A Deep Learning Approach for IP Hijack Detection Based on ASN Embedding

Tal Shapira
talshapira1@mail.tau.ac.il
School of Electrical Engineering, Tel-Aviv University

Yuval Shavitt
shavitt@eng.tau.ac.il
School of Electrical Engineering, Tel-Aviv University

ABSTRACT

IP hijack detection is an important security challenge. In this paper we introduce a novel approach for BGP hijack detection using deep learning. Similar to natural language processing (NLP) models, we show that by using BGP route announcements as sentences, we can embed each AS number (ASN) to a vector that represents its latent characteristics. In order to solve this supervised learning problem, we use these vectors as an input to a recurrent neural network and achieve an excellent result: an accuracy of 99.99% for BGP hijack detection with 0.00% false alarm. We test our method on 48 past hijack events between the years 2008 and 2018 and detect 32 of them, and in particular, all the events within two years from our training data.

CCS CONCEPTS

• **Networks** → *Routing protocols*; **Network security**; • **Security and privacy** → **Security protocols**; • **Computing methodologies** → **Knowledge representation and reasoning**; *Supervised learning by classification*; **Neural networks**.

KEYWORDS

Deep Learning, BGP, BGP hijacking, AS embedding, routing, security

ACM Reference Format:
Tal Shapira and Yuval Shavitt. 2020. A Deep Learning Approach for IP Hijack Detection Based on ASN Embedding. In *Workshop on Network Meets AI & ML (NetAI'20), August 14, 2020, Virtual Event, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3405671.3405814

1 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. 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 prefix and carries one or more BGP attributes. BGP allows each AS to choose its own policy on selecting the best routes, accepting routes (according to its import policy), and announcing routes (according to its export policy). An AS routing policy is usually determined by the commercial contractual relationship with other administrative domains.

The BGP connection between two autonomous systems usually called an AS link and is determined by the contractual commercial agreements between two connected ASes, also called Type of Relationships (ToR). ASes ToR are broadly classified into three symmetric types: (1) Peer-to-peer (P2P) - two ASes freely exchange traffic between themselves and their customers, but do not exchange traffic from or to their providers or other peers, (2) Provider-to-customer (P2C) - the customer AS pays the provider AS for transit traffic from and to the rest of the Internet. In most cases, the customer AS does not forward traffic between its providers, and (3) Sibling-to-sibling (S2S) - two ASes, that usually belongs to the same administrative domain, freely transit traffic for each other.

In recent years, there have been many reports of BGP Prefix hijacking [9, 27], e.g., more than 40% of the network operators reported that their organization had been a victim of a hijack in the past [27]. BGP Prefix hijack attacks have become a commonly employed technique by hostile governments and criminal organizations. This attack allows the attacker to eavesdrop, record and manipulate Internet traffic, and also to implement various man-in-the-middle attacks against the victim network, even when strong encryption is used. The attack is made by a BGP speaker that advertises illegitimate routes for IP prefixes or sub-prefixes that are own by the victim AS. Note that BGP prefix hijacking, can also be caused by router misconfiguration.

Several reactive mechanisms or modifications to BGP have been proposed to deal with the attack (such as RPKI [15] and BGPsec [20]), however most operators have not deployed them and are reluctant to deploy them due to technical and financial costs. Defending against hijacking consists of detection and mitigation. Current solutions for BGP hijacking detection are based on BGP routing databases, detect only simple attacks, and suffer from large delayed response time, and lack of accuracy (prone to false positives).

In this paper, we introduce a novel approach for BGP hijack detection, using deep learning methods. Our datasets consist of BGP path announcements obtained from RouteViews [23] and labeled by a VF algorithm (i.e., standard routes and hijacked routes) that is based on Shavitt *et al.* [33] with manual corrections.

