A Deep Learning Framework for BGP Anomaly Detection and Classification

Paulo César da Rocha Fonseca

Instituto de Computação

Universidade Federal do Amazonas

Manaus - AM March 2020

A Deep Learning Framework for BGP Anomaly Detection and Classification

Paulo César da Rocha Fonseca

Advisor: Prof. Edjard Souza Mota, Ph.D.

Instituto de Computação

Universidade Federal do Amazonas

A thesis submitted to the Post-graduate Program in Informatics of the Institute of Computing of the Federal University of Amazonas in partial fulfillment of requirements for the degree of *Doctor of Science*.

Manaus - AM February 2020

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Fonseca, Paulo César da Rocha
F676d A Deep Learning Framework for BGP Anomaly Detection and Classification / Paulo César da Rocha Fonseca. 2020
117 f.: il. color; 31 cm.

Orientador: Edjard Souza Mota
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Machine Learning. 2. Border Gateway Protocol. 3. Anomaly Detection. 4. Deep Learning. I. Mota, Edjard Souza II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"A Deep Learning Framework for BGP Anomaly Detection and Classification"

PAULO CÉSAR DA ROCHA FONSECA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Edjard de Souza Mota - PRESIDENTE

Prof. Eduardo Luzeiro Feitosa - MEMBRO INTERNO

Prof. André Luiz da Costa Carvalho - MEMBRO EXTERNO

Prof. José Neuman de Souza - MEMBRO EXTERNO

Prof. Ítalo Fernando Scotá Cunha - MEMBRO EXTERNO

Manaus, 18 de Novembro de 2019

Agradecimentos

Primeiramente, agradeço a minha família por ter me apoiado em todos os sentidos por esses longos anos. Em especial à minha mãe e ao meu pai, por serem tão excepcionais no que fazem, por terem me ensinado através do exemplo. Nominalmente, agradeço ao meu irmão, Jonas Caio, às minhas avós, Francisca e Francisca, às minhas tias, Maria Auxiliadora e Maria Cleonice. Estendo esse agradecimento a todos os membros da minha família, que sempre desejam o melhor para mim.

Às pessoas que estiveram comigo esses anos, com as quais compartilhei tanto deste processo e compartilharam palavras de incentivo comigo. Em especial à Nani, por estar comigo em todos os momentos, sentindo tanto, me ajudando tanto e me ensinando a ser uma pessoa melhor. Nominalmente, também agradeço ao Leandro Machado, à Caroline Silva, ao Emory Raphael, à Ana Paula, ao Jonathan Byron, ao Bruno Dias, ao Flávio Montenegro, ao Cleumir Souza, ao Rômulo Araújo, ao Bruno Mandell, ao Fidel Graça.

À Raissa, que me ajudou a chegar ao fim, sendo caprichosa, cuidadosa e atenciosa. Pela paciência e compreensão, pela aceitação e apoio incondicional.

Ao meu orientador, Edjard Mota, pela mentoria acadêmica e em outros inúmeros aspectos da vida. Sou muito grato por todas lições transmitidas nos últimos 11 anos, desde quando a primeira proposta de PIBIC estava relacionada com programação orientada a aspectos, até a proposta do presente trabalho. Sou grato pela sua visão, que me levou a trabalhar com redes neurais anos antes da sua popularização, e pela sua rigorosidade, que me levou a publicar em veículos científicos de alta relevância. Estendo o agradecimento ao Alexandre Passito, por sempre ter feito contribuições valiosas ao trabalho desenvolvido desde a graduação, sendo parte importante na escrita e apresentação do artigo apresentado no ISCC 2019 com

resultados parciais deste trabalho. Também agradeço ao Ricardo Bennesby, pelo apoio, conversas e amizade nos últimos 11 anos.

Aos professores presentes em minha qualificação e defesa de doutorado, prof. Dr. Italo Cunha, prof. Dr. Neuman Souza, prof. Dr. André Carvalho e prof. Dr. Eduardo Feitosa pelos comentários valorosos que foram essenciais para o versão final da tese.

À FAPEAM, pela bolsa de estudo que me auxiliou financeiramente durante o desenvolvimento deste trabalho.

Por fim, sou grato a todas as dificuldades que tive nesse caminho, pois me fizeram ser a pessoa que chegou até aqui.

Table of contents

List of figures

List of tables

Abstract

The Border Gateway Protocol (BGP) is the default Internet routing protocol that manages connectivity among Autonomous Systems (ASes). Although BGP disruptions are rare when they occur the consequences can be very damaging. Consequently, there has been a considerable effort aimed at understanding what is normal and abnormal BGP traffic and, in so doing, enable potentially disruptive anomalous traffic to be identified quickly. Even though there is an extensive research on anomaly detection, there are two major gaps in current literature: the scarcity of public datasets for all types of events and the lack of a BGP anomaly classification framework that differentiates anomaly classes. Since that there are no public datasets of labeled BGP anomalous events, each model was validated using different datasets, which had to be individually generated for each approach. The absence of common groundwork dataset increases the difficulty in comparing different approaches. The lack of a classification framework hinders the deployment of specific mitigation measures to each anomaly class in an automated fashion. In the current work, we address both problems: 1) We provide a BGP dataset generation tool and publicly available datasets for different anomaly classes. These datasets contain the most used features by previous research efforts and additional novel features; 2) We address the BGP anomaly classification problem by developing a framework that uses deep learning as the core engine of an anomaly detection and classification mechanism. We built a model that exploits different neural network architectures advantages. Both novel features and the BGP anomaly detector and classifier were evaluated and it was demonstrated that they can be used to react to anomalies in real-time and leverage the deployment of different mitigation and coordination strategies to different anomaly classes in an autonomous fashion.

Chapter 1

Introduction

The Internet is a structural part of the lives of almost two billion users worldwide and critical to many services, such as financial transactions, business operations, and many other applications [22, 132]. The Internet is also highly dynamic, extremely heterogeneous, has a planetary-scale, and a largely opaque ecosystem of networks [25].

The *de facto* Internet routing protocol is the BGP [126], which is responsible for the exchange of reachability information among computers and routers of different domains. A number of works [129, 98, 10] explored the possibilities that a logically centralized control plane brings to accomplish BGP's purpose of furthering the business goals of an organization in providing its routing information to other organizations.

There has been a considerable effort aimed at understanding what is normal and abnormal BGP behavior and enabling potentially disruptive anomalous traffic to be identified quickly. Many efforts with the goal of detecting BGP anomalies were proposed throughout recent years [4], varying with respect to the used approach, level of automation achievable, type of detected anomaly, and many other factors. In the current work, we provide a BGP dataset generation tool, publicly available datasets for different anomaly classes and a framework that uses machine learning as the core engine of an anomaly detection and classification mechanism. This mechanism can be used to react to anomalies in real-time and leverage the deployment of different mitigation and coordination strategies to different anomaly classes in an autonomous fashion.

This chapter is organized as follows. In Section 1.1, we discuss the challenges of detecting BGP anomalies and introduce our definitions of unstable BGP traffic and anomalous BGP traffic. In Section 1.2, we enunciate the thesis statement of this work. In section 1.3, we formulate the research questions that guided the current work. In Section 1.4, we list the objectives that were derived from our research questions. In Section 1.5, we list the contributions produced throughout the development of this work. Finally, we present the organization of this thesis in Section 1.6.

1.1 Motivation

Along with the increasing necessity of ubiquitous Internet connectivity, it is possible to observe the growing complexity of management of networked environments due to a number of factors (e.g. the need to configure a large number of devices, the inherent complexity of policies themselves [11]). Nonetheless, simplified management is a key enabler to realize current network applications' requirements.

Among proposals that aimed to simplify network management and facilitate innovation, software-defined networking (SDN) has emerged as one of most important networking paradigms in the recent years [43]. SDN aims to simplify management and provide flexibility to the network through a clear separation of *control plane* from *data plane*. The control plane is composed by high-level functionality, such as network policies and intrusion detection, whereas the data plane comprises low-level forwarding capabilities at network-device-level. In SDN, the control plane is moved to a logically centralized controller which provides a northbound interface, allowing the construction of high-level network management applications, and a southbound interface, which manages network devices through a vendor-independent protocol.

SDN was initially aimed for enterprise networks [108], however, many approaches that integrate and leverage current Internet infrastructure with SDN [129, 61] were proposed. Such solutions have demonstrated that bringing SDN to inter-domain routing can improve the convergence performance of routing mechanisms [81], offer new routing capabilities [61],

or lay the groundwork for new services and markets [82, 53]. Lin et al. [98] proposed the SDN-IP architecture, depicted in Figure 1.1, which integrates the SDN control plane with BGP speakers in order to access external legacy networks. In such architecture, BGP policies may be implemented as high-level intents in the control plane, which translates them to BGP announcements and withdrawals.

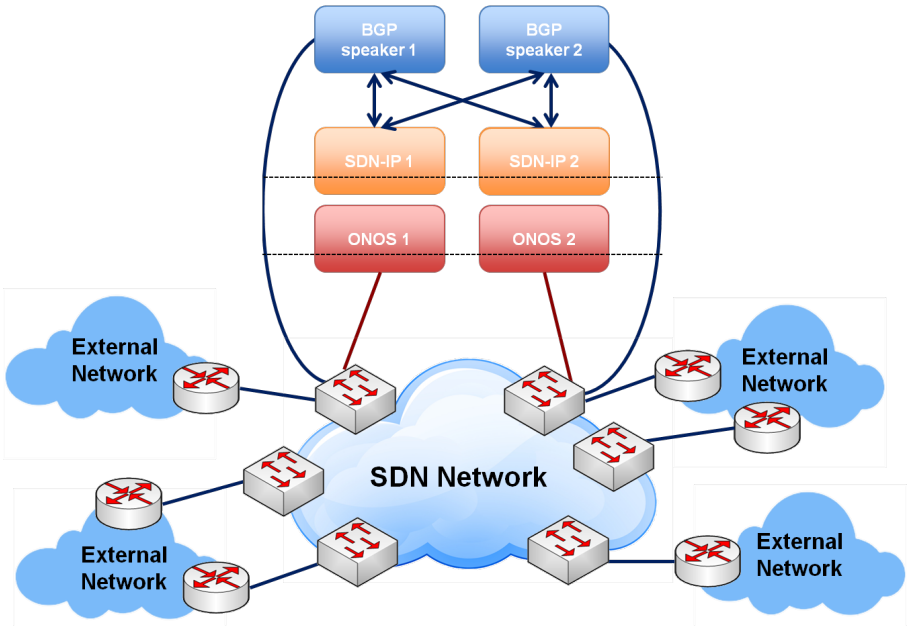


Fig. 1.1 Integration of SDN and BGP

Any BGP activity that does not contribute to the business goals or undermines them can be considered anomalous. Unfortunately, it can be very difficult to determine whether a particular activity is furthering those goals. For example, BGP updates that do not reflect underlying topology changes may be a legitimate policy changes or they might be the consequence of route flapping, where the same path is repeatedly announced and quickly withdrawn [37].

A number of studies [86, 93, 44, 142] showed that many of these events are short-lived and do not disrupt Internet connectivity. However, larger scale events may be able to cause Internet outages and instabilities that affect quality of service for hours [28]. These large scale events may have a number of causes: (i) natural disasters, such as earthquakes and hurricanes; (ii) software attacks, such as worms or prefix hijacking; (iii) physical attacks, such as military

attacks, terrorism, or electromagnetic pulse attacks; (iv) accidental misconfiguration; (v) certain forms of censorship such as the one adopted during the Arab spring; (vi) software bugs (e.g., in the routers); (vii) network equipment hardware failures, and many other factors.

An autonomous system with a logically centralized control plane provides as an advantage the complete view of the routing information of multiple BGP routers and the possibility of mitigation strategies that affect traffic of specific applications, allowing network management to achieve a greater level of automation. Feamster and Rexford [42] argue that networking research must move towards the realization of self-driving networks [80], empowered by machine-learning-based models. In these networks, applications must be developed using high-level policy goals and a holistic view of the underlying components. For example, anomaly classification and detection mechanisms must be able to make real-time, closed-loop decisions, rather than only detect anomalies through offline analysis of network traces.

One of the key enablers of self-driving networks is deep learning [89], a trend that has gained a lot of attention in recent years. Numerous efforts showed its effectiveness to tackle a wide range of problems [67, 134, 157]. In the last decade, networking research has applied neural networks to areas such as traffic prediction [95, 122], congestion control [40, 62] and fault management [84, 114]. During our research, we identified that SDN and deep learning methods have the potential to overcome the limitations of previous approaches, automating the process of detection and classification through systematic extraction of BGP features with a high accuracy rate, and supporting models that are able to capture complex behavior and differentiate anomaly classes.

A major challenge in applying deep learning methods is to obtain training data that is representative of the real distribution and in large volume, in order to train models that generalize better for future occurrences and fit to complex behavior. There is a number of public BGP data sources [128][130] that contain control plane information from anomalous events, however the raw BGP data is not suited for machine learning methods. Pre-processing of raw data includes the definition and extraction of relevant features and transformation into tabular data, which is used as input to machine learning models.

As previously discussed, there is an extensive research on BGP anomaly detection [4], however, we identified that a major gap in current literature is that there is a scarcity of public datasets of labeled BGP anomalous events, thus, each model was validated using different datasets, which had to be individually generated for each approach. The absence of common groundwork dataset increases the difficulty in comparing different approaches.

Another gap is the lack of a classification framework that systematically differentiates between different types of anomalies, allowing specific mitigation measures to each anomaly class. Few efforts [138, 88] towards this goal are specialized to identify only one type of anomaly.

Besides, most efforts are based on approaches that depend mostly on human interaction in order to either analyze the suspicious identified behaviors (*e.g.* time series analysis methods) or to fine tune the parameters of a model or define a "magic number" that defines the threshold to detect anomalies (*e.g.* statistical pattern recognition), which hinders the possibility of network automation without network operator intervention.

This work discusses which features may be extracted from BGP control plane messages in order to detect and classify anomalies, as well as proposes novel features that yield better detection results than features available in the literature. I constructed datasets of BGP anomaly events by extracting these features and made them publicly available [45]. Then, I developed a deep learning framework for detection and classification of BGP anomalies that achieved good performance in a significant subset of such datasets.

1.2 Thesis Statement

Considering the presented motivation, our thesis statement is as follows:

Deep learning approaches are able to effectively detect and classify large scale anomalous events in BGP traffic, overcoming current limitations that lead to lack of generalization, in order to make closed-loop decisions.

1.3 Research Questions

We enumerate the research questions of investigation in this thesis as follows:

RQ1 Which features can be extracted from BGP traffic that better represent the difference between anomalous and normal behavior?

RQ2 Which features can be extracted from BGP traffic that better represent the difference between different types of anomaly?

RQ3 Which deep learning architecture can capture the relevant behavior in order to effectively *detect* and *classify* a wide range of anomalies?

1.4 Objectives

The main objective of this work is to *develop a framework that detects and classifies anomalous behavior in the BGP traffic in a timely fashion through the use of deep learning models, enabling the application of different mitigation techniques suited to each anomaly class*. The specific objectives of this work are as follow:

1. Identify which features can be used to detect and classify anomalous behavior
2. Develop a mechanism that extract time series of features from BGP messages
3. Label anomaly datasets of different classes through the use of public data

1.5 Contributions

The main contributions of this work are as follows:

1. Novel features that capture the relationship of complementary messages and allows easier detection and classification of anomalous behavior.
2. Publicly available BGP feature extraction tool [117] that allows easy integration of new network features.

3. Publicly available datasets of network features [45] during BGP anomalies of different classes, which can be used to train customized machine learning models.

4. Novel deep learning framework capable of detecting a wide range of BGP anomalies.

5. Classification mechanism capable of categorizing different types of anomalies.

Publications:

1. Due to the denial of access to data centers to collect data about failures the subject of this research had to be changed from fault management in software-defined networking to the current line of work. During this former phase, we developed a comprehensive survey [30] which identified main gaps in current research and guided our future steps.

2. A paper based on the developed dataset generation tool, along with the generated datasets and trend analysis. This paper [46] was presented at the IEEE Symposium on Computers and Communications 2019 and it is the base material of Chapter 4.

3. Final results of the current work will be submitted to the *Journal of Networks and Computer Applications*.

1.6 Thesis Organization

Chapter 2 presents a general overview on the main topics of this work, elaborating concepts that will be used throughout this thesis, namely: Border Gateway Protocol, BGP anomalies, Machine Learning and Software defined networking.

Chapter 3 provides a summarized review on anomaly detection and classification efforts, which are categorized in four main areas: Time series analysis, comparison with historical data, statistical pattern recognition and machine learning. We also identify the main gaps in current research, such as, lack of anomaly classification and high difficulty to reproduce and compare approaches.

Chapter 4 discusses which features may be extracted from BGP messages in order to detect and classify anomalies, as well as proposes novel features. It also presents a

framework for BGP anomaly dataset generation and generated datasets, both of which are publicly available.

Chapter 5 discusses which properties are present in BGP message dynamics and which neural network models are capable of capturing such properties. Then, we present our proposed deep neural network architectures for anomaly detection and classification. Finally, we propose a SDN architecture with a anomaly detection and classification modules. This chapter also presents the evaluation of the proposed method and describes advantages of different metrics and which are adequate to the BGP anomaly detection problem. Then, we elaborate on how the machine learning models were trained, which hyperparameters were used and the achieved results.

Chapter 6 provides a discussion on how our BGP dataset generation tool can be used as a benchmark for future anomaly detection and classification methods. It also discuss the deep learning model results and how it can be improved. On future directions, we also discuss how the current work may leverage self-driving networks and be improved by explainable AI.

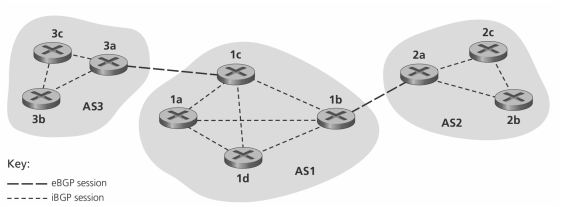
Chapter 2

Background

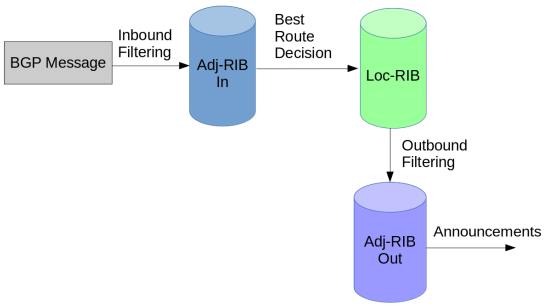
This chapter presents a general overview on the main topics and concepts that will be used throughout this work. Section 2.1 provides an overview of Border Gateway Protocol functioning, describing the organization of the Routing Information Base, types of BGP messages and its respective attributes. Section 2.2 describes different types and sub-types of BGP anomalous behavior, varying with respect to granularity, size, root causes and effects. Section 2.3 describes the events which originated our datasets. Section 2.4 provides an overview of the Software-Defined Networking architecture and an explanation of OpenFlow [108], the SDN *de facto* protocol. Section 2.5 introduces the core concept of machine learning approaches and deep learning, focusing on neural networks architectures that were used in this work.

2.1 Border Gateway Protocol

The Internet is a global network composed by tens of thousands of interconnected administrative domains, called autonomous systems (ASes). An AS comprises a collection of routers, which are responsible for the distribution of a set of IP routing prefixes. These routers exchange information through two classes of routing protocols, Internal Gateway Protocol (IGP), responsible for communication inside a single AS, and Exterior Gateway Protocol



(a) BGP architecture [85]



(b) BGP speaker internal tables

Fig. 2.1 BGP speaker internal organization and inter-AS architecture

(EGP), which allows routers from different ASes to exchange data. IGP is used to propagate the best paths inside an AS, whereas EGP computes the best routes across different ASes.

The Border Gateway Protocol (BGP) [126] is the *de-facto* Internet EGP, responsible for the maintenance of updated information among routers about selected paths to any given routing prefix. Each BGP router (hereafter interchangeably called *BGP speaker*) keeps its own *Routing Information Base* (RIB), which consists of three tables: *Adj-RIB-In*, *Adj-RIB-Out* and *Loc-RIB*. The Adj-RIB-In stores unedited routing information sent by neighboring routers, Loc-RIB keeps the best routes available, derived from the routes stored in Adj-RIB-In and internal routing policies, and Adj-RIB-Out stores the routes that will be sent to neighboring routers. Figure 2.1 depicts an example of intra and inter-domain topology.

