26] (a metric that quantifies the likelihood of an AS to lie on paths toward particular destination IP prefixes), and used a Random Forest classifier, which achieved an accuracy of 95.71% over their dataset.

In recent work, Moriano et al. [12] proposed an algorithm that leverages the burstiness of disruptive BGP updates to provide early detection of large-scale malicious incidents using local collector data. They validated the effectiveness of their method by analyzing four cases of large-scale routing anomalies. However, their method is less effective in detecting small-scale events.

III. THE DATASET

As mentioned in Section I, our approach is based on analyzing BGP announcements. For this end, we use the RouteViews' BGP paths dataset (RV) [27], which contains BGP paths collected from about 40 vantage points.

A challenge in studying BGP hijacking is the limited availability of ground truth hijack events. Therefore, in order to validate our results, we collected 13 BGP AS events from several blogs including Dyn [28], RIPE [29], BGPMon [30], APNIC [31], ORACLE [32], MANRS [33], and BGProtect [34] as sources for hijack/leak events that were manually checked by experts and reported as suspicious. The events are from February 2008 to June 2019 (see Table III). All are long enough to be captured in the RV oix files. Among the

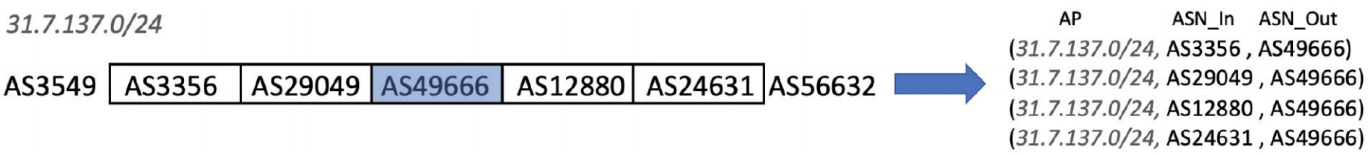

Fig. 1: An example for generating input/output ASNs and AP for the training process of AP2Vec.

events, 7 events were also analyzed by Cho et al. [4], and 3 were analyzed by Moriano et al. [12]. Furthermore, we also test our approach using several arbitrary samples from October 2020 (see Table IV) in order to test the False Alarm behavior.

For each event, we retrieve both the RV's file corresponding with the start time of the event and one RV file before. For example, for an event that started at 2018-04-24 11:05, we use the files of 2018-04-24 12:00 and 2018-04-24 10:00; each contains BGP announcements from two hours.

IV. MOTIVATION AND PROBLEM STATEMENT

We consider the problem of BGP hijacking detection by detecting hijacked APs and a hijacker ASN. More formally, let $\mathcal{P}_t$ be a collection of AS-level paths(e.g., BGP announcements) and $\mathcal{T}_t$ their corresponding origin APs, which are collected at time t, such that each path $p \in \mathcal{P}_t$ consists of a series of ASNs and has a corresponding origin AP $\in \mathcal{T}_t$, as demonstrated in Figure 1. Our objective is to detect a collection of hijacked paths $p_i, \ldots, p_{1+l}$, their origin AP and the hijacker ASN, which is involved in all of the hijacked paths.

Shapira and Shavitt [13] showed that they could use supervised learning to differentiate between a BGP hijack event and a benign route change since in a benign routing, changed ASNs are replaced with functionally similar ones. However, in a hijack attack, the sequence of ASN functions along the path changes. In many cases, this may result in the violation of valley-free routing [10], so this structural change detection can be viewed as a generalization of detecting valley-free violation.

We extend the method that was introduced in [13] and propose an unsupervised method, which learns simultaneously representations of ASNs and APs that capture the AS-level graph structure based entirely on BGP announcements. Consequently, we utilize the significant information about the dependence of ASNs to capture the functional role of each ASN, as well as the similarity between a path and an origin AP.

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 [35]. One of them is embedding discrete values to continuous N-dimensional vectors. This technique is broadly used in the field of Natural Language Processing (NLP), also known as word embedding [14], [36] or Neural Language Modeling, which 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 both syntax and semantic.

Word2Vec [14] is one of the most popular unsupervised data-driven techniques to learn word embeddings. It uses a shallow neural network trained on large amounts of unstructured text data. Word2Vec has two important benefits; first, its representations exhibit linear structure that makes precise analogical reasoning possible. For example: $\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman}$ resembles the closest vector to the word Queen [36]. The second benefit is that in contrast to one-hot encoding, which maps 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 maps 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 opposite training objectives: 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 CBOW model. In this model, we analyze each sentence in the corpus and train the network to predict each word in the sentence using the words within a certain range before and after the current word as context. 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 a unique one-hot encoding vector represents each word. The hidden layer is a fully-connected layer of size equals to the embedding size d. Within a fully-connected layer, neurons between two adjacent layers are fully connected, meaning that each neuron in the current layer is connected to every neuron in the previous layer, and each connection has its own weight. The hidden layer weight matrix $W_{|\mathcal{V}|\times d}$ and the output layer weight matrix $W'_{d\times|\mathcal{V}|}$ 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.

