SASA: Source-Aware Self-Attention for IP Hijack Detection

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

***Abstract*— IP hijack attacks deflect traffic between endpoints through the attacker network, leading to man-in-the-middle attacks. Current detection solutions are only based on AS-level path analysis, while attacks that include data-plane manipulations may exhibit only geographic anomalies and preserve the AS-level route, or hide the problematic AS in the path. Thus, there is a need to develop data-plane analysis frameworks that examine the actual route packets traverse. We introduce here a deep learning system that examines the geography of traceroute measurements to detect malicious routes. We use multiple geolocation services, with various levels of confidence; each also suffers from location errors. Moreover, identifying a hijacked route is not sufficient since an operator presented with a hijack alert needs an indication of the cause for flagging out the problematic route. Thus, we introduce a novel deep learning layer, called Source-Aware Self-Attention (*SASA*), which is an extension of the attention mechanism. *SASA* learns each data source's confidence and combines this score with the attention of each router in the route to point out the most problematic one. We validate our IP hijacking classification method using two router data types: coordinates and country location, and show that *SASA* outperforms the regular self-attention layer, using the same neural network architecture, and achieves extremely high accuracy.**

***Index Terms*— Internet, BGP, IP hijack, routing, security, IP geolocation, deep learning, attention mechanism, noisy data, dataset.**

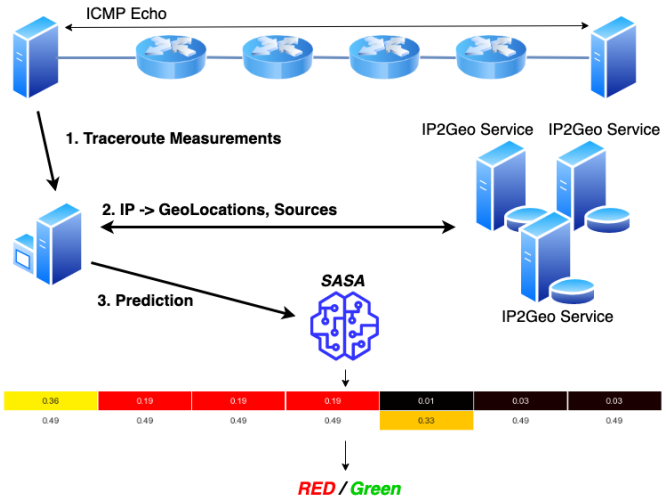


Fig. 1. Inference information flow.

I. Introduction

IN RECENT years, there have been many reports of IP hijack attacks of nations and large companies, as more than 40% of the network operators reported that their organization had been a victim of a hijack in the past [1], [2]. In an IP hijack attack, the attacker diverts the traffic to its own network and then forwards it to the original destination, forming a man-in-the-middle (MITM) attack. This allows espionage, traffic manipulation, network penetration, and more. Since such attacks are hard to perform, they are mostly used by governments and large criminal organizations.

Current solutions for IP hijack detection [3]–[5] are based on monitoring BGP routing announcements, and mostly detect changes due to change of origin AS, the first upstream providers, and some obvious malicious route changes. However, hijack attacks are not limited to BGP manipulations and can also be performed with stealthier methods, e.g., by manipulating routing at the data plane in IXPs or inserting static entries to key ISPs. We show in this paper (see section VII-G), for the first time in the literature, an example of a suspected IP hijack attack that has no BGP signature and seem to be a result of BGP entry manipulation at the source ISP. Thus, we need to develop IP hijack detection tools that examine the actual route packet traverse, which, during a data-plane attack, may not be the one announced in BGP.

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 [6] and throughout this paper we will use IP hijack and deflection interchangeably.

In this paper, we introduce a novel *data-plane* approach for IP hijack detection based on geographical data, using deep learning methods. We rely on the actual route that the packets traverse, obtain through traceroute measurements, rather than the path advertised by BGP [7]. Given the routers' IP addresses along the route, we obtain their geographic location and analyze the route geography.

Manuscript received April 6, 2021; revised July 24, 2021, September 6, 2021, and September 17, 2021; accepted September 21, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Liu. This work was supported in part by the Grant on Cyber Research through the Israeli Prime Minister's Office (PMO) and in part by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University. *(Corresponding author: Tal Shapira.)*

Tal Shapira is with the School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: talshapira1@mail.tau.ac.il).

Yuval Shavitt is with the School of Electrical Engineering, Tel-Aviv University, Tel Aviv 69978, Israel, and also with BGProtect Ltd., Ra'anana 43000, Israel (e-mail: shavitt@eng.tau.ac.il).

Digital Object Identifier 10.1109/TNET.2021.3115935

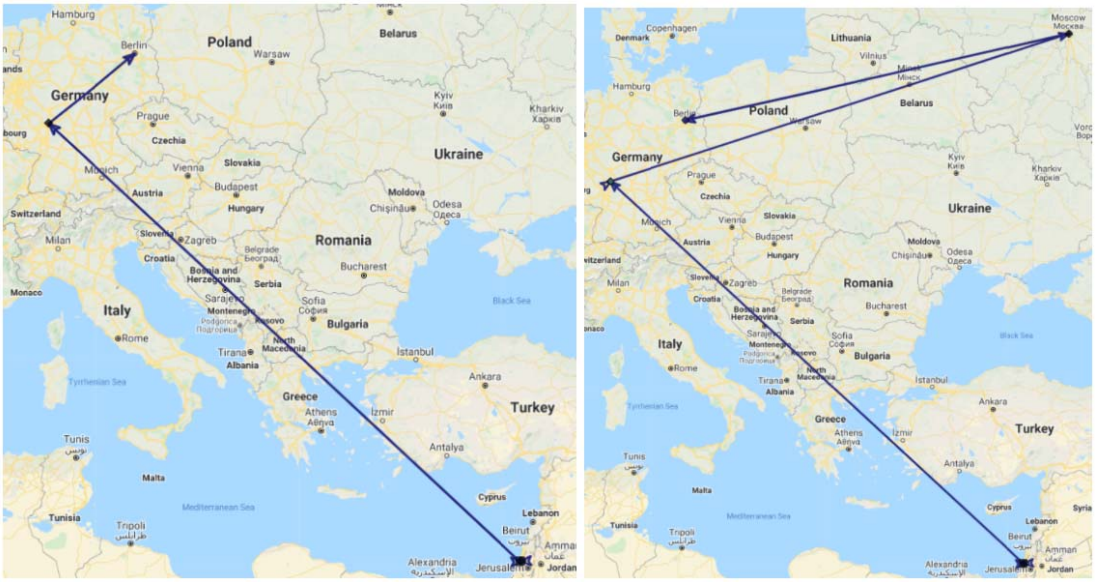


Fig. 2. A comparison between (LEFT) a regular route from Israel to Germany, and (RIGHT) a Russian Hijacking from March 2021 of this route with clear Geographical footprint: IL → DE → **RU** → DE.

Our motivation for using geography for hijack detection is that although the primary routing decision criteria in BGP mostly derive from economic agreements between ASes, these contracts also reflect geographical and geopolitical constraints, as it is reflected in Figure 2. Hence, it is less likely to select paths that significantly deviate from the ideal direct geographical route, and certainly, it is unlikely for a route to traverse unfriendly countries. There have been only a few works that examined using geographical data for characterizing international detours in the Internet [7], [8] and some for visualization purposes [9]–[12]; however, none of them aimed at IP hijack detection.

Our data is obtained from large traceroute measurement campaigns from multiple agents around the world. The IP addresses are converted to geographical information using several geolocation services. Each route is labeled as hijacked or benign by three analysis algorithms: BGP Valley-free (VF) analysis that is based on Shavitt *et al.* [13] with manual corrections, geographical analysis, and ASN ownership analysis (see Sec. III).

The introduction of the Attention Mechanism [14] in deep learning has improved the success of various NLP models, by mapping the essential and relevant words from the input sentence and assign higher weights to these words, enhancing the accuracy of the output prediction. We build on the excellent results achieved for time series tasks and design a new layer that is based on the attention layer. Because there are many different services, with various levels of confidence [15], we design our layer to incorporate the data source confidence level of each data sample. Thus, we called our layer Source-Aware Self-Attention (*SASA*). Just like any other parameter, our layer also learns the confidence of each data source.

We test the *SASA* module in two scenarios. First, we examine a *solid* monitoring campaign from a few tens of locations to a limited set of about 2000 Address Prefixes (APs) worldwide. This campaign was active for many months, and as a result, its related geographic data is more accurate since operators manually corrected geographic errors. The second *noisy* scenario examines routes from 7 monitoring points to the entire IPv4 address space. As a result, it represents a case of high geolocation inaccuracy. In addition, we test the model on a few interesting deflection cases that were not part of the training datasets, all of them are successfully flagged by the model. We make the *noisy* dataset as well as some of the other events data and our code **publicly available** for the community.