BGP routers exchange four types of messages: OPEN, KEEPALIVE, UPDATE and NOTIFICATION. BGP speakers at the borders of two adjacents ASes establish a peering session by sending an OPEN message containing AS number and IP. The session is maintained as long as BGP speakers reply to KEEPALIVE messages and NOTIFICATION messages, used in case of error or graceful disconnection, are not sent. At first, the routers communicate their full routing table (Adj-RIB-Out). Then, upon modifications, an UPDATE message is sent to announce a new route, withdraw a route that was advertised previously, or update an existing route with new parameters. These parameters include the AS path to a prefix, the next hop, prefix aggregation information and other extensions [17] [8] [7]. Figure 2.2 exemplifies the propagation of a BGP update message.

The main UPDATE attributes are AS-PATH, ORIGIN, LOCAL-PREF, AGGREGATOR and MULTI EXIT DISCRIMINATOR (MED).

- AS-PATH is a list of ASes that must be traversed to reach a prefix. Before a BGP speaker propagates a path to its peers, it appends its own AS number to the AS-PATH attribute.
- ORIGIN refers to whether the message was originated from IGP, EGP or another routing protocol (*e.g.* static route).
- LOCAL-PREF is an integer value that represents a preference of a network operator for a route and can be used to express business policies (*e.g.* prioritize routes from peers over providers).
- A BGP speaker can aggregate multiple prefixes in a single update message, in this case, the AGGREGATOR attribute contains information about the BGP speaker that aggregated the route.
- In case there are multiple entry points to an AS, the MED attribute indicates a metric that is used by the external decision process to choose the route with lower MED value.

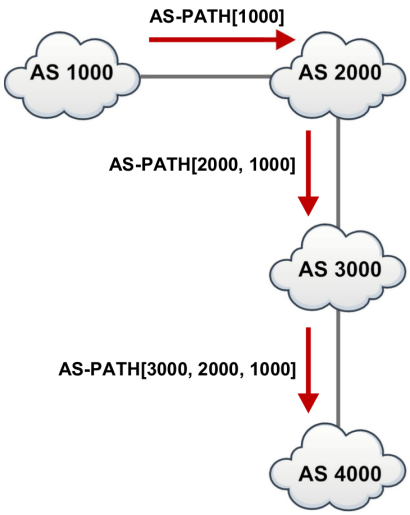


Fig. 2.2 BGP update message propagation example [4]

Considering these attributes, the BGP speaker selects the best advertised route. Table 2.1 summarizes the order of criteria used during the BGP path selection process.

Table 2.1 BGP path selection process

Priority	Attribute
1	Highest LOCAL-PREF value
2	Lowest AS-PATH length
3	Lowest Origin Type
4	Lowest MED value
5	EBGP learned over IBGP learned
6	Lowest IGP cost
7	Lowest Router ID

2.2 BGP Anomalies

Ideally, during its normal operation, BGP would only be used by ASes in order to exchange information that represented real changes in inter-domain infrastructure and enforce policies of network administrators. Therefore, as long as both of them remained the same, BGP traffic volume should remain low. However, a number of studies [93, 86] on BGP behavior through the years showed that BGP traffic has plenty of dynamics, for instance:

- **Forwarding instabilities**: Transmission of information due to *forwarding plane changes*. For example, an AS outage forces its peers to send a backup route to their respective peers.
- **Policy fluctuations**: Transmission of information due to *policy changes*. For example, an AS announces the same AS-path to reach a given prefix but with different policy attributes, e.g. MED, communities, localpref.

- **Pathological updates**: Transmission of redundant or inconsistent information due to network device malfunctioning, misconfiguration, or other pathological behavior. Identify the root cause of pathological updates may be hard, or even not possible.

Also, there are some assumptions about BGP operation, such as destination-based routing and valley-free routing, however, studies showed that the following anomalous states are observable in the Internet:

- **Violations of destination-based routing**: Ideally, a router forwards all packets with the same destination address to the same next hop. However, Flach et al. [44] found that, from a set of approximately 40k routers probed, 29% of them forward traffic going to a single destination via different next hops, and in some cases, even to different ASes.

- **Detoured routes**: The BGP specification defines that, in the absence of any policy that enforces the contrary, the route selection process must choose the shortest-path. However, ASes' policies and configuration errors may lead to non-optimal routes being selected, even though there exists other policy-compliants routes available [102]. In an early study [142], Tangmunarunkit et al. showed that about 20% of Internet paths are inflated by 50%

- **Violations of valley-free routing**: AS-paths are assumed to be *valley-free*, i.e. considering *provider-to-customer* as downward communication, siblings and peers at the same level, once a route goes downwards, it never goes up again. However, Giotsas et al. [55] inferred through the COMMUNITY attribute of BGP messages that, from the set of routers analyzed, 4% and 22% of IPv4 and IPv6 routes are valley paths, respectively.

Large scale anomalous behavior refers to a set of BGP updates that represent a significant deviation from the normal BGP dynamics distribution, that affects multiple ASes for a determined period of time, after which the BGP returns to its normal previous state. Al-Musawi et al. [4] classifies BGP anomalies in four main categories: direct intended, direct unintended, indirect and network failures.

- **Direct intended**: The most common type of direct intended anomaly are BGP hijacking attacks. This anomaly occurs when an attacker claims to own a prefix or sub-prefix that belongs to another AS in order to redirect routes from the AS to the attacker, as illustrated in figure 2.3. The attacker may have the goal to intercept the traffic to the AS victim, or black hole and discard traffic. This category classifies further considering the following events.

  Prefix hijack: A BGP router announces the exact prefix already announced by another AS. This is likely to cause the attacker's neighbors to adopt this new route, thus raising a Multiple Origin AS conflict (MOAS). There are many cases in which MOAS conflict are caused by non-malicious BGP operation, making it more difficult to detect this kind of hijacking.

  Prefix and AS hijack: In order to avoid a MOAS conflict, the attacker announces the targeted prefix and a (fake) direct connection between its AS and the victim AS. This may cause the peers to adopts the fake route and redirect traffic to the hijacker.

  Sub-prefix hijack: Exploiting the fact that BGP prefers more specific addresses over more general addresses, an attacker may announce sub-prefix(es) of prefix(es) announced by the victim AS. If the announced sub-prefix is not advertised by other legitimate ASes, the attacker's announcement can be propagated globally.

  Sub-prefix and AS hijack: Similar to prefix and AS hijacking, in this type of hijacking the attacker announces a fake path between its AS and the victim AS and further increases the difficult of detection.

  Hijacking a legitimate path: The attacker manipulates received updates by ommiting some intermediate ASes from the $\mathrm{AS\text{-}path}$, in order to increase the likelihood of being chosen as the best route.

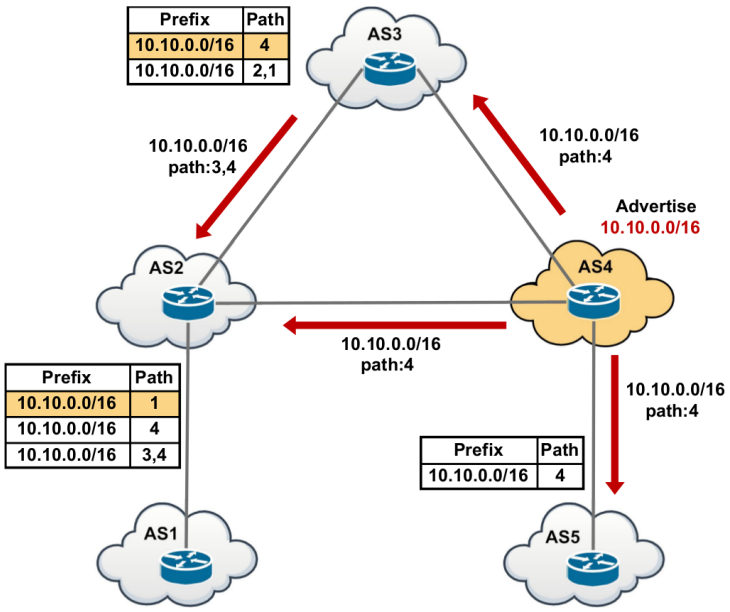


Fig. 2.3 Example of a direct intended anomaly (prefix hijacking) [4]

- **Direct unintended**: Due to misconfigurations, BGP routers might announce anomalous prefixes and/or sub-prefixes, either because they are already being announced by legitimate ASes or because they are private and/or unused. Even though the effect of these anomalies are similar to hijacking, these events *usually* are short-lived, since all parties (AS origin and affected peers) are interested in interrupting the malfunctioning as soon as possible.

  Origin misconfiguration: An origin misconfiguration is the unintentional insertion of a route into the global BGP tables.

  Export misconfiguration: An export misconfiguration is an inadvertent export of a route to a BGP peer in violation of the exporter's policy.

- **Indirect anomalies**: There are events which are not related to BGP or Internet routing directly but might greatly affect BGP dynamics and AS reachability. The most common cause of an indirect anomaly are attacks that generate a high volume of traffic, thus causing the congestion and unresponsiveness of AS routers. Failure to respond to KEEPALIVE messages for short periods will cause flapping routes. In order to

alleviate the impact of this problem, most providers yield higher priority to BGP traffic. Two common types of indirect anomalies are:

Worm events: Malicious software which goal is to infect as many computers as possible, propagating itself through the network. Depending on the severity of an worm attack, massive traffic may be generated leading to critical instability, such as experienced during Nimda [104], Code Red II [112] and Slammer [111] attacks. Current implementations of BGP protocol prevent an overload by a massive amount of data traffic through the separation of BGP control traffic from data traffic.

(Distributed) Denial-of-Service: Traffic incoming from DDoS attacks may also lead to overload of network edge routers and their subsequent unresponsiveness. Another type of overload occurs when DDoS attackers aim an address of a network device of an IXP. Usually, these addresses are not globally reachable but it may be case if a misconfiguration happens. This was observed in the Spamhaus DDos attack [145, 73]. The latter example is an instance of *direct intended anomaly*.

- **Network failures**: The Internet is composed by many types of network entities and devices, ranging from links and routers to ASes and operators, each one of them may fail and affect the Internet in a different way [2].

  Link and router outages: The impact of a link/router outage depends on whether they are *single-points of failure* for a given prefix or AS [101]. If they are not, the underlying routing protocols will handle the failure and a BGP UPDATE message with the same $\mathrm{AS\_PATH}$ and a different $\mathrm{NEXT\_HOP}$ will be sent to its neighbors.

  AS outage: The impact of an AS outage is proportional to how many ASes use the AS in its preferred route to a set of prefixes and the size of such set. Even the outage of a relatively small AS for a short period of time may lead to a high volume of BGP update messages and have a lasting effect in BGP convergence.

  Operator-level: Often, an operator administers multiple sibling ASes sharing the same infrastructure [14], leading to a high correlation during outages related to these infrastructures.

Peering infrastructure outage: The trend towards the flattening of the Internet have increased the number of ASes that relies on peering infrastructures, such as *IXPs* and *colocation facilities*. A recent study [54] showed that outages involving theses infrastructures may have unexpected consequences in ASes at diverse locations, due to interconnection strategies such as remote peering.

The table 2.2 summarizes the characteristics of anomaly categories.

Table 2.2 BGP anomalous events

Event	**Type**	**Direct**	**Intended**	**Malicious**
Prefix hijack	Hijacking	Yes	Yes	Yes
Prefix and AS hijack	Hijacking	Yes	Yes	Yes
Sub-prefix hijack	Hijacking	Yes	Yes	Yes
Sub-prefix and AS hijack	Hijacking	Yes	Yes	Yes
Hijacking a legitimate path	Hijacking	Yes	Yes	Yes
Origin misconfiguration	Misconfiguration	Yes	No	No
Export misconfiguration	Misconfiguration	Yes	No	No
Prefix outage	Outage	Yes	No	No
Link and router outage	Outage	Yes	No	No
AS outage	Outage	Yes	No	No
Operator outage	Outage	Yes	No	No
Peering infrastructure outage	Outage	Yes	No	No
Worm events	Attack	No	Yes	Yes
(Distributed) Denial-of-Service	Attack	Possible	Yes	Yes
Violations of destination-based routing	Property violation	Yes	Yes	No
Detoured routes	Property violation	Yes	Yes	No
Violations of valley-free routing	Property violation	Yes	Yes	No

2.3 BGP Anomaly Events

Next, we present the events that we used to build our datasets.

2.3.1 Direct

AS9121 Routing Table Leak

On December 24, 2004, the AS9121 announced to new paths through BGP sessions that were used to reach almost 70% of all prefixes, which amounted to 106k prefixes [121]. This misconfiguration resulted in a loss for tens of thousands of networks, which traffic was either lost or diverted. AS9121 started to announce the leaked prefixes to its neighbors around 9:20 GMT, and the event lasted until just after 10:00 GMT. AS9121 continued announcing the prefixes for the rest of the day and reached a second peak of announcement rate at 19:47 UTC.

AWS Leak

The AWS Route Leak started at 17:10 UTC on April 22, 2016 and resulted in loss of traffic and connectivity for a number of networks with high traffic prefixes, such as Google, Amazon, and Twitter. The event occurred due to maintenance issues on Innofield AG (AS 200759) that is connected to Swiss Internet eXchange (SwissIX). During maintenance reactivation of BGP sessions, AS 200759 distributed prefixes belonging to Amazon as belonging to private AS 65021. Prefix announcements were propagated through AS 6939 Hurricane Electric (HE) that peers at SwissIX.

Malaysian Telecom Leak

The Malaysian Telecom (AS 4788) leaked one third of all IP prefixes in the global routing table to the backbone provider Level3 (AS 3549), starting on June 12, 2015 at 8:43 UTC and lasted until 11:45 UTC. The event was triggered by routers misconfiguration at Telecom Malaysia. Level3 (AS 3549) propagated traffic from its peers and customers via Telecom Malaysia, which was overwhelmed by the massive volume of data, resulting in major packet loss and performance degradation.

Indosat Telecom Leak

On April 2, 2014, AS4761 (Indosat) leaked around 320,000 routes, which happened during scheduled maintenance starting at 18:25 UTC. Due to a misconfiguration, BGP was redistributed with bad upstream filtering which caused Indosat originate prefixes that were not assigned to it. Several hundreds of those prefixes were widely accepted, and services of some networks such as Akamai, a leading content delivery network (CDN) and cloud service provider, were disrupted.

2.3.2 Indirect

Indirect anomalies studied have a similar context: malicious worm propagating themselves through the network cause a large surge in Internet traffic. The following episodes were analysed and collected.

Code Red II

On 19 July, the Code Red II worm started its spreading across the global network. Since the beginning of the anomaly it was observed an exponentially growing eight-fold increase in the BGP advertisement rate, over a period of about eight hours. This surge faded over the same time scale as it arrived.

Nimda

On Tuesday, 18 September, as the Nimda worm started its propagation, it was observed another BGP storm. Over a period of roughly two hours, starting at about 13:00 UTC, the rrc00 collector perceived BGP advertisement rates exponentially ramped up by a factor of 25, from 400 per minute to 10,000 per minute, with sustained spikes to more than 200,000 per minute. The advertisement rate then decayed gradually over several days, reaching regular levels by September 24.

Slammer

The Slammer worm was released on Jan 25th, 2003 5:31 UTC, infected at least 75,000 hosts in just over 30 minutes and it was reported that a number of critical AS-AS peering links were operating above critical load thresholds during the attack period. Besides, anomalous behavior was observed in specific ASes, as with two small edges ASes, which announced less than 0.25% of BGP routing table entries, but, during the worm attack, contributed to over 6% of total update messages observed at monitoring points.

2.3.3 Outages

Japan Earthquake

The 2011 earthquake off the Pacific coast of Tōhoku was a magnitude 9.0–9.1 undersea megathrust earthquake off the coast of Japan that occurred at 05:46 UTC on Friday 11 of March, 2011. Fiber optic network of undersea cables that connect Japan to the rest of the world was damaged when the earthquake struck beneath the Pacific seafloor. As a result, the Internet relying on this underlying infrastructure to deliver data traffic was affected by this disaster [99]. Many telecommunication operators reported service disruptions and performance degradations in the communications with either Japan or other places around the world.

Moscow Blackout

The Moscow Power Blackout occurred on May 25, 2005 and lasted several hours. Moscow Internet exchange was shut down during the power outage. Routing instabilities were observed due to loss in connectivity of some ISPs peering at this exchange. This effect was apparent at the RIS remote route collector in Vienna (rrc05) through a surge in announcement messages arriving from peer AS 12793.

2.4 Software defined networks

In traditional networks, control plane and data plane are tightly coupled in network devices. Conversely, SDN keeps only data forwarding functionality in network devices, whereas it delegates control plane functionality to a physically separate layer, composed of one or more network entities called *controllers*, which provide the interface between the high-level network applications and the network devices. As depicted in Figure 2.4 [50], SDN architecture comprises three layers (in the course of this work, we use the terms *layer* and *plane* interchangeably):

- *Infrastructure layer*: Also known as the *data plane*, it is responsible for data forwarding and statistics storage; it may include physical and virtual switches. Network devices must comply with a standard interface (e.g. OpenFlow protocol [49]) that is used by the control plane to manage the devices.

- *Control layer*: The control layer consists of one or more SDN controllers. The main function of the control layer is to maintain a *logically centralized network view* that allows network applications reason about network properties and behavior. The control layer observes network state through the open interface with the devices, and provides an *application programming interface* (API) to construct network applications using high-level terms (e.g. host names instead of IP addresses).

- *Application layer*: Uses the API provided by the control layer to implement network business applications (e.g. firewall, load balancer), enforcing networking policies and requirements.

Even though first proposals of control and data plane separation are not recent [146, 154], discussions about its implications and real deployments have not gained traction until about the past few years [137, 135, 19, 148] with the proposal and the subsequent large adoption of the OpenFlow protocol [109], a southbound protocol that provides a communication interface which can be used to configure switches' forwarding policies. The OpenFlow matching process is depicted in Figure 2.5: When a switch receives a packet, (1) it checks if the

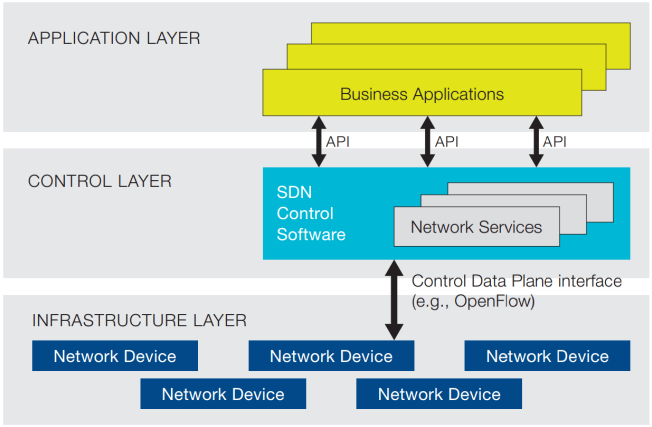


Fig. 2.4 Software defined networking architecture [50].

packet's header fields matches with one or more rules; (2) If no match occurs, the packet is forwarded to the network controller, (3) which will handle the packet and generate a new flow rule; (4) otherwise, if there is a matching rule, (5) a list of actions is applied to the packet. Figure 2.6 lists which header fields are available for matching in OpenFlow 1.1.0. Actions that can be applied to flows include basic forwarding, packet modification, and QoS support. The set of possible actions and fields that can be used for rule matching increases with newer specifications of OpenFlow protocol. Although there are alternatives to OpenFlow, such as ForCES [36] and OpFlex [140], most of SDN solutions are implemented using OpenFlow, including those related to fault management.

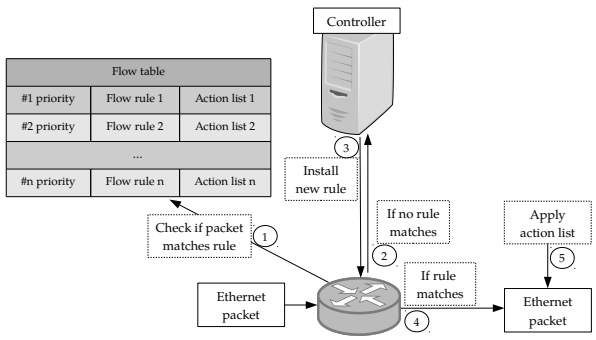


Fig. 2.5 Overview of OpenFlow rule-matching process

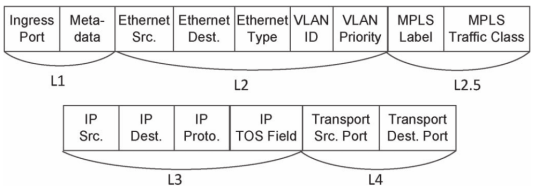


Fig. 2.6 Match fields of a flow table entry in OpenFlow 1.1.0 [72]

SDN offers three main features to network administrators: *centralized management*, *flexibility*, and *programmability*. Centralized control simplifies the management because it allows the construction of control logic from a complete view of the network. A designated controller to which all network devices must report their states decreases convergence time and coordination complexity. The southbound API gives flexibility to network operators by allowing them to separately handle each type of traffic, according to network needs. Since network devices from different vendors must agree on a standardized specification, network operators can build network applications on a well-defined common ground. At last, programmability is achieved through a northbound API that provides a set of abstractions. Network services and business applications are programmed using these high-level abstractions, which may implement entities from network resources to business policies and more. Also, programmability allows network applications to change its behavior at run-time, reacting to network events.

