Detecting and Diagnosing Anomalies in Cellular Networks using Random Neural Networks

Pedro Casas*, Alessandro D'Alconzo*, Pierdomenico Fiadino†, Christian Callegari‡

*AIT Austrian Institute of Technology – email: {forename.surname}@ait.ac.at
†Eurecat Technology Centre of Catalonia – email: pierdomenico.fiadino@eurecat.org
‡CNIT and Dept. of Information Engineering, University of Pisa – email: christian.callegari@cnit.it

***Abstract*—Despite a large body of literature and methods devoted to the analysis of network traffic, the automatic detection and classification of network traffic anomalies still represents a major issue for network operators. The problem becomes even more challenging for cellular ISPs, both due to the ever growing number of connected devices and to the constant deployment of new applications and services prone to performance issues. In this paper we tackle this problem using Machine Learning (ML) approaches: in particular, we devise a system based on Neural Networks to unveil the relations between several monitored traffic features and network anomalies impacting a large number of customers in an operational cellular network. By training a model based on Random Neural Networks (RNN), we provide a fast and accurate anomaly detector and classifier, capable to pinpoint anomalies without assuming any specific traffic model or particular network behavior. The proposed solution is evaluated using synthetically generated data from an operational cellular ISP, drawn from real traffic statistics to resemble the real cellular network traffic. Our RNN model is capable to detect and classify different classes of anomalies with high accuracy and low false alarm rates, even when the volume of such anomalies is small.**

***Index Terms*—Anomaly Detection; Network Measurements; Machine Learning; Random Neural Networks; DNS Traffic; Cellular ISP.**

I. Introduction

During the last decade, a plethora of new, heterogeneous Internet-services have become highly popular and omnipresent, imposing new challenges to network operators. The complex provisioning systems used by these services induce continuous changes that impact both operators and customers. Indeed, efficient traffic engineering becomes a moving target for the operator [1], and management of Quality of Experience (QoE) gets more cumbersome, potentially affecting the end customer [2]. Furthermore, due to their traffic characteristics, applications that provide continuous online presence (e.g., messaging services) might severely impact the signaling plane of the network, especially in cellular networks [4]. In such a complex scenario, it is of vital importance to promptly detect and classify the occurrence of abrupt changes that could result in anomalies for some of the involved stakeholders.

In this paper we propose a simple yet effective approach to detect and classify network and traffic anomalies using supervised Machine Learning (ML) techniques. Supervised ML offers algorithms which can learn from and make predictions on data, building models from labeled input data to take data-driven decisions instead of static ones. ML techniques provide a promising alternative for detecting and classifying anomalies based on large sets of traffic descriptors or features. ML has been largely used in the field of automatic network traffic classification, and to a lesser extent also applied in the anomaly detection domain. The literature offers multiple types of ML-based classifiers, covering a very wide range of approaches and techniques [27]. We particularly propose a detection and classification system based on a new kind of neural network, introduced in recent years by E. Gelenbe [29]: the Random Neural Network (RNN). As it has been shown in many previous applications [31], RNNs are a very powerful tool to capture the intrinsic model behind the data. Using a standard three-layers neural network topology, we show that the proposed solution can automatically detected and diagnose (i.e., classify) different classes of network traffic anomalies with high accuracy and low false alarm rates, increasing the visibility and easing the daily management tasks of network operators, specially in current context where traffic complexity keeps growing.

While the approach we consider is not tied to a specific type of network and can be generalized to any kind of communication system, results presented in this paper consider the analysis of data captured in an operational cellular network, and therefore use some features exclusively available in cellular contexts. Anomalies observed in a cellular network are normally different from those observed in typical fixed-line networks [3], given that the type of end-user traffic is quite different due to the overwhelming usage of smartphone apps. Given that the types of anomalies we are interested in are mostly related to issues impacting the end customers of a cellular network, we particularly focus on the study of application-specific anomalies. Such anomalies refer to the occurrence of anomalies linked to popular applications (e.g., YouTube, Facebook, WhatsApp, etc.) and/or application providers (e.g., Google, Facebook, Apple Services, etc.). From our operational experience and our previous studies [3], application-specific anomalies are particularly visible in the DNS traffic of a network, as current apps and services distributed by omnipresent Content Delivery Networks (CDNs) extensively rely on a heavy usage of DNS for content access and location. As so, abrupt changes in the DNS queries count can be considered as a symptom of such anomalies [3]. Besides DNS query counts, our approach relies on the availability of related *meta-data*, which can also be observed in the analyzed cellular ISP. These meta-data may include information related to the end-device (e.g., device manufacturer, Operative System), the access network (e.g., Radio Access Technology – RAT, Access Point Name – APN, IP address of the DNS resolver), and the requested service (e.g., requested Fully Qualified Domain Name – FQDN).

The remainder of this paper is organized as follows: Sec. II briefly reviews the related work. Sec. III describes the proposed anomaly detector and classifier, and briefly overviews the underlying theoretical concepts behind the considered RNN model. Sec. IV presents the characterization of the cellular traffic and the generation of synthetic datasets used for evaluation purposes. In Sec. V we discuss the obtained results, considering both detection and classification of anomalies, additionally comparing the achieved performance to that of other ML-based systems proposed in the literature. Finally, Sec. VI concludes this work.