Our approach achieves excellent results and also indicates the cause for flagging out problematic routes. We detect IP hijack attacks with an accuracy of 99.24% with 0.80% False Alarm and a detection rate of 99.52% on the *solid* dataset, and accuracy of 90.19% with 9.50% False Alarm and a detection rate of 84.54% on the *noisy* dataset. Finally, as far as we know, we are the first to use geographical route information to detect IP hijack attacks.

The rest of the paper continues as follows. After describing related work in Sec. II, we describe the datasets in Sec. III. In Sec. IV, we introduce some preliminaries on Recurrent Neural Networks and Attention Mechanism, then Sec. V introduces the *SASA* layer and describes the neural network's architecture, and Sec. VI presents our experiments and their results. In Sec. VII we introduce interesting deflection cases, and in Sec. VIII we discuss several deployment issues. Finally, the last section concludes the paper.

II. Related Work

There are many different approaches for the detection of IP hijacking. We divide these approaches into three main categories, based on the type of information they use: 1) Control-plane approaches [3]–[5] - 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 [2], [16], [17] - 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 [18]–[20] - these approaches use both control-plane and data-plane information and sometimes also use external databases to perform joint analysis.

There are several works that involved the use of geographical data, mainly for visualization purposes [9]–[12]. Theodoridis *et al.* [11] introduced an unsupervised method that is based on three features related to the frequency of appearance and the geographic deviation of each intermediate AS towards a given destination country: the probability of an intermediate country appearance along a route toward a specific Origin-Country (CAP), the geographic length which is the ratio of the length of the path against the ideal direct path (CGL), and the Z-score of geographic lengths for all the intermediate countries of a certain Origin-Country (CGLZ). Following their work, Papadopoulos *et al.* [12] presented BGPfuse, a scheme for visualizing and exploring BGP path change anomalies, which used the three features which were mentioned before (CAP, CGL, CGLZ) with the addition of CAPZ, which is the Z-score of CAP. They used these features to quantify the degree of the anomaly of each BGP hijacking event.

Several recent works [6], [21]–[24] examined machine learning techniques with manually generated features to identify malicious origin ASes; mainly leveraging historical BGP data from RouteViews [25] and RIPE. Cho *et al.* [6] classified detected hijack events. Unlike previous works, their work aimed to classify detected hijack events and not detect new hijack events. Shapira and Shavitt [26] have introduced a new method, called *BGP2VEC*, that uses BGP route announcements as sentences to embed each AS number (ASN) to a vector that represents its latent characteristics. Using these vectors as an input to a recurrent neural network, they achieved an excellent result for IP hijack detection.

Shah *et al.* [8] characterized international detours in the Internet, i.e., traffic that leaves a country crossing international boundaries and return to the same country. To detect detours, they used BGP RIB (based on RouteViews [25] and RIPE) and map each AS to its corresponding country using geolocation services. They analyzed each global BGP RIB entry looking for detours and showed that more than 5K unique BGP prefixes experienced a detour. Shah *et al.* [8] claim that over 11% of the ASNs are mapped to more than one countries. Indeed many of the geographic deflection we discovered involve large ASNs that span many countries and thus are not amendable to their analysis. In addition, we aim to discover only deflections that may be due to malicious intent, and the source and origin may not be the same country.

As opposed to Shah *et al.* [8], Edmundson *et al.* [7] used router-level forwarding paths (via traceroutes measurements) to generate country-level paths using MaxMind (which is termed as data source 'A' in our study). They generated paths to popular domains for five chosen countries, characterized transnational routing detours, and compared different country avoidance techniques, i.e., for preventing routing through unfavorable countries, using the open DNS resolver. While MaxMind has very good accuracy for end systems, their router level accuracy has significant errors [15].

In 2004, Zhu and Wu [27] presented a systematic evaluation of the effect of class noise and attribute noise in machine learning, and analyze their impacts on the system performance. Following their work, many advances have been made in dealing with label noise [28]–[31]. However, as far as we know **we are the first to present a deep learning method for dealing with unreliable data in multi-source datasets.**

III. The Datasets

Our data is constructed (see Figure 1) by executing traceroutes from DIMES [32] style software agents (the agents were installed on paid VPSs without volunteer machines) in two different ways: 1) 'Dataset A' - On May 2019, we used 78 agents worldwide (see Table VIII), each performed about 30,000 traceroutes to roughly 2000 destination IPs, overall there were 8213 destinations in 434 ASes worldwide; overall 2,113,049 traceroutes. 2) 'Dataset B' On Dec. 2019, we used 7 agents, each traceroute one IP from every announced AP in the IPv4 range; overall 5,291,799 traceroutes

The IPs in the traceroutes were converted to two types of paths: coordinates (latitude, longitude) and the country location. The conversions were done using three types of databases: (M) a proprietary high accuracy dataset of about 20 million IPs, mostly routers, where geolocations are generated by multiple methods based on traceroute from hundreds of monitors worldwide, (R) a proprietary good accuracy dataset of about 700 million IPs of routers, cloud blocks, and end systems, (A) MaxMind was used for IPs not in the other datasets, mostly for IPs in stub networks. We denote by (X) private IPs and non-responding routers. The total numbers of non-unique usages for each dataset are summarized in Table II.

We generate three labels for each path, using RED for a suspected hijacked route and GREEN otherwise, by applying the followings three rule-based algorithms:

1) Geo - an intermediate country is suspicious, e.g., a route between Italy and Spain traverses Russia.
2) Owner - intermediate network ownership is suspicious, e.g., a route between British and Italian networks traverses a Chinese network in the UK.
3) VF - BGP 'valley-free' analysis [13] using the CAIDA AS Relationships dataset [33].

The rules for both Geo and Owner labeling need to take into account common routing practices, e.g., a route between Australia and Japan may traverse West Coast USA although it is longer than other options, while using a Chinese provider (e.g., China Telecom PoPs in West Europe) for domestic traffic in West Europe is suspicious, since such routes hardly exist. We use RouteViews [25] to map each IP address to its corresponding ASN, and use the CAIDA AS Rank [34] to map each ASN to its owner country both with manual corrections.[1]

The 'combined' label for a path is GREEN if none of its three labels are RED. The number of labels for each algorithm are summarized in Table I. Note that each algorithm has some unclassified routes; therefore, the total number of routes per algorithm may differ. In contrast, the algorithm proposed in this paper classifies all routes.

[1]For example, AS5080 (Aramco) appears as American in public databases, but we correct it to be Saudi.

TABLE I
NUMBER OF SAMPLES IN OUR DATASETS

	Dataset A: 'solid'		Dataset B: 'noisy'	
Algorithm	**GREEN**	**RED**	**GREEN**	**RED**
VF	1,977,640	274	3,666,786	47,254
Geo	2,088,768	3,529	4,826,201	108,196
Owner	2,106,455	5,326	522,9781	24,243
Combined	1,953,252	8,350	3,334,759	170,425

TABLE II
DISTRIBUTION OF DIFFERENT SOURCES IN OUR DATASETS

	Dataset A: 'solid'		Dataset B: 'noisy'	
Source	**Count**	**%**	**Count**	**%**
M	16,214,895	74.02	30,821,949	58.80
R	2,985,041	13.62	12,457,239	23.77
A	1,822,587	8.32	5,851,491	11.16
X	883,532	4.04	3,286,212	6.27

The routes in 'Dataset A' were used continuously for several months for monitoring specific routes and thus were inspected manually. As a result, IP geolocations were corrected manually (either individually or by specially crafted algorithms); indeed, Table II shows that over 74% of the geolocations are marked with 'M.' In addition, the continuous manual route labeling inspection results in the identification of cases where the rule-based labeling is incorrect, and the monitoring system is capable of handling the various level of exception rules that are inserted in these cases by a team of specialists. As a result, the system labeling deviates from the simple rule-based analysis. In fact, this manual exception process is one of the motivations for this paper since we believe we managed to train the model to generalize the specific manual alterations.

We also examined a few interesting deflection cases discovered by our monitoring system during the years 2018-2020. None of the events occur during the time we collected our training datasets.

We make 'Dataset B' as well as several additional deflection cases publicly available.[2] For each hop, we provide its corresponding latitude, longitude, location country code, ASN, and geolocation source. For each path, we provide two types of labels; 'VF' and 'combined.'