These features open possibilities for addressing network requirements and challenges in novel ways. A complete view of the network facilitates routing decisions, traffic engineering, and fault localization because it delegates the handling of partial information and communication asynchrony to the network controller. Centralized management also allows network applications to detect complex network behavior, such as periodic traffic patterns and distributed denial-of-service attacks [13]. The ability to handle traffic using fine-grained criteria helps traffic slicing and management of multi-tenant environments [136]. Taking advantage of programmability, it is much more simple to manage virtual networks using high-level terms and abstracting the details of the underlying infrastructure. Also,

applications that require frequent reconfigurations, such as big data processing [149], can take advantage of SDN's ability to program the network at runtime.

To further discussion, a number of works provide detailed investigation on SDN architecture [152, 115, 72], on how SDN may help the realization of current network requirements [71, 94, 29, 153, 125].

2.5 Machine Learning

Given a specific task and input data, a machine learning method seeks to improve its performance on the task based on knowledge extracted from its previous experiences with similar data [110]. There is a number of different machine learning algorithms and techniques that can be used to tackle tasks that differ with respect to many factors, including the type of available data, expected output and computational constraints. There are three main classes of learning paradigms:

- **Supervised learning** algorithms learn from labeled data that includes the expected output (*e.g.*, class in classification problems, value in regression problems) and its performance is evaluated based on how well it can approximate to the expected value.
- **Unsupervised learning** methods extract commonalities from unlabeled data in order to identify an underlying structure or distribution. The goal is to devise the correct placement of new data amongst the known data, for example, clustering of user personas according to navigation patterns.

Machine learning methods can be used to solve many kinds of tasks. The most common examples are listed below [56]:

Classification: The model must specify which of k categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : Rn \rightarrow 1,...,k$. When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y. For example, in an object recognition[83],

where the input is an image and the output is a numeric code identifying the object in the image.

Classification with missing inputs: In some cases, there is no guarantee that every measurement in the input vector will always be provided. When some of the inputs may be missing, the learning algorithm must learn a set of functions rather than providing a single classification function. Each function corresponds to classifying x with a different subset of its inputs missing. This challenge is common in medical diagnosis, because many types of medical tests are expensive or invasive. For instance, Goodfellow et al. [57] proposed an robust method to deal with missing data.

Regression: The task is to predict a numerical value given some input, through an output function $f : Rn \rightarrow R$. This type of task is similar to classification, except that the format of output is different. An example of a regression task is to predict the selling price of a house given features such as location, number of rooms and built area.

Transcription: The task is to observe an unstructured representation of some type of data and transcribe it into textual form. For instance, in character recognition, the model is fed with a photograph containing an image of text and returns this text in the form of a character sequence. Google Street View uses this approach in order to process address numbers in this way [58].

Machine translation: The task is to convert an input that already consists of a sequence of symbols in some language into a sequence of symbols in another language. Deep learning has recently begun to have an important impact on this kind of task [103]

Anomaly detection: The algorithm process a sequence of events or objects, and flags some of them as being anomalous. Examples of anomaly detection tasks include credit card fraud detection and distributed Denial-of-Service attack detection [16];

Most machine learning algorithms perform some kind of optimization, aiming to minimize the output value of a cost function $f(x)$ by altering x. The most popular optimization method in machine learning is the gradient descent technique [15].

Given a function $y = f(x)$, the derivative $f(x)$ gives the slope of $f(x)$ at the point x, which can be used to specify how to scale a change in the input that gives a corresponding change in the output: $f(x+\varepsilon) \approx f(x) + \varepsilon f'(x)$. Thus, the derivative can assist the minimizing of a function because it indicates how to change x in order to make a small improvement in y. Figure 2.7 depicts an example of this technique. The gradient descent technique exploits this property of derivatives in order to find the model that better fits to the training data.

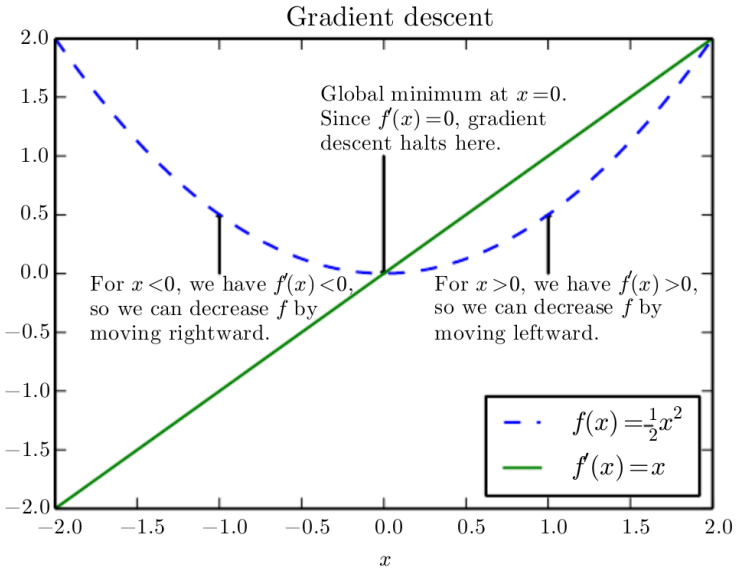


Fig. 2.7 A depiction of how the derivatives of a function can be used to minimize its output [56].

Two central concepts in machine learning are bias and variance. Bias is the dichotomy between the average prediction of a model and the target value. When a model has high bias, it means that it is not learning the representative behavior from the training data and oversimplifies the model. It always leads to high error on training and test data. Variance is the variability of model prediction for a given data point or a value. A model with high variance learns the behavior of the training data flawlessly but does not generalize on the data which it has not seen before. Such models perform very well on training data but has high error rates on test data. When a model have a high bias, it means that is suffering from underfitting, whereas high variance means overfitting. Figure 2.5 depicts the trade-off between bias and variance and Figure 2.5 shows how the model complexity usually affects

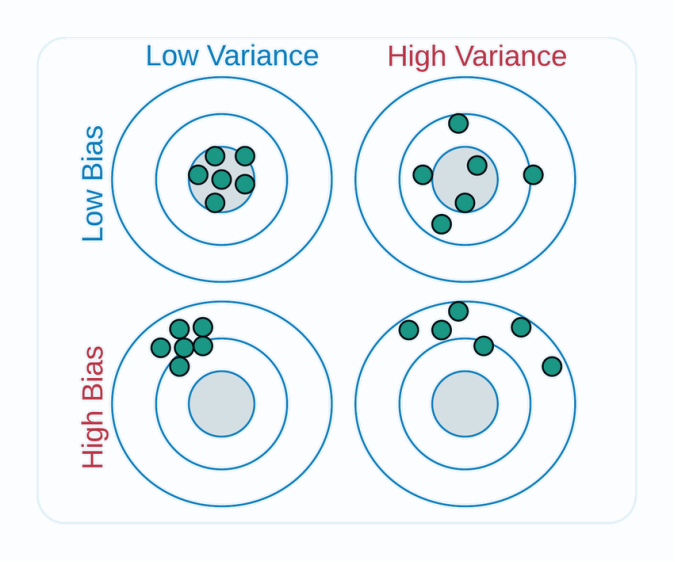


Fig. 2.8 Visualization of bias and variance trade-off using a bulls-eye diagram [59]

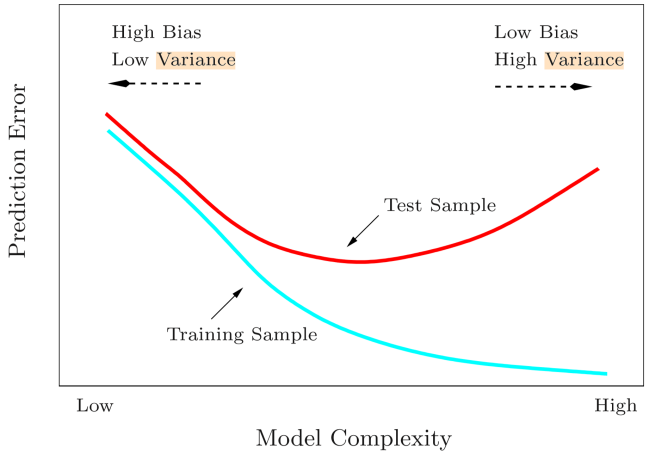


Fig. 2.9 Model complexity is proportional to the chances of overfitting [51]

bias and variance. The main goal during model tuning is to find the *sweet spot* that balances bias and variance.

2.5.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) [147] is a well-known machine learning approach widely used for classification and time-series prediction [133]. SVMs are learning systems of machines trained with a Mathematical Organization algorithm which implements a limit

derived from the Statistical Learning Theory. SVM is noteworthy for its solid theoretical basis and it can reach a high performance in practical applications. A SVM model projects the input data into a linear space and tries to find a hyperplane which best separates a set of sample vectors with the minimum error and maximum distance between the vectors (opposite classes). Figure 2.10 depicts different separating hyperplanes.

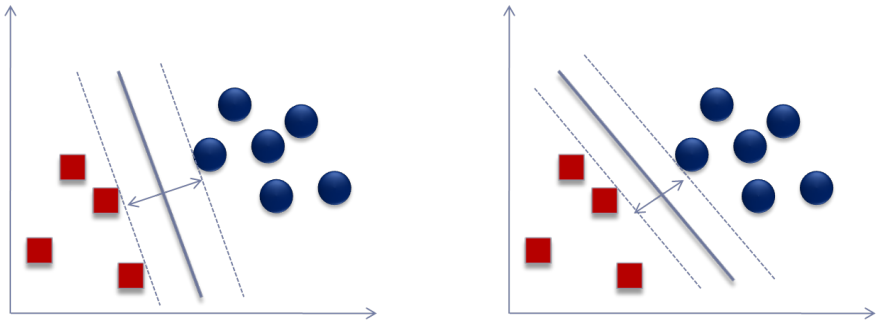


Fig. 2.10 Examples of separating hyperplanes

An advantage of SVM is that most of the data is ignored, because the relevant data is only the points that end up exactly on the margin of the optimal classifier and these are often a very small fraction of the sample dataset. Initially proposed to linearly separable problems, SVM can be used for problems not separable in a 2D space through the use of a kernel function. First, the input data is mapped to a higher dimensional space which have a separating hyperplane, then a kernel function maps the higher dimensional data to a linear space and finds a separating hyperplane with maximal margin. The kernel function of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ may be of four basic types:

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$;
- Polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$;
- Radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2), \gamma > 0$;
- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$;

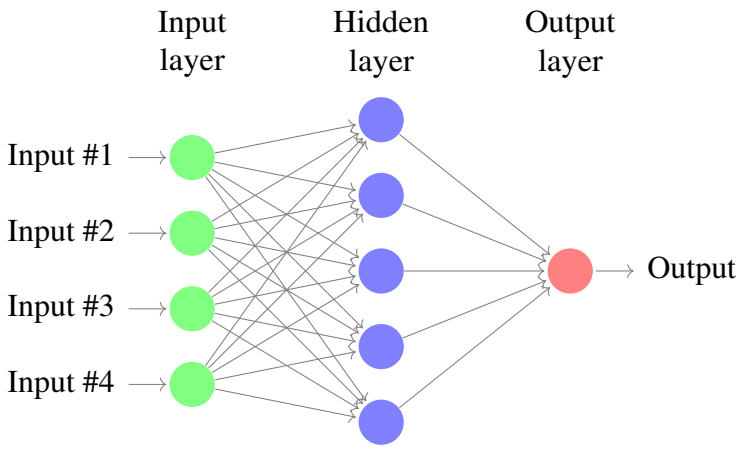
The γ hyperparameter is used to control the radius of influence that a single training example reaches. Another important hyperparameter for SVM is the penalty parameter

$\mathbf{C} > 0$. It adjusts how much a hyperplane must be corrected in order to correctly classify an incorrect prediction. Both parameters are related to the bias-variance trade-off: A value of C or γ too large might lead to overfitting, whereas a small value might cause underfitting.

2.5.2 Neural Networks and Deep learning

Neural networks have gained a lot of attention in recent years and its variations encompass all three learning paradigms. A neural network is composed by weighted interconnections of data structures called *neurons*, as depicted in Figure 2.11. Neurons receive an input signal, apply a function on its value and, then, output a transformed value. The output value of a neuron is forwarded to adjacent neurons through connections that transform the value by multiplying it by a adjustable parameter. As depicted in Figure 2.11, a neural network is composed by three types of layers: an *input layer*, which receives sample data; *hidden layer(s)*, which may comprise multiple layers of interconnected neurons that are not visible outside of the model; and an *output layer*, which presents the result of the neural network processing. When this process occurs from the input layer to the hidden layers to the output layer, without any cycles, the network is a *feedforward neural network*.

Fig. 2.11 Neural network example



The connection weights of a neural network are randomly initialized and during the training they are adjusted in order to generate a network that better fits the training data. Neural networks architectures vary with respect to their internal organization, how the

neurons interconnect and transform the values, and has proven to be an effective approach to model a wide range of problems [67, 134, 157]. A neural network architecture might be designed to capture a specific behavior. For instance, neurons might be endowed with memory capabilities in order to capture temporal properties of input data. In the last decade, networking research has applied neural networks to areas such as traffic prediction [95, 122], congestion control [40, 62] and fault management [84, 114].

Most of deep learning methods relies on a variation of gradient descent called stochastic gradient descent (SGD) [12]. SGD was designed to work well with large datasets. In this approach, only a subset of the whole training dataset is used, allowing that a model receives a training set with billions of examples and only uses hundred of examples. Besides SGD, there are other gradient descent variants [144, 38] that are used to adjust neural network parameters in order to minimize the cost function. Besides the model parameters that are adjusted through an optimizer model, it is also necessary to tune the neural network hyperparameters, which include the number of neurons of each hidden layer, number of hidden layers, optimization approach, and optimizer learning rate. The fine tuning of hyperparameters requires the empirical experimentation of different values.

Next, we describe neural network architectures that are relevant to our work:

Wavelet networks:

Whereas, most of neural networks use the sigmoid function as the activation function to be applied to the input of a neuron, wavelet networks (WN) [159] use wavelet transforms. This type of network might be used as a signal denoising layer [160]. In WN, the learning process searches for the optimal weights for the input values, as well as for the scaling and translation parameters of the wavelet function. Figure 2.12 depicts how a wavelet neural network may adjust its parameters in order to extract a denoised version ofthe input signal.

Convolutional networks:

Convolutional neural networks (CNN) [90] are composed by a conjunction of convolutional layers and pooling layers. *Convolutional layers* are divided into local units that are connected

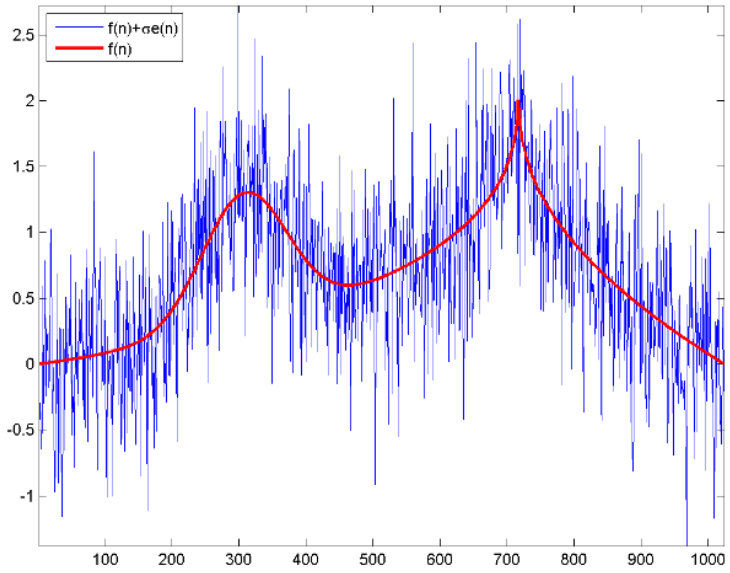


Fig. 2.12 Wavelet neural network as a denoising layer. The blue line represents a noisy input signal and the red line is the attenuated output signal. [106]

only to a set of neurons in the previous layer, through a set of weights called kernel. This organization allows these local units to learn behaviors that are spatially correlated in the signal. *Pooling layers* merge the values of the local units of a convolutional layer through a pooling function (*e.g.* average and max pooling [91]). Pooling reduces the dimension of the representation and creates an invariance to small shifts and distortions. Figure 2.13 depicts a CNN architecture example.

Although convolutional neural networks (CNNs) are mainly used for the processing of 2-dimensional data (e.g. images) it is possible to use it for the classification of 1-dimensional time series [41]. The kernel moves in one direction from the beginning of a time series towards its end, performing convolution and pooling operations explained in Section 5.1, instead of moving to the left, right and down as it does when the usual 2D convolution is applied to images. When a CNN is applied to a time series it can extract shorter features of the overall data, regardless their position in the sequence.

In multivariate time series classification [161], each feature is mapped to a channel of an independent convolution and pooling layers in order to learn features individually. Then, the

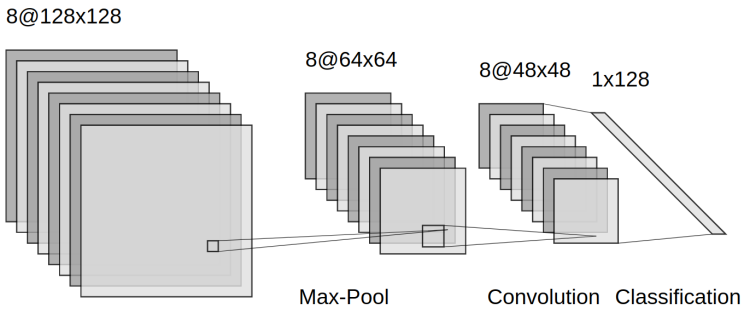


Fig. 2.13 Convolutional network architecture example

learnt features of each channel are passed into a Multilayer Perceptron (MLP) to perform classification. The tuning of a CNN includes the kernel width, stride and pooling function. The kernel width indicates the size of the sliding window that will traverse the time series, whereas the stride controls how many steps the sliding window will take while it moves on the time series. Both parameters influences the size of the network, as a large kernel width and stride will encompass and merge a larger number of values in a time series. The pooling function performs an operation over the values on the sliding window, the more popular being average pooling, which takes the average of the sliding window values, and max pooling, which extracts the maximum value from the sliding window.

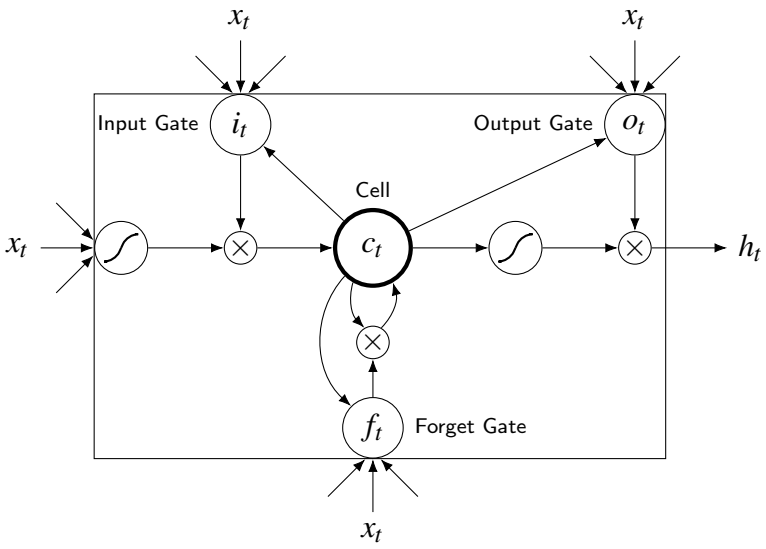
Long Short-Term Memory Networks:

Recurrent neural networks (RNNs) are specialized in processing sequential data and may consider the current sample, as well as past samples. Long Short-Term Memory (LSTM) networks [65] add the capability of memory persistence to the network, by utilizing a modified neuron called LSTM cell, depicted in figure 2.14. A LSTM cell have additional weights that control how much from previous examples must be forgotten, the *forget gate*, and how much must be considered in the processing of the current example, the *output gate*. A LSTM network is composed by multiple LSTM cells, as depicted in Figure 2.15.