The research leading to these results has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "BigDAMA". Christian Callegari has received funding from the European Union under the FP7 Grant Agreement n. 607019, "SCOUT".

Table I
FEATURES USED IN THE ANALYSIS.

Field Name	Description
Manufacturer	Device manufacturer
OS	Device operating system
APN	Access Point Name
FQDN	Fully Qualified Domain Name of remote service
Error Flag	Status of the DNS transaction

II. RELATED WORK

There has been considerable amount of research about anomaly detection in network traffic. A large set of papers apply concepts and techniques imported from fields like Data Mining [15], Machine Learning [16], Self-Organizing Maps [12], Genetic Algorithms [13], Fuzzy Logic [14], etc. Focusing on statistical-based methods, most work rely on the analysis of scalar time-series, typically of total volume. They adopt various techniques like Wavelet Transform [9], [17], CUSUM [18] and others. It is commonly accepted that information-theoretic concepts, and in particular entropy measures, are well-suited for anomaly detection [6], [7]. Distribution-based approaches such as [8] are intrinsically more powerful, as they look at the entire distribution, rather than only at some specific mode or aggregation. A comprehensive survey on multiple anomaly detection techniques applied to different fields beyond network communications is available in [11].

In terms of ML-based approaches for classifying anomalies, the field of automatic traffic analysis and classification trough ML techniques has been extensively studied during the last half-decade. A standard non-exhaustive list of supervised ML-based approaches includes the use of Bayesian classifiers [19], linear discriminant analysis and k-nearest-neighbors [20], decision trees [23] and feature selection techniques [21], and support vector machines [22]. Unsupervised and semi-supervised learning techniques have also been used before for traffic analysis and classification, including the use of k-means, DBSCAN, and AutoClass clustering [24], sub-space clustering techniques [26], [28], and a combination of k-means and maximum-likelihood clusters labeling [25]. We refer the interested reader to [27] for a detailed survey on the different ML techniques applied to automatic traffic classification.

III. RNNS FOR ANOMALY DETECTION AND CLASSIFICATION

The RNN-based approach introduced in this work has its origins in the statistical learning field. The method uses a RNN to learn the relations between a set of n monitored traffic descriptors or features and the nature or class of the corresponding monitored traffic (i.e., normal operation or anomaly type). Our approach works in batch-mode, continuously analyzing and predicting the class of the traffic observed during a time bin of length t. Every t seconds, a set of n input traffic features $X_t = \{x_1(t), x_2(t), \ldots, x_n(t)\}$ is computed, and a label y_t is predicted, using a pre-trained RNN model. This RNN model basically provides a certain non-linear transfer-block $f_k(\cdot) : \mathbb{R}^n \rightarrow \mathbb{K}$, such that

$$y_t = f_k\left(X_t\right) = f_k\left(x_1(t), x_2(t), .., x_n(t)\right), \qquad (1)$$

where $\mathbb{K} = \{0, 1, 2, \ldots, i, \ldots, m-1\}$ defines the set of m predefined classes which the detector and classifier has been trained to assign (e.g., 0 for anomaly-free traffic, i for anomaly of type i).

In this paper we take as main traffic feature the total number of DNS requests issued within a time bin. As we said before, perturbations in this feature indicate that a device sub-population deviates from the usual DNS traffic patterns, thus pointing to potential anomalies. For the sake of better detecting and diagnosing the anomalies, we additionally take the distributions of DNS query counts across the fields described in Tab. I. From these distributions, we compute a set of features describing their shape and carried information, such as various percentiles and entropy values. Tab. II describes the specific set of 36 features, which are computed for every time bin. The set includes the number of observed DNS requests, as well as multiple percentiles of fields such as associated APN, device OS and manufacturer, requested FQDN and number of DNS error messages. We also take as input the average values of these fields, as well as their entropy, the latter reflecting the dispersion of the observed samples in the corresponding time bin. As we explain in Sec. IV, the training of the RNN model is done on top of synthetically generated datasets, which are by default labeled datasets. Given that the RNN model is relatively new within the networking community, we provide next an overview on the basic principles of RNNs, as well as some details on how we use it in our detection scheme.

Table II
INPUT FEATURES FOR THE RNN-BASED DETECTOR AND CLASSIFIER.