Over the past few years, advances in deep learning [19] have driven tremendous progress in many fields; one of them is Natural Language Processing (NLP). In our recent work [29], we built on the excellent results achieved for NLP tasks (Word2Vec [21]) and introduced an extension of word vectors for creating a dense vector

NetAI'20, August 14, 2020, Virtual Event, NY, USA

ACM ISBN 978-1-4503-8043-0/20/08…$15.00
https://doi.org/10.1145/3405671.3405814

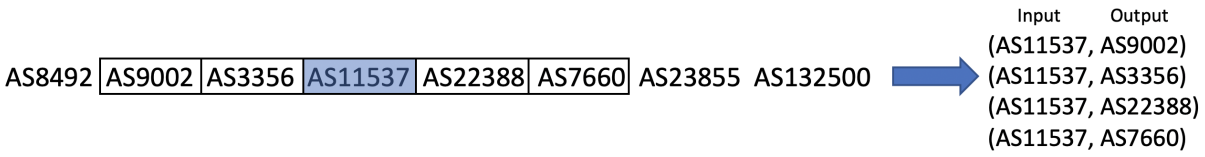

Figure 1: An example for generating input/output ASNs for the training process of BGP2VEC.

representation of ASNs (we called it *BGP2VEC*). This method is trained on unlabeled BGP routes dataset in order to embed each ASN to a dense vector based on hierarchical contexts of ASNs.

Using the *BGP2VEC* embedding, we apply a recurrent neural network to classify BGP routes as standard or hijacked routes. Our approach achieved excellent results. We detect BGP hijacking with an accuracy of 99.99% and a False Alarm of 0.00% on our dataset. Furthermore, as we examined our misclassified predictions, we found that more than 50% of our misclassified predictions were wrong, i.e., we found errors in the labeled dataset. Additionally, we tested our algorithm on 48 past documented hijack events [6] between 2008-2018, and classified correctly all the events within 2 years of our training data, or 2/3 of all the events. Finally, as far as we know, we are the first to apply an end-to-end deep learning approach to detect IP hijack based on only BGP routes.

The rest of the paper continues as follows. After describing related work in Sec. 2, we describe the dataset in Sec. 3. In Sec. 4 we describe our method, and in sec. 5 we explore the latent characteristics of ASN embedding. Sec. 6 presents our experiments and their results, and Sec. 7 presents several interesting examples. Finally, the last section concludes the paper.

2 RELATED WORK

Approaches for defending against BGP hijacking can be sorted into two categories: prevention and detection. BGP prefix hijacking prevention solutions, also sometimes called reactive solutions, are based on cryptographic authentications such as RPKI and BGPsec [15, 16, 20, 31]. There are many different approaches for the detection of BGP hijacking. We divide these approaches into three main categories, based on the type of information they use: (1) Control-plane approaches [18, 24, 28] - also called passive solutions, these methods analyzed BGP routing information from a distributed set of BGP monitors and route collectors to detect anomalous behavior, (2) Data-plane approaches [9, 34, 35] - only relies 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 (3) Hybrid approaches [14, 26, 30] - these approaches use both control-plane and data-plane information and sometimes even use external databases to perform joint analysis.

Several recent works [1, 5, 6, 10, 32] examined machine learning techniques with manually generated features to identify malicious origin ASes; mainly leveraging historical BGP data from RouteViews [23] and RIPE. Testart *et al.* [32] 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 and 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 found about 900 ASes with similar behavior.

Cho *et al.* [6] classified detected hijack events in order to understand the nature of reported events. They introduced four categories of BGP hijack: 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 (a metric that quantifies the likelihood of an AS to lie on paths toward certain destination IP prefixes), and use Random Forest classifier, which achieved 95.71% accuracy over their dataset.

As far as we know, there is no prior work in the field that involves the use of end-to-end (namely, without manually generated features) **deep learning approach and by using only BGP routes data.** As we will show, without any features engineering, our deep learning method produced excellent results.

3 THE DATASETS

In our experiments, we use data that was collected in March 2018. We use two types of datasets: (1) RouteViews's BGP paths (RV) [23] - contains BGP paths 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, and (2) Labeled BGP routes - consists of approximately 2,648,900 standard routes (labeled 'GREEN') and 47,800 hijacked routes (labeled 'RED'). The remaining roughly 900,000 routes which had missing ToRs were classified as 'undecided' and were only used in the unsupervised embedding. The labeling was generated by combinations of VF algorithms [33] and manual work. Manual auditing of routes was done by sampling GREEN routes at a low probability and RED routes at a high probability. Manual auditing could result in identifying wrongly inferred ToRs (and re-evaluating all the routes with corrected ToRs), or by manually changing the label of routes whose VF labeling is incorrect. Thus, 'RED' labeled routes are only proxy for hijacking and misconfigurations, although our large scale manual examination, removed many valley free violations that were benign.