An additional hyperparameter in LSTM networks is the *lag* value, which indicates how many steps in the past each sample contains, i.e., each feature of input sample of a LSTM

layer with $lag = 1$ will include the feature value at the current t instant and at the last $t-1$ instant.

Fig. 2.14 LSTM cell structure



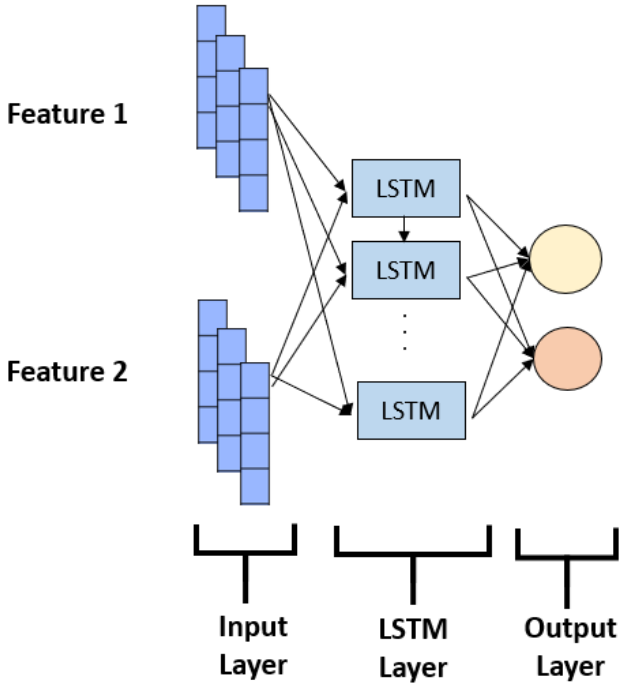


Fig. 2.15 Long Short-Term Memory network example

Chapter 3

Related Work

Since earlier discussions of BGP instabilities by Labovitz et al. [86, 87], many works have expanded their analysis [93], larger scale anomalies have occurred [150] and mechanisms to detect and classify anomalies were proposed [4]. This chapter discusses selected works focused on detection and classification of BGP abnormal events. We seek to identify general trends that these works develop, which types of anomalies they detect and/or classify, and their limitations.

There are four main approaches to BGP anomaly detection and classification: Time-series analysis, machine learning, statistical pattern recognition and comparison with historical data. We provide a brief description of representative works from each class.

3.1 Time series analysis

Approaches based on time series analysis observe the evolution of BGP dynamics through time and associate specific changes of traffic behavior with anomalous events. For example, the number of BGP update messages exchanged among peers may increase drastically during an anomaly.

Zhang et al. [158] propose the use of wavelet transforms (WT) [31] as tool to observe BGP features across different time scales and frequencies. Wavelet transforms decompose a signal into subsequences at different resolution scales, transforming a given time series into

high and low-frequency components. At high frequency (shorter time intervals), the wavelet can capture discontinuities, ruptures and singularities in BGP data. At low frequency (longer time intervals), the wavelet reveals the coarse structure of the data to identify the long-term trends. Wavelet transforms can be used as a denoising tool and the extracted subsequences used as a smoothed version of fine-grained BGP dynamics, as depicted in figure 3.1.

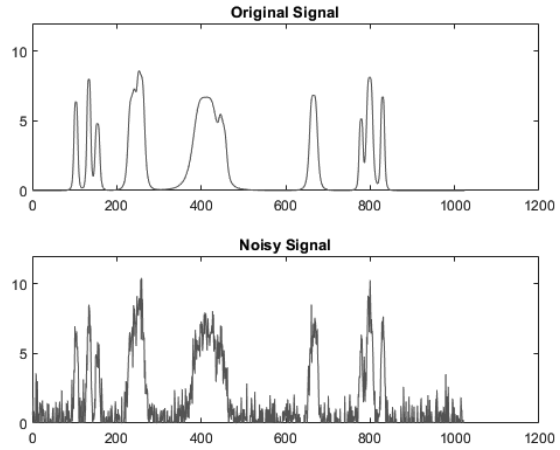


Fig. 3.1 Signal denoising using wavelet transform

In this work, Zhang et al. construct time series daily episodes of the number of BGP updates for each prefix and for each peer, then a wavelet transform is used to extract peaks at different time scales and frequencies. These episodes are grouped using the *k-means* clustering algorithm [63]. Zhang et al. observed that the largest cluster dominates over the others, as depicted in Figure 3.2 for different number of clusters, validating the idea that BGP dynamics are consistent at prefix- and peer-level. However, due to a high number of false positives, it is required that small clusters are manually inspected in order to investigate possible anomalies.

Mai et al. [105] proposed an anomaly detection framework called *BAlet*, which extended previous work by increasing the robustness of the time series analysis, automating detection and providing anomaly localization. BAlet uses Maximal Overlap Discrete WT [120], a type of WT that is translation-invariant and yields the same results for anomalies that start at different points. Traffic episodes are clustered using a *k-means* variation that detects and generates new clusters automatically. When a new cluster is generated, the mechanism

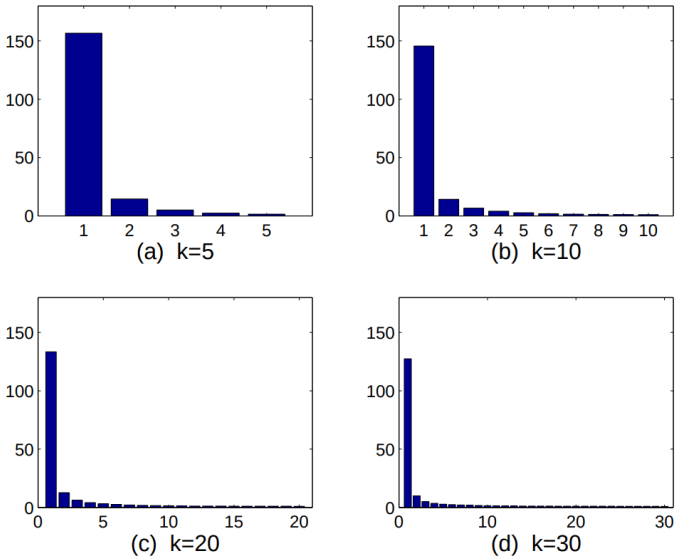


Fig. 3.2 Episode clusters distribution for different k values. [158]

raises an alert indicating a possible anomaly. BAlet also groups updates according to their originating AS and uses this information to suggest possible origins of the anomaly. Similarly to Zhang et al. [158], BAlet still requires human intervention due to a high number of false positives.

Prakash et at. [123] further investigated BGP dynamics temporal properties and demonstrated that regular BGP traffic shows self-similarity, and leverages this fact in its anomaly detection mechanism, called BGP-lens. They observed two types of anomalies, *prolonged spikes*, anomalies that last for hours, and *clothesline* anomalies, low-frequency trends that may last for days or weeks, both of which are depicted in Figure 3.1. Similarly to previous time series analysis methods, BGP-lens is rather focused on providing a number of GUI tools in order to help network administrators to analyze BGP traffic and trends than focused on automate BGP anomaly detection and mitigation.

Summary: These methods have the advantage to inherently detect anomalies at different scales, but require that the network administrator takes action and further investigate whether the anomaly corresponds harmful anomalous event. Besides, they were not tested against direct anomalies or outages, nor are specialized to classify different anomaly classes.

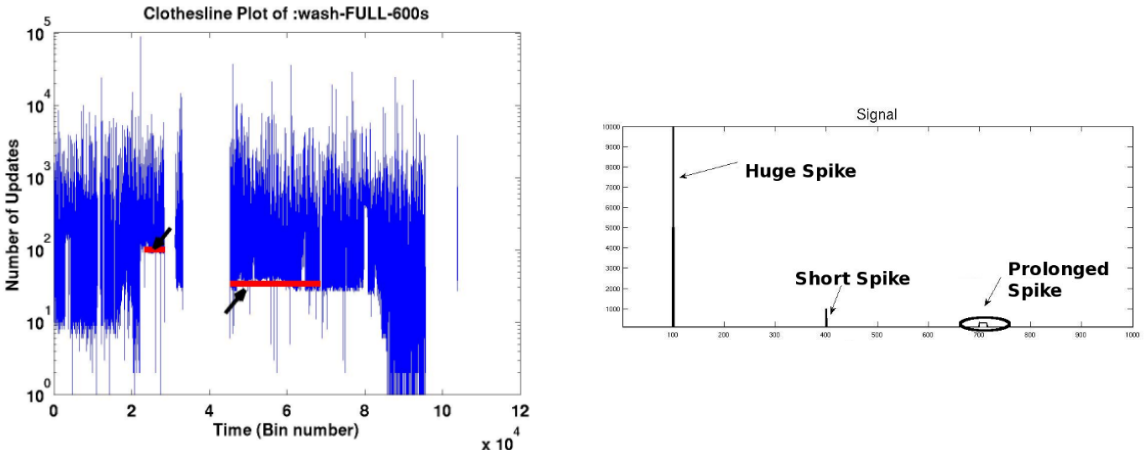


Fig. 3.3 BGP clothesline pattern and spike patterns [123]

3.2 Comparison with historical data

These methods improve time-series approaches by incorporating expert knowledge into the detection mechanism. The following mechanisms assume that the Internet topology does not change frequently and anomalies may be detected by validating new updates against the history of BGP messages.

Lad et al. [88] presented a prefix hijacking system that uses the history of BGP updates to detect suspicious behavior and warn the original prefix owner. The mechanism monitors BGP messages in public repositories and detect changes announced for a set of preconfigured prefixes, through the architecture depicted in Figure 3.4. The system uses an adaptive window scheme, which increases the window size when there are many changes in the origin of a prefix and decreases it in case of a small number of changes. This system requires that network administrators register their prefixes manually, in order to use it as a base point and only provides notifications of possible hijacks.

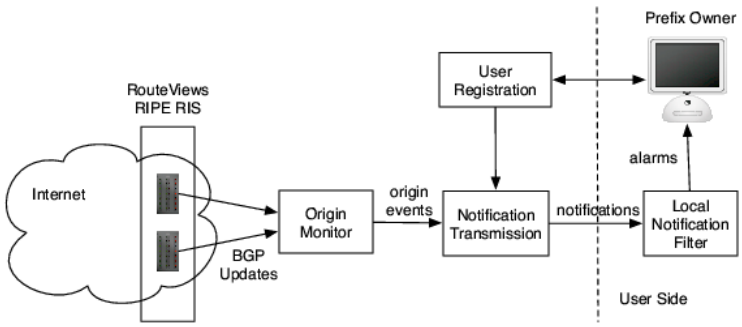


Fig. 3.4 PHAS hijack detection architecture. [88]

Shi et al. [138] extends previous work by adding data plane and Internet Route Registry (IRR) [69] information a distributed routing database development effort constructed with public collaboration, to its hijacking detection system. Their method, called Argus, keeps track of control plane messages and stores the normal origin AS of each prefix. When a change occurs, Argus retrieve live IPs within the prefix range using available traceroute information [47] and uses ping to check their reachability. They also use the IRR data, in order to improve the false positive rate.

Summary: These methods automate part of the decision process of the network administrator when troubleshooting anomalies and may be used with time series analysis methods, but they still may suffer from false positives due to unexpected changes and inconsistent information in IRR. Besides, the proposed methods only detect hijackings (direct anomalies).

3.3 Statistical pattern recognition

The following methods handle some of the limitations of previous methods by increasing the generalization provided through the use of statistical models. In these methods, an expert builds a model that is adjusted according to real data from the problem domain. An expert must analyze the resulting model and observe how it fits to the data, in order to adjust parameters and thresholds to detect future anomalies.

Deshpande et al. [33] developed an anomaly detection mechanism that leverages both traffic volume and historical data. The mechanism extracts BGP volume, AS-PATH length, and edit distance between paths from BGP update messages every 5 minutes. They also

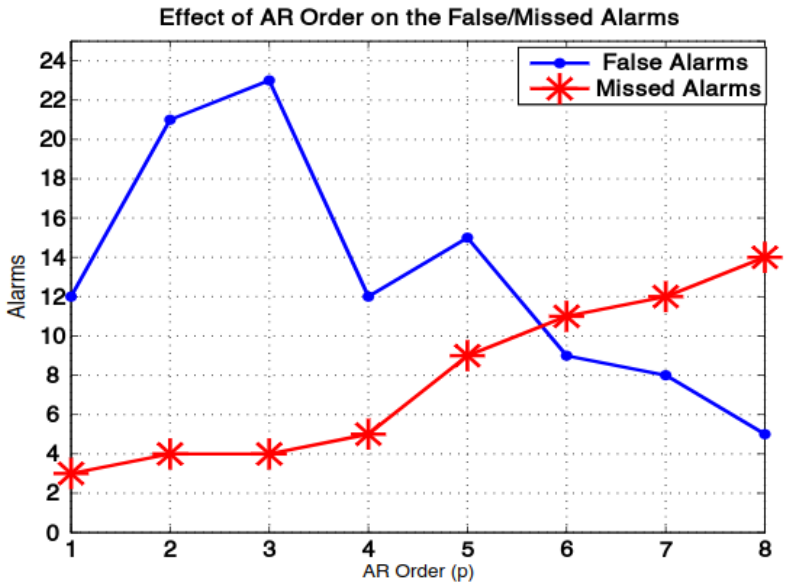


Fig. 3.5 Autoregressive order parameter selection [33]

evaluate the significance of features using the Fisher score [39]. The detection method is deployed at router-level and is based on the Generalized Likelihood Ratio Test (GLRT), a statistical technique used in hypothesis testing. GLRT helps to detect and locate change in time series data, based on threshold values parameterized by an expert. They trained the model using data from all three types of events: direct, indirect and outages. However, they do not differentiate among the classes of anomalies and the system may take hours to detect anomalies, not being suitable for real-time detection. Besides, the model is highly sensitive to parameter tuning, Figure 3.5 shows the variation of false alarms and missed anomalies with respect to different values of AR order [68], a parameter that must be tuned in autoregressive models.

Huang et al. [66] used information from different data sources to perform detection and classification of BGP outages. They observed that not all anomalies caused changes in BGP volume data and explored two additional data source: operational mailing list and router configuration. As with other methods, Huang et al. construct a time series using the number of BGP messages, then use a Principal Component Analysis (PCA) [74] to build traffic subspace. The subspace that contains the largest fraction of traffic is the *normal subspace* and

others are regarded as *anomalous subspaces*. They use the network configuration files to infer the topology and relationships among network devices. From this information, the method extracts a decision tree that identifies different network disruption types, as depicted in figure 3.6. The outages identified were manually validated against reports in the mailing list. This method only detects network outages and also may take an hour to detect an anomaly.

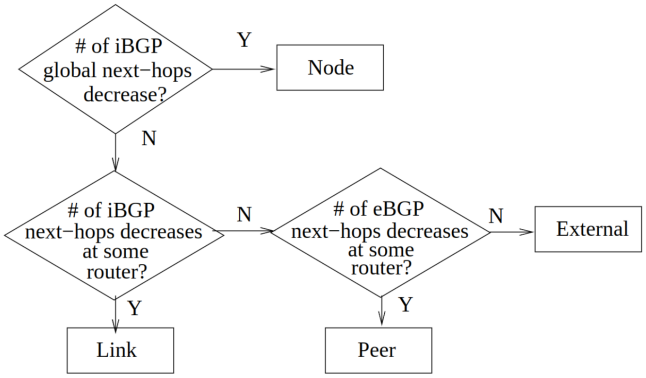


Fig. 3.6 Decision tree for classifying outage type [66]

Summary: Approaches based on statistical models have the disadvantage of requiring an expert to tune the model parameters. As time passes, the model may need to be readjusted in order to cope with network changes. Huang et al. also require human operators to extract information from router configuration and mailing lists. Furthermore, they are slow and not well suited for online detection.

3.4 Machine learning

Machine learning methods are similar to statistical pattern recognition approaches, but have the advantage of not requiring an expert to tune the learning model, which can be done by a generalist through adjustment of hyper-parameters and training/test error observation. Besides, deep learning may be used to define complex models that can capture behaviors that statistical pattern recognition models can not fit, due to being more shallow.

Li et al. [92] presented early efforts in integrating machine learning and BGP anomaly detection. They proposed a framework that extracted 35 features from control plane mes-

	Rules for the normal class		Rules for the worm class
1.	IF Announce <= 236 THEN class = "normal" [99.8%]	1.	IF Announce > 236 AND Update_prefix > 720 AND WADiff > 47 AND AW > 84 AND WADup > 28 THEN class = "worm" [89.1%]
2.	IF Updates <= 358 AND Withdraw_prefix <= 106 THEN class = "normal" [99.6%]		
3.	IF WADiff <= 34 THEN class = "normal" [99.5%]		

Fig. 3.7 Classification rules generated by data mining in Li et al. [92]

sages, including volume of traffic and graph properties from topology, and applied the C4.5 algorithm [151], which automatically calculates thresholds for each feature that define an anomaly. They used outages and indirect anomalies datasets to train the model. A sample of generated rules is depicted in Figure 3.7.

Multiple works [34, 162, 26, 96, 107] investigated different machine learning methods through a similar workflow. They extract BGP features, based on message volume and AS paths announced, and a classifier that learns how to detect anomalies. These works use a wide range of machine learning algorithms: naive Bayes, decision trees, hidden Markov models (HMM), support vector machine (SVM) and neural networks. For feature selection, they used Fisher score and minimum Redundancy Maximum Relevance (mRMR) [119], a scoring algorithm that maximizes variance among features. These mechanisms were trained and tested using direct and indirect anomalies, but their detection models do not differentiate anomaly classes. Besides, each approach is trained with a different subset, for example, Cosovic et al. [26] only uses routing table leak events (direct anomalies). It is noteworthy that these approaches used vanilla methods, without any customization, focusing on the feature engineering and selection in order to achieve good performance.

Other efforts proposed customized deep learning model architecture, in order to endow the model with characteristics that matched with the anomaly detection characteristics. Cheng et al. [20] argued that BGP anomaly detection has temporal features that might be decomposed in different frequencies. They proposed MS-LSTM, a detection mechanism that integrates wavelet transforms and neural networks. Initially, the mechanism extracts time

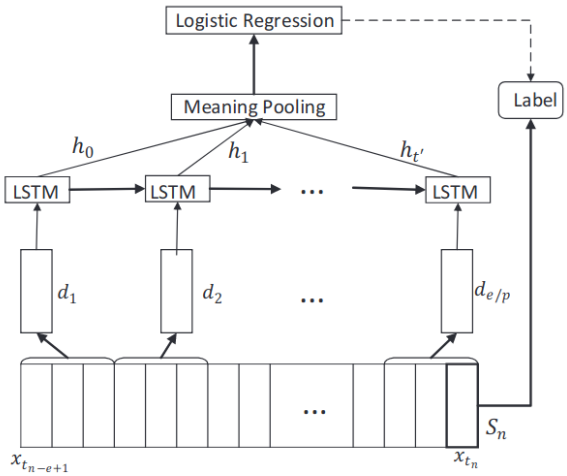


Fig. 3.8 MS-LSTM network architecture

series for 33 features using control plane data, then wavelet transforms are applied to the time series in order to obtain sequences from multiple resolutions. Transformed sequences are fed into a Long Short Term Memory (LSTM) network [65], a recurrent neural network used for sequence classification and capable of detecting long range dependencies. The proposed architecture is depicted in Figure 3.8. The model was trained and tested only with indirect anomaly events, thus, it is not possible to assess whether the MS-LSTM model may generalize and perform well on different classes.

Kim et al. [78] proposed a neural network architecture that integrates convolutional neural networks and LSTM networks in order to extract spatial and temporal features from the training data. This model is aimed to anomaly detection in web servers, rather than BGP anomalous events and it uses a univariate time series of traffic volume as its training data. The proposed architecture, depicted in Figure 3.9, is composed by convolution layers followed by pooling layers, which are responsible for detecting spatial properties in the web traffic volume, such as short spikes, then it is integrated to a LSTM layer which detects long-term dependency features. This approach was capable of achieving an accuracy higher than 98%, but it was not applied to classification or BGP anomaly detection.