IV. PRELIMINARIES

In this section, we introduce some preliminaries on Recurrent Neural Networks (RNN) and Attention Mechanism, which are the two main ingredients of our approach.

A. Recurrent Neural Networks

Recurrent Neural Networks or **RNNs** [35] are a class of Artificial Neural Networks (ANNs) for processing sequential data. In a traditional ANN, we assume that all inputs (and outputs) are independent of each other. RNNs perform the same task for every element of a sequence, with the output being depended on the previous computations, such that each member of the output is a function of the previous members of the output, and is produced using the same update rule applied to the previous outputs. This recurrent formulation results in the sharing of parameters through a very deep computational graph when each node represents the state at time t. RNNs are trained in the same way as MLPs, using back-propagation [36].

The Long Short Term-Memory neural networks, or **LSTMs** [37], are a specific kind of RNNs, which are capable of learning long-term dependencies and have achieved state of the art results in many fields such as speech recognition, translation, image captioning, etc. These networks are composed of units called memory cells that have an internal recurrence (a self-loop), in addition to the outer recurrence of the network. Each cell contains three 'gates'; 1) an input gate to control the flow of inputs into the memory cell, 2) an output gate to control the output flow of cell activations into the rest of the network, and 3) a forget gate, which controls the self-loop weight and set it to a value between 0 and 1 via a Sigmoid unit.

B. Attention Mechanism

An attention Mechanism is a method that is broadly used in the field of Natural Language Processing (NLP) [38], including the problem of long sequences in machine translation [14], [39], [40], by creating a unique mapping between each time step of the decoder output to all the encoder hidden states. Attention mechanisms help deep-learning algorithms to achieve better performance by paying greater attention to certain factors when processing the data. There are many types of attention mechanisms. Among them, there are Bahdanau Attention [39], which commonly referred to as Additive Attention and Luong Attention [40], which often referred to as Multiplicative (dot-product) Attention. Self-attention [14] (or intra-attention) is a specific type of the attention mechanism, which only requires a single sequence to compute its representation, and quantifying the interdependence within the input elements. Since self-attention is applied to both each element and all elements together, no matter how distant they are, the system can capture distant dependency relationships.

The following steps can describe the general process: First, the encoder produces hidden states of each element in the input sequence. Second, the similarity between the encoder hidden states and the decoder hidden states is computed using a similarity function such as a dot product, to obtain an alignment score. In the case of self-attention, the alignment score is computed by the similarity between the input sequence and the hidden states (or between query-key pairs as presented in Figure 3). In the third step, a *softmax* [41] function is applied to normalize these alignment scores and get the attention weights, and finally, the attention weights and their corresponding values are multiplied to form the context vector.

V. METHOD

In this section, we describe in detail the implementation of the *SASA* layer and the architecture of the deep neural network we designed for the classification.

A. SASA

Our Source-Aware Self-Attention layer is based on the Scaled Dot-Product Attention layer (*SDPA*), that was introduced by Vaswani *et al.* [14], which is a variant of the dot-product attention [40]. Let n be the number of elements in the

[2]https://github.com/talshapira/SASA

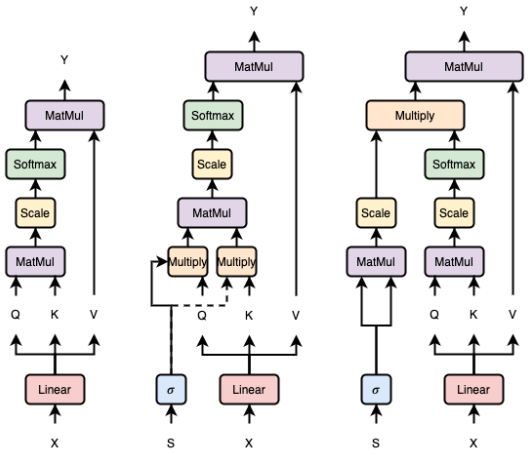


Fig. 3. (left) Scaled Dot-Product Attention. (middle) Scaled Dot-Product Attention using Queries and Keys multiplied by the Sources. (right) Source-Aware Self-Attention.

input sequence. Given input matrices of n query vectors $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, keys vectors $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{n \times d_v}$, the scaled dot-product attention outputs are computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \quad (1)$$

where d_k is the dimension of keys and queries vectors, d_v is the dimension of values vectors, and the softmax [41] function is defined by

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}, \quad \text{for } i = 1, \ldots, n. \quad (2)$$

In the case of self-attention, the queries, keys, and values are obtained by mapping the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{n \times d}$ using different linear projections, where d is the number of hidden units of the input layer.

Given an input of n sources (a source per element), we map each source to a one-hot vector of size m, such that m is the total amount of possible sources, and each vector consists of 0s in all cells except for a single 1 in the cell that is used uniquely to identify the word. Then we embed each one-hot vector to an embedding vector of size d_s, by using the linear projection $\mathbf{W}^S \in \mathbb{R}^{m \times d_s}$, and obtain a matrix of n source vectors $\mathbf{S} \in \mathbb{R}^{n \times d_s}$. Finally, a confidence score is calculated for each source using the Sigmoid function, which is defined by $\sigma(x) = 1/(1 + \exp(-x))$, and maps each value to a confidence value between 0 to 1.

Using the n source vectors, we suggest two variations of layers, which are presented in Figure 3. In the first variation (we termed it (*SDPA-QS-KS*)), we first multiply the keys and the queries vectors with the source vectors using element-wise multiplication (or Hadamard product), and then we used the Scaled Dot-Product Attention as is, i.e.,

$$\text{Attention}(\mathbf{Q} \odot \sigma(\mathbf{S}), \mathbf{K} \odot \sigma(\mathbf{S}), \mathbf{V}). \quad (3)$$

The second variation is our Source-Aware Self-Attention layer (*SASA*). We first multiply (using element-wise multiplication) the scaled squared source scores matrix, which is an $n \times n$ matrix, with the regular attention scores matrix, and then we multiply the product with the values to obtain the attention outputs based on the following equation:

$$\text{SASA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{S}) = (\frac{\sigma(\mathbf{S})^2}{d_s} \odot \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}))\mathbf{V}. \quad (4)$$

In this work, we choose to use $d_k = d_v = d_s = d = 32$. Furthermore, we choose to use a single value for each source, such that we first embed each one-hot vector to a single value (i.e., $d_s = 1$), apply the Sigmoid function for each value, and then we duplicate each value d_k to match the dimensions of keys and queries vectors.

TABLE III
THE ARCHITECTURE OF OUR ATTENTION-LSTM NETWORK

Layer	Size
Embeddings:	Input: 40, 2/1 Output: 40, 32
Attention:	Output: 40, 32
BLSTM:	Output: 200
FC + Sigmoid:	Output: 1

B. Network Architecture

The purpose of this study is to examine the viability of attention mechanisms in two aspects: how to treat multiple information sources with different reliability, and how to highlight the cause of flagging an anomaly. Thus, we concentrate on the design of the attention layer, and do not attempt to optimize the rest of the architecture.

As depicted in Table III, our LSTM architecture comprises four layers, not counting the input. Our input layer consists of 40 entities, which is the maximum length of routes in our datasets. In the case of a shorter route, we pad the remaining entities with 0s. As mentioned in Sec. III, in this work, we use two types of inputs: 1) coordinates, the latitudes and longitudes pairs divided by 90 and 180, correspondingly, such that each entity has a size of 2, and 2) countries, each country is indexed with a number between 0 and 246 (The datasets have 78 and 247 countries, respectively).

A sequence of coordinates or countries is fed into the first layer of the network, which is an embedding layer (We omit the details). The next layer is the attention layer. In our experiments, we compare between different attention layers as described in Sec. V-A. The inputs of each variation of the *SASA* layer also includes the source vectors, as described in Sec. V-A. We use $d_k = d_v = d_s = d = 32$ such that the output remains with the same size as the input. The next layer is the Bidirectional-LSTM layer, which consists of 100 LSTMs cells with a default configuration [37], and produces a 200-size output vector. Finally, our output layer is a single neuron with a Sigmoid activation function, which produces a value between 0 and 1.