Furthermore, in order to validate our results, we use the *BGP data for ground truth events* dataset that was introduced by Cho *et al.* [6]. This dataset contains 70 events from February 2008 to July 2018; 35 events from Dyn [11] and 35 events from BGPMon [2]. The historical BGP data was collected using CAIDA's BGPStream [3]. The dataset contains 21 typo events, 15 prepending mistakes, 16 origin change events (MOAS), and 18 forged AS paths events with an average number of 669 AS paths per event. **As we will show later, we don't aim to classify these events into 4 classes, but to actually detect these hijack events.**

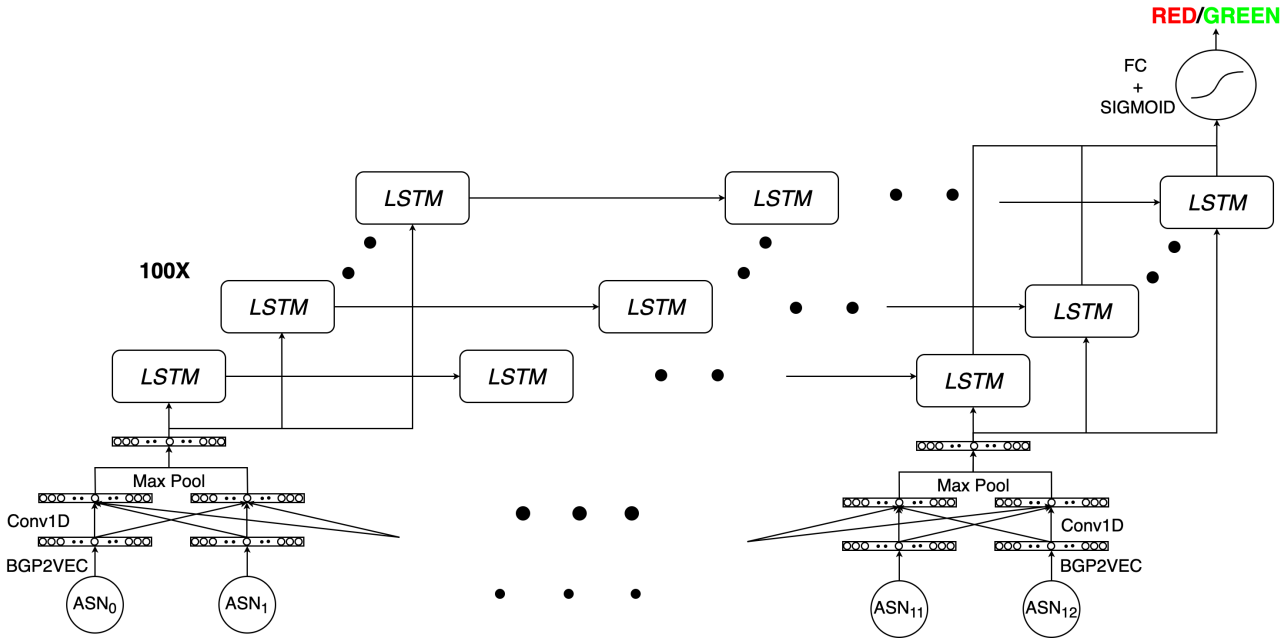

Figure 2: Our LSTM architecture.

4 METHOD

Our method works as follows: at the first stage, using a shallow neural network (BGP2VEC [29]), we map each ASN to a 32-dimensional representative vector. Then, for the BGP hijacking detection task, we activate an LSTM network that receives the vectors from the previous stage. In this section, we will introduce in detail both stages.

4.1 ASN Embedding

As mentioned in [29], in the first stage, we produce a 32-dimensional continuous vector representation for each ASN. We apply a similar skip-gram model as introduced in [21], which contains an input layer of size 62,525 (which is the number of distinct ASes in our dataset), one FC hidden layer with a size of 32, and an output layer whose size is determined by the window size. We choose to apply a window of size 2 (see Figure 1), which is the maximum distance between the input ASN and a predicted ASN (the output) within an AS path, which results with an output layer with a maximum size equals to 4x62,525. In order to improve the representation, we use negative sampling [21] to distinguish the target ASN from the noise distribution using 5 negative samples for each target ASN. We build and run our network using the *Gensim* [25] library.