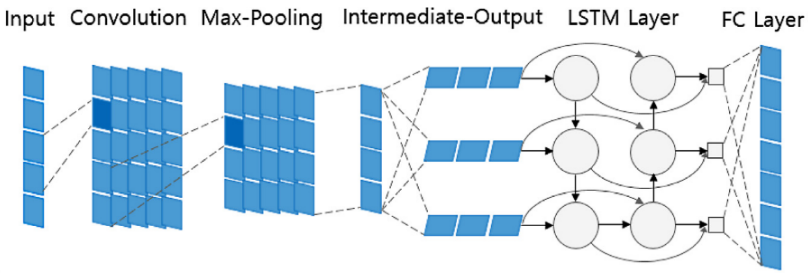


Fig. 3.9 C-LSTM architecture [78]

3.4.1 Datasets

Besides the model architecture, another important aspect of deep learning is the training data used to fit the proposed model. When different neural networks architectures are trained with different datasets, any comparison between them is hampered because there is no guarantee that their results were not influenced by the training data distribution. Therefore, a current concern in deep learning is to provide common datasets for different domains.

In handwritten digit recognition, the MNIST database [32] is a widely used dataset which consists of a set of images of digits written by high school students and employees of the United States Census Bureau. The MNIST database contains 60,000 training images and 10,000 testing images, where the digits have been size-normalized and centered in a fixed-size image. A large number of efforts were evaluated using the MNIST database, based on a wide range of methods, such as linear classifiers, k-nearest neighbors, SVMs and neural networks.

In intrusion detection, mechanisms are trained in order to detect anomalous behavior and/or malicious attack in the network. The KDD Cup 1999 dataset [143, 139] includes a wide variety of network data and contains intrusions simulated in a military network environment. Nine weeks of raw TCP dump data were collected from a network simulating a U.S. Air Force LAN, with fabricated attacks. The KDD training dataset consists of 494,020 labeled samples each of which contains 41 features, being split in 19.69% normal traffic and 80.31% attack connections. KDD CUP 99 has been widely used for evaluation of intrusion detection systems [18, 5, 48].

	Nimda	Slammer	Code Red	Moscow Black-out	Japan Earth-quake	AWS Leak	Malaysian Telecom	AS9121 Leak
Li et al. [123]	✓	✓	✓	✓				
Lad et al. [88]						✓	✓	✓
Deshpande et al. [33]	✓	✓	✓			✓	✓	
Cheng et al. [20]	✓	✓	✓					
Cosovic et al. [26]						✓	✓	✓
Current work	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.1 Comparison of events used to evaluate the proposed efforts

These datasets play an important role in research efforts because they allow comparison when a novel method is proposed. In BGP anomaly detection, there is a scarcity of public datasets with most used features. To the best of our knowledge, the only dataset publicly available was provided by Cheng et al. [21]. However, the provided datasets only comprises 3 events: Slammer, Nimda and Code Red II, all of which belong to the indirect class. Table 3.1 compares which events were used to evaluate some of the discussed efforts, we can observe that different approaches used different datasets than those provided by Cheng et al. Thus, it is necessary to have a common view of all event classes in order to consistently compare the previously proposed approaches and allow future methods to assess their own performance without the need to re-implement feature extraction and dataset generation.

Summary: Machine learning detection mechanisms available in the literature achieved acceptable detection accuracy, precision and recall for the evaluated datasets and, once trained, the detection models have potential to provide results in real-time. However, few works explored temporal and spatial properties of BGP data, only performed anomaly detection, without exploring anomaly classification and location. Additionally, few data sources were explored and only shallow models were used.

3.5 Summary

Table 3.2 BGP anomaly detection and classification efforts events

Method	Detec.	Classif.	Online	Direct	Indirect	Outage	Approach	Data Source
Zhang et al. [158]	✓			✓			Time series analysis	Control plane
Mai et al. [105]	✓			✓			Time series analysis	Control plane
Prakash et al. [123]	✓			✓			Time series analysis	Control plane
Deshpande et al. [33]	✓			✓	✓	✓	Statistical pattern recognition	Control plane
Huang et al. [66]	✓					✓	Statistical pattern recognition	Control plane Router configuration
Lad et al. [88]	✓	✓					Historical data	Control plane
Shi et al. [138]	✓	✓		✓			Historical data	Control plane Data plane IRR
Li et al. [92]	✓		✓	✓		✓	Machine Learning	Control plane
Ding et al. [34]	✓		✓	✓	✓		Machine Learning	Control plane
Li et al. [96]	✓		✓	✓	✓		Machine Learning	Control plane
Ćosović et al. [162]	✓		✓	✓	✓		Machine Learning	Control plane
Cheng et al. [20]	✓		✓		✓		Machine Learning	Control plane
⋆ **Current work**	✓	✓	✓	✓	✓	✓	Machine Learning	Control plane

The discussed efforts differ with respect to many aspects, next we summarize the main gaps and characteristics. Time-series analysis and methods based on historical behavior analysis only filter events that are possible candidate anomalies, have a higher false positive rate than others, thus they require that a network administrator takes further investigation. Statistical models make this process more transparent and have higher precision and recall, however, they still need a domain-specific knowledge in order to correctly fine tune the models. Machine learning approaches require less human intervention, since their hyper-parameters can be adjusted without, necessarily, domain-specific knowledge, although, most

of discussed efforts use vanilla models without customizations that are suited for BGP anomaly detection main characteristics, such as spatial and temporal properties.

In our literature survey, we identified two major gaps in previously proposed efforts. These open issues were used to guide the development of this work. First, an important gap is the lack of a classification framework that systematically differentiates between direct, indirect and outage anomalies. Different types of anomalies require specific mitigation measures. For example, when a network administrator identify a routing table leak (direct anomaly), it may localize which routes were inadvertently announced and modify them in order to stop their propagation. There are few efforts [138, 88] that perform classification of anomalies, but they are specialized in only one type of anomaly and also suffer from previously cited disadvantages.

Another major gap in previous efforts is that each model was validated using different datasets, which had to be generated for each approach. The absence of common groundwork dataset increases the difficulty in comparing different approaches. For example, a model trained and validated against a small amount of datasets might perform better than a model trained with a larger amount of datasets, but it does not guarantee that the former model is better than the latter, because it might not have the generalization necessary to perform well on a bigger dataset pool.

In the current work, our main contribution is two-fold: a BGP dataset generation tool and anomaly classifier. We provide a tool which generates tabular data containing time series of features from raw BGP messages. These features encompass metrics used by other efforts and novel metrics introduced in this work. The extracted features are capable of capturing relevant behavior in the context of anomaly detection, and possibly in other areas as well. Using this tool, we generated datasets of the most used anomaly events, which comprises all different anomaly classes. The generated datasets were made publicly available and are suited to be used by most of machine learning models [20][34]. The feature extraction and dataset generation tools were also made publicly available in an open-source platform and can be extended in order to support more features. Since there are no similar BGP datasets, nor feature extraction tools, publicly available, we believe that this contribution will assist

future research in BGP behavior and support the growing trend of applying machine learning methods to networking problems.

We also provided a anomaly classifier, through two-layered anomaly categorization. Such classification is the first step towards specialized mitigation techniques, endowing networks with the ability to perform closed-loop decisions in a timely manner. Our classifier also has a low number of false positives, decreasing the need of operator intervention. We propose an integration of our detection and classification mechanisms in a SDN ecosystem, where a logically centralized layer will process and reply to BGP messages from different peer ASes, in a seamless manner to the external observer, at the same time it exploits the abstractions provided by SDN to reason about network behavior and enforce its policies. Another advantage of the proposed architecture is that the use of SDN may leverage the realization of self-driving networks [6, 70].

Table 3.2 compares current work and previous efforts with respect to the task performed (detection and classification), the types of anomalies handled, the approach and data sources used. Table 3.1 compares which datasets where used by different approaches, even though this table does not include all discussed efforts, it lists one method from each discussed general approach (time series analysis, comparison with historical data, statistical pattern recognition and machine learning). To the best of our knowledge, none of the previously proposed methods used anomalous events from all of the anomaly classes to evaluate the accuracy of their detection.

Chapter 4

Feature Extraction and Dataset Generation

This chapter is organized as follows: Section 4.1 describes the process of feature extraction and describes volume, AS path features, as well as our proposed distribution features. Section 4.2 describes the workflow that we use to generate and label the datasets. Section 4.3 describes the general trends in anomalous behavior in comparison with regular traffic and among different anomalies classes.

4.1 Feature Extraction

BGP control plane messages are the most used to perform anomaly detection [4], due to its fine temporal granularity, high-level of detail and availability from multiple points-of-view across the Internet. BGP public data relies mainly on *route collectors*. A route collector runs a process, which emulates a router and establishes BGP peering sessions with one or more real routers. Each router sends to the collector update messages each time the Adj-RIB-out changes, reflecting changes to its Loc-RIB. The collector periodically dumps a snapshot of the Adj-RIB-Out tables and update messages received since last dump. These dumps offer a complete view of the routing dynamics. The most popular projects operating route

collectors and making their dumps available in public archives are RouteViews [130] and RIPE RIS [128]. This architecture is depicted in Figure 4.1.

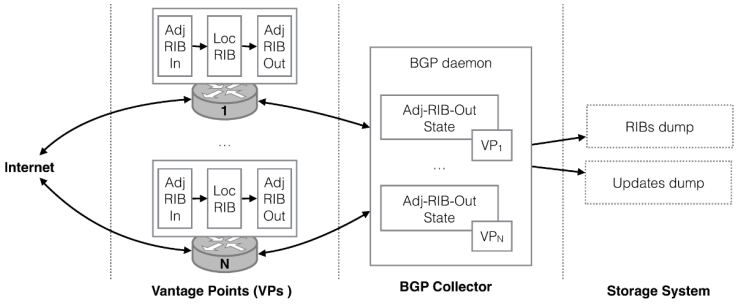


Fig. 4.1 BGP collection process [116].

BGP messages only represent incremental changes, not providing information about past states and large-scale behavior changes across different ASes. In order to extract meaningful information from control plane messages, a feature extraction process must be applied upon BGP updates.

Most of the BGP anomaly detection methods extract features as time series and observe deviation patterns that indicate anomalous behavior. In current literature [4], features extracted from BGP messages are mainly classified in two classes: *Volume* and *AS path* features. Volume features provides information about the absolute quantity of each type of message with respect to that kind of information that it communicates (*e.g.* re-announcements, withdrawals). AS path features give insight into behavior of announced AS paths (*e.g.*, longer paths, rare ASes in paths). Some works use BGP attributes (such as communities attributes [54]) in order to extract information that do not fit in these two classes (volume and AS path), but these extracted features are not represented as time series, thus, they are out of the scope of this discussion.

4.1.1 Volume features

Our BGP update message classification is depicted in figure 4.2, based on the taxonomy proposed by Wang et al. [150]. Given a time window *w*, the mechanism counts the number of occurrences of each message type of each node from the classification tree (Figure 4.2).

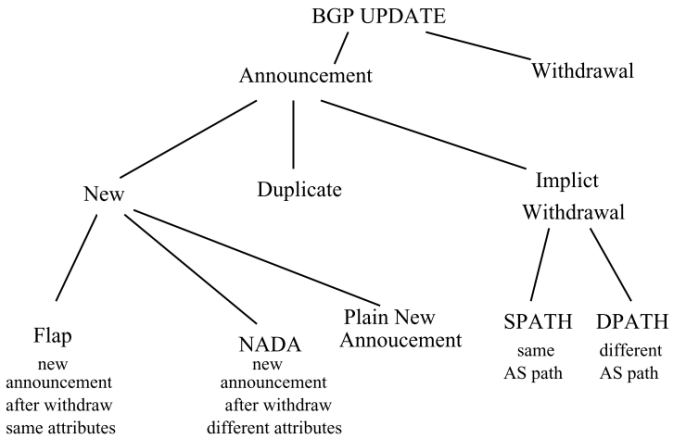


Fig. 4.2 BGP update message classification

For example, given $w = 5$ *minutes*, in the most general level, we count the number of announcements and withdrawals in the period. Upon a deeper analysis on announcements, we also count the number of new announcements, duplicate announcements and implicit withdrawals.

- **Number of announcements:** General number of messages received that contains a request to *include* a BGP AS path in the RIB.

$$\sum_{t}^{t+w} BGP\ Message_{announcement} \tag{4.1}$$

- **Number of withdrawals:** General number of messages received that contains a request to *remove* a BGP AS path in the RIB.

$$\sum_{t}^{t+w} BGP\ Message_{withdrawal} \tag{4.2}$$

- **Number of duplicate announcements:** Our mechanism keeps track of previously announced paths and marks consecutive announcements with the same attributes as duplicates.

$$\sum_{t}^{t+w} BGP\ Message_{duplicateann.} \tag{4.3}$$

- **Number of duplicate withdrawals:** Consecutive announcements/withdrawals with the same attributes.

$$\sum_{t}^{t+w} BGP\ Message_{duplicatewd} \tag{4.4}$$

- **Number of non-duplicate announcements:** Announcements that contains information different from the stored in the RIB.

$$\sum_{t}^{t+w} unique(BGP\ Message_{announcement}) \tag{4.5}$$

- **Number of flaps:** When a path is announced, succeeded by a withdrawal message that removes it from the RIB and, then, the same path is announced again with the same attributes.

$$\sum_{t}^{t+w} BGP\ Message_{flap} \tag{4.6}$$

- **Number of new announcements after withdraw:** When a path to a prefix is announced, then removed from the RIB and then a path to the same prefix is announced with different attributes.

$$\sum_{t}^{t+w} BGP\ Message_{announcement\ after\ withdrawal} \tag{4.7}$$

- **Number of plain new announcements:** Announcements for a prefix that was never stored in the RIB.

$$\sum_{t}^{t+w} BGP\ Message_{plain\ new} \tag{4.8}$$

- **Number of implicit withdrawals with same path:** Consecutive announcements for the same prefix with the same path but with at least one different attribute (e.g. MED, LOCAL PREF, ORIGIN)

$$\sum_{t}^{t+w} BGP\ Message_{implicit\ withdrawal_1} \tag{4.9}$$

- **Number of implicit withdrawals with different path:** Consecutive announcements for the same prefix with a different path.

$$\sum_{t}^{t+w} BGP\ Message_{implicit\ withdrawal_2} \tag{4.10}$$

- **Number of** IGP**/**EGP**/**INCOMPLETE **messages:** The ORIGIN attribute indicates which protocol generated the BGP message, our mechanism keeps a counter for each ORIGIN type.

$$\sum_{t}^{t+w} BGP\ Message_{IGP},\ \sum_{t}^{t+w} BGP\ Message_{EGP},\ \sum_{t}^{t+w} BGP\ Message_{INCOMPLETE} \tag{4.11}$$

- **Number of** ORIGIN **changes:** Counts when a re-announcement changes the ORIGIN attribute previously announced.

$$\sum_{t}^{t+w} BGP\ Announcement_{diff\ \mathrm{origin}} \tag{4.12}$$

- **Number of announced prefixes:** Number of different prefixes announced in time window w.

$$\sum_{t}^{t+w} unique(Prefix) \tag{4.13}$$

- **Average announcements per prefix:** Average number of BGP messages across all prefixes.

$$\frac{\sum_{t}^{t+w} BGP\ Message}{Number\ of\ prefixes} \tag{4.14}$$

- **Maximum announcements per prefix:** Maximum number of BGP messages related to a prefix during a time window w.

$$max(\sum_{t}^{t+w} BGP\ Message_p, where\ p \in prefixes \tag{4.15}$$

- **Average announcements per peer:** Average number of BGP messages across all peers of a BGP router.

$$\frac{\sum_{t}^{t+w} BGP\ Message}{Number\ of\ peers} \tag{4.16}$$

- **Maximum announcements per AS:** Maximum number of BGP messages related to a BGP router peer during a time window w.

$$max(\sum_{t}^{t+w} BGP\ Message_{as}, where\ as \in ASes \tag{4.17}$$

4.1.2 AS path features

Considering the AS-PATH attribute and announcement history, we extract the following features, based on the classification proposed by Wang et al. [150].

- **Announcements to longer/shorter paths:** Our extractor identifies when a new path is announced to a prefix that already exists in the RIB. This features tracks whether this new path is longer or shorter than the path previously stored.

$$\sum_{t}^{t+w} BGP\ Msg^{t}_{prefix}, \quad where\ path\_length(BGP\ Msg^{t}_{prefix}) > path\_length(BGP\ Msg^{t-1}_{prefix}) \tag{4.18}$$

$$\sum_{t}^{t+w} BGP\ Msg^{t}_{prefix}, \quad where\ path\_length(BGP\ Msg^{t}_{prefix}) < path\_length(BGP\ Msg^{t-1}_{prefix}) \tag{4.19}$$

- **Average AS path length:** Average AS path considering the path announced in all BGP messages during a time window w.

$$\frac{\sum_{t}^{t+w} path\_length(BGP\ Message)}{\sum_{t}^{t+w} BGP\ messages} \tag{4.20}$$

- **Maximum AS path length:** Maximum AS path length found in received BGP messages during a time window w.

$$max(path\_length(BGP\ Message^{i}),\ where\ t < i < t + w \tag{4.21}$$

- **Average AS path length:** Some ASes artificially inflate the path length of an announced path by including redundant hops in a path (*e.g.*, path = (A B B B C)). This strategy is widely used in order to decrease the chances of route being chosen by a peer (e.g. backup path). Our extractor keeps metrics that ignore prepending in order to focus on the underlying infrastructure rather than network policies. This metric calculates the average AS path ignoring prepending and considering the paths announced in all BGP messages during a time window w.

$$\frac{\sum_{t}^{t+w} path\_length(BGP\ Message_{u}nique)}{\sum_{t}^{t+w} BGP\ message_{u}nique} \tag{4.22}$$

- **Maximum AS path length:** Maximum AS path length, ignoring prepending, found in received BGP messages during a time window w.

$$max(path\_length(BGP\ Message^{i}_{u}nique),\ where\ t < i < t + w \tag{4.23}$$

- **Average edit distance:** When a AS path is announced to a prefix, we measure its difference in comparison with the previously stored path by calculating the edit distance. This metric, which is also called Levenshtein distance, is a sequence comparison method that indicates the minimum number of single-character edits (i.e. insertions and deletions) required in order to turn a sequence into the other. The average edit distance is calculated over all path changes.

$$\frac{\sum_{t}^{t+w} edit\_distance(BGP\ msg^{t}_{path_{c}hange}, BGP\ msg^{t+1}_{path_{c}hange})}{\sum_{t}^{t+w} BGP\ message_{path_{c}hange}} \tag{4.24}$$

- **Maximum edit distance:** This feature stores the highest edit distance in a given time window w.

$$max(edit\_distance(BGP\ msg^{i}_{path_change}, BGP\ msg^{i+1}_{path_change})),\ where\ t < i < t+w \tag{4.25}$$

- **Edit distance with k value:** Counts the number of message updates with a given k distance with respect to the previous known route.

$$edit\_distance(BGP\ msg^{i}_{path_change}, BGP\ msg^{i+1}_{path_change}) = k,\ where\ t < i < t+w \tag{4.26}$$

- **Edit distance with k value (unique):** Similar to the previous feature, but removes duplicate ASes in the path.

- **Number of rare ASes:** We keep track of how many times each AS appear in all the announced paths. Usually an AS appear multiple times in a given time window, we classify an AS as *rare* if its number of appearances is below a given percentage threshold of ASes appearances (we set the default at $95th$ percentile).

- **Average number of rare ASes:** The average number of ASes that appear in a given time window w.

$$\frac{\sum_{t}^{t+w} rare\_paths(BGP\ Message)}{\sum_{t}^{t+w} BGP\ messages} \tag{4.27}$$

- **Maximum number of rare ASes:** The maximum number of ASes that appear in a message during a given time window w.

$$max(rare\_paths(BGP\ Message^{i})),\ where\ t < i < t+w \tag{4.28}$$

4.1.3 New features

During the analysis of the behavior of volume and AS path features, we observed that the distribution of update types may change significantly when anomalies occur, *e.g.* the

proportion of announcements to shorter and longer paths. Leveraging this observation, we propose novel features that represent the distribution of message types, which may be used to distinguish different anomaly classes, as will be discussed in Section 4.3. In the next subsections, we provide more detail on the extracted features.