Field	Feature	Description
DNS_query	querycnt	total num of DNS requests
APN	apn_h	$H(\text{APN})$
	apn_avg	$\overline{\text{APN}}$
	apn_p{99,75,50,25,05}	percentiles
Error_flag	error_code_h	$H(\text{Error\_flag})$
	error_code_avg	$\overline{\text{Error\_flag}}$
	error_code_p{99,75,50,25,05}	percentiles
Manufacturer	manufacturer_h	$H(\text{Manufacturer})$
	manufacturer_avg	$\overline{\text{Manufacturer}}$
	manufacturer_p{99,75,50,25,05}	percentiles
OS	os_h	$H(\text{OS})$
	os_avg	$\overline{\text{OS}}$
	os_p{99,75,50,25,05}	percentiles
FQDN	req_fqdn_h	$H(\text{FQDN})$
	req_fqdn_avg	$\overline{\text{FQDN}}$
	req_fqdn_p{99,75,50,25,05}	percentiles

The RNN Model

The RNN model can be described as a merge between the classical Artificial Neural Network (ANN) model and queuing networks. RNNs are, as ANNs, composed of a set of interconnected neurons. Each neuron exchanges impulse signals with other neurons and with the environment, and has a potential associated with it, which is an integer random variable. The potential of neuron i at time t is denoted by $q_t(i)$. If the potential of neuron i is strictly positive, the neuron is *excited*; in this state, it randomly sends signals according to a Poisson process with rate r_i. In this model, neurons exchange *positive* and *negative* signals. The probability that a signal sent by neuron i goes to neuron j as a positive/negative signal is denoted by $p^+_{i,j}/p^-_{i,j}$. The signal leaves the network with probability d_i. When a neuron receives a positive signal, its potential is increased by 1; if it receives a negative signal or if it sends a signal, its potential decreases by 1. The lowest potential is 0. The flow of positive and negative signals arriving from the environment to neuron i is also a Poisson process of rate λ^+_i and λ^-_i respectively. Finally, instead of working with branching probabilities $p^+_{i,j}$ and $p^-_{i,j}$, we use the neural network *weights* $w^+_{i,j} = r_i p^+_{i,j}$ and $w^-_{i,j} = r_i p^-_{i,j}$, in analogy to standard ANNs. In this context, let us define ρ_i as the limit probability in which neuron i is excited, which corresponds to a strictly positive potential:

$$\rho_i = \lim_{t\rightarrow\infty} \Pr\left(q_t(i) > 0\right) \qquad (2)$$

Similar to the classical Jackson's result for open queuing networks, E. Gelenbe proved in [29] that this RNN model allows a simple system of equations with unique solution ρ_i, given the rates λ_i^+ and λ_i^- of incoming signals. In a traditional statistical learning application, a RNN with N interconnected neurons can be seen as a black-box, where the incoming signal rates $\boldsymbol{\lambda}^+ = \left(\lambda_1^+, \lambda_2^+, .., \lambda_N^+\right)$ and $\boldsymbol{\lambda}^- = \left(\lambda_1^-, \lambda_2^-, .., \lambda_N^-\right)$ are the inputs, and the probabilities of neuron excitement $\boldsymbol{\rho} = (\rho_1, \rho_2, .., \rho_N)$ are the outputs. As in most RNN applications, we shall consider that λ_i^- is 0 for every neuron. In this context, this black-box has a certain transfer-block $f(\cdot)$ that relates the N inputs with the N outputs:

$$\boldsymbol{\rho} = f\left(\boldsymbol{\lambda}^+\right) \tag{3}$$

where $f(\cdot)$ depends on the number of neurons N, the connection topology of the RNN, and the neural network weights $\mathbf{w} = \left\{w_{i,j}^+, w_{i,j}^-\right\}$. The weights $\mathbf{w}$ are thus the free parameters of the RNN model, which can be calibrated to build a non-linear transfer-block $f(\cdot)$ as the one we need. In general, some λ_i^+ in (3) are set to 0, and only a subset of $\boldsymbol{\rho}$ is used as output. In the proposed RNN-based detector and classifier (1), the block $f(\cdot)$ has n inputs and multiple outputs, one per output class (e.g., normal traffic and anomaly type). The n inputs correspond to the traffic features in Tab. II, whereas the outputs correspond to the labels $y \in \mathbb{K}$. Given that the outputs of the RNN ρ_i are probabilities, the set $\mathbb{K}$ is re-mapped to a binary set where each class is expressed as an m-dimensional binary vector, where all elements are 0 except from the one indicating the desired class, which is set to 1. As such, the neuron with the highest potential, i.e., the highest ρ_i, indicates the class i predicted by the model. The calibration of $f(\cdot)$ is performed by supervised learning, using a *learning dataset* composed of T input-output pairs $\{n_t, y_t\}$, $t = 1, .., T$. We do not provide the details of the learning algorithm in this paper, but we refer the interested reader to [30].

As in most applications of neural networks for learning purposes, we use a three-layers feed-forward network topology, which simplifies the RNN model and speeds-up computations. In such a topology, the set of N neurons is divided into three subsets: a set of I input neurons, a set of H hidden neurons, and a set of O output neurons. Input neurons receive positive signals from the environment and send signals to hidden neurons. Hidden neurons do no interact with the environment and only send signals to output neurons. Output neurons only send signals to the environment. The number of input neurons I is equal to n. The number of output neurons is $O = m$. The number of hidden neurons H is not a-priori fixed, and there is no foolproof method for setting it [32]. Too many degrees of freedom may cause over-fitting problems, and too few may reduce the *expressive* power to capture the underlying model. A convenient heuristic to choose H in an ANN is that the total number of weights is roughly $T/10$ [32] (recall that T is the size of the training dataset). The number of weights in a RNN is twice that of an ANN (considering the negative weights), and thus this relation reduces to $T/5$.