VI. METHOD

We aim to identify BGP hijack attacks where the hijacker injects an ASN into the route to deflect the traffic toward its network. To this end, we perform the following stages(see Figure 3):

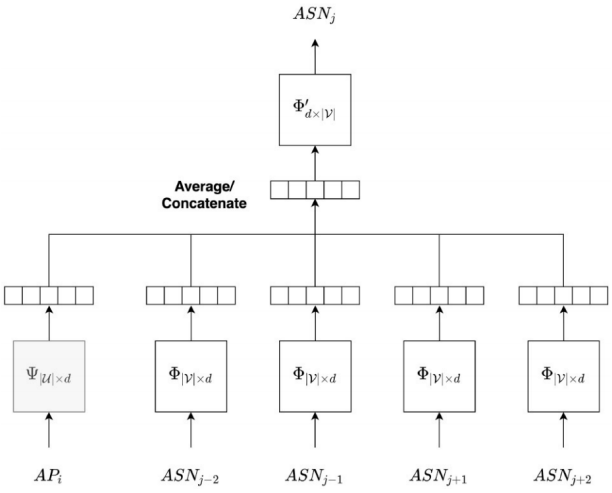

Fig. 2: Our AP2Vec architecture, where $|\mathcal{V}|$ is the number of ASNs, $|\mathcal{U}|$ is the number of APs, $d = 32$, and $w = 2$.

A. AP2Vec

As mentioned in Section I, in the first stage, we produce a d-dimensional continuous vector representation for each ASN and each AP. As in the training process of document embedding in NLP, i.e., Doc2Vec [15], we train our network over a large corpus of APs $\mathcal{T}$, such that each AP has several BGP paths from $\mathcal{P}$ (described in Section III), one for each vantage point. The APs and their paths are equivalent to documents and their sentences in NLP tasks. We apply a similar Distributed Memory Model of Paragraph Vectors (PV-DM) as introduced in [15], such that we predict each ASN in a certain AS-path based on the corresponding AP and ASNs within a certain range before and after the target ASN (as shown in Fig. 2). As a result, the model learns in parallel both to characterize an AP by the ASNs used to reach it and an ASN by its context (a combination of neighboring ASNs and the origin AP), i.e., where Φ_v and Φ'_v are the input and output vectors of AS v simultaneously. This is one of the main reasons for choosing the PV-DM model for this task.

As presented in Algorithm 1, we begin by building vocabularies of all the ASNs (line 2) and the APs (line 3) in the dataset and randomly initializing the embeddings for all ASNs ($\Phi \in \mathbb{R}^{|\mathcal{V}|\times d}$) (line 4) and APs ($\Psi \in \mathbb{R}^{|\mathcal{U}|\times d}$) (line 5) in the vocabularies. Then for each path and its corresponding AP, we apply the PV-DM model (Algorithm 2) as follows. Given an AS-level path $\{v_1, v_2, v_3, \ldots, v_N\}$ and a context window of size w, the PV-DM model attempts to predict each origin AS v_j by its context C, which is defined by its surrounding ASes and corresponding AP u_i:

$$C = \{v_{j-w}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_{j+w}, u_i\}. \tag{1}$$

More formally the objective of the PV-DM model is to learn the latent representations $\Phi \in \mathbb{R}^{|\mathcal{V}|\times d}$ and $\Psi \in \mathbb{R}^{|\mathcal{U}|\times d}$, which maximize the average log co-occurrence probability:

$$\frac{1}{N} \log \Pr(v_j \mid C; \Phi, \Psi). \tag{2}$$

As presented in [14], the basic neural-network based Skip-Gram formulation defines the conditional probability $\Pr(v_{j+k} \mid v_j)$ using the softmax function:

$$\Pr(v_{j+k} \mid v_j) = \frac{\exp(\Phi'^{\top}_{v_{j+k}} \Phi_{v_j})}{\sum_{v\in\mathcal{V}} \exp(\Phi'^{\top}_{v} \Phi_{v_j})} \tag{3}$$

where Φ_v and Φ'_v are the input and output vectors of AS v. In our case of the PV-DM model there is a hidden layer that averages the context vectors:

$$h = \frac{1}{|C|} \sum_{v\in C} v. \tag{4}$$

Then the conditional probability $\Pr(v_j \mid C)$ is defined by:

$$\Pr(v_j \mid C; \Phi, \Psi) = \frac{\exp(\Phi'^{\top}_{v_j} h)}{\sum_{v\in\mathcal{V}} \exp(\Phi'^{\top}_{v} h)} \tag{5}$$

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 [14], as a more efficient way of deriving ASN embeddings, by replacing the log-likelihood term in the PV-DM objective with the term:

$$\log \sigma({\Phi'_{v_j}}^{\top} h) + \sum_{i=1}^{k} \mathbb{E}_{v_i\sim P_n(v)}[\log \sigma(-{\Phi'_{v_i}}^{\top} h)], \tag{6}$$

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 [14].