C. Training Specifications

The training of the networks is done by optimizing the *binary cross entropy* [42] cost function, which is a measure of

TABLE IV
A SUMMARY OF OUR RESULTS USING DIFFERENT TYPES OF ATTENTION LAYERS, OVER DIFFERENT DATASETS BASED ON THE 'COMBINED' LABELING METHOD

Attention	Data	Sources	Acc.	FA	Rc	AUC	TPR@X%FA		
							10%	1%	0.1%
Dataset A: 'solid', 78 Agents									
SDPA	Coordinates	No	98.93%	1.10%	**99.52%**	99.77%	99.76%	99.39%	**69.43%**
SDPA-Q-S	Coordinates	Yes	98.33%	1.70%	97.67%	99.53%	99.70%	85.59%	39.35%
SDPA-Mul-S	Coordinates	Yes	98.98%	1.00%	99.45%	99.73%	**99.82%**	99.39%	57.93%
SDPA-QS-KS	Coordinates	Yes	98.82%	1.20%	99.33%	99.79%	**99.82%**	99.03%	66.77%
SASA	Coordinates	Yes	**99.10%**	**0.90%**	**99.52%**	**99.80%**	**99.82%**	**99.58%**	66.95%
SDPA	Countries	No	99.11%	0.90%	97.82%	99.72%	99.64%	97.94%	69.73%
SDPA-QS-KS	Countries	Yes	99.06%	0.90%	99.09%	**99.83%**	**99.88%**	99.15%	55.57%
SASA	Countries	Yes	**99.24%**	**0.80%**	**99.52%**	99.80%	99.82%	**99.76%**	**80.81%**
Dataset B: 'noisy', 7 Agents									
SDPA	Coordinates	No	87.51%	12.00%	79.51%	91.75%	76.81%	47.69%	18.31%
SDPA-QS-KS	Coordinates	Yes	86.87%	13.00%	80.85%	92.29%	77.44%	48.61%	19.03%
SASA	Coordinates	Yes	**89.47%**	**10.30%**	**82.43%**	**93.95%**	**82.27%**	**58.11%**	**25.87%**
SDPA	Countries	No	88.58%	11.20%	83.74%	93.69%	82.42%	59.82%	**32.99%**
SDPA-QS-KS	Countries	Yes	89.22%	10.50%	83.65%	94.10%	83.20%	60.61%	32.07%
SASA	Countries	Yes	**90.19%**	**9.50%**	**84.54%**	**94.56%**	**84.95%**	**63.02%**	31.75%

the difference between the *Sigmoid* layer output and the true label of the sample. For the optimization process we use the *Adam* [43] gradient-based optimizer. Because our datasets are imbalanced, i.e., the 'RED' classes are mostly less than 1% of the datasets, we use a balanced generator that randomly samples the training set, such that in each batch, we have the same amount of 'RED' and 'GREEN' paths.

We build and run our networks using the *Keras* [44] library with *Tensorflow* [45] 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 60 epochs of the training set. We save the result, which achieves the best accuracy during the training process.

VI. EXPERIMENTS AND RESULTS

In this section, we report our experimental results. Since we have not found any previous work to compare our routes with, we compare our *SASA* layer with the regular Scaled Dot-Product Attention (*SDPA*) layer, as well as other variants we suggest. Since both *SDPA* and *SASA* present good attention performance, we will devote most of the evaluation section to the gain in accuracy of hijack detection obtained by the *SASA* layer.

A. Evaluation Criteria

Any classification system for anomaly detection (in our case route deflection) would like to maximize its detection rate and minimize the false alarm rate. For this purpose, we introduce three criteria as presented in Table IV:

1) **Accuracy** (Acc.), 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

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

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,

2) **False Alarm** (FA), which is the False Positive Rate (FPR), defined as

$$FPR = \frac{FP}{FP + TN}, \tag{6}$$

3) **Detection Rate** or **Recall** (Rc), which is defined by

$$Rc = \frac{TP}{TP + FN}. \tag{7}$$

In many cases, one would like to control the trade-off between the false alarm rate and the detection rate. Namely, set the accepted false alarm (which can be done here by setting the threshold for the prediction score) and aim at the highest possible detection rate. Thus, we introduce two additional evaluation criteria:

4) **AUC**, which is the area under the ROC curve (a plot of the true positive rate (TPR) against the false positive rate (FPR) at various thresholds), as displayed in Figures 4 and 5, and
5) **True Positive Rate** (TPR) as defined by

$$TPR = \frac{TP}{TP + FN}, \tag{8}$$

for different FPR values. In real-world deployments, the number of routes to be classified by our models may be quite high. As a result, even a moderate FPR may result in too many false alarms, which will make the detector unuseful. Consequently, we evaluate the TPR of the detectors while enforcing FPR levels (0.1%, 1%, and 10%).

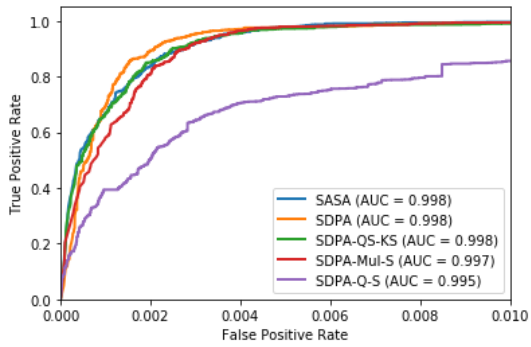


Fig. 4. ROC curves of models on 'Dataset A' - test set for FPR 0.01.

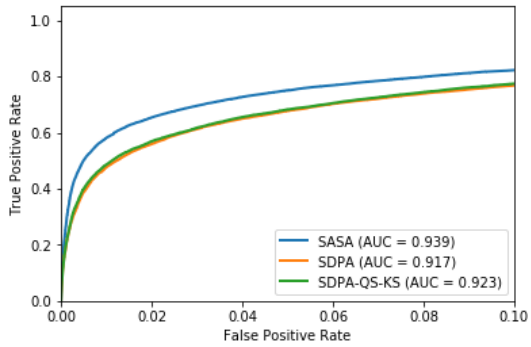


Fig. 5. ROC curves of models on 'Dataset B' - test set for FPR 0.10.

B. Results on Geo Routes Classification

A comparison of our results using different types of attention layers is presented in Table IV. For each dataset, we conducted multiple experiments using different data types; coordinates and countries as described in Sec. III, and different Attention layers; Scaled Dot Product Attention (*SDPA*), *SDPA-QS-KS* and Source-Aware Self-Attention (*SASA*), as described in Sec. V-A. Furthermore, we also conducted one experiment using the *SDPA* as is, by just replacing the keys (or queries) with the sources (denoted by *SDPA-Q-S*), i.e.,

$$\text{Attention}(\mathbf{Q}, \sigma(\mathbf{S}), \mathbf{V}), \tag{9}$$

and another experiment by using the *SDPA* as is and just multiplying it with the sources (denoted by *SDPA-Mul-S*), i.e.,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \odot \mathbf{S}. \tag{10}$$

As Table IV shows all three *SDPA-QS-KS*, *SDPA-Q-S*, and *SDPA-Mul-S* led to performance degradation (even relative to the regular *SDPA* without the use of sources). For each experiment in Table IV, after training our neural network over the corresponding training set, we evaluate our method over the test set, which consists of 20% of the dataset and based on the 'combined' labeling.

Given the imbalance of the datasets, an "always GREEN" classifier will achieve an accuracy of 99.57% and 94.89% on datasets A and B respectively, with 0 false alarm. Not surprisingly, all the tested methods achieved similar results, with *SASA* showing (slightly for 'Dataset A') better accuracy and better false alarm rate: an accuracy of **99.24%** with a false alarm of **0.8%** on 'Dataset A', and an accuracy of 90.19%, with 9.5% FA on 'Dataset B', as presented in Table IV. All the methods achieve high Recall values, where *SASA* achieves the highest results with 99.52% on 'Dataset A', while the "always GREEN" classifier would get 0%. Namely, looking at all three parameters, we can see that learning was achieved despite the large imbalance. Notice that on 'Dataset A', which is less noisy (the 'M' sources comprises about 74% of the sources), the *SASA* layer achieves 0.1-0.2% improvement, while on 'Dataset B', which is noisier (the 'M' sources comprises less than 59% of the sources), the *SASA* layer achieves 1.6-1.9% improvement.

It can also be noticed that the use of countries achieves better accuracy than coordinates; this may be explained by the fact that borders are far from convex, and it is hard to learn cases when one country stretches into another. For example, the Vladivostok area can be easily mistaken to be part of China, and distinguishing between Singapore and Batam, ID that are only 30Km apart is hard since Indonesia has territories that are engulfing Singapore from almost any direction.

Figures 4 and 5 present ROC curves of all models on 'Dataset A' and 'Dataset B', respectively, using coordinates as entities and based on the 'combined' labeling method. It can be seen that both *SDPA*, *SASA*, and *SDPA-QS-KS* achieve great TPR results on 'Dataset A' even for low FPR below 0.5%. On 'Dataset B' it can be seen that *SASA* outperforms all other models, and for low FPR, achieve TPR values higher than the rest by about 10%. The AUC value for *SASA* (see Table IV) is 99.80 and 94.56, for datasets A and B, respectively.