4.2 LSTM Architecture

At the second stage, as can be seen in Figure 2, we activate an LSTM network that receives the vectors produced in the previous stage and classifies BGP routes. As depicted in Table 2, our LSTM architecture comprises five 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 (labeled as ConV1D) with 32 filters of length 3. *A convolutional layer* [8] produces layers that are called features map. Each unit in a feature map is connected to a local region of the previous layer through a set of weights called filter, such that all units in a feature map share the same filter. The outputs of our ConV1D are 32 feature maps of size 13. ConV1D contains a total number of 3,104 trainable parameters (3072 weights and 32 bias parameters). The result of the convolution in each unit is passed through the ReLU activation function [22]. 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 the LSTM layer with a default configuration [13], which produces a 100-size output vector. The layer consists of 100 LSTMs cells, which contains a total number of 5,320 trainable parameters. Finally, our output layer is a single neuron with a Sigmoid activation function, which produces a value between 0 and 1, and contains 101 trainable parameters.

4.3 Training Specifications

The training of the LSTM networks was done by optimizing the *categorical cross entropy* [4] 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 true label of the sample. For the optimization process we use the *Adam* [17] gradient-based optimizer. The Adam optimization algorithm is an extension to the stochastic gradient descent algorithm, which achieves better results than other optimization algorithms. We used the default hyper-parameters as provided in Kingma *et al.* [17] and set our batch size to 64.

Table 1: Exploration of the 5 nearest neighbours for AS3356 (Level3) and AS15169 (Google).

Neighbor	ASN	Owner	Cosine Similarity	Degree	hops from Tier-1	AS Class	Country	ToR	Similarity 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
-	**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 2: The architecture of our LSTM network.

BGP Routes Classification	
Embeddings:	Input: 13, 1 Output: 13, 32
Conv1D:	Output: 13, 32
MaxPool:	Output: 6, 32
LSTM:	Output: 100
FC + Sigmoid:	Output: 1

We build and run our networks using the *Keras* [7] library with *Tensorflow* [12] as its back-end. We use 80% of the samples as a training set and 20% of the samples as a test set. We run our network for 10 epochs of the training set for the BGP routes classification. We save the result, which achieves the best accuracy during the training process. During the test time, our network classifies an AS sequence of ASNs in an average time of 0.1mSec on a single Intel CPU.

5 EXPLORATION OF ASN EMBEDDINGS

By exploring the generated vectors, we show that the representations exhibit linear structure, and specifically, we can characterize an ASN by its closest ASNs. Table 1 displays the 5 nearest neighbours for 2 chosen ASes. For each explored AS, we find its 5 nearest neighbors using cosine similarity, and for each 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. In addition, for each neighbor, we assigned a grade 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 similarity of 0.897. The next 3 nearest neighbors are other tier-1 providers, while the 5th nearest neighbor does not seem to be related to AS3356 and thus counted as zero; therefore, the total similarity grade is 14.

For the second 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 many latent characteristics of the ASNs.

6 EXPERIMENTS AND RESULTS

In this section, we report our experimental results for BGP hijack detection. Due to the lack of standard datasets, we have not found any previous work for comparing our BGP routes classification results. However, as we will show, our method succeeds in finding many mistakes in the dataset, which emphasizes the strength of our results.

6.1 Evaluation Criteria

We use the **accuracy** criteria 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 binary classification is

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP, TN, FP and FN are the true positive, true negative, false positive, and false negative, respectively. In our case the positive class corresponds with 'RED' routes.

For visualizing the results, we use the normalized **confusion matrix** (Figure 4). In a confusion matrix, each row represents the actual class, while each column represents the predicted class. In a normalized confusion matrix, the diagonal values represent the true positive rate (TPR) and the true negative rate (TNR) correspondingly.