Our neural network has two types of inputs: $2w$ inputs for the ASNs in the context window and one for the AP. Each input is embedded into a d-feature vector with an embedding matrix $\Phi_{|\mathcal{V}|\times d}$ for ASNs and $\Psi_{|\mathcal{U}|\times d}$ for APs; these are the ASN-vectors and AP-vectors that are learned by the model. The embedding layers are connected by an average operation (the hidden layer) of size d. The last layer is the output layer of size $|\mathcal{V}|$ for the desired output ASN, with a weight matrix $\Phi'_{d\times|\mathcal{V}|}$.

The hyper-parameters we choose, are based on optimization for BGP2Vec [13]: the window w is set to 2 (see Figure 1), and the embedding size is set to 32. In order to improve the representation, we use 5 negative samples [14] for each target ASN. We build and run our network using the Gensim [37] library. The training procedure is done by feeding the network with pairs of context C and target ASN v_j; the inputs are one-hot vectors representing the input ASNs and AP and the training output, which is also a one-hot vector representing the output ASN. Then applying gradient descent learning [38] (also known as back-propagation) to adjust the weights of the network, in order to maximize the log probability of any target ASN given the context ASNs and corresponding AP based on Equations (2), (6). We repeat this process for 3 epochs.

Algorithm 1 AP2Vec $(\mathcal{P}, \mathcal{T}, w, d, \alpha)$

Require: $\mathcal{P}$: AS paths list, $\mathcal{T}$: corresponding origin APs, w: window size, d: embedding size, α: learning rate
Ensure: Matrix of ASN representations $\Phi \in \mathbb{R}^{|\mathcal{V}| \times d}$, matrix of AP representations $\Psi \in \mathbb{R}^{|\mathcal{U}| \times d}$

```
Generate vocabulary V from P
Generate vocabulary U from T
Sample Φ from R^{|V|×d}
Sample Ψ from R^{|U|×d}
for each path ∈ P & u_i ∈ T do
    PVDM(Φ, Ψ, path, u_i, w, α)
end for
return Φ, Ψ
```

Algorithm 2 PVDM $(\Phi, \Psi,$ path, u, w, $\alpha)$

```
for each v_j ∈ path do
    J(Φ, Ψ) = −log Pr(v_j | Φ(v_{j−w}), ..., Φ(v_{j+w}), Ψ(u))
    Φ = Φ − α ∗ ∂J/∂Φ
    Ψ = Ψ − α ∗ ∂J/∂Ψ
end for
```

B. BGP Hijacking Detection

Based on the AP2Vec model, we can now describe the main method for BGP hijacking detection. Given two consecutive RV records, we apply the AP2Vec model on the first record and obtain both ASN-vectors and AP-vectors. For each AP $u' \in \mathcal{T}_t$ we define its parent AP $u \in \mathcal{T}_{t-1}$ as the minimal super-prefix of the current AP. This is because one of the most common and problematic hijack attacks is the 'more-specific origin change,' in which a hijacker announces a sub-prefix of the prefix announced by the legitimate AS, therefore in this case, the AP is not part of the preceding record $\mathcal{T}_{t-1}$. Be aware that in the case that the AP is part of the preceding record, it is also equal to the parent AP. It is worth mentioning that our method can not be applied in a case where there is no parent AP, which is the case of unallocated or unused APs.

At the first phase of our unsupervised method, we compare each path $p' \in \mathcal{P}_t$ in the second record, with the path $p \in \mathcal{P}_{t-1}$, which shares the same vantage point (ASN) and parent AP, from the first record. If the paths differ $(p' \neq p)$, we propose two methods to detect a structural change in the path (see Figure 3):

1) **Similarity between a path and an AP (PAPES)**: as mentioned in Section V, one of the main advantages of the Doc2Vec model is its ability to generate vectors for new paragraphs, and in our case a new AP or even specific paths. This made by adding more columns to the hidden layer of the AP vectors $(\Psi_{(|\mathcal{U}|+1)\times d})$ and applying gradient descent while holding the rest of the neural network's weights fixed $(\Phi_{|\mathcal{V}|\times d}$ and $\Phi'_{d\times|\mathcal{V}|})$. Then, for a given path, we generate its embedding vector (we choose to run 50 epochs of inference) and calculate the similarity between the generated vector and the AP2Vec vector of its parent AP Ψ_u. We define similarity between two AP2Vec vectors $\Psi_{u_i}, \Psi_{u_j} \in \mathbb{R}^d$ of the corresponding APs $u_i, u_j \in \mathcal{U}$ as the cosine similarity:

$$\text{sim}(\Psi_{u_i}, \Psi_{u_j}) = \frac{\Psi_{u_i}^T \Psi_{u_j}}{\|\Psi_{u_i}\|_2 \cdot \|\Psi_{u_j}\|_2}, \tag{7}$$

and choose a maximal threshold th_{cs} which determines if the structural change of the path is significant, i.e., if