IV. Anomaly Modeling and Data Generation

We have evaluated different anomaly detectors for longer than six months in 2014 with DNS traffic from the operational cellular network of a nationwide European operator. While the extensive experimentation allowed us to collect results in a number of paradigmatic case-studies, the number of traffic anomalies observed in the corresponding period was relatively low, limiting as such the performance analysis of the proposed approach exclusively to those few real cases. In principle, one could resort to test traces obtained in a controlled environment (laboratory) or by simulations, but these approaches would miss the complexity and heterogeneity of the real traffic.

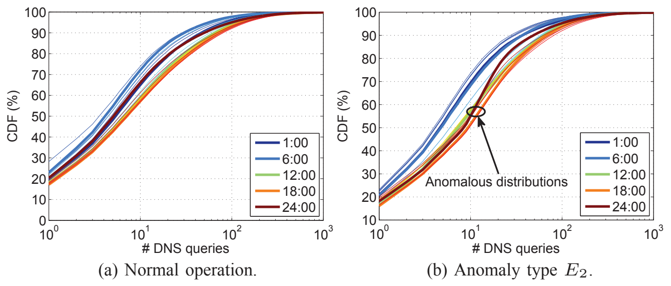


Figure 1. Hourly trend of the distribution of number of devices across query count over one day.

To bypass this hurdle, we adopted a methodology based on semi-synthetic data, derived from real traffic traces as suggested in [10]. Such an approach does not only allow to extensively analyze the performance of the framework with a large number of synthetic, yet statistically relevant anomalies, but also permits to protect the operator's business sensitive information, as neither real data traces nor real anomalies are exposed. To illustrate the procedure, next we explain how to generate semi-synthetic background DNS traffic, as well as how to replicate some the observed DNS-related anomalies.

A. Construction of Semi-synthetic Background Traffic

The procedure for constructing the semi-synthetic dataset is conceived with the objective of maintaining as much as possible the structural characteristics of the real, normal operation (i.e., anomaly-free) traffic, while eliminating possible (unknown) anomalies present in real traces. Exploring real traces, we observed that the traffic yields some fundamental temporal characteristics. In particular, the traffic is non-stationary due to time-of-day variations. This effect is not limited to the number of active devices, but rather applies to the entire distribution. Distribution variations depend on the change of the applications and terminals mix, which in turn induce modifications in the traffic patterns. Furthermore, we found that, besides a strong 24-hours seasonality, the traffic exhibits a weekly pseudo-cycle with marked differences between working days and weekends/festivities [5]. Finally, traffic remains pretty similar at the same time of day across days of the same type.

The first step of the construction procedure consists of manually labeling and removing possible anomalous events. However, as the complete ground truth is unknown in real traffic, we cannot completely rely on individual labeling of alarms. Therefore, we have to accept that minor anomalies may go undetected if their effect is comparable with purely random fluctuations. Then, the dataset is transformed to eliminate possible residual (unknown) anomalies present in the real traffic, while preserving the above mentioned structural characteristics. The transformation procedure is described as follows.

Let consider a real dataset spanning a measurement period of a few weeks, for a total of m consecutive one-day intervals (e.g., $m = 28$ in our case). Each one-day period starts and ends at 4:00 am local time: this is the time-of-day where the number of active devices reaches its minimum (considering a single time-zone). Denote by m_W and m_F the number of working and festivity (W- and F-) days, respectively, in the real dataset (e.g., $m_F = 8$ and $m_W = 20$), and by K the total number of 1-min time bins ($K = 28 \cdot 24 \cdot 60 = 40320$). For each device i consider the vector $\mathbf{d}_i \equiv \{c_i^{\tau_0}(k),\ k = 1, 2, \ldots, K\}$ at the minimum timescale ($\tau_0 = 1$ minute) across the whole real trace duration, where each element $c_i^{\tau_0}(k)$ is the list of the DNS measurements related to device i at time k. For those time bins where device i is inactive, the corresponding element in $\mathbf{d}_i$ is empty. We now divide this vector into m blocks, each one corresponding to a single one-day interval. Each block is classified as W- or F-block based on the calendar day. At this point we apply a random *scrambling* within the W class: each W-block element of $\mathbf{d}_i$ is randomly relocated at the same time position selected among all W-days. The same scrambling is applied independently to the F-blocks. In this way we obtain a new vector $\widetilde{\mathbf{d}}_i$ where the position of the blocks has been scrambled, separately for W- and F-blocks, but the time location and the F/W intervals have been maintained. Finally, from the set of scrambled vectors $\widetilde{\mathbf{d}}_i$ we can derive a new set of distributions for each time bin k and timescale τ, for all the considered traffic features.