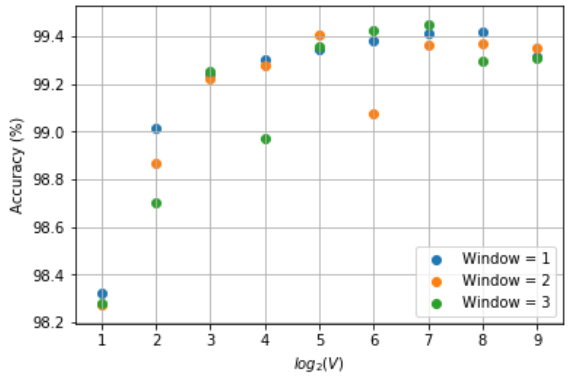

Figure 3: Grid search results for BGP routes classification accuracy as a function of the embedding size (V) and the window size of the BGP2VEC.

6.2 Hyperparameter Optimization

Figure 3 shows the results of a grid search over different ASN embedding sizes (V) and window sizes, in order to optimize these parameters to achieve the best accuracy for the classification problem. We performed a grid search over window sizes 1, 2, and 3; and multiples of 2 embedding sizes in a range between 2 and 512. In each experiment, we used the same architecture as depicted in Table 2, with only one difference in the output of the embedding layer.

Figure 3 shows that we achieve the best accuracy results for BGP routes classification with an embedding size of 128 and a window size of 3. However, there is only a slight difference in window sizes in the range [32,256]. Thus, for the rest of the paper, we select an embedding size of 32 and a window of 2. Note that by always classifying as 'Green' we get 98.23% accuracy due to the unbalanced dataset.

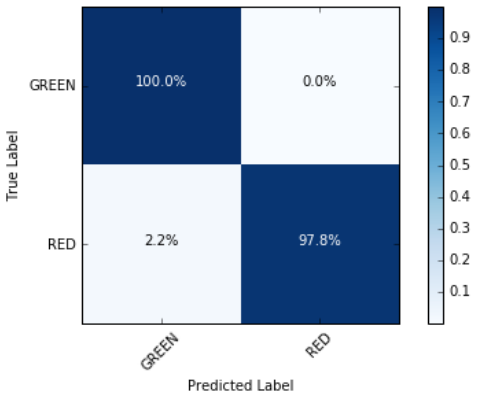

Figure 4: A confusion matrix of the BGP routes classification problem.

6.3 Results on BGP Routes Classification

As mentioned above, after training our LSTM network over the training set, we evaluate our method over a test set consisting of 20% of the dataset, and achieve an accuracy of **99.98%**. Figure 4 shows that our method correctly classifies 97.8% of the hijacked routes (RED) and almost completely succeeds to classify legitimate routes (GREEN) with a false alarm (FP) of **0.00%**, that is much lower than the 1.77% of hijacked routes in our dataset. By manually exploring our 211 misclassified test results, we found that 152 of them (or **72.0%**) were indeed classified correctly, which increases the mentioned accuracy to **99.99%**. Moreover, our method was able to find BGP hijacking that has not been found using other methods, as described in Sec. 7 below.

6.4 Results on Ground Truth Events

For each event in the *BGP data for ground truth events* we apply our pre-trained LSTM network (that was trained over the labeled BGP routes dataset from March 2018) over all the related hijacked BGP routes. Unlike [6], we do not aim to classify these events into 4 classes but to detect these hijack events. A summary of our results is displayed in Table 3. Among these 70 events, in 22 cases, the hijack AS was not found in our training set, these are ASes that were present in the routing system only for a short term, such as the malicious Bitcanal AS (AS197426), or AS57807 that almost always announced using its sibling AS58321; therefore, we can not apply our LSTM network for these cases. Overall, among the remaining 48 events (termed as valid events), our method succeeds in detecting 32 events (66.7%) with a high mean average prediction score (MAPS) of 0.985. Among the detected event, we calculated the prevalence of paths that were identified as hijacked by our method, out of all paths associated with that event, and achieve an average prevalence (denoted as 'Red Prevalence') of 47.1%.

As Table 3 shows, our method deals well with prepending mistakes and forged AS paths while achieves less accurate results for typos and origin changes. It worth mentioning that **although our method was trained with a dataset from March 2018, it succeeds in detecting hijacked routes from past events** (2008-2018). Furthermore, our method succeeds in detecting all recent hijack events (2016-2018).