$$\text{sim}(\Psi_u, \Psi_{p'}) < th_{cs} \tag{8}$$

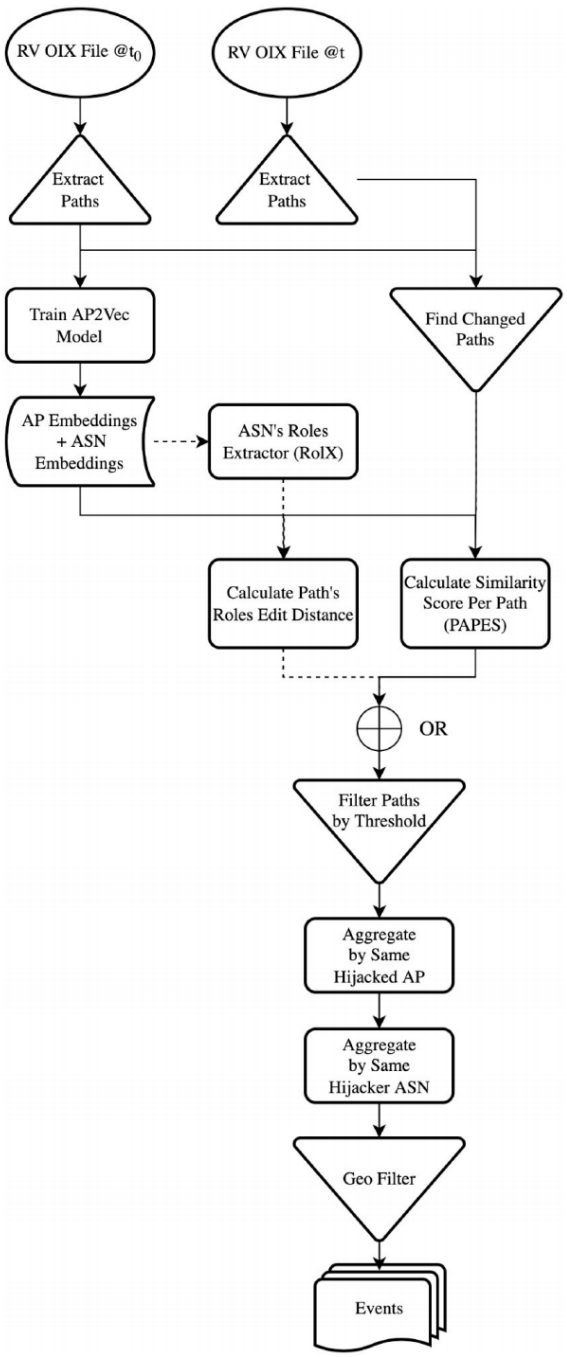

Fig. 3: A block diagram of our BGP hijacking detection system. The dashed arrows depict an alternative(with slightly higher FA rate) method.

2) **Path's roles edit-distance (Roles-ED)**: we replace each ASN in a path p' with its corresponding role and remove consecutive duplicates. We apply the same for the preceding path p, compute the edit-distance between the two role-paths p'_r, p_r using the Levenshtein distance [39] ($\mathcal{LD}$), and choose a minimal threshold th_{ld} which determines if the structural change of the path is significant. In order to generate the functional role of each ASN, we apply the unsupervised method for role extraction RolX described in [40] with a slight modification. Instead of using the ReFeX algorithm [41] for recursive structural feature extraction, we use the ASN-vectors as a set of d structural features; one feature for each dimension of the embedding. In Section VIII we compare between the two methods and show that the usage of ASN-vectors as features outperforms the regular features. Using the Minimum Description Length criterion [42], RolX automatically determines the number of roles that results in best compression. In all of our experiments, over the years 2008-2020, we obtain the same number of roles (in both methods) - 8.

As we present in Section VIII-A, we choose both thresholds based on the mean and standard deviations of the distribution of each metric.

In the second phase, we generate a list of candidate hijacked APs. These are APs that all their corresponding changed paths (denoted by $C_{u'}$) are due to a structural change. We denote by $L_{u'}$ the set of routes with a structural change from the previous phase and keep only the APs where $L_{u'} = C_{u'}$. Namely, we require that all routing changes for this AP will have a structural change. This requirement may be relaxed to $|L_{u'}| \geq \alpha |C_{u'}|$, for some α close to 1. Then, for each AP in the remaining list, we look for a newly seen ASN that appears in all the paths in $L_{u'}$ (lines 21-22 in Algorithm 3). If such an ASN does not exist, we remove the AP from the candidate list. This phase's outcome is a list of ASN hijackers L_{ASN} and their corresponding hijacked APs L_{AP}. The rationale for this phase is that a successful hijack attack should be observed from several vantage points, and no other routing change should occur concurrently.

At the last stage, we look for ASNs that appear as suspected attackers for several APs. We unite all these APs to a single hijack event.