The purpose of detecting deflected routes is to be able to react in case of an IP hijack attack. If a system creates too many false alarms (FAs), the load on the responsible team may be too high, and system credibility will be hurt. Thus, we would like to control the false alarm rate and achieve the highest possible detection rate. Table IV shows trade-off points for moderate FA of 1%, and low FA rate of 0.1%. For countries, which is the better option as we have already seen, *SASA* has a significantly better detection rate, over 80.81% detection rate, and a gap above 10% for 0.1% FA for 'Dataset A'. For 'Dataset B', 0.1% FA is not a possible working point since all methods detect less than a third of the deflection events; for 1% FA, *SASA* detects 63% of the deflections, about 3% better that *SPDA*.

Table V displays a comparison of experiments using different labeling methods: Geo, Owner, VF, and Combined as described in Sec. III. In each experiment, we trained our network with coordinate data using one of the specific labeling methods, and evaluate its performance based on a unified test set using the 'combined' labeling method. The results highlight the generalization ability of our method; by training our network using 'Geo' and 'Owner' labels, our method can achieve high accuracy for the 'combined' labeling method, in some cases even better results than by training using the 'combined' labeling method. It can also be seen that based on the VF labeling, our network achieves lower accuracy, which may be explained by the fact that VF is not based on geographical data.

C. Exploration of Source Scores

Table VI displays a comparison of source scores that were calculated by *SASA* for the different path types. Interestingly,

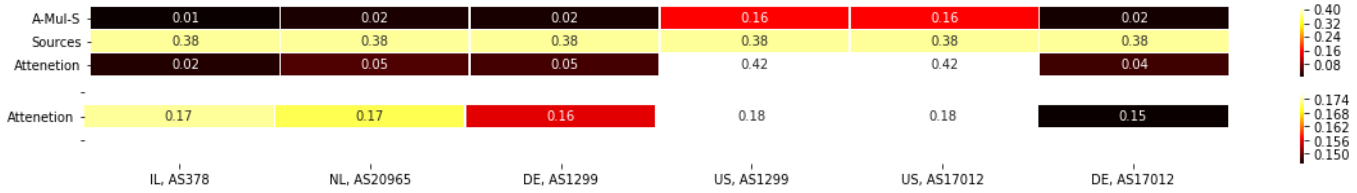


Fig. 6. Rows from top: *SASA* attention scores (A-Mul-S), source scores, *SASA* intermediate attention scores (without sources), and *SDPA* attention scores.

TABLE V

A COMPARISON OF *SPDA* AND *SASA* COORDINATES-ROUTES CLASSIFICATION RESULTS USING DIFFERENT LABELING METHODS BASED ON THE 'COMBINED' LABELING METHOD (DENOTED AS 'C')

Attention	Label	Acc.	FA	Rc	AUC	'C' Acc.	'C' FA	'C' Rc	'C' AUC
Dataset A: 'solid', 78 Agents									
SDPA	Geo	99.43%	0.60%	99.57%	**99.92%**	99.21%	0.60%	44.43%	**83.74%**
SASA	Geo	**99.64%**	**0.40%**	**99.86%**	99.88%	**99.42%**	**0.30%**	**45.46%**	79.33%
SDPA	Own	**99.39%**	**0.60%**	98.11%	99.78%	**99.25%**	**0.60%**	**63.74%**	75.85%
SASA	Own	99.27%	0.70%	**99.72%**	**99.80%**	99.14%	0.70%	65.43%	**81.85%**
SDPA	VF	94.16%	5.80%	**98.04%**	**98.65%**	93.89%	5.80%	**19.31%**	**67.07%**
SASA	VF	**95.84%**	**4.20%**	**98.04%**	98.46%	**95.45%**	**4.20%**	4.60%	44.40%
Dataset B: 'noisy', 7 Agents									
SDPA	Geo	86.87%	13.00%	84.68%	93.82%	85.77%	**12.00%**	61.80%	80.30%
SASA	Geo	**89.70%**	**12.00%**	**90.36%**	**96.58%**	**88.48%**	**12.00%**	**64.07%**	**81.79%**
SDPA	Own	89.14%	10.90%	89.99%	95.94%	86.71%	10.40%	**30.95%**	**61.71%**
SASA	Own	**90.38%**	**9.60%**	**94.32%**	**97.39%**	**87.72%**	**9.20%**	29.73%	61.23%
SDPA	VF	86.52%	13.40%	83.24%	92.83%	84.58%	13.30%	42.79%	71.37%
SASA	VF	**88.01%**	**12.00%**	**88.75%**	**95.03%**	**86.02%**	**11.80%**	**43.30%**	**72.06%**

TABLE VI

A COMPARISON OF DATA SOURCE SCORES THAT WERE CALCULATED BY *SASA*

Data	Dataset	M	R	A	X
Coordinates	A	0.45	0.15	0.75	0.62
Countries	A	0.41	0.82	0.21	0.49
Coordinates	B	0.49	0.33	0.15	0.50
Countries	B	0.20	0.11	0.09	0.62

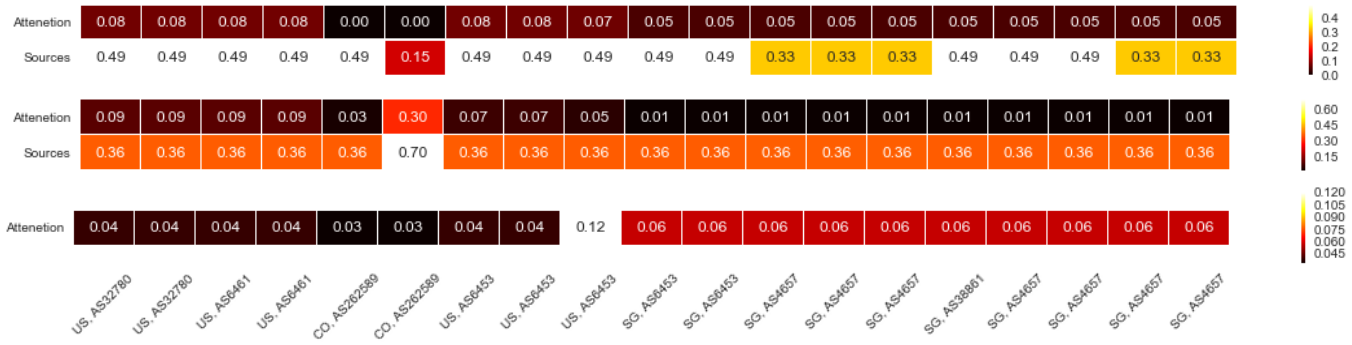
the network gave a higher score to untrusted than to the trusted sources. The reason for this is that MaxMind is quite accurate for stub networks, while badly performs for routers and clouds. However, the MaxMind data is hardly used for routers, especially in the first dataset, since the proprietary databases are used for these. As a result, for 'Dataset A', MaxMind is mostly used where it is in fact accurate, leading to a high score. Furthermore, consistently, the 'X' source, which stands for restricted IPs, gained a high source score. This may be because 'RED' routes have a relatively higher amount of X (and A) sources than 'GREEN' routes.

D. Comparison With Other Methods

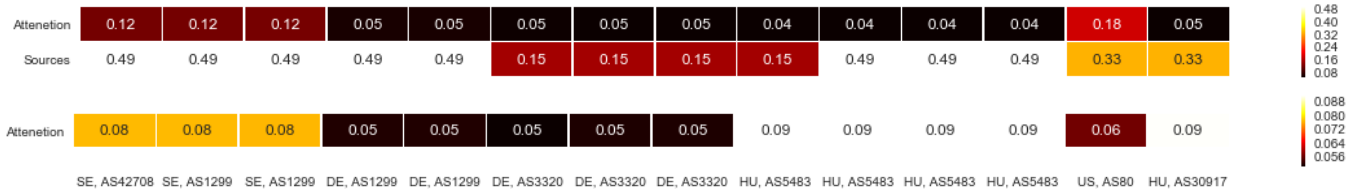
We compare the performance of *SASA* with two AS-level path analysis methods: 1) *VF* - the Valley-Free approach, which is a 'classic' AS-level path analysis method for BGP hijacking detection [19], [46], [47]. The computation of the 'VF' is based on the CAIDA AS Relationships dataset [33]. The comparison is performed using a subset of the test set which has both coordinates and ASN-level routes, for a fair comparison with the 'VF' approach. 2) *BGP2VEC* - we use the same deep learning model that was used in [26]. The model was trained based on a labeled dataset of approximately 3,600,000 BGP route announcements that was collected in March 2018.