The dataset obtained in this way retains certain characteristics of the real dataset, while others are eliminated. The most important change is that the random scrambling of the individual components $\mathbf{d}_i \to \widetilde{\mathbf{d}}_i$ results in the *homogenization* of the individual daily profiles — separately for W- and F-days. This eliminates any minor residual local anomaly that survived the manual labeling by spreading it out across all one-day intervals of the same F/W type. In other words, all W-days in the new dataset share the same (synthetic) aggregate daily profile. Same applies to F-days. Note however that the synthetic dataset retains the most important characteristics of the real process. In the first place, it keeps the time-of-day variations of the number of active devices. Secondly, the semi-synthetic dataset maintains the differentiation between the two classes of W- and F- days, although it eliminates any differentiation *within* each class (e.g., between Saturday and Sunday). Thirdly, it keeps the differentiation between distributions for different time-of-day. This is clear from Fig. 1(a), which shows the hourly Cumulative Distribution Functions (CDFs) of the number of devices across query count during one day of the semi-synthetic dataset. The result of the procedure is an anomaly-free DNS dataset *structurally similar* to the real trace.

B. Modeling and Generation of Synthetic Anomalies

During six months of experimentation we encountered a few recurring large-scale DNS traffic anomalies. Investigating these events we found some common traits and we conceived a procedure for reproducing them along with their most relevant characteristics. In particular, we identified two exemplary event types, E_1 and E_2 from now. In both cases, we model an outage of an Internet service for a specific sub-population of devices, which react by repeatedly and constantly issuing DNS queries to resolve the requested service throughout the anomaly. Involved devices are identified by fixing a specific OS (with its different versions). Moreover, we aim at modeling the correlation between the selected sub-population and the unreachable service. Therefore, we separately rank the 2nd-Level Domain (2LD) of the FQDNs for anomalous and background traffic, and select the most popular 2LD of the former that is not in the latter. As a simple example of such types of anomalies, we have observed events in which Apple devices running a specific version of iOS lost their persistent connectivity to certain servers providing the Apple push-notification service (which is the core of the remote notifications used in virtually every iOS App), resulting in a surge of DNS requests to locate new servers, and the resulting "scanning" of the complete IP address space of Apple push-notification service. Such an event was perceived by the cellular ISP as an internal sort of DDoS attack, as a large population of their own customer devices starting "bombarding" the network, starving resources at the access in some specific regions.

Event E_1*:* This type models the case of a short lived (i.e., hours) high intensity anomaly (e.g., 10% of devices repeating a request every few seconds), where all the involved devices are produced by a single manufacturer and run the same OS. In this case, the number of involved terminals and the overall number of additional queries is such to overload the local DNS servers. The latter effect is modeled by increasing the number of `time-out` codes in the Error Flag field.

Type	E_1	E_2	E_3
Start time t_1	9:00	13:00	18:00
Duration d	2h	1 day	1h
Involved devices D	10%	5%	3%
Back-off time	5 sec	180 sec	20 sec
Manufacturer	single popular	multiple	multiple
OS	single	single	multiple
Error flag	+5% timeout	—	—
FQDN	top-2LD	top-2LD	top-2LD

Table III
ANOMALOUS DNS TRAFFIC FEATURES FOR TYPES E_1, E_2, E_3.

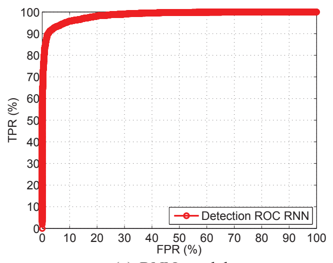
Event E_2*:* This type models a long lasting (i.e., days) low-intensity anomaly (e.g., 5% of devices repeating requests every few minutes). Differently from the previous case, the involved terminals are produced by multiple manufacturers, even if they share the same OS. Given the low-intensity, we did not introduce a modification in the distribution of the Error Flag. Fig. 1(b) shows the changes in the distribution of number of devices across query counts introduced by this event (wrt Fig. 1(a)). Note that although E_2 type anomalies are of relatively low intensity, their identification is important as, in our experience, they may lead to problems on the signaling plane.

Event E_3*:* We additionally introduce a third class of anomalies type E_3 which models a scenario in which all the customers of certain virtual operators (reflected by specific APNs) are affected by short lived service outages, responding with a surge in the number of DNS queries.

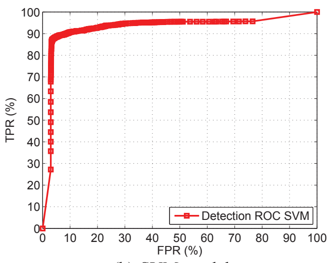
Tab. III summarizes the characteristics of the three event types and the actual values used for generating the anomalous dataset in the experiments discussed next. To illustrate the anomaly generation procedure, we consider an event of type E_1 of duration $d = 2h$, starting at $t_1 = 9:00$. Starting from t_1 at each time bin, $D = 10\%$ of all the active terminals are randomly extracted from the semi-synthetic background traffic, such that the OS is the selected one and the manufacturer is always the same. For each involved terminal, we generate one additional DNS query every 5 seconds, which are then added to the semi-synthetic dataset. The corresponding FQDN is randomly chosen among the domains in the 2LD identified as explained above. Finally, the Error Flag is changed to `time-out` in 5% of the overall DNS queries, so as to model the resolver overload. The last step consists of mangling both anomalous and background traffic. The procedure for generating types E_2 and E_3 is analogous, but differs in the selection of the specifically impacted features.