Algorithm 3 HijackDetection $(\mathcal{P}_{t-1}, \mathcal{P}_t, \mathcal{T}_{t-1}, \mathcal{T}_t, w, d, \alpha, th_{cs}, th_{ld}, th_{c_u}, th_{c_v})$

Require: $\mathcal{P}_{t-1}, \mathcal{P}_t$: AS path lists at times $t-1, t$; $\mathcal{T}_{t-1}, \mathcal{T}_t$: corresponding origin APs; w: window size; d: embedding size; α: learning rate; thresholds $th_{cs}, th_{ld}, th_{c_u}, th_{c_v}$

Ensure: L_{ASN}: list of suspicious ASNs, L_{AP}: corresponding potential victim APs

1: Generate vocabulary $\mathcal{V}$ from $\mathcal{P}_{t-1}$
2: Generate vocabulary $\mathcal{U}$ from $\mathcal{T}_{t-1}$
3: $\Phi, \Psi \leftarrow$ AP2Vec$(\mathcal{P}_{t-1}, \mathcal{T}_{t-1}, w, d, \alpha)$
4: Generate parent APs $\mathcal{S} : u' \in \mathcal{T}_t \mapsto u \in \mathcal{U}$
5: **for** each $p' \in \mathcal{P}_t$ & $u' \in \mathcal{T}_t$ **do**
6: Get corresponding $p \in \mathcal{P}_{t-1}$ & $u \in \mathcal{U}$
7: **if** $p' \neq p$ **then**
8: $C_{u'} \leftarrow p$
9: Generate $\Psi_{p'}$ for p'
10: **if** sim$(\Psi_u, \Psi_{p'}) < th_{cs}$ **then**
11: $L_{u'} \leftarrow p$
12: **end if**
13: **end if**
14: **end for**
15: **for** each $u' \in \mathcal{T}_t$ **do**
16: **if** $L_{u'} == C_{u'}$ **then**
17: **for** each $v \in$ set$(L_{u'})$ **do**
18: **if** $L_{u',v} == L_{u'}$ **then**
19: $L_{AP} \leftarrow u'$
20: $L_{ASN} \leftarrow v$
21: **end if**
22: **end for**
23: **end if**
24: **end for**
25: **return** L_{AP}, L_{ASN}

VII. EXPLORATION OF AP EMBEDDINGS

We base AP2Vec on BGP2Vec [13], [43] where ASN embedding using BGP announcements was thoroughly explored. They showed that the ASN representations exhibit linear structure, and specifically, an ASN could be characterized by its closest ASNs. We claim here that these results also apply to the AP representations and show in this section qualitative examples (Evaluation of the embedding for hijack detection will be done in Section VIII). We found high cosine similarity between APs, which share the same function: universities, organizations, IXPs, content providers, geographical locations, and ASNs. We give two examples to illustrate the latent characteristics of the obtained AP vectors.

We define similarity between two AP2Vec vectors according to Equation (7) and find the nearest neighbor of AP2Vec vector:

$$\text{NearestNeighbor}(u_i) = \arg\max_{u_j \in \mathcal{U}} \text{sim}(\Psi_{u_i}, \Psi_{u_j}). \quad (9)$$

The same is also applied for ASN vectors. We calculate the average cosine similarity correlation of a specific group of APs or ASNs by taking the average of all the cosine similarities of the pairs in this group.

We use to collect the list of Akamai and DigitalOcean APs and ASNs. For Akamai, we select three APs; '184.24.0.0/13', '2.16.0.0/13', and '23.0.0.0/12' to a short-list termed 'Akamai Short,' and term the rest of the APs as 'Akamai Long' list. We then find the 10 nearest neighbors for each of the three APs in 'Akamai Short.' Figure 5 presents a 2-dimensional UMAP projection of the obtained vectors. As presented, among the nearest neighbors, we can find 10 additional APs which belong to 'Akamai Long,' and there is only one AP among the nearest neighbors which do not belong to Akamai. Furthermore, the two ASNs which own the three Akamai APs are located close to Akamai's APs. The average cosine similarity correlation of Akamai APs is 0.595, whereas the average cosine similarity correlation of a random sample of 1000 APs is 0.317, which

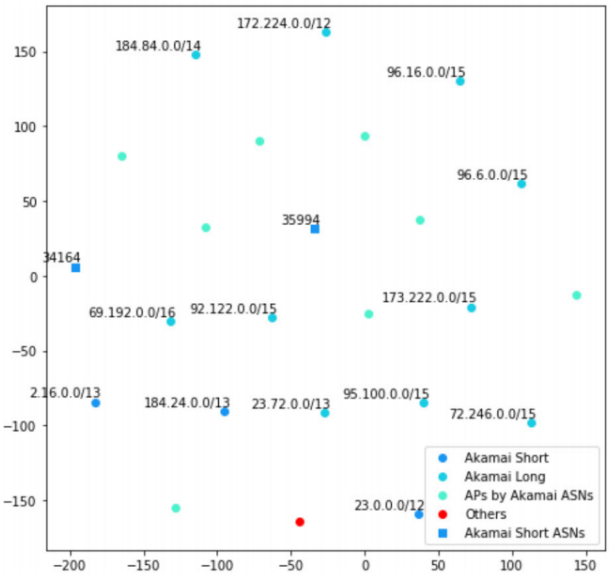

Fig. 4: Two-dimensional UMAP projection of 32-dimensional vectors of DigitalOcean APs and their nearest neighbours.

means the representations capture the closeness of Akamai APs.