Furthermore, we compare our Attention model with other state of the art Machine Learning techniques using the coordinates routes. We use the following ML methods: Logistic Regression (LR), Support Vector Machine (SVM), Naive Bayes classifier (NB), a single hidden layer neural network (MLP), Decision Tree (DT), and Random Forest (RF). We used the python *scikit-learn* package [48] to run the ML models with default parameters.

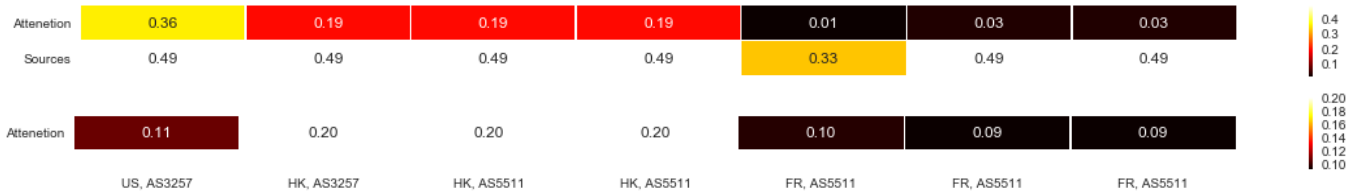
Table VII presents the comparison using two labeling methods; 'Geo' and 'Owner', using 'Dataset B'. Clearly, the 'VF' algorithm detects only a low percentage of the deflected routes. We suspect that this is due to the usage of tier-2 providers that are controlled by state actors. As a result, the deflections do not create a valley to be detected by the VF algorithm. Note, that the 'VF' algorithm does not provide prediction score, and therefore the last four columns are empty. BGP2Vec was trained on AS level data (without geography), thus its close to 50% success rate is impressing since the number of VF and Owner tagged RED routes is less than 50% of the combined RED routes (see Table I).



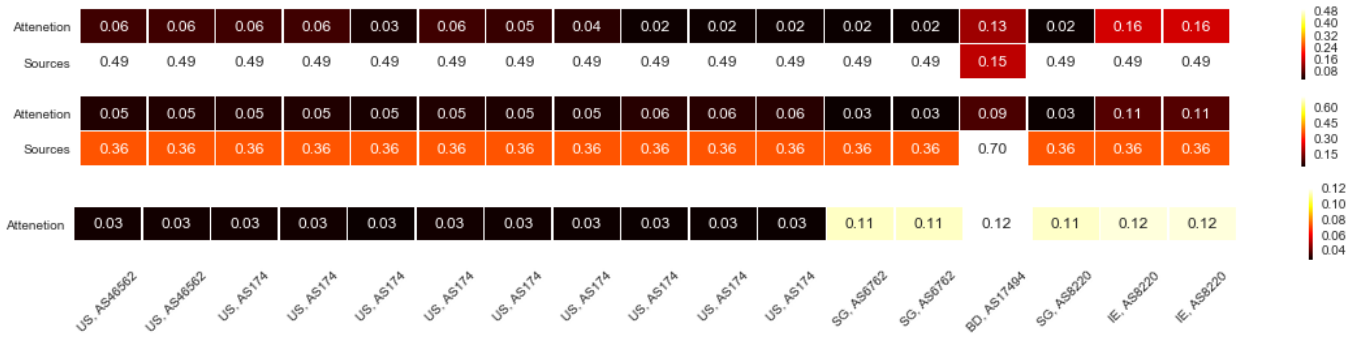
(a) 'Deflection to Columbia' - rows from top: *SASA* attention scores and source scores, *SDPA-QS-KS* attention scores and source scores, and *SDPA* attention scores.



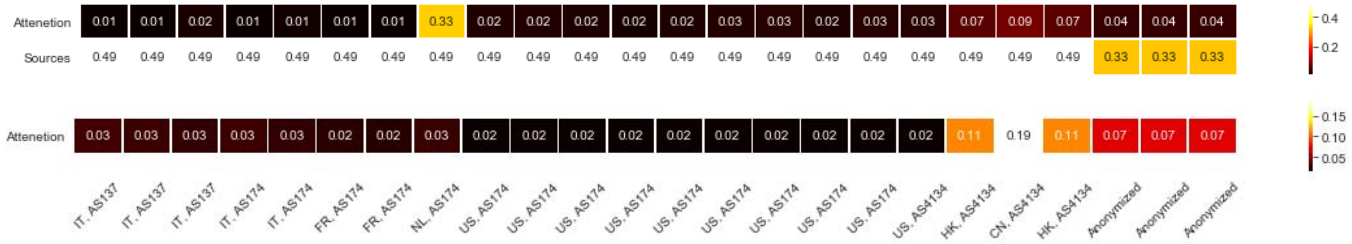
(b) 'Usage of Grey IP Space' - rows from top: *SASA* attention scores and source scores, and *SDPA* attention scores.



(c) 'GTT and Orange Peering' - rows from top: *SASA* attention scores and source scores, and *SDPA* attention scores.



(d) 'Bangladesh Leak' - rows from top: *SASA* attention scores and source scores, *SDPA-QS-KS* attention scores and source scores, and *SDPA* attention scores.



(e) 'Asian Hijack' - rows from top: *SASA* attention scores and source scores, and *SDPA* attention scores.

Fig. 7. Examples of deflected routes.

The SASA detection rate is significantly higher than any other ML method for both labeling. We suspect that the noise in the data is confusing classic methods but SASA manages to clean it at the cost of an increase in the false alarm. The AUC of SASA is also the best of all methods, but here DT and RF are doing quite well. In particular, for some of the example points given in the right-most columns *SASA* does not achieve the best results.

TABLE VII

A SUMMARY OF OUR RESULTS COMPARED WITH 'VF' METHOD (BASED ON BGP ROUTES) AND OTHER COMMON MACHINE LEARNING TECHNIQUES BASED ON THE COORDINATES ROUTES. FOR THE EVALUATION WE USE 'DATASET B' USING BOTH 'GEO' AND 'OWNER' LABELING METHODS

Method	**Acc.**	**FA**	**Detection Rate (Rc)**	**AUC**	**TPR@X%FA**		
					10%	**1%**	**0.1%**
Dataset B: 'noisy', 7 Agents, 'Geo'							
SASA	89.81%	10.16%	**88.27%**	**95.96%**	88.12%	60.78%	23.84%
VF	96.77%	1.25%	2.41%	-	-	-	-
BGP2VEC	87.08%	12.45%	49.54%	80.58%	45.75	12.44%	2.80%
LR	97.95%	0.00%	0.00%	61.95%	15.83%	1.73%	0.06%
SVM	97.95%	0.00%	0.00%	61.64%	15.45%	1.51%	0.16%
NB	95.71%	2.36%	3.69%	61.78%	14.06%	1.68%	0.12%
MLP	98.13%	0.06%	11.44%	85.87%	62.96%	34.86%	15.50%
DT	98.85%	0.42%	64.18%	87.04%	75.94%	70.57%	45.00%
RF	99.02%	0.16%	60.06%	94.92%	90.58%	76.10%	53.78%
Dataset B: 'noisy', 7 Agents, 'Owner'							
SASA	90.54%	9.47%	**94.34%**	**97.52%**	95.22%	50.53%	14.75%
VF	98.36%	1.31%	14.33%	-	-	-	-
BGP2VEC	87.05%	12.45%	49.57%	80.57%	45.75%	12.44%	2.81%
LR	99.62%	0.00%	0.00%	79.69%	38.22%	5.88%	0.34%
SVM	99.62%	0.00%	0.00%	79.24%	36.81%	3.13%	0.34%
NB	99.13%	0.49%	1.14%	80.48%	31.00%	3.09%	0.26%
MLP	99.67%	0.01%	14.87%	93.14%	77.25%	40.55%	25.45%
DT	99.75%	0.10%	85.97%	85.97%	72.55%	72.43%	62.31%
RF	99.81%	0.02%	58.86%	94.04%	89.02%	81.07%	67.50%

VII. EXAMPLES

We tested here *SASA* and the other models on a few interesting deflection cases that were not part of the datasets. These events are all from recent years, 2017-2020 and we make their data available.[3] All three models that were trained using the noisy dataset with coordinates identified all the examples as hijacked.

As our primary motivation for using attention was to gain the ability to explain the classification of our neural network, figures 6 and 7 display examples of attention scores using different attention layers.