V. Anomaly Detection and Classification Performance

In this section we assess the proposed RNN-based approach, firstly by evaluating its detection performance, and then by comparing its classification performance against other ML-based approaches. For evaluation purposes and following the data generation approach described in Sec. IV, we construct a fully labeled dataset consisting of a full month of synthetically generated cellular network DNS measurements, reported with a time granularity of 5 minutes. The dataset contains normal operation traffic, with multiple instances of the aforementioned E_1, E_2 and E_3 anomalies, as specified in Tab. III. To perform a better evaluation, we introduce multiple instances of each anomaly type with a different fraction of the device population involved in the anomaly. In total we include 16 different variations of these anomalies, added on top of the 1-month anomaly-free traffic.



(a) RNN model.



(b) SVM model.

Figure 2. ROC curves for the detection of anomalies types E_1, E_2 and E_3. For each of the anomaly types E_1 and E_2 we include 7 anomaly variations, involving a number of devices going from 0.5% to 20% of the overall population (0.5%, 1%, 2%, 5%, 8%, 10%, 20%). For E_3 we include two different intensities, considering a population of 3% and 12%, which correspond to the actual size of virtual-operator customer populations, as observed from our real measurements. Each time bin is assigned a class, either normal - label 0, or anomalous - label 1, 2 or 3 for the three anomaly types respectively.

A. *Detection Performance*

Let us first get an initial picture of the detection capabilities of the RNN-based approach, by testing the detection and false alarm rates on the complete set of 16 variations of anomalies. To reduce biased results, the training and testing of the RNN model is performed by 10-fold cross validation, using a custom implementation of the RNN model in Java. Detection performance is evaluated in a time bin basis and not in an event basis: this means that there are 24 anomalous time slots for each anomaly variation of type E_1 (i.e., $7 \times 24 = 168$ time slots in total), $12 \times 24 = 288$ time slots for each variation of type E_2 (i.e., $7 \times 288 = 2016$ in total) and 12 time slots for each variation of type E_3 (i.e., $2 \times 12 = 24$ time slots in total). We follow such a direction as we are not only interested in detecting the occurrence of an event, but also its full span/duration. Note that anomaly classes are highly imbalanced, which in principle imposes major challenges in the training phase [32]. To counterbalance this problem, we resort to a standard over-sampling approach, in which we add copies of instances from the under-represented classes, in this case anomalies of types E_1 and E_3, to perform the training.

Fig. 2 depicts the Receiver Operating Characteristic (ROC) curves obtained for the detection of the complete set of anomalies. Fig. 2(a) provides the results obtained for the RNN model, whereas Fig. 2(b) shows the comparative results obtained for a Support Vector Machine (SVM) model. We selected this approach for comparison based on the a-priori good performance shown by the application of SVMs in previous work on anomaly detection [11] and traffic classification [27]. The RNN model can correctly detect more than 90% of the anomaly instances with different intensities with a false alarm rate below 1%, but it is not capable to properly detect part of the slowest intensity anomalies, as we show next. The SVM model achieves slightly worse detection performance, resulting in a similar true positives rate but a false alarm rate above 3%.

Fig. 3 splits the results obtained for the RNN model by anomaly type and by intensity level. Whereas Fig. 3(a) and Fig. 3(c) show that anomalies of type E_1 and E_3 are detected with almost no false alarms for all the evaluated intensity levels, Fig. 3(b) shows that the lowest intensity anomalies of type E_2, i.e., those affecting only 1% and 0.5% of the devices population, are partially missed by the RNN-based detector. Nevertheless, we can claim that detection performance is very high, even in the case of low intensity anomalies, starting at 2% for all the evaluated anomaly types, and even as low as 0.5% for anomalies of type E_1.

B. *Classification Performance vs ML Approaches*

We move on now to the evaluation of the anomaly classification capabilities of the RNN model. We additionally evaluate other ML-based classifiers typically used in the literature, for comparison purposes. In particular, we consider the following classifiers: : Support Vector Machines (SVM), C4.5 Decision Trees (C4.5), Naive Bayes (NB), and Locally-Weighted-based Learning (LWL). We use the well-known Weka Machine-Learning software tool[1] to calibrate these ML-based algorithms and to perform the evaluations. We address the interested reader to the survey [27] and to the Weka documentation for additional information on the different configuration parameters of each algorithm.

To evaluate and compare the performance and virtues of the classification models, we consider three standard metrics: Global Accuracy GA, Recall and Precision. GA indicates the percentage of correctly classified instances (time bins) among the total number of instances. Recall R_i is the number of instances from class $i = 0, \ldots, 3$ correctly classified (TP_i), divided by the number of instances in class i (n_i). Precision P_i is the percentage of instances correctly classified as belonging to class i among all the instances classified as belonging to class i, including true and false positives (FP_i). Recall and precision are two widely used performance metrics in classification. Precision permits to measure the fidelity of the classification model regarding each particular class, whereas recall measures the per-class accuracy.