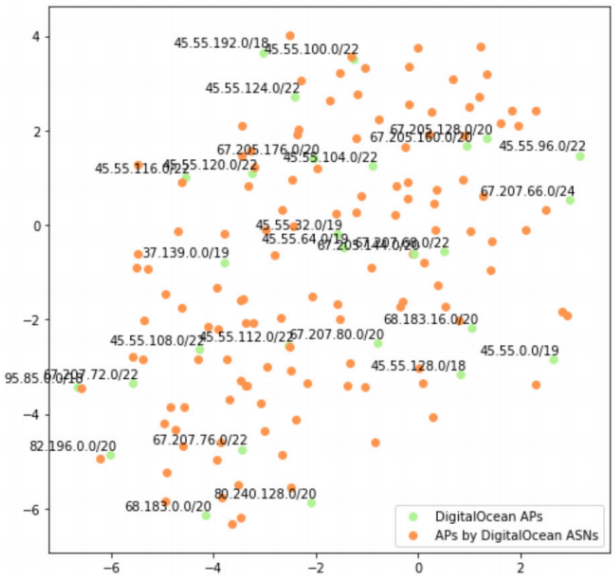

Fig. 5: Two-dimensional UMAP projection of 32-dimensional vectors of DigitalOcean APs and their nearest neighbours.

For DigitalOcean, we find the 10 nearest neighbors for an arbitrary set of 28 of its APs. Figure 5 depicts a 2-dimensional UMAP [44] projection of the obtained vectors. As presented, all the nearest neighbors are owned by DigitalOcean ASNs. This result is consistent with the high cosine similarity correlation obtained for DigitalOcean APs, which is equal to 0.96.

Figure 6 depicts a UMAP projection of both Akamai and DigitalOcean APs. There is a clear separation between the APs of these ASNs, which can be explained by the low average similarity between them, only 0.18; much lower than 0.312, which is the average similarity between random ASNs. The low similarity between these two ASNs is surprising since both are global CDNs.

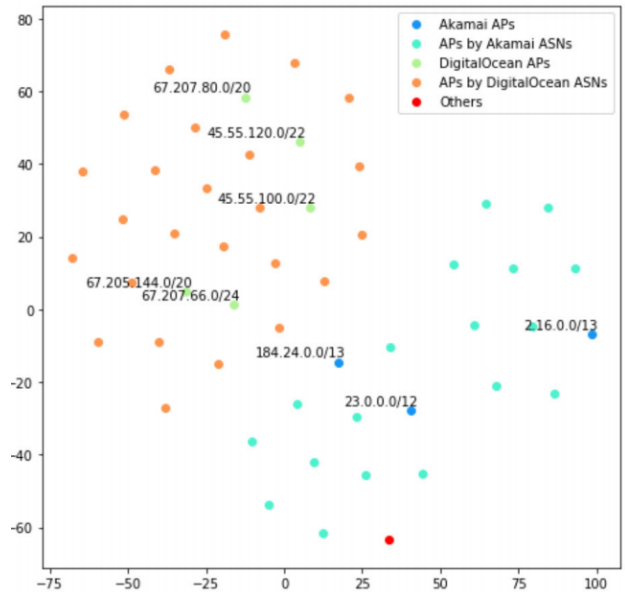

Fig. 6: Two-dimensional UMAP projection of 32-dimensional vectors of DigitalOcean APs and their nearest neighbours.

VIII. EXPERIMENTS AND RESULTS

In this section, we report our experimental results for BGP hijacking detection. As mentioned in Section III, we evaluate our two method's ability to detect real events by testing it using a list of ground truth past events. To evaluate the false alarm rate, we test our methods using several arbitrary RV records from October-November 2020.

To show the advantage of using deep learning for this task, we compare our two suggested methods (see Section VI-B) with recently proposed path-based metrics, which are entirely based on BGP paths without any side information. First, as mentioned in Section VI-B, for role extraction, we compare the usage of ASN-vectors as features and the usage of the default recursive structural features (ReFeX [41]) as used in RolX. Second, we propose two other methods to detect a structural change in a path, based on the global AS hegemony [26] and local AS hegemony that were presented in [4]. The global AS hegemony is computed based on paths to all APs reported by the vantage points, while the local AS hegemony is computed based on paths towards only one origin AS. We calculate AS hegemony based entirely on RV records and do not use any external source. Therefore we compare a total of 5 metrics for the detection of structural changes:

1) Path and AP Embedding Similarity (PAPES) - as defined in Section VI-B. 2) Path's roles edit-distance using the ASN-vectors as features (Roles-ED) - as defined in Section VI-B. 3) Path's roles edit-distance using ReFeX [41] features (F-Roles-ED). 4) AS local hegemony similarity (Local Hegemony Sim) - as described in [4]. We select the top three ASes in terms of local hegemony scores from the preceding paths and current

TABLE I: Summary of Metric Statistics Based on 4 Chosen Timestamps. Metrics Are Calculated Only for APs With Changed Paths, and Only for the Paths That Were Changed. We Present the Mean and Standard-Deviation ($\mu \pm \sigma$) Across All APs