A. PayPal

The route from Israel's academic network to PayPal in Germany is routed through a PayPal gateway in Eastern US and thus flagged as suspicious (we see many such PayPal routes from Europe). As Figure 6 shows, both layers highlighted the US IPs that are responsible for the deflection; however, the use of sources gains higher explainability: 0.16 scores for the US IPs compared to 0.02 of the second-highest score, while for *SDPA* the US scores are 0.18, but other IPs have similar score: 0.17 and 0.16. Also, note the high score of the US IPs in the intermediate calculation of the *SASA* layer (before multiplying by the source score) 0.42 compared to 0.05 of the second highest.

B. Deflection to Columbia

A /24 AP, which is owned by StarHub in Singapore, is only announced through TATA as an upstream provider, and as a result, has low visibility. The TATA announcement to Internexa (AS262589) is leaked to Zayo (AS6461) that exports it due to no other alternatives. As a result, many routes from the US to this Singaporean block are routed through Columbia in a route with a long geographic deflection and valley-free violation.

All three methods (while only *SDPA* and *SDPA-QS-KS* shown) successfully identified the route deflection with an almost perfect prediction score (above 99.9%). *SDPA-QA-KS* identified the problematic route hop with a very distinctive high attention score, while *SDPA*, which does not use the source score failed. *SASA* also failed in identifying the deflection root score.

C. Usage of Grey IP Space

A route from Stockholm, Sweden to Budapest, Hungary (measured in June 2020, but exist for many months) shows a General Electric unannounced IP address between Magyar Telekom (AS5483) and Budapest Hitel es Fejlesztesi Bank (AS30917). This is most likely a loopback interface configuration of Magyar Telekom.

SASA successfully flagged the US hop, with a relatively high score, while *SDPA* flagged the Hungarian portion of the route with a score similar to the Swedish portion of the route.

D. GTT and Orange Peering

The example shows a route from Seattle, WA, US to Paris, France that involve only two tier-1 providers: GTT (AS3257) and Orange (AS5511). Until recently, GTT Carried the traffic to Paris, where it peers with Orange. However, starting from June 24th, 2020, slightly after 16:00 UTC, the peering point

[3]https://github.com/talshapira/SASA

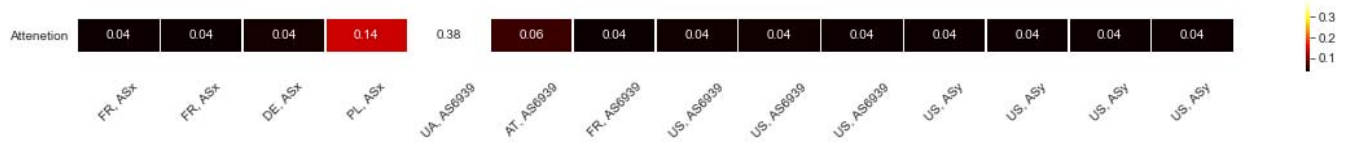


Fig. 8. Geo-deflection in Europe - an example of a deflected route that cannot be observed via BGP analysis.

moved to Hong Kong. While we do not suspect this routing change is malicious due to the parties involved, this is exactly the type of misconfiguration a hijack alert system should flag.

Figure 7(c) shows that *SPDA* managed to highlight HK as the problematic part of the route, while *SASA* flagged the US origin.

E. Bangladesh Leak

On September 26th, 2018, between about 7:45 UTC and 14:45, BTCL (AS17494) of Bangladesh leaked over 3500 APs to Telecom Italia (AS6762) that exported to its peers. As a result, many routes worldwide were diverted to Bangladesh.

The example in Figure 7 shows a route between the Total cloud in Los Angles, California, to a Colt IP address in Dublin, Ireland. The route before the leak was comprised of Total (AS46562) in LA, GTT in LA, Level 3 from LA to Paris, and Colt (AS8220) from Paris to Dublin.

Both the *SASA* and *SDPA-QA-KS* layers successfully highlighted the Bangladeshi hop as the deflection source. *SDPA* had the same score for the Bangladeshi and Irish portions of the route with a similar score for the Singaporean portion.

F. Asian Hijack

In 2016 China Telecom hijacked traffic from several European countries to an Asian government network (details are anonymized). In this example, we show the route from the GARR academic network, where Cogent carry the traffic from Rome, Italy to Los Angeles; there it is peering with China Telecom that hijacks the traffic through Guangdong, China.

Both the *SASA* and *SDPA-QA-KS* layers successfully classified the route as hijacked. Furthermore, figure 7(e) shows that *SPDA* managed to highlight China as the problematic part of the route, with a relatively high attention score.

G. Geo-Deflection in Europe

On April 2017, a geographic analysis detected a deflection of a route towards a single AP that belongs to a tier-2/3 provider in New England from a large cloud provider in France. The AS-level route, both the one obtained from traceroute and from BGP announcements, was benign: Hurricane Electric (AS6939) connects two customer networks. Therefore, an analysis based on the AS-level route (such as 'VF') would not flag this route. However, the geography of the route, traversing Kiev, was highly unusual and extremely suspicious.

The route was compared to many other routes between the French provider in France and other destinations in New England, all these routes where geographically confined to West Europe and North America. Following a message we sent to the French provider NOC, the route was immediately corrected.

As presented in Figure 8, *SASA* successfully classified the route as hijacked and highlighted Ukraine as the problematic part of the route, with a very high attention score.

H. Examples Summary

In general, both the *SASA* and *SDPA-QA-KS* layers were successful in highlighting the cause of the deflection in the routes, but *SASA* was **better at detecting deflection** events.

There is a subtle issue regarding the highlighting of the problematic portion of the route. Consider the American hop on the route from Sweden to Hungary (Figure 7(b)). The reason the US hop is problematic is that it appears in a continental route in Europe, and there is nothing wrong with its location on its own. In other words, what is problematic in the route is the triplet ⟨Sweden (source), US (middle), Hungary (destination)⟩. In practice, this is what we would like the network to learn. Remember, that nowhere in the training process we directed the network what is wrong, each route was simply labeled as 'GREEN' or 'RED'.

VIII. Discussion

We showed that by introducing an attention mechanism to the model, deflected routes' detection rate improves. We also showed that in many cases, the attention mechanism highlights the problematic portion of the route successfully. However, as discussed in Sec. VII-H, the reason to flag a route as deflected is not a single segment in it, but the combination of this segment with source and destination location. As future work, it will be interesting to force the system to output a triplet, or alternatively, to disallow it to consider either end of the route as problematic.

Selecting the right false-alarm rate for the system is not trivial. If the route monitoring system generates too many false alarms, it will lose operators' credibility or simply overwhelm them with work. Assume that an organization wishes to protect 200 APs each from 50 locations, this results in 10,000 routes to monitor, and with FA $= 0.01$, it may result in 100 FAs. However, routes are highly correlated since routes towards a destination share the final portions of the route, and routes emanating from a monitoring point share the same start. In addition, organizations tend to have several APs in the same location connected to the same upstream providers. This will significantly reduce the number of events for the SOC team to handle, where an event is the collection of all flagged routes with the same routing problem at the same time. Of course, with time, a noisy dataset like 'Dataset B' will gradually be cleaned since false alarms will trigger database corrections.

TABLE VIII
AGENT DISTRIBUTION BY COUNTRY IN OUR DATASETS

Country	Dataset A: 'solid'	Dataset B: 'noisy'
AT	1	0
AU	3	0
BE	1	0
BR	1	0
CA	6	0
CH	2	0
CN	1	0
CZ	2	0
DE	4	1
ES	2	0
FR	3	1
GB	4	1
ID	0	1
IL	2	1
IN	1	0
IT	5	0
JP	2	0
KZ	1	0
LT	1	0
NL	1	0
PL	3	0
RO	1	0
RU	1	0
SE	1	0
SG	7	1
TW	1	0
US	20	1
ZA	1	0
Total	**78**	**7**

We need to acknowledge that traceroute measurements come with their own problems. Some networks block ICMP probes [32, Sec. 4.3], but this mostly happens at stub ASes, which are less important for detecting deflections. Table II shows that only 4-6% (X data source) of the routers are either private IP addresses or do not return our probes. This number is quite small, and is mostly due to a few transit networks. SASA associates a weight to these type of routers, and it may be interesting in the future to separate this 'dataset' to non-responsive and private IPs.

Another problematic issue in this paper is the lack of external ground truth. We strongly believe that our labeling, at least for the 'solid' dataset, is mostly reliable, since this data is used and eye-balled continuously for monitoring routes. Indeed, the results for the solid dataset are significantly better than for the noisy dataset.