7 EXAMPLES

7.1 Violation of VF

The BGP route: [23673, 1299, 3267, 31500, 57363] was labeled as 'RED' in our dataset because it violates the "VF Routing," due to the fact that GlobalNet (AS31500) and RUNNet (AS3267) are classified as peers, while Telia (AS1299) is a provider of RUNNet (AS3267). The relations between RUNNet and GlobalNet do not seem to follow the VF model: RUNNet is an academic network, but it seems to provide some transient service to GlobalNet. Our deep learning method managed to capture this complex relationship and classified the path correctly as 'GREEN'.

7.2 Iranian Hijacker

As described in Sec. 5, by exploring the generated vectors of the BGP2VEC, we can characterize an ASN by its most similar ASNs

Table 3: Results over the *BGP data for ground truth events* dataset.

Category	Prepending	Typo	MOAS	Forged Path	All
Total Events	21	15	17	17	70
Valid Events	21	14	9	4	48
Detected Events	**18**	**8**	**3**	**3**	**32**
Success	**85.7%**	**57.1%**	**33.3%**	**75.0%**	**66.7%**
MAPS	0.986	0.985	0.982	0.988	0.985
Red Prevalence	50.1%	46.3%	27.5%	46.3%	47.1%

(in term of cosine similarity). For example, we examined AS41881, which is assigned to Fanava Group and involved in many hijacked routes (~200). We found that most of its nearest neighbors are also involved in many hijacked routes, such as AS202788, which is also an Iranian AS, and all the routes that it is part of are hijacked (~40). Furthermore, the nearest neighbor of AS41881 is AS1756, which in our dataset is involved only in legitimate routes. However, by running our model over its 'undecided' routes, we found that 66 out of 108 were classified as hijacked routes, which makes all the three nearest neighbors of AS41881 being involved in hijackings.

7.3 Amazon's Route 53 DNS Service

On April 24th, 2018 between 11AM to 3PM UTC, eNet (AS10297) announced five more specific APs, which belongs to AWS

$$205.251.\{192, 193, 195, 197, 199\}.0/24,$$

while Amazon (AS 16509) continued to announced

$$205.251.\{192, 194, 196, 198, 200\}.0/23.$$

As a results of the attack, a relatively small amount of currency from MyEtherWallet.com was stolen.

Running our network, which was trained on data from March 2018, on data from the time of the attack, the network detects 2 hijacked routes out of the 7 unique BGP routes that are associated with the 5 hijacked APs. Both routes [23673, 55329, 4788, 6939, 10297] with a prediction score of 0.9892 and [47872, 6939, 10297] with a prediction score of 0.3343 do not violate VF routing; namely, the network learned additional features on top of VF. These results demonstrate the strength of our approach.

8 CONCLUDING REMARKS

We introduce a novel approach for control-plane BGP hijacking detection (by classifying BGP routes as hijacked or benign), using a deep learning method. As far as we know, we present the first work in the field that involves the use of an end-to-end deep learning model.

Our method achieves excellent results on our dataset and very good results on a small set of documented past events without any prior assumptions. This is done using a highly unbalanced labeled dataset, which is known to be challenging. In addition, our method detected hijacked routes for routes with missing ToRs and found mistakes in the labeled dataset. This means that we learn additional features beyond VF routing.

We believe that we present a strong case to make deep learning an important tool for BGP hijack detection. We plan to investigate the non-overlapping regions between our method and other well-known tools.

ACKNOWLEDGMENT

This research was funded in part by the Israeli PMO cyber grant program, and by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