We group the feature extraction of complementary message types, for instance, all BGP messages might be separated into announcements and withdrawals, thus we keep track of two complementary distribution features that represent the percentage of each type at a given time (e.g. 90% announcements and 10% withdrawals).

Usually, during a large-scale anomaly, we observe a massive increase in the updates, thus, all features increase accordingly. However, this trend makes it harder to isolate the effect upon the distribution of a given type of message and compare with the previous values. Using these distribution features allow us to infer whether the proportion of a type of message (*e.g.* shorter paths) increased or decreased by looking to a single variable.

We propose the following distribution features:

- **Announcements vs withdrawals:** General proportion between announcements and withdrawals.
- **Origin types:** Distribution among IGP, EGP and INCOMPLETE messages.
- **Announcements types:** Distribution among flaps, re-announcements with new paths, new announcements and implicit withdrawals.
- **Longer vs Shorter:** Proportion between changes to paths with longer and shorter paths. We generate a pair of features considering the proportion with respect to the total number of announcements, and other pair considering only the announcements that changed the path length.
- **Implicit withdrawals vs explicit withdrawals:** Distribution between the withdrawal types.
- **Withdrawals w/ same path vs withdrawals w/ different path:** Distribution between the implicit withdrawal types.

New features evaluation

We evaluate how the new features correlate with the output classes by applying two methods: univariate selection and feature importance. Univariate feature selection examines each feature in order to determine the strength of the relationship of the feature with the target variable. There is a lot of different options for univariate. We used the chi-squared statistical test for non-negative features to select 15 of the best features from the extracted datasets, which are listed in Table 4.1. We can observe that volume features rank in the first positions due to the high spike in data that occurs during anomalies. Distribution features comprise almost half of the top 15 features, which indicates that they are relevant to the classification task, given that volume features are not enough to detect and classify anomalies.

Another way to assist the task of feature selection is by extracting the feature importance from a trained model. Feature importance gives the score for each feature of the dataset, the higher the score more important or relevant is the feature towards the target variable. We trained models of a Decision Tree Classifier and extracted 10 top features for the dataset. Figure 4.3 shows the results of the extracted feature importances. Similar to the univariate selection, we identified that volume features ($\mathrm{nlri\_ann}$ and $\mathrm{announcements}$ are the number of announced prefixes and number of announcements, respectively) are the most important, due to the large increase in the number of BGP messages during anomalies. However, it also possible to observe that distribution features (features that have $\mathrm{ratio\_}$ prefix in the vertical label) occupy a large part of the the top 10 most important features.

Summary: Our framework connects to a BGP collector, which is selected according to empirical observations, and generates a time series for each feature described in this section, in given a period of time. First, we initialize the RIB state by downloading the RIB dump that is closer to the beginning of the desired period. Then, each message is processed and classified according to its type, its attributes are extracted and related features are updated accordingly. As an example, Figure 4.4 shows a time series for the number of announcements during the Slammer event, from January, 23, 2003 to January, 27, 2003. We proposed novel features that extract the distribution of volume among complementary BGP message types

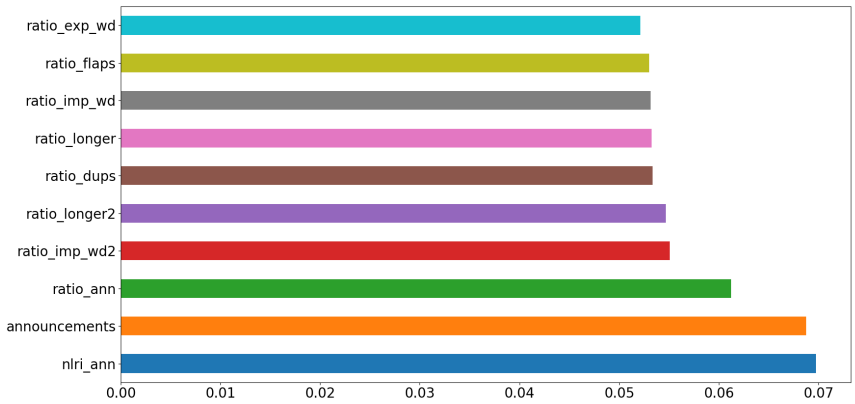


Fig. 4.3 Feature importance extracted from trained models.

and evaluated their importance to the classification task, which showed that they rank close to significant volume features.

4.2 Dataset generation

Even though there are many data sources of BGP control plane messages that encompasses periods of anomaly, there are no public datasets that are in a format that is suited to feed machine learning models, *e.g.* labeled time series of BGP features. Therefore, each anomaly detection method must re-implement a feature extraction and generate its own dataset. Besides, the process of generating datasets may be laborious because there is not a consensus of when an anomaly starts and ends, from which BGP collectors the anomaly can be observed and the duration of an anomaly can differ among different collectors.

Below, list the steps necessary to build a dataset.

1. Search for event reports that indicated approximate times of start and end of the anomaly, as well as which ASes were involved in the anomaly event.

2. Make a list of point-of-view candidates considering ASes and collectors that are close to the anomaly origin.

Rank	Feature
1	Announcements
2	Number of prefixes
3	Withdrawals
4	Origin changes
5	Number of rare ASes
6	Maximum length of AS path
7	(Distr.) Implicit withdrawal
8	Maximum number of rare ASes
9	(Distr.) Explicit withdrawal
10	(Unique) Average length of AS path
11	(Distr.) Withdrawal
12	(Distr.) New announcement after withdr.
13	(Distr.) Flaps
14	(Distr.) Duplicates
15	(Distr.) New announcements

Table 4.1 Ranking of features with the strongest relationship with the output class

Table 4.2 BGP volume features

Features	Type	Subtype
Number of duplicate annoucements/withdrawals	Volume	Stateful
Number of new annoucements	Volume	Stateful
Number of re-announcements after a withdrawal	Volume	Stateful
Number of withdrawals after announcing the same path	Volume	Stateful
Total number of annoucements/withdrawals/updates per AS	Volume	Stateless
Total number of annoucements/withdrawals/updates per prefix	Volume	Stateless
Number of IGP, EGP and INCOMPLETE in the Origin attribute	Volume	Stateless
Maximum/average announcements per prefix	Volume	Stateless

3. Download BGP raw data from public repositories [130, 128] of the collectors identified in the last step.

4. Implement a feature extraction mechanism and generate time series features from all candidates considering a period that comprises the anomaly, as well as regular traffic before and after the anomaly.

Table 4.3 BGP AS-path features

Features	**Type**
Maximum/Average AS-PATH length	AS path
Maximum/Average unique AS-PATH length	AS path
AS-Path length	AS path
Maximum/Average of rare ASes in the path	AS path
Observation of rare ASes in the path	AS path
Announcement to longer/shorter path	AS path
Prefix origin change	AS path
Number of new paths announced after withdrawing an old path	AS path
Number of new-path announcements	AS path
Maximum/Average edit distance	AS path
Maximum edit distance equals n ($n = 1,2,...$)	AS path
Maximum AS-path edit distance equals n ($n = 1,2,...$)	AS path

Table 4.4 BGP distribution features

Features	**Type**
Announcements vs Withdrawals	Distribution
ORIGIN types	Distribution
Announcement classes	Distribution
Longer vs Shorter	Distribution
Exp. Withdrawals vs Imp. Withdrawals	Distribution

5. Analyze the extracted series in order to observe trends that matched those reported (*e.g.* increase in number of announcements around a specific time of day).

6. Label the identified period as an anomaly event.

Figure 4.5 illustrates the described workflow. During the generation of datasets, we developed a set of tools that aid the visual identification and labeling of anomalies. Since we assume that large-scale BGP anomaly events change the normal volume of features, our tool

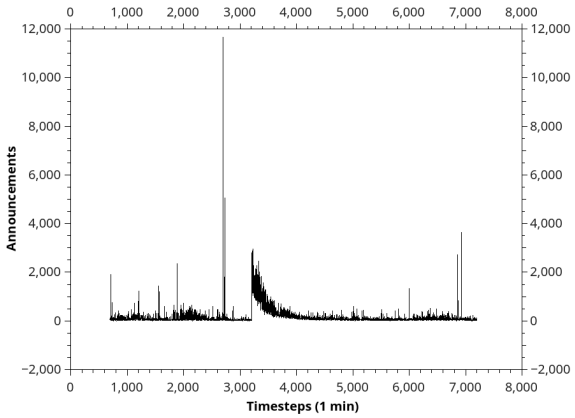


Fig. 4.4 Number of announcements timeseries during an anomaly

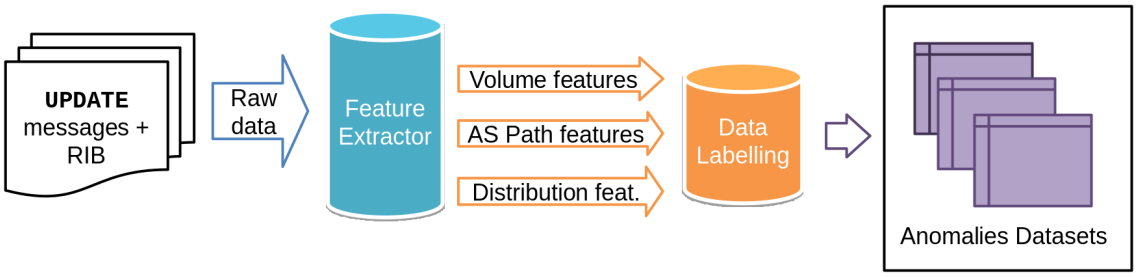


Fig. 4.5 BGP dataset generation framework

performs peak detection in the time series of each feature, applying the method used by Lo et al. [100]. We provide a plot of each feature along with detected peaks. It also accepts as input a period of time, which were collected in step 1, and plot it along with the peaks in order to analyze which peaks match to the proposed anomaly period. Figure 4.6 depicts an example, where the black line is the number of announcements, the dotted blue line is the proposed anomaly period and the diagonal cross marks are the detected peaks. It is possible to observe that the number of peaks increases inside the proposed window, which can be adjusted in order to avoid mislabeling of samples.

Considering that there are 45 features, as described in Section 4.1, (plus the *edit distance with k-value*, which is configurable and will add k more features), the process of analyzing

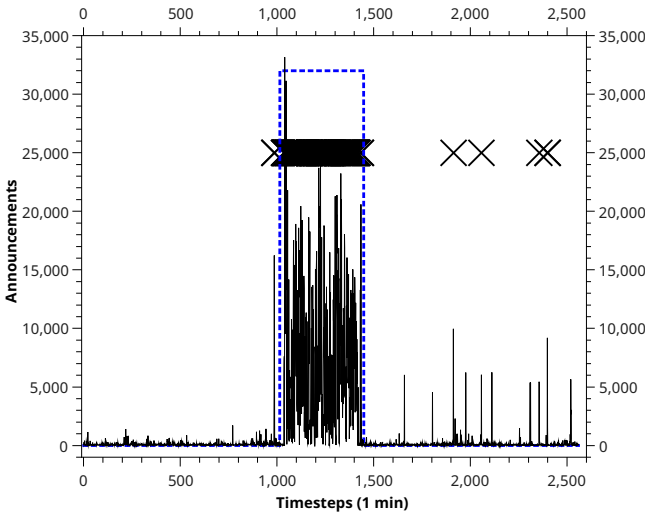


Fig. 4.6 Example of peak detection

each feature time series may be time consuming. In order to assist this task, we applied feature importance methods in order to prioritize the most relevant features and discard features with low variance.

Using this workflow, we generated labeled datasets for the events described in section 2.3 and found the collectors, peers, start and end time described in table 4.5. In order to add diversity to the data, we augmented our datasets by generating them with different timescales, considering time bins of 1, 5 and 10 minutes. At the end of this process, 9 events were considered, 27 collectors and ASes were capable of observing the anomalies, which generated 81 datasets with different timescales and 86400 samples.

4.3 Trend Analysis in Traffic Behavior

In this section we describe the general trends that we observed in anomalous behavior in comparison with regular traffic and among different classes of anomalies.

4.3.1 Anomaly and regular traffic

We compared the mean and median of each feature before, during and after an anomaly. Next, we list the features that showed significant changes in their behavior during the anomalies in most of generated datasets, along with our observations:

- **Announcements:** All datasets showed a significant increase in the number of announcements during anomalies, however, other features are necessary to detect an anomaly, since there are non-anomalous events that may cause a surge in this feature (*e.g.*, after a switch re-connection, the full RIB is sent).
- **Withdrawals:** The number of withdrawals increases along with the number of announcements.
- **Origin changes:** We did not observe a dominance of a specific ORIGIN type during anomalies, but most of dataset showed an increase in the number of origin changes indicating that during anomalies the distribution of origin types often changes.
- **Rare ASes average:** Normally, an anomaly involves a large volume of BGP updates involving a limited number of ASes, which causes decrease in the rare ASes average.
- **Distribution - New announcements:** The proportion of plain new announcements tends to decrease, suggesting that most message updates are flaps, re-announcements or implicit withdrawals.

4.3.2 Direct, indirect and outage anomalies

We applied the same method described in the last subsection to analyze which trends could be observed among the anomaly classes listed in section 2. These observations can be used to leverage anomaly classification and enforce mitigation techniques to each type of anomaly.

Outages

- **Number of rare ASes:** During outages, the quantity of rare ASes increases, possibly because BGP speakers start to announce rarely used backup paths.

- **Distribution - Announcements vs Withdrawals:** During outages, a surge of announcements to backup routes decreases the proportion of withdrawals.
- **Distribution - Explicit Withdrawals and Flaps:** Besides explicit withdrawals, the proportion of flaps also decreases because the paths affected by the outage remains unavailable during the anomaly period.
- **Distribution - Longer paths and Shorter paths:** We observed that during outages, the proportion of longer and shorter paths decreases, while the number of paths with the same length increased. This is possibly because the high volume of backup paths announced have the same length of primary paths.

Direct

- **Distribution - Shorter paths:** During direct anomalies, the proportion of shorter paths increases. Possibly, this occurs because in this type of anomaly (*e.g.*, hijacking and misconfiguration) bogus routes are propagated inadvertently, which can be achieved only if the new routes are more attractive than the current ones.

Indirect

- **Distribution - Implicit Withdrawals:** The only trend that could be observed only in indirect anomalies is that the distribution of implicit withdrawals remained the same in most of datasets, whereas most of datasets from other types of anomalies increase the proportion of implicit withdrawals and a minority where it decreases.

Summary: In this chapter, we described an implementation of the dataset generation tool that converts raw BGP control plane messages to timeseries of features. These features are capable of capturing relevant behavior in the context of anomaly detection, and possibly in other areas as well. The generated datasets were made publicly available and are suited to be used by most of machine learning models [20][34]. The feature extraction and dataset generation tools were also made publicly available in an open-source platform and can be extended in order to support more features.

Table 4.5 Anomalies datasets

Anomaly	**Class**	**Collec.**	**Peers**	**Start**	**End**
AS9121 RTL	Direct	rrc05	13237, 12793, 1853	9:20 UTC 24/12/2004	10:03 UTC 24/12/2004
AWS Route Leak	Direct	rrc04	6939	17:10 UTC 22/04/2016	20:00 UTC 22/04/2016
Malaysian Telecom	Direct	rrc04	513, 2509, 20932	8:42 UTC 12/06/2015	10:24 UTC 12/06/2015
Code Red v2	Indirect	rrc04	513, 559, 6893	10:00 UTC 19/07/2001	20:00 UTC 19/07/2001
Nimda	Indirect	rrc04	513, 559, 6893	13:00 UTC 18/07/2001	12:00 UTC 21/07/2001
Slammer	Indirect	rrc04	513, 5596, 893	5:31 UTC 25/01/2003	19:59 UTC 25/01/2003
Moscow Blackout	Outage	rrc05	1853, 12793, 13237	4:40 UTC 25/05/2005	7:40 UTC 25/05/2005
Japan Earthquake	Outage	rrc06	10026, 2497	9:13 UTC 11/03/2011	15:39 UTC 11/03/2011

This is a major contribution of the current work, given that there are no comprehensive BGP anomaly datasets, nor feature extraction tools, publicly available. We believe that this contribution will assist future research in BGP behavior and support the growing trend of applying machine learning methods to networking problems. We also proposed novel features that allows intuitive observation of BGP behavior trends, as depicted in section 4.3. These observations can be used to leverage anomaly detection and classification mechanisms and insights that can be used in the mitigation of each type of anomaly.

Chapter 5

Anomaly Detection and Classification

In this chapter we present our proposed deep neural network architectures for anomaly detection and classification. We decided to develop two separate models for detection and classification, rather than a single model that performed both tasks end-to-end. This decision was made in order to alleviate the complexity of both tasks, as early experiments and data analysis showed that each problem have different characteristics and relevant features. We also present an evaluation of the proposed methods and comparison with different baselines.

5.1 Anomaly Detector

During our research and analysis of BGP data collected, we identified three desirable features for an effective anomaly detection architecture: 1) It must be able to detect traffic trends of different scales, high- and low-frequency components; 2) detect spatial correlations in a given time series without considering its location in the sequence, 3) detect short- and long-term dependence in sequential data.

Our proposal is to compose a deep learning model using different neural network architectures, extracting the desirable features from the input data by exploiting the advantages specific to each architecture. Previous works obtained effective results from the integration of different neural network architectures, such as recurrent neural networks and convolutional networks [97] for object recognition and wavelets and convolutional networks [9, 52].

We compose a neural network with layers based on the neural networks architectures discussed in Chapter 2 because their properties match with the desired features. Namely, we compose our architecture by the following layers:

Wavelet deconvolution layer:

This layer serve as a signal denoising layer. In the forward pass, this layer calculates the wavelet transform on the input signal and pass the transformed signal to the next layers. In the backward pass, it computes the gradients of the loss function with respect to the scale parameters, adjusting the parameters to values that best fit the target data. Adding a wavelet layer as the first layer in our model allows the model to learn the relevant spectral content in the input with backpropagation, leading to a reduction of the number of hyperparameters to be optimized. As Figure 5.1 depicts, each feature is processed separately by a wavelet neuron and produces a denoised version of the signal.

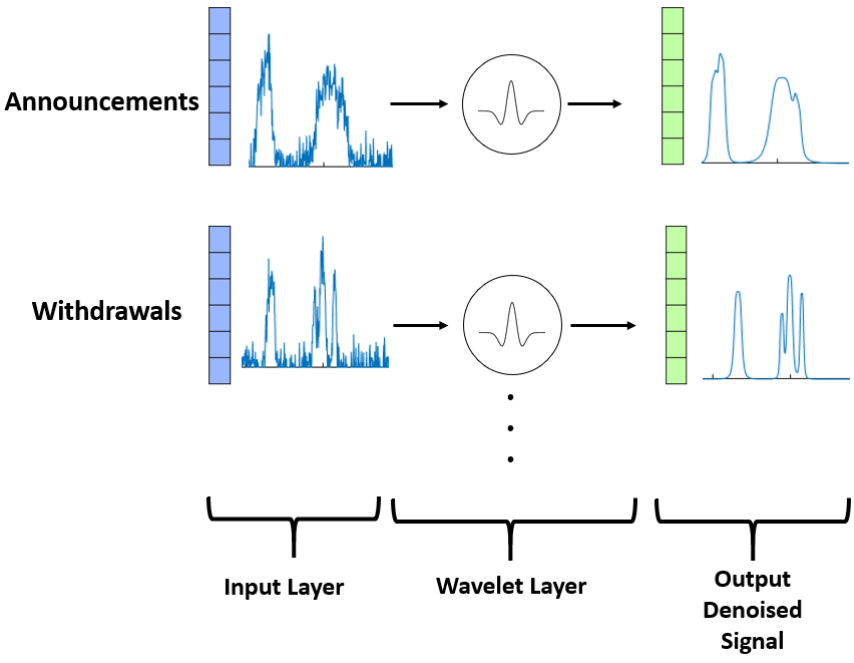


Fig. 5.1 Wavelet transform layer

Convolutional neural networks (CNN):

We based our CNN model in the approach used by Zheng et al. [161], each feature is mapped to a channel of an independent convolution and pooling layers in order to learn features individually, as depicted in Figure 5.2. Then, the learnt features of each channel are passed to the next layer. A deep CNN will comprise repetitions of the convolutional layer and pooling layer combination. In each pooling layer, the data dimension is reduced, enabling the model to learn more general properties of the data. This type of characteristic is suited to the BGP anomaly detection problem because anomalous data have spatial features (e.g. sudden spikes, prolonged peaks) of different scale.

In our model, we must tune a set of hyperparameters, such as the number of layers, number of neurons, kernel width, stride and pooling function, which were explained in Chapter 2.