IX. CONCLUDING REMARKS

We introduced a novel approach for data-plane (namely traceroute based) IP hijack detection by classifying geographic routes, using a deep learning method. As far as we know, our work is the first to detect IP hijacked routes based on geography.

In order to geolocate the IP addresses, we used multiple geolocation services, with various levels of confidence; all suffer from geolocation errors. To take advantage of the knowledge of the sources, we developed an attention-based layer that aimed to deal with multi-source data; we termed it Source-Aware Self-Attention, *SASA*. This layer also highlights the cause of flagging out a problematic route.

We showed that both the *SASA* and *SDPA* layers were successful in highlighting the cause of the deflection in the routes, but *SASA* was better at detecting deflection events. We showed that by training our network on an unbalanced dataset, we could detect hijacked routes with an accuracy of 99.24% with 0.80% False Alarm and a detection rate of 99.52%, based only on geographical data. We also tested *SASA* and the other models on a few interesting deflections cases from 2017-2020, and correctly identified all of them as hijacked.

Finally, a sophisticated attacker may attempt to hide the attack by sending ICMP replies that impersonate a legitimate route. This makes the attack significantly harder to design and perform. However, an interesting future direction is to extend the input SASA with delay and TTL in order to make such impersonation harder.

APPENDIX
AGENT DISTRIBUTION

The location of the agents we used for the generation of the two datasets is presented in Table VIII.

REFERENCES

[1] 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, 2018.

[2] C. Demchak and Y. Shavitt, "China's maxim–leave no access point unexploited: The hidden story of China telecom's BGP hijacking," *Mil. Cyber Affairs*, vol. 3, no. 1, p. 7, Jun. 2018.

[3] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A prefix hijack alert system," in *Proc. USENIX Secur. Symp.*, 2006, vol. 1, no. 2, p. 3.

[4] J. Qiu, L. Gao, S. Ranjan, and A. Nucci, "Detecting bogus BGP route information: Going beyond prefix hijacking," in *Proc. Secur. Privacy Commun. Netw.*, 2007, pp. 381–390.

[5] P. Sermpezis *et al.*, "ARTEMIS: Neutralizing BGP hijacking within a minute," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2471–2486, Dec. 2018.

[6] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "BGP hijacking classification," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2019, pp. 25–32.

[7] A. Edmundson, R. Ensafi, N. Feamster, and J. Rexford, "Characterizing and avoiding routing detours through surveillance states," 2016, *arXiv:1605.07685*. [Online]. Available: http://arxiv.org/abs/1605.07685

[8] A. Shah, R. Fontugne, and C. Papadopoulos, "Towards characterizing international routing detours," in *Proc. 12th Asian Internet Eng. Conf.*, New York, NY, USA, Nov. 2016, pp. 17–24.

[9] M. Syamkumar, R. Durairajan, and P. Barford, "Bigfoot: A geo-based visualization methodology for detecting BGP threats," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2016, pp. 1–8.

[10] S. Papadopoulos, K. Moustakas, and D. Tzovaras, "BGPViewer: Using graph representations to explore BGP routing changes," in *Proc. 18th Int. Conf. Digit. Signal Process. (DSP)*, Jul. 2013, pp. 1–6.

[11] G. Theodoridis, O. Tsigkas, and D. Tzovaras, "A novel unsupervised method for securing BGP against routing hijacks," in *Computer and Information Sciences*. London, U.K.: Springer, 2013, pp. 21–29.

[12] S. Papadopoulos, G. Theodoridis, and D. Tzovaras, "BGPfuse: Using visual feature fusion for the detection and attribution of BGP anomalies," in *The 10th Workshop Vis. Cyber Secur.*, Oct. 2013, pp. 57–64.

[13] Y. Shavitt, E. Shir, and U. Weinsberg, "Near-deterministic inference of AS relationships," in *Proc. ConTel*, Zagreb, Croatia, Jun. 2009, pp. 191–198.

[14] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[15] Y. Shavitt and N. Zilberman, "A geolocation databases study," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 10, pp. 2044–2056, Dec. 2011.

[16] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting ip prefix hijacks in real-time," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 277–288, 2007.

[17] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: Detecting IP prefix hijacking on my own," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 327–338, 2008.

[18] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 3–17.

[19] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the internet with Argus," in *Proc. Internet Meas. Conf.*, 2012, pp. 15–28.

[20] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. E. Biersack, "HEAP: Reliable assessment of BGP hijacking attacks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 6, pp. 1849–1861, Jun. 2016.

[21] 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 *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–6.

[22] Q. Ding, Z. Li, P. Batta, and L. Trajkovic, "Detecting BGP anomalies using machine learning techniques," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, p. 003.

[23] H. Alshamrani and B. Ghita, "IP prefix hijack detection using BGP connectivity monitoring," in *Proc. IEEE 17th Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2016, pp. 35–41.

[24] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling BGP serial hijackers: Capturing persistent misbehavior in the global routing table," in *Proc. Internet Meas. Conf.*, Oct. 2019, pp. 420–434.

[25] Oregon Advanced Network Technology Center. *Route Views Project*. Accessed: May 2019. [Online]. Available: http://www.routeviews.org/

[26] T. Shapira and Y. Shavitt, "A deep learning approach for IP hijack detection based on ASN embedding," in *Proc. ACM SIGCOMM Workshop Netw. Meets AI*, Aug. 2020, pp. 35–41.

[27] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, Nov. 2004.

[28] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, 2012, pp. 567–574.

[29] A. J. Bekker and J. Goldberger, "Training deep neural-networks based on unreliable labels," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Feb. 2016, pp. 2682–2686.

[30] Y. Dgani, H. Greenspan, and J. Goldberger, "Training a neural network based on unreliable human annotation of medical images," in *Proc. IEEE 15th Int. Symp. Biomed. Imag.*, Apr. 2018, pp. 39–42.

[31] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 967–972.

[32] Y. Shavitt and E. Shir, "DIMES: Let the internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 71–74, 2005.

[33] (2018). *TCR Dataset*. [Online]. Available: http://www.caida.org/data/active/as-relationships/

[34] (Jul. 2018). *Caida as Rank*. [Online]. Available: http://as-rank.caida.org/

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[36] 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.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4929–4936.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[40] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*. [Online]. Available: http://arxiv.org/abs/1508.04025

[41] Y.-T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 36, no. 7, pp. 1141–1151, Jul. 1988.

[42] 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 Austral. Conf. Neural Netw.*, Melbourne, VIC, Australia, 1997, pp. 181–185.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Oct. 2014.

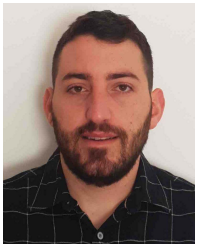
[44] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[45] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[46] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, Dec. 2001.

[47] S. Sundaresan, R. Lychev, and V. Valancius, "Preventing attacks on BGP policies: One bit is enough," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-CS-11-07, 2011.

[48] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.



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, and the M.Sc. degree *(magna cum laude)* in electrical engineering from Tel Aviv University, Tel Aviv, Israel, in 2018, where he is currently pursuing the Ph.D. degree with the School of Electrical Engineering.

From 2009 to 2012, he served as a Cadet for the Talpiot Elite Israeli Defense Forces Program. From 2013 to 2015, he served for the Israeli Prime Minister Office as an RF and Antennas Engineer, from 2015 to 2018, he served as a Data Scientist and the Data Science Team Leader, and in the last two years of his military service, he served as the Head for the Cybersecurity Research and Development Group. After graduation, he worked as the Head of the 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. His research focuses on deep learning, computer networks, and cybersecurity.

Yuval Shavitt (Senior Member, IEEE) received the B.Sc. degree *(cum laude)* in computer engineering, 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 for the Israel Defense Forces first as a System Engineer and the last two years as the Software Engineering Team Leader. After graduation, he spent a year as a Post-Doctoral Fellow at the Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA. From 1997 to 2001, he was a member of Technical Staff at the Networking Research Laboratory at Bell Laboratories, Lucent Technologies, Holmdel, NJ, USA. Since October 2000, he has been a Faculty Member with the School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel. In 2014, he co-founded BGProtect Ltd., a company that monitors the internet for IP hijack attacks, where he serves as the CTO. His recent research focuses on internet measurements and deep learning solutions for various networking problems.

Prof. Shavitt served as a TPC member for many networking conferences. He served on the Executive Committee for INFOCOM 2000, 2002, and 2003. He was the TCP Co-Chair for TMA 2011 and a Keynote Speaker at PAM 2012. He was an Editor of *Computer Networks* from 2003 to 2004. He served as a Guest Editor for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and *JWWW*.