REFERENCES

[1] Hussain Alshamrani and Bogdan Ghita. 2016. IP prefix hijack detection using BGP connectivity monitoring. In *IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*. 35–41.
[2] BGPMon. 2019. BGPStream. https://bgpstream.com/about/.
[3] CAIDA. 2019. CAIDA BGP Stream. https://bgpstream.caida.org/.
[4] Dunne Campbell, R. A. Dunne, and N. A. Campbell. 1997. On The Pairing Of The Softmax Activation And Cross–Entropy Penalty Functions And The Derivation Of The Softmax Activation Function. In *8th Australian Conference on Neural Networks*. Melbourne, Australia, 181–185.
[5] Min Cheng, Qian Xu, Jianming Lv, Wenyin Liu, Qing Li, and Jianping Wang. 2016. MS-LSTM: A multi-scale LSTM model for BGP anomaly detection. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 1–6.
[6] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill. 2019. BGP hijacking classification. In *Network Traffic Measurement and Analysis Conference (TMA)*.
[7] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.
[8] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket, and H. S. Baird. 1990. Handwritten zip code recognition with multilayer networks. In *10th International Conference on Pattern Recognition*, Vol. ii. 35–40 vol.2.
[9] Chris C. Demchak and Yuval Shavitt. 2018. China's Maxim - Leave No Access Point Unexploited: The Hidden Story of China Telecom's BGP Hijacking. *Military Cyber Affairs* 3 (Oct. 2018). Issue 1.
[10] Qingye Ding, Zhida Li, Prerna Batta, and Ljiljana Trajković. 2016. Detecting BGP anomalies using machine learning techniques. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 003352–003355.
[11] Dyn. 2019. Dyn blog. https://dyn.com/blog/category/research/.
[12] Martín Abadi et al. 2015. TensorFlow. https://www.tensorflow.org/.
[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[14] Xin Hu and Z Morley Mao. 2007. Accurate real-time identification of IP prefix hijacking. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 3–17.
[15] Geoff Huston and Randy Bush. 2011. Securing BGP and SIDR. *IETF Journal* 7, 1 (2011).
[16] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2006. Pretty good BGP: Improving BGP by cautiously adopting routes. In *icnp*. IEEE, 290–299.
[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980
[18] Mohit Lad, Daniel Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. 2006. PHAS: A Prefix Hijack Alert System.. In *USENIX Security symp.*, Vol. 1.
[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[20] Matt Lepinski and K Sriram. 2017. BGPSEC protocol specification. IETF RFC 8205.
[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[22] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *The 27th International Conference on International Conference on Machine Learning (ICML'10)*. Omnipress, USA, 807–814.

[23] University of Oregon Advanced Network Technology Center. [n.d.]. Route Views Project. http://www.routeviews.org/.
[24] Jian Qiu, Lixin Gao, Supranamaya Ranjan, and Antonio Nucci. 2007. Detecting bogus BGP route information: Going beyond prefix hijacking. In *Security and Privacy in Communications Networks (SecureComm 2007)*. IEEE, 381–390.
[25] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.
[26] Johann Schlamp, Ralph Holz, Quentin Jacquemart, Georg Carle, and Ernst W Biersack. 2016. HEAP: reliable assessment of BGP hijacking attacks. *IEEE Journal on Selected Areas in Communications* 34, 6 (2016), 1849–1861.
[27] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos. 2018. A Survey among Network Operators on BGP Prefix Hijacking. *ACM SIGCOMM Computer Communication Review (CCR)* 48, 1 (Jan 2018), 64–69.
[28] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti. 2018. ARTEMIS: Neutralizing BGP Hijacking within a Minute. *IEEE/ACM Transactions on Networking* (Oct 2018).
[29] T. Shapira and Y. Shavitt. 2020. Unveiling the Type of Relationship Between Autonomous Systems Using Deep Learning. In *IEEE/IFIP NOMS - International Workshop on Analytics for Network and Service Management (AnNet 2020)*.
[30] Xingang Shi, Yang Xiang, Zhiliang Wang, Xia Yin, and Jianping Wu. 2012. Detecting prefix hijackings in the internet with argus. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 15–28.
[31] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy Katz. 2004. Listen and whisper: Security mechanisms for BGP. In *The First Symposium on Networked Systems Design and Implementation (NSDI)*.
[32] Cecilia Testart, Philipp Richter, Alistair King, Alberto Dainotti, and David Clark. 2019. Profiling BGP Serial Hijackers: Capturing Persistent Misbehavior in the Global Routing Table. In *ACM Internet Measurement Conference (IMC '19)*. 420–434.
[33] U. Weinsberg, Y. Shavitt, and E. Shir. 2009. Near-Deterministic Inference of AS Relationships. In *ConTel 2009*. Zagreb, Croatia.
[34] Zheng Zhang, Ying Zhang, Y Charlie Hu, Z Morley Mao, and Randy Bush. 2008. Ispy: detecting IP prefix hijacking on my own. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 327–338.
[35] Changxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis. 2007. A lightweight distributed scheme for detecting IP prefix hijacks in real-time. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 277–288.