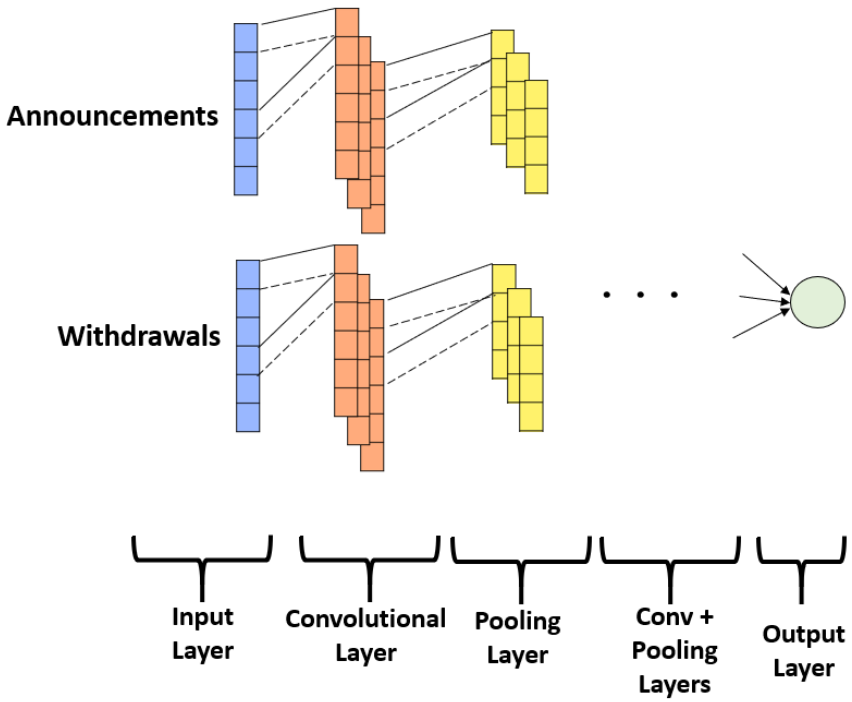


Fig. 5.2 Convolutional neural network for 1D classification

Long Short-Term Memory Networks:

Whereas the CNN layer is capable of detecting spatial features of the data, the Long Short-Term Memory (LSTM) layer add the capability of detecting long-term temporal dependence. The last layer of the CNN layer serves as the input layer to the LSTM layer(s). Similarly to the CNN, increasing the number of LSTM layers allow the capturing of more complex behavior. Figure 5.3 depicts an example of the LSTM network of our model with two hidden layers.

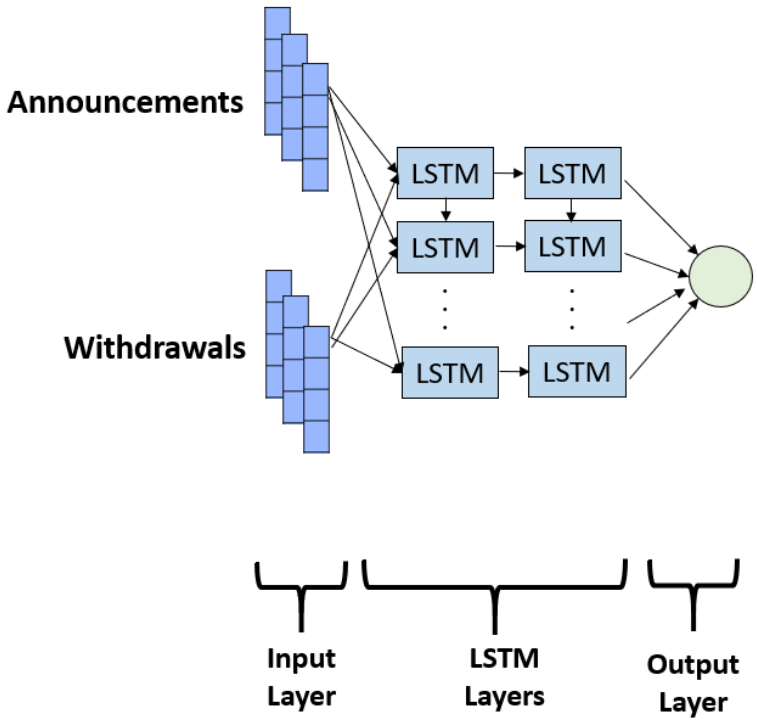


Fig. 5.3 Long-Short Term Memory network with multiple layers

Our architecture combines these layers in order to take advantage of their discussed features. Figure 5.4 depicts the proposed neural network architecture. We also enable a dropout parameter[141] in our model, which is a regularization method, where each neuron has a p probability to turn off inputs from a layer. This method prevents the network from overfitting to training data during the network learning process and not generalizing well for test data.

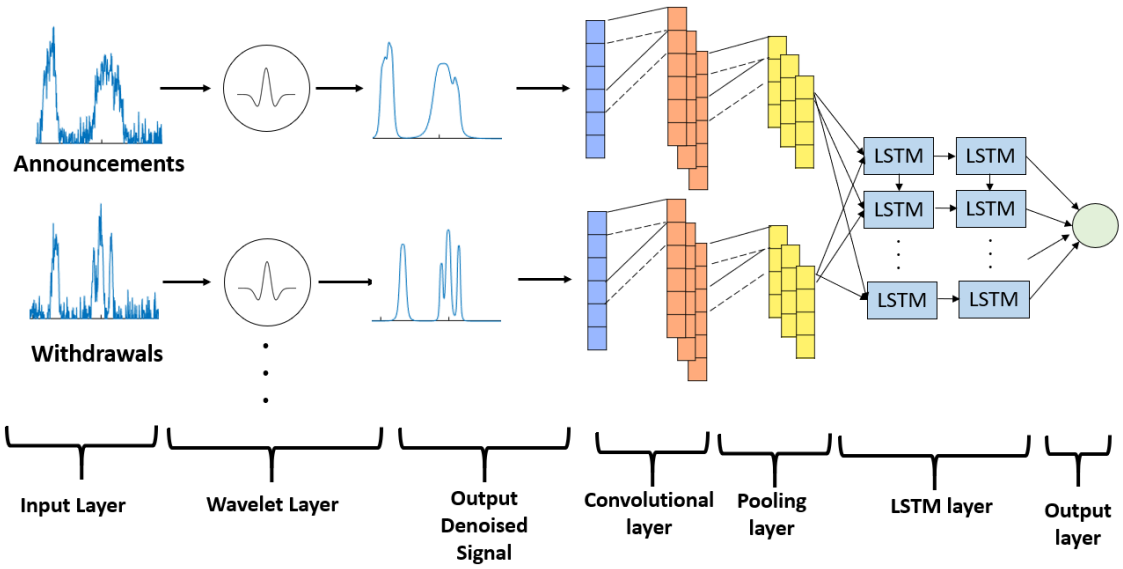


Fig. 5.4 Possible configuration of the proposed architecture

5.2 Anomaly Classifier

Once an anomaly is detected, the task of anomaly classification takes place. We observed that in this task the most dominant factor is the distribution of types of BGP updates in a time span, for example, during outages the proportion of announcements for paths that are *longer* or with *same distance* increases, whereas during hijacking anomalies (*i.e.* direct anomalies) the proportion of *shorter paths* tends to increase. For anomaly classification we propose a stacked LSTM network.

A simple LSTM network is proposed because we observed that anomalies from the same class may have different spatial features, which may cause a convolutional layer to classify two datasets from the same type of anomaly as different. We used the following architecture, depicted in figure 5.5.

Even though this model is simpler than the model used for BGP anomaly detection, experiments showed that it was capable of achieving good results, most likely due to the importance of the distribution features, which were proposed in this work. These results are also important because, to the best of our knowledge, this is the first method that differentiate BGP anomaly classes, addressing a major gap in current BGP anomaly research.

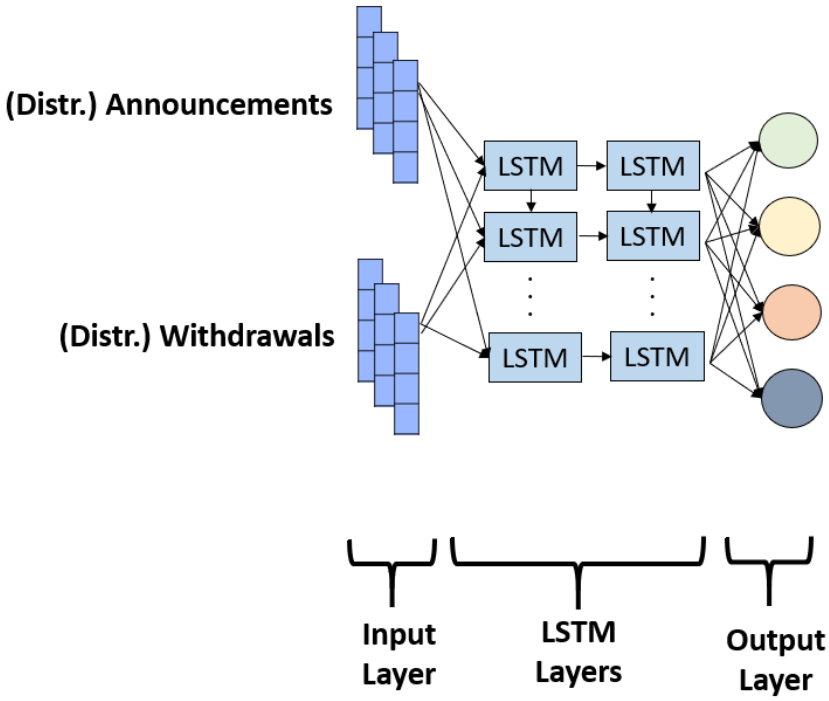


Fig. 5.5 LSTM network for BGP anomaly classification

5.3 Results and Discussion

This section is organized as follows. In Subsection 5.3.1, we describe the metrics chosen to evaluate the models and our reasoning. In Subsection 5.3.2, we describe the techniques used during training and validation of the developed models, such as grid search and early stopping [113]. In Subsections 5.3.3 and 5.3.4, we describe the neural network architectures used, the best hyper-parameters found, the training process and results found for anomaly detection and classification.

5.3.1 Metrics

The predicted values of a classifier are categorized as true positives, true negatives, false positives and false negatives. A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts

the negative class. Conversely, a false positive is an outcome where the model incorrectly predicts the positive class and a false negative is an outcome where the model incorrectly predicts the negative class. Hereafter, these metrics will be referred as TP, TN, FP and FN, respectively.

Accuracy is a metric that summarizes the performance of a classifier against all classes and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

In unbalanced datasets, the accuracy may be deceiving because a high accuracy does not necessarily mean that the number of true positives and false negatives is acceptable for a given purpose. For example, anomalous behavior seldom occurs, thus a model that always classifies traffic as not-anomalous will be correct for the most part of time, leading to a high accuracy, but it will have no use since an actual anomaly will never be detected. Precision, recall and F1-score aim to cope with this bias. Precision and recall increase when the number of true positives gets higher, but precision (equation 5.2) is affected when the number of false positives decreases, whereas recall (equation 5.3) is affected by the number of false negatives. Therefore, precision indicates whether a model hits *only* the right positive cases and recall evaluates whether a model hits *all* the right positive cases. Unfortunately, precision and recall are often in tension. Improving precision frequently reduces recall and vice-versa. To fully evaluate the effectiveness of a model, it is necessary to examine both precision and recall. F1-measure (equation 5.4) is a metric that aims to summarize both measures by calculating the harmonic mean of recall and precision.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

When we have multiple classes, computing the precision, recall and F1-score is not so straightforward as in binary classification. There are two approaches to compute these metrics, F1-micro and F1-macro, defined as follows:

In micro-averaging, precision, recall and F1-score are computed globally over all category decisions. Micro-averaged F-measure gives equal weight to each class and, therefore, it calculates an average over all the classes. It tends to be dominated by the classifier's performance on common categories. Below we present the formula of micro-averaged recall, given that $l = number\ of\ classes$ (precision and F1-score might be derived using the same reasoning):

$$Recall_{macro} = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} TP_i + FN_i} \tag{5.5}$$

Conversely, in macro-averaging, precision, recall and F1-score are computed locally over each category first and then the average over all categories is taken. Macro-averaged F-measure gives equal weight to each category, regardless of its frequency. It is influenced more by the classifier's performance on rare categories. The macro-averaged recall formula is:

$$Recall_{micro} = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i + FN_i}}{l} \tag{5.6}$$

For example, given a multi-class classification model with the following results: Class 1 have 1 TP, 1 FP, precision = 0.5; class 2 have 1 TP, 1 FP and precision = 0.5; class 3 have 10 TP, 90 FP and precision = 0.1.

A macro-average precision will then compute $\frac{0.5+0.5+0.1}{3} = 0.367$, whereas the micro-average will compute $\frac{1+1+10}{2+2+100} = 0.115$

Summary: Given that anomaly detection is a intrinsically unbalanced problem, because anomalies are deviations of a regular, much more frequent, behavior, we include precision, recall and F1 in our evaluation and analysis. Although F1 is useful to summarize the performance of a model, during our experiments we faced scenarios where models with similar F1-score had significantly different precision and recall values. Thus, a complete

analysis must include all metrics. Macro and micro measures might greatly differ if the classes are unbalanced, if it is important to perform well on all classes, regardless the size of each class, or if the classes are balanced, macro is preferred. If the classifier must perform better on bigger classes, micro average must be used.

5.3.2 Methodology

During our experiments we used a set of techniques to optimize the models and to understand their behavior and dataset trends.

When evaluating a model, we split our datasets in train, validation and test sets then we feed the model with the training set and evaluate its performance on validation and test sets. The *training set* is used to train the model initially. Once the model is trained, its prediction is evaluated using the *validation set*. If the model performs poorly, the model is adjusted until its accuracy, precision and recall achieve an acceptable level. At last, in order to guarantee that the model is not biased towars the training and validation sets, the model is evaluated using data that was not used before through a *test set*. The obtained results in each set were used to decide whether a following action must be taken.

If a model does not perform well on training data, it means that the model was not able to capture the data behavior, leading to underfitting. If a model performs well on training and poorly on validation set, it means that it may have a variance problem, being too complex and suffering from overfitting. If a model achieves good results in train and validation sets but does not perform well on test set, it means that train, validation and test sets have a different data distributions, which indicates that the model may not perform well on real data.

When those behaviors are found, the following actions must be taken:

- High *training error* may be mitigated by using a more sophisticated model (e.g. adding layers, adding neurons), capable of capturing more complex behavior, or by training the model longer.

- High *validation error* indicates overfitting, thus, we use regularization methods (e.g. adding dropout layers), simplify the architecture, and generate more datasets and events.
- In order to cope with high *test error*, we add more data and make sure that train, dev and test sets have the same distribution, which is representative of real-world data.

Figure 5.6 depicts a flowchart of these actions in a time sequential order.

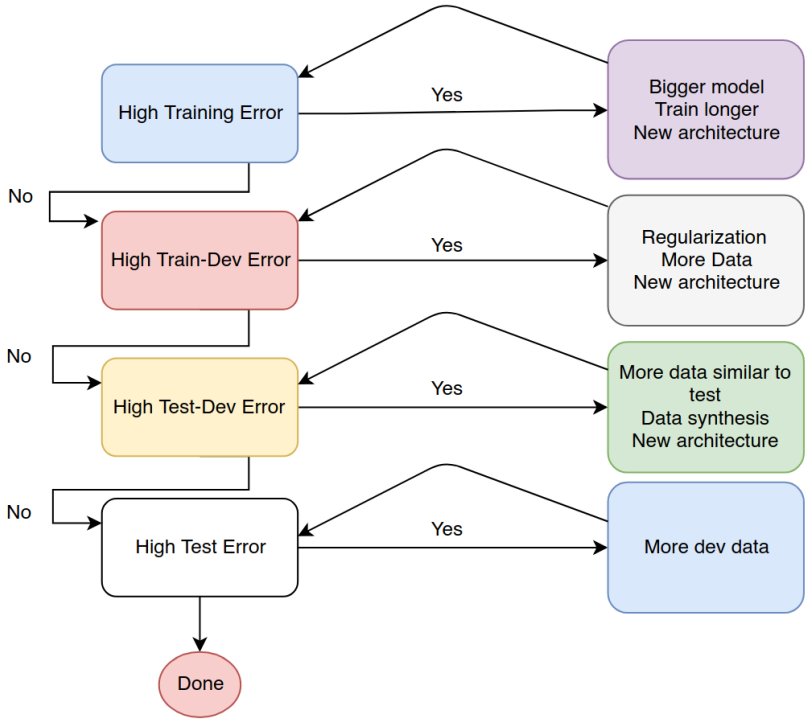


Fig. 5.6 Machine learning training flowchart [156]

5.3.3 Anomaly Detection

We performed experiments with different models and observed their behavior during training and validation for different combinations of features, hyperparameters and datasets. Even though previous machine-learning-based efforts suffered from problems previously discussed in Chapter 3, initially, we assumed that a result for a model would be acceptable if it achieved similar values to those previously found.

Hyperparameter	Values
C (Penalty parameter)	0.0001, 0.001, 0.01, 0.1, 1, 10
γ (gamma)	0.0001, 0.001, 0.01, 0.1, 1
Kernel	Linear, Polynomial, RBF, Sigmoid

Table 5.1 SVM hyperparameters

	Precision	Recall	F1-score
Code Red II	8.80%	26.55%	13.22%
Nimda	10.03%	9.65%	9.83%
Slammer	27.21%	23.19%	25.04%
Japan Earthquake	19.01%	44.25%	26.60%
Moscow Blackout	27.02%	29.82%	28.36%
AS 9121 Route Leak	16.19%	37.61%	22.64%
Malaysian Telecom	4.30%	14.21%	6.60%
AWS Leak	3.17%	19.48%	5.45%

Table 5.2 SVM results

Support Vector Machine (SVM)

As a baseline, we used a SVM model [147], replicating experiments performed by Cosovic et al. [27]. Because SVM was not designed for handling sequential data, each training sample is processed individually. Experiments used the SVM implementation from scikit-learn toolbox [118] and all combinations of the hyperparameters listed in table 5.1 were performed:

We have chosen this initial set of hyperparameters values empirically, considering the functioning of the model (e.g. a penalty value too large will likely lead to overfitting because outliers will greatly affect the model training) and most commonly used values. We trained the models with all available datasets, with the exception of the validation dataset. For each scenario, the model processed 86,400 samples (a sample contains the features calculated during a given time bin, e.g. 1 minute or 5 minutes) minus the samples from validation and test datasets, and took approximately 130 minutes to converge for each set of hyperparameters. The best parameters found were $C = 0.1$, $\gamma = 0.1$, $kernel = RBF$ and the results are listed in table 5.2.

These results are inferior to those previously reported by Ding et al. [34]. As discussed in Chapter 3, Ding et al. does not consider a comprehensive range of dataset events, it just considers Slammer, Nimda and Code Red II, which belong to the *indirect* class. Besides, SVM, as well as many classic machine learning approaches, handles each training sample independently and ignores sequential dependency among feature values. The results show that the SVM model is not able to consistently detect anomalies, with the precision and recall for all datasets indicating that the model produces a high number of false positives and false negatives. This behavior might be observed in figure 5.7, which depicts the prediction of a model that was tested against the Slammer dataset. The black lines and dots represent the predicted anomalies and the red line is the real anomaly. We can observe that the model was capable of detecting the majority of the anomaly (increases *recall*), however, it also produced a large number of *false positives* (decreases *recall*).