Record Time	#APs	#ASNs	#APs w/ Changed Paths	PAPES	Roles-ED	F-Roles-ED	Local Hegemony Sim	Valley Diff
2020-10-18 16:00	883,337	70,334	10,208	0.46 ± 0.25	0.82 ± 0.77	0.57 ± 0.76	0.82 ± 0.17	-0.02 ± 0.24
2018-04-24 12:00	753,146	61,206	15,605	0.51 ± 0.26	0.56 ± 0.70	0.17 ± 0.46	0.87 ± 0.12	0.13 ± 0.24
2014-11-15 00:00	542,384	48,881	11,404	0.51 ± 0.25	0.96 ± 0.49	0.72 ± 0.68	0.90 ± 0.09	-0.02 ± 0.23
2008-02-24 20:00	246,271	27,478	16,043	0.25 ± 0.26	1.20 ± 0.63	0.93 ± 0.81	0.91 ± 0.09	0.00 ± 0.15

TABLE II: SUMMARY OF THE NUMBER OF FLAGGED CANDIDATES BASED ON DIFFERENCE METRICS FOR THE 4 CHOSEN TIMESTAMPS. HERE, WE DID NOT REMOVE EVENTS WITH ASNS WHICH SIGNIFICANTLY APPEARED IN THE PREVIOUS TIME-SLOT ANALYSIS

Record Time	PAPES			Roles-ED			F-Roles-ED			Local Hegemony Sim			Valley Diff		
	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs	Events	APs	ASNs
2020-10-18 16:00	10	48	16	41	163	67	42	101	65	20	54	33	4	8	4
2018-04-24 12:00	16	73	21	60	223	88	39	134	55	40	174	60	6	50	6
2014-11-15 00:00	5	14	6	23	52	31	30	71	42	26	62	35	2	2	2
2008-02-24 20:00	2	10	5	46	121	70	31	93	43	40	74	61	6	10	7

paths and calculate the cosine similarity between them. Note that, unlike the other metrics that are computed on a single path, this metric is calculated over the entire list of changed paths of the origin AP. 5) AS global hegemony valley depths difference (Valley Diff) - we extend the method described in [4] for calculating the valley depth for a path based on global hegemony and subtract the depth of the preceding path from the depth of the current path. Since Cho et al. [4] mentioned several limitations of using AS hegemony for BGP hijacking classification, we choose to use the difference between two consecutive valleys and not the absolute value to minimize these limitations.

A. Parameters Exploration

As mentioned in Section VI-A, we choose the hyper-parameters for the AP2Vec model based on the hyper-parameters optimization conducted in [13].

For the hijack detection stage, we explore our chosen metrics based on 4 experiments with the following timestamps: 2020-10-18 16:00, 2018-04-24 12:00, 2014-11-15 00:00, and 2008-02-24 20:00. Table **??** presents statistics for the five metrics. Notice that for each AP, we calculate the average metric only across its changed paths. We use these statistics to define the thresholds mentioned in Section VI-B.

Then, for each similarity metric (see Table **??**), we choose an upper threshold of $\mu - \sigma$, and for each distance metric, we choose a lower threshold of $\mu + \sigma$ (in the case of perfect normal distribution, this leaves us with 15.86% of the results). Since the Edit-Distance is an integer, we round down the threshold (e.g., for Roles-ED, we round down to 1, meaning that we consider a structural change if the edit distance is at least 1). Additional analysis of the statistics of the metrics appears in the Appendix.

Based on the chosen thresholds for each metric (excluding local hegemony similarity) and for each AP, we count the number of structural-changed paths. For all the metrics we use, more than 80% of the APs with path changes have less than 5 paths with structural roles changes. The presented distributions are similar to other time slots.

Since we are interested in hijack attacks, there should be a cause for the route change. Thus, we require that in all the changed paths, there will be a role change and that the suspected (namely newly seen) ASN will be the same. In practice, this requirement may be relaxed and allow a few outliers. We identify all the APs with the same suspected ASN as a single event.

Table II summarizes the result statistics for the experiments we conducted for several time slots (same as Table **??**). We choose times at different parts of the day and from different years. The most impressive aspect of the table is that all metrics manage to identify up to a few tens of suspected events from an original batch of over 10,000 APs with route change. PAPES and Valley Diff stand out with 2-16 events for each experiment. This is already a reasonable number to be handled by a response team.

To prevent repeated flagging of the same event, we filter out route changes that are caused by ASNs that were already flagged in the previous time slot. In continuous deployment, these APs have already been examined in the preceding analysis. We also remove events based on geo-filtering, where specific countries appear as both the hijack victims and attackers: Brazil, Iran, and Bangladesh. In these countries, ASes do not seem to obey the common BGP concatenation rules (aka valley-free routing [10]) and have complex relationships that our embedding does not manage to learn. As a result, our algorithm flags many local hijack attacks on ASes in these countries. We use [45] to obtain the owner-country of each ASN.

In the following sections, we will compare the detection results for the different metrics. If only a small number of paths were changed, it means that the reflection/hijack did not succeed in propagating through the Internet. Thus, we consider only APs with a minimum of 5 changed routes. We validated that all the ground-truth events are above this threshold.