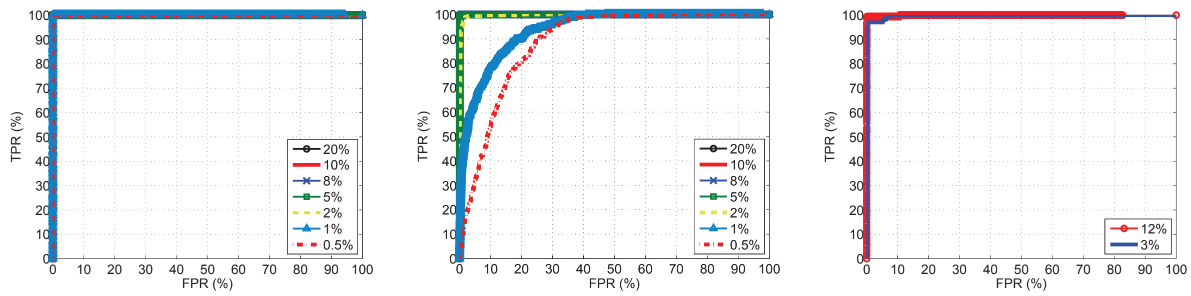
$$\mathrm{R}_i = \frac{TP_i}{n_i}, \quad \mathrm{P}_i = \frac{TP_i}{TP_i + FP_i}, \quad \mathrm{GA} = \frac{\sum_{i=1}^{M} TP_i}{n} \tag{4}$$

Fig. 4 reports the performance of the five compared classifiers in the classification of all the 5-minutes time bins. To limit biased results, all the evaluations presented use 10-fold cross-validation. Reported results refer to optimal parameter settings, after thorough testing. According to Fig. 4(a), RNN, SVM and C4.5 achieve high overall classification accuracy, above 90% in the three cases, and with a slightly better performance for the RNN model. The NB and LWL models achieve worse results, clearly suggesting that the underlying hypotheses of both models do not hold in this case. In terms of precision and recall, depicted in Figs. 4(b) and 4(c) respectively, the RNN model outperforms the other classifiers in all the classes, evidencing the nice properties introduced by such a model. Still, as already evidenced in the results presented in Fig. 3(b), the RNN model can not correctly deal with all the intensities of the E_2 anomalies, resulting in worse performance for this specific class. The SVM model achieves high precision for anomalies of type E_2, but results in a very poor accuracy for this specific type of anomalies, with a recall as low as 57%. Due to page-length limitations we do not include the confusion matrix for each classifier, but the main source of under-performance for the SVM model in this case comes from completely missing the anomalies, and not by misclassifying them into another anomaly class. All in all, results clearly suggest that the proposed RNN-based anomaly detection and classification approach is highly accurate and can diagnose the studied anomalies properly. As future work, we shall dig deeper into feature selection approaches to improve the performance of the model with the lowest intensity anomalies of type E_2.

VI. Conclusions

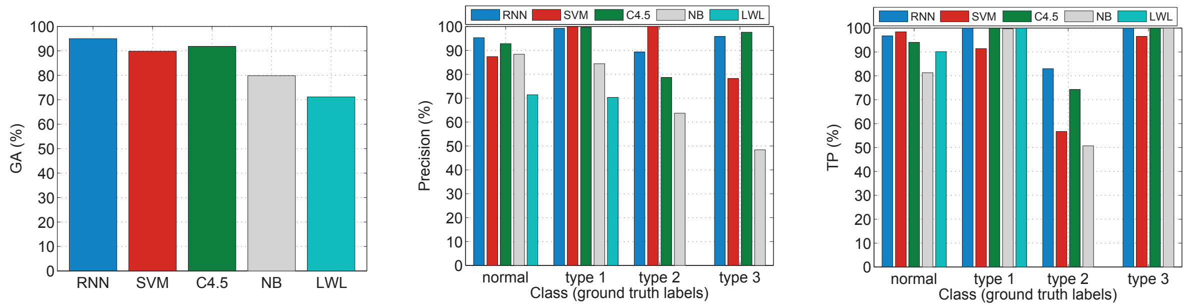
In this paper we have presented a RNN-based approach for detection and classification of large scale Internet anomalies based on the analysis of passively captured network data. The approach is

[1]Weka Data Mining, at http://www.cs.waikato.ac.nz/ml/weka/.



(a) Anomaly type E_1. (b) Anomaly type E_2. (c) Anomaly type E_3.

Figure 3. ROC curves for the detection of anomalies, split by anomaly types and different intensity levels. Whereas anomalies of type E_1 and E_3 are detected with almost no false alarms, the low intensity anomalies of type E_2 are partially missed by the RNN-based detector.



(a) Global Accuracy. (b) Precision. (c) Recall.

Figure 4. Classification Accuracy, True Positives Rate, Precision, and Recall for normal operation instances and different anomaly type events. The performance of the RNN model is almost perfect for normal traffic and anomalies of type 1 and 3, but quality significantly drops for the anomaly type 2.

based on a new breed of neural networks, which has shown high performance in many other applications besides anomaly detection and classification. We believe that ML-based approaches can provide high insights and visibility for daily network operations, specially in current context where traffic complexity keeps growing. Given the general lack of large-scale ground-truth datasets to test the performance of systems like ours, we developed an approach to generate semi-synthetic data, derived from real traffic traces. We believe that this is also an important contribution, as it would help the owners of real data to make such datasets available for the research community without disclosing any privacy or business sensitive information. As future work, we are conducting Out-of-Sample (OSS) tests to verify the generalization and applicability of the trained model in a real setup, by assessing its performance with other classes of anomalies not available in the training set.

References

[1] P. Fiadino et al., "Characterizing Web Services Provisioning via CDNs: The Case of Facebook", in *TRAC*, 2014.
[2] P. Casas et al., "When YouTube doesn't Work – Analysis of QoE-relevant Degradation in Google CDN Traffic", *IEEE TNSM*, vol. 11 (4), 2014.
[3] M. Schiavone et al., "Diagnosing Device-Specific Anomalies in Cellular Networks", in *ACM CoNEXT Student Workshop*, 2014.
[4] A. Aucinas et al., "Staying Online While Mobile: The Hidden Costs", in *CoNEXT*'13.
[5] P. Romirer-Maierhofer et al., "Device-specific Traffic Characterization for Root Cause Analysis in Cellular Networks", in *TMA*, 2015.
[6] G. Nychis et al., "An Empirical Evaluation of Entropy-based Traffic Anomaly Detection", in *ACM IMC*, 2008.
[7] B. Tellenbach et al., "Beyond Shannon: Characterizing Internet Traffic with Generalized Entropy Metrics", in *PAM*, 2009.
[8] A. D'Alconzo et al., "Distribution-based Anomaly Detection in 3G Mobile Networks: from Theory to Practice", *Int. Journal of Net. Mng.*, vol. 20 (5), 2010.
[9] A. Dainotti et al., "A Cascade Architecture for DoS Attacks Detection Based on the Wavelet Transform", *J. Comput. Secur.*, vol. 17 (6), 2009.
[10] H. Ringberg et al., "The need for simulation in evaluating anomaly detectors", in *ACM CCR*, vol. 38 (1), 2008.
[11] V. Chandola et al., "Anomaly Detection: a Survey", in *Com. Sur.*, vol. 41 (3), 2009.
[12] A. Mitrokotsa et al., "Detecting denial of service attacks using emergent self-organizing maps", in *IEEE ISSPIT*, 2005.
[13] M. Ostaszewski et al., "A non-self space approach to network anomaly detection", in *IEEE IPDPS*, 2006.
[14] W. Chimphlee, et al., "Integrating genetic algorithms and fuzzy C-means for anomaly detection", in *IEEE Indicon*, 2005.
[15] G.Prashanth, et al., "Using random forests for network-based anomaly detection", in *IEEE ICSCN*, 2008.
[16] Y. Li et al., "An efficient network anomaly detection scheme based on TCM-KNN algorithm and data reduction mechanism", in *IAW*, 2007.
[17] M. Raimondo et al., "A peaks over threshold model for change-point detection by wavelets", *Statistica Sinica*, vol. 14, 2004.
[18] P. Lee, et al., "On the Detection of Signaling DoS Attacks on 3G Wireless Networks", in *IEEE INFOCOM*, 2007.
[19] A. Moore et al., "Internet Traffic Classification using Bayesian Analysis Techniques", in *Proc. ACM SIGMETICS*, 2005.
[20] M. Roughan et al., "Class-of-Service Mapping for QoS: a Statistical Signature-Based Approach to IP Traffic Classification", in *IMW*, 2004.
[21] N. Williams el al., "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification", in *CCR*'06.
[22] S. Valenti et al., "Accurate, Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets", in *TMA*, 2009.
[23] V. Carela-Español et al., "K-Dimensional Trees for Continuous Traffic Classification", in *TMA*, 2010.
[24] J. Erman et al., "Traffic Classification using Clustering Algorithms", in *MineNet*'06.
[25] J. Erman et al., "Semi-Supervised Network Traffic Classification", in *Sigmetrics*'07.
[26] P. Casas et al., "MINETRAC: Mining Flows for Unsupervised Analysis & Semi-Supervised Classification", in *ITC*, 2011.
[27] T. Nguyen et al., "A Survey of Techniques for Internet Traffic Classification using Machine Learning", in *IEEE Comm, Surv. & Tut.*, vol. 10 (4), 2008.
[28] P. Casas et al., "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge", in *Com. Comm.*, vol. 35 (7), 2011.
[29] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution", in *Neural Computation*, v. 1, pp. 502-511, 1989.
[30] E. Gelenbe, "Learning in the Recurrent Random Neural Networks", in *Neural Computation*, vol. 5, 1993.
[31] H. Bakircioglu et al, "Survey of Random Neural Network Applications", in *E.J.O.R.*, vol. 126 (2), 2000.
[32] C. Bishop, "Neural Networks for Pattern Recognition", *Oxford Uni. Press*, 1995.