REFERENCES

[1] P. Sermezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, "A survey among network operators on BGP prefix hijacking," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 1, pp. 64–69, Jan 2018.

[2] 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.

[3] Y. Gilad, T. Hlavacek, A. Herzberg, M. Schapira, and H. Shulman, "Perfect is the enemy of good: Setting realistic goals for BGP security," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, 2018, pp. 57–63.

[4] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "BGP hijacking classification," in *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA)*, Jun 2019, pp. 1–8.

[5] G. Huston and R. Bush, "Securing BGP and SIDR," IETF Journal, Tech. Rep. 1, 2011.

[6] M. Lepinski and K. Sriram, "BGPSEC protocol specification," Internet Engineering Task Force, RFC 8205, 2017.

[7] Y. Gilad *et al.*, "Are we there yet? on rpki's deployment and security," *Proc. NDSS*, pp. 1–15, 2017.

[8] T. Chung *et al.*, "RPKI is coming of age: A longitudinal study of RPKI deployment and invalid route origins," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 406–419.

[9] P. Sermezis *et al.*, "ARTEMIS: Neutralizing BGP hijacking within a minute," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2471–2486, Dec 2018.

[10] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec 2001.

[11] K. Balu, M. L. Pardal, and M. Correia, "DARSHANA: Detecting route hijacking for communication confidentiality," in *Proceedings of the IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 52–59.

[12] P. Moriano, R. Hill, and L. J. Camp, "Using bursty announcements for detecting BGP routing anomalies," *Computer Networks*, vol. 188, p. 107835, Apr 2021.

[13] T. Shapira and Y. Shavitt, "A deep learning approach for IP hijack detection based on ASN embedding," in *Proceedings of the Workshop on Network Meets AI & ML*, 2020, pp. 35–41.

[14] 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*, 2013, pp. 3111–3119.

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 1188–1196.

[16] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and whisper: Security mechanisms for BGP," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[17] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: Improving BGP by cautiously adopting routes," in *Proceedings of the International Conference on Network Protocols (ICNP)*, 2006, pp. 290–299.

[18] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2007, pp. 3–17.

[19] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "ISPY: Detecting IP prefix hijacking on my own," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 327–338, 2008.

[20] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the Internet with argus," in *Proceedings of the Internet Measurement Conference*, 2012, pp. 15–28.

[21] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. W. Biersack, "HEAP: Reliable assessment of BGP hijacking attacks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1849–1861, Jun 2016.

[22] M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, "MS-LSTM: A multi-scale LSTM model for BGP anomaly detection," in *Proceedings of the IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, pp. 1–6.

[23] Q. Ding, Z. Li, P. Batta, and L. Trajkovic, "Detecting BGP anomalies using machine learning techniques," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 3352–3355.

[24] H. Alshamrani and B. Ghita, "IP prefix hijack detection using BGP connectivity monitoring," in *Proceedings of the IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, 2016, pp. 35–41.

[25] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling BGP serial hijackers: Capturing persistent misbehavior in the global routing table," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2019, pp. 420–434.

[26] R. Fontugne, A. Shah, and E. Aben, "The (thin) bridges of AS connectivity: Measuring dependency using as hegemony," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, R. Beverly, G. Smardagdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 216–227.

[27] University of Oregon Advanced Network Technology Center, "Route Views Project," http://www.routeviews.org/, accessed: Nov. 18, 2020.

[28] Dyn, "Dyn Blog," https://hub.dyn.com/dyn-research, 2020.

[29] Ripe NCC, "Ripe NCC Publications," https://www.ripe.net/publications/, 2020.

[30] BGPMon, "BGPMon Blog," https://www.bgpmon.net/blog/, 2020.

[31] APNIC, "Apnic Blog," https://blog.apnic.net/, 2020.

[32] Oracle, "Oracle Internet Intelligence," https://blogs.oracle.com/internetintelligence/, 2020.

[33] MANRS, "Manrs Blog," https://www.manrs.org/news/, 2020.

[34] BGProtect, "BGProtect Case Studies," https://www.bgprotect.com/case-studies, 2020.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[37] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC Workshop on New Challenges for NLP Frameworks*, May 2010, pp. 45–50.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[39] H. Hyyrö, "Explaining and extending the bit-parallel approximate string matching algorithm of Myers," Dept. of Computer and Information Sciences, University of Tampere, Tampere, Finland, Tech. Rep. A-2001-10, 2001.

[40] K. Henderson *et al.*, "RolX: Structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1231–1239.

[41] ——, "It's who you know: Graph mining using recursive structural features," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 663–671.

[42] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[43] T. Shapira and Y. Shavitt, "Unveiling the type of relationship between autonomous systems using deep learning," in *Proceedings of the IEEE/IFIP NOMS International Workshop on Analytics for Network and Service Management (AnNet)*, Apr 2020, pp. 1–6.

[44] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[45] CAIDA, "CAIDA AS Rank," http://as-rank.caida.org/, 2020, accessed: Oct. 2020.