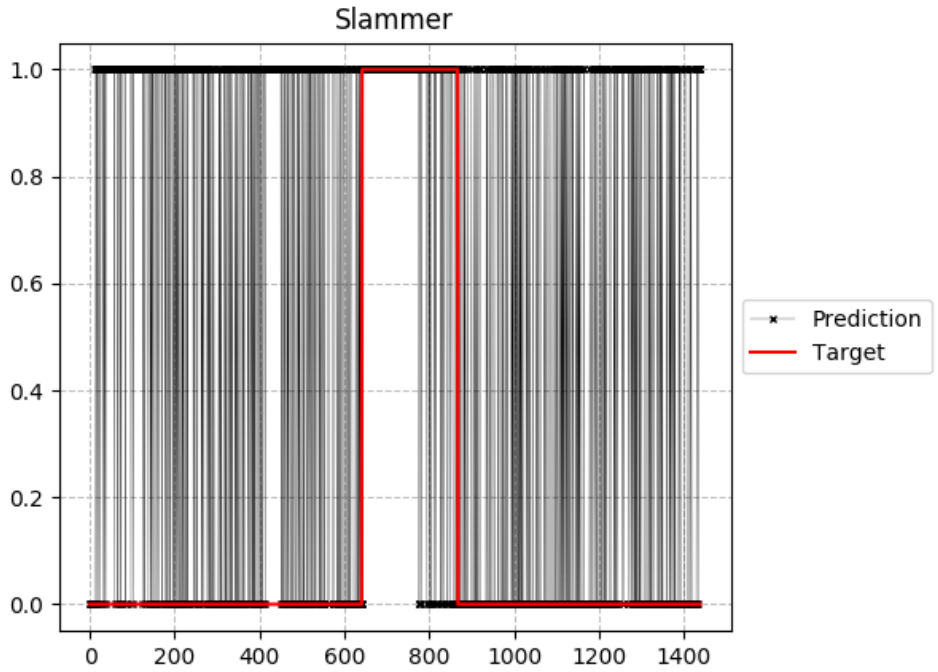


Fig. 5.7 SVM prediction with Slammer event as the testing dataset

Deep Learning

The training of deep learning models is time-consuming and the amount of time to train a network is directly proportional to the network size. Thus, we used increasingly bigger models, in order to assess whether such complexity is necessary to achieve good results. We used Long-Short Term Memory (LSTM), Convolutional Neural Networks (CNN) and a Wavelet Layer (WL). We implemented our models using the Keras Library [23, 60] with the TensorFlow [1] backend. The wavelet layer was adapted from the implementation of the WaveletDeconv library [76], which was also built on top of Keras.

Long-Short Term Memory

We performed experiments with a simple LSTM network in order to replicate the approach used by Ding et al. [34]. Differently from SVM, LSTM models can receive data sequentially and detect temporal properties of a feature. In order to exploit this LSTM characteristic, we feed the model with the whole sequence of one event once, the model adjust its weights through backpropagation, then another event is passed to the model, and so on. One training *epoch* is the time necessary for a model processing all samples from all events once. In early experiments, we observed that if a fixed sequence of events is passed to the model, the last events have greater impact than the first events. We handled this by shuffling the order of events in each round of training. We also parameterized our LSTM layers to be stateful [127], so the last state of a LSTM cell for each sample in a batch will be used as initial state for the sample of the following batch.

As previously discussed, the fine tuning of hyperparameters requires the empirical experimentation of different values. The hyperparameters include the number of neurons of each hidden layer, number of hidden layers, learning rate, lag, gradient descent optimizer and features used.

We selected the possible number of neurons using a rule of thumb that the number of hidden neurons should be between the size of the input layer and the size of the output layer [64]. Many problems can be efficiently modeled using up to two layers, because they are sufficient to represent an arbitrary decision boundary with rational activation functions [64].

Considering this, up to three layers were used, in order to evaluate whether a higher number of layers yields better results. Many approaches to optimize the gradient descent were proposed [131] and the type of optimizer chosen by the network is also a hyperparameter of the network. For many classification problems, the current state-of-the-art are optimizers that consider the momentum and context of the gradient descent, in order to converge more quickly. Two optimizers with these properties were used RMSProp [144] and Adam [79] and their parameters were set o their default values [24].

We used early stopping to control the duration of training, i.e., we evaluated the F1-score for the validation set during the training and as long as it enters in a downward trend, training is interrupted. This strategy led to different durations depending on the models and training/validation/test datasets, ranging from 100 to 200 epochs. We performed experiments with the parameters listed in table 5.3 and each round of training took from 9 hours for the smallest models to 18 hours for the large models (3 layers, 40 neurons, $lag = 15$).

Hyperparameter	**Values**
Number of neurons	10, 20, 40
Hidden layers	1, 2, 3
Learning rate	0.001, 0.01, 0.1, 1
Lag	0, 5, 10, 15
Optimizer	RMSProp Adam
Features	AS path and volume Distribution Distribution and volume AS path, volume and distribution

Table 5.3 LSTM hyperparameters

The best hyperparameters found were $learning\ rate = 0.01$, $lag = 10$, Adam optimizer and volume + distribution features. The number of neurons and number of hidden layers yielded similar results, however, in order to secure the ability of the network to generalize is

preferable that the number of nodes and layers is minimized. If the network is too large, it increases the chances that the network will overfit, becoming a memory bank that can recall the training set to perfection, but does not perform well on samples that were not part of the training set. The increase in performance considering only distribution and volume features shows that these novel features are able to capture the anomaly behavior as well as it reduces the input size from 42 to 27 features, reducing the network complexity. Table 5.4 lists the results of the LSTM model after fine-tuning, a network with 1 hidden layer with 20 neurons.

Table 5.4 Results from LSTM after hyperparameter fine-tuning

	Precision	**Recall**	**F1-score**
Code Red	74.8%	97.7%	84.7%
Nimda	94.2%	43.4%	59.4%
Slammer	87.7%	89.2%	88.5%
Moscow Blackout	74.1%	89.5%	81.1%
Japan Earthquake	87.1%	97.4%	92.0%
AS9121 Leak	37.9%	100.0%	55.0%
Malaysian Telecom	21.3%	100.0%	35.1%
AWS Leak	34.1%	98.6%	50.7%

Summary: In order to compare with SVM, we plot the prediction of the LSTM that was tested against the Slammer dataset, after hyperparameter tuning, in Figure 5.8 of a model. We can observe that the model was capable of detecting the majority of the anomaly with a low number of false positives and negatives. The results of test datasets are comparable to those found on the literature with a $F1 > 80\%$ for half of the datasets.

Even though the precision for the Nimda dataset is high, the recall is low, which also yields a F1-score lower than others results found. Upon further analysis, we found that most of anomalous data points that are close to the end of the anomaly event were predicted as regular traffic, leading to a high number of false negatives, thus the low *recall*. Since the Nimda event is the longest anomaly in our datasets, lasting 3 days whereas other events lasted less than 3 hours, we hypothesize that a) the end of the dataset might contain regular traffic when the BGP volume slowly returns to its previous regular state, or b) the LSTM network have difficulty in detecting dependencies . In case of a) further refinement of the

dataset might be necessary and in case of b) anomaly events with duration of days must be added to our dataset pool.

AS9121 leak, Malaysian Telecom leak and AWS leak have high recall and low precision, thus, the model can predict correctly all anomalies but also yields a high number of false positives. These events belong to the *direct* class, indicating that the LSTM is not able to effectively capture all characteristics of such class. Further discussion will be provided in next subsections.

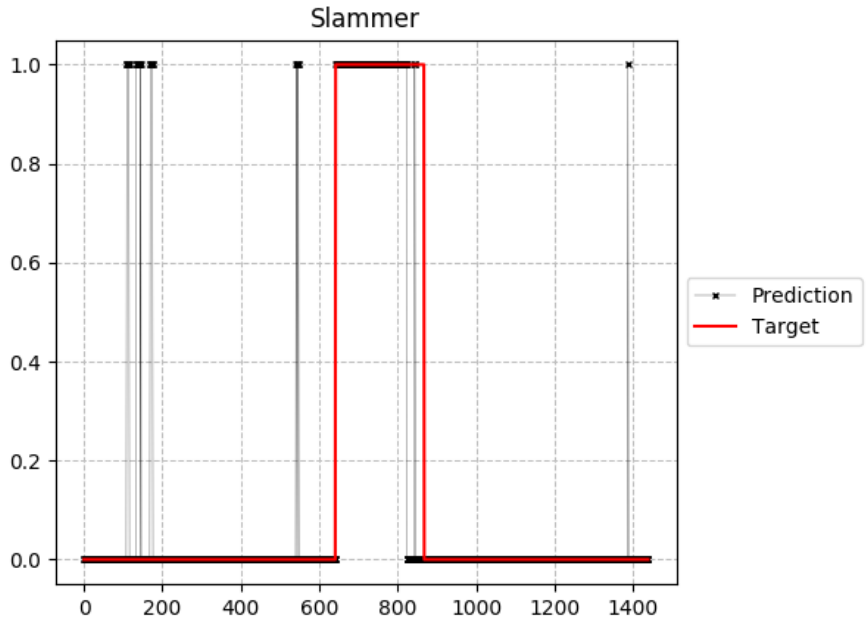


Fig. 5.8 LSTM prediction with Slammer event as the testing dataset

Convolutional Neural Networks

Similarly to LSTM, we feed the CNN model with one event at a time, the model adjust its weights through backpropagation, then another event is passed to the model, and so on. We also consider one training *epoch* as the time necessary for a model process all events once.

Besides the hyperparameters listed for LSTM (i.e. learning rate, optimizer, number of neurons, hidden layers, and features), the tuning of a CNN also includes the kernel width, stride and pooling function. We also used early stopping during the training of the CNN

model, which led to different durations, ranging from 150 to 200 epochs. We performed experiments with the parameters listed in table 5.5 and each round of training took from 6 hours for the smallest models to 16 hours for the largest models.

Hyperparameter	**Values**
Convolutional + pooling layers	1, 2, 3
Learning rate	0.001, 0.01, 0.1, 1
Kernel size	3, 5, 9, 11
Stride	1, 2, 3
Pooling	Average pooling Max pooling
Optimizer	RMSProp Adam
Features	AS path and volume Distribution AS path, volume and distribution

Table 5.5 CNN hyperparameters

The best hyperparameters found were $learning\ rate = 0.01$, $stride = 1$, $kernel\ width = 5$, Adam optimizer, max pooling and volume + distribution features. We achieved best results with one convolutional layer and pooling layer. Table 5.6 lists the results of the CNN model after fine-tuning.

Table 5.6 CNN results after hyperparameter fine-tuning

	Precision	**Recall**	**F1-score**
Code Red	69.1%	83.3%	75.5%
Nimda	66.2%	68.3%	67.2%
Slammer	78.7%	71.2%	74.8%
Moscow Blackout	81.3%	76.3%	78.7%
Japan Earthquake	68.7%	74.9%	71.7%
AS9121 Leak	45.7%	48.6%	47.1%
Malaysian Telecom	31.5%	100.0%	47.9%
AWS Leak	41.2%	47.6%	44.2%

Summary: With respect to F1-score, the CNN model produced similar results to those of LSTM for most part of the datasets, however further analysis of precision and recall showed noteworthy differences. In CNN results it is possible to observe an increase in precision

	Precision	**Recall**	**F1-score**
Code Red	80,3%	82,1%	81,2%
Nimda	75,4%	85,8%	80,3%
Slammer	90,1%	87,4%	88,7%
Moscow Blackout	65,8%	75,2%	70,2%
Japan Earthquake	81,8%	78,2%	79,9%
AS9121 Leak	45,1%	40,1%	42,5%
Malaysian Telecom	34,1%	100,0%	50,9%
AWS Leak	31,9%	97,9%	48,1%

Table 5.7 MS-LSTM results

in comparison with the LSTM results with respect to the AS9121 and AWS leak datasets. We hypothesize that this indicates that model can capture spatial properties, such as sudden spikes and prolonged large volumes of messages, that the LSTM model is not able to detect. Given the nature of CNN, the model might achieve a higher results if it could learn an optimal time scale decomposition of the input signal, or if more layers were added. Our results also indicates that it may be necessary a larger training set and time in order to improve accuracy, precision and recall.

MS-LSTM

The related work that most closely relates to our model is the Multi-Scale LSTM (MSLSTM) proposed by Cheng et al. [20]. The authors implemented [21] a detection mechanism that integrates wavelet transforms and neural networks. Differently from our model, MS-LSTM apply wavelet transforms before feeding the data to the LSTM network. Instead of learning the best time resolution, MS-LSTM extract generates multiple time resolutions. The MS-LSTM architecture is depicted in Figure 3.8. The model was trained and tested only with indirect anomaly events, and with volume and AS path features, thus, it was not possible to compare it directly with our model. We performed adaptations on the original code in order to support the distribution features and trained for our generated datasets.

The best hyperparameters found were $learning\ rate = 0.01$, $lag = 40$, Adam optimizer and volume + distribution features. The results found are presented in the Table 5.7.

Summary: The MS-LSTM approach yielded better results than using LSTM layers with raw input data without preprocessing. This improvement in results shows that include different time resolutions may help the model to detect temporal pattern with optimal time scale. Zheng et al. argue that the selection of time scale has great impact on the model performance. One disadvantage of this method is that it still requires the operator to use fixed time scales, which increase the model complexity and may cause the optimal scale to be missed if it is not in the fixed set of time scales.

Our model

Our CNN results indicate that a CNN model can capture behaviors that a single LSTM network is not able, but more data may be necessary in order to extract a robust time resolution. This might be achieved through the use of wavelet transforms. As previously discussed, the integration of wavelet transforms and LSTM is capable of improving the anomaly detection model performance [20]. However, such model require a preprocessing of the input data using fixed translation and scaling parameters of wavelet transform. The tuning of these hyperparameters would increase the complexity of the model. Instead, in our proposed model, we added a Wavelet deconvolution layer, based on the implementation of Khan et al. [77], which is capable of automatically finding the best scaling and translation parameters through backpropagation.

Considering the desired properties of Section 5.1, we propose a deep learning model that integrates a wavelet deconvolution layer with CNN and LSTM. We connect these models in a linear fashion. Previous work [78] indicate that the integration of CNN and LSTM can yield robust performance for anomaly detection.

The wavelet deconvolutional layer does not require any additional hyperparameter. For the already discussed parameters, we have used the optimal values used for LSTM and CNN that achieved best results. We also used early stopping during the training and duration ranged from 200 to 250 epochs. Each round of training ranged from 14 to 16 hours, depending on the size of the training datasets. The results are listed in Table 5.7.

Table 5.8 WD + CNN + LSTM results

	Precision	**Recall**	**F1-score**
Code Red	81.1%	89.5%	85.1%
Nimda	73.3%	87.6%	79.8%
Slammer	88.1%	90.3%	89.2%
Moscow Blackout	74.3%	81.4%	77.7%
Japan Earthquake	91.7%	87.9%	89.8%
AS9121 Leak	60.1%	65.3%	62.6%
Malaysian Telecom	33.9%	100.0%	50.6%
AWS Leak	65.8%	91.0%	76.4%

Table 5.9 Comparison of BGP anomaly detection methods

	LSTM	**CNN**	**MS-LSTM**	**Our model**
Code Red	84,74%	75,54%	81,19%	85,09%
Nimda	59,44%	67,22%	80,28%	79,79%
Slammer	88,46%	74,76%	88,71%	89,19%
Moscow Blackout	81,06%	78,72%	70,16%	77,69%
Japan Earthquake	91,96%	71,67%	79,94%	89,76%
AS9121 Leak	54,97%	47,09%	42,49%	62,59%
Malaysian Telecom	35,12%	47,91%	50,86%	50,62%
AWS Leak	50,67%	44,17%	48,12%	76,37%

Summary: With respect to F1-score, we can observe an increase in the majority of datasets, indicating that the model was capable of retaining, at least, part of the advantages from both CNN and LSTM. However, results for part of the datasets indicate that improvement is still needed. Upon closer analysis of the datasets with bad results ($F1 < 70\%$), we identified two problems: the duration of some direct anomalies is short and some collectors produce noisy data. The AS9121 Route Leak lasted for less than an hour, which might lead to confusion with BGP messages surge caused by peers reconnecting and sending their full RIB. During the labeling of some datasets, we also observed that traffic spikes are present for the most part of the observed time period, differing from the behavior of other collectors for the same time period, indicating the possible influence of unknown factors. Since both of these problems are observed in real-world data, a possible mitigation is the dataset generation for more events and collectors. We compare the F1-measure of all neural network models trained in Table 5.9.

5.3.4 Classification

Initially, in order to have a baseline performance we trained a SVM model for classification, however, the model was not able to converge. After multiple runs, in all cases, we observed that the model was dominated by the larger class, predicting all samples to the most populated class. We experimented with the hyperparameters 5.1 and higher values of the penalty parameter in order to enforce a correct classification for the underrepresented classes but no significant result was achieved. Considering more complex models, a LSTM network was

Hyperparameter	**Values**
Number of neurons	10, 20, 40
Hidden layers	1, 2, 3
Learning rate	0.001, 0.01, 0.1, 1
Lag	0, 5, 10, 15, 20, 25
Optimizer	RMSProp Adam
Features	AS path and volume Distribution Distribution and volume AS path, volume and distribution

Table 5.10 LSTM hyperparameters

trained and capable of yielding better results. The model was trained with the combination of the hyperparameters in table 5.3. The best hyperparameters found were $learning\ rate = 0.01$, $lag = 20$, Adam optimizer and volume + distribution features. As in the LSTM detection, the number of neurons and number of hidden layers achieved similar results, using the same reasoning, the chosen model was composed by 1 hidden layer with 20 neurons.

The increase in performance considering only distribution and volume features indicates that the network was capable of finding the trends discussed in Section 4.3.2. For classification, the optimal lag hyperparameter value ($lag = 20$) is higher than for detection ($lag = 15$), indicating that it is necessary to consider a wider window to detect the trends that differentiate anomaly classes. Table 5.11 lists the results obtained. The validation sets were split in the following tuples, containing one dataset from each class: Code Red, Moscow

Blackout, AS9121 Leak; Nimda, Japan Earthquake, Malaysian Telecom Leak; Slammer, Japan Earthquake, AWS leak.

Summary: F1-micro indicates that even though larger datasets performs better than smaller datasets (e.g. AS9121 has the lowest F1-score), almost all datasets have F1-scores higher than 80%. Given that for classification, we assume that the input samples are anomalies, we must improve our detection mechanism. Surprisingly, classification of direct classes performed better than detection of the same classes, we hypothesize that the model is able to detect differences among samples of different classes when it knows they are anomalies, because it does not need to consider the noisy data, or this might be a gap in our model that ends up mixing properties of regular traffic with misclassified direct classes. For classification, Nimda dataset also showed the behavior where samples towards the end of the dataset are misclassified, indicating that further refinement of the dataset might be necessary and more anomaly events must be added to our dataset pool.

Table 5.11 LSTM classification results

	F1-micro	**F1-macro**
Code Red – Indirect	96.4%	88.2%
Nimda – Indirect	67.2%	89.1%
Slammer – Indirect	97.1%	90.7%
Moscow Blackout – Outage	87.8%	88.2%
Japan Earthquake – Outage	90.2%	89.1%
AS9121 Leak – Direct	79.0%	88.2%
Malaysian Telecom – Direct	83.6%	89.1%
AWS Leak – Direct	84.0%	90.7%

5.4 Use Case: SDN architecture

Even though the initial training of our models are performed offline, we propose an integration of these methods in a SDN ecosystem. Figure 5.9 depicts the external view of this architecture, where a logically centralized layer will process and reply to BGP messages from different peer ASes, in a seamless manner to the external observer, at the same time it leverages the abstractions provided by SDN to reason about network behavior and enforce its policies.

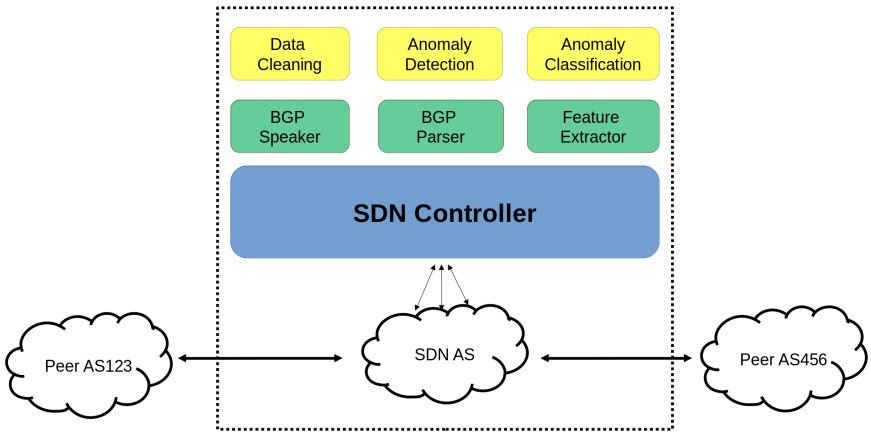


Fig. 5.9 Proposed solution data extraction and anomaly classification

Figure 5.10 expands the internal workflow of this model. After the receipt of the BGP raw messages from the BGP speaker, the BGP parser module is used to convert the messages to MRT format to enable feature extraction. After extracting the features, the data is normalized in a data cleaning step. Then, the anomaly detection module is responsible for detecting the anomaly. If an anomaly is found, the anomaly classification model identifies the type of anomaly.

The proposed SDN framework, due to its modularization, can be used with any machine learning model that is parameterized to a matching input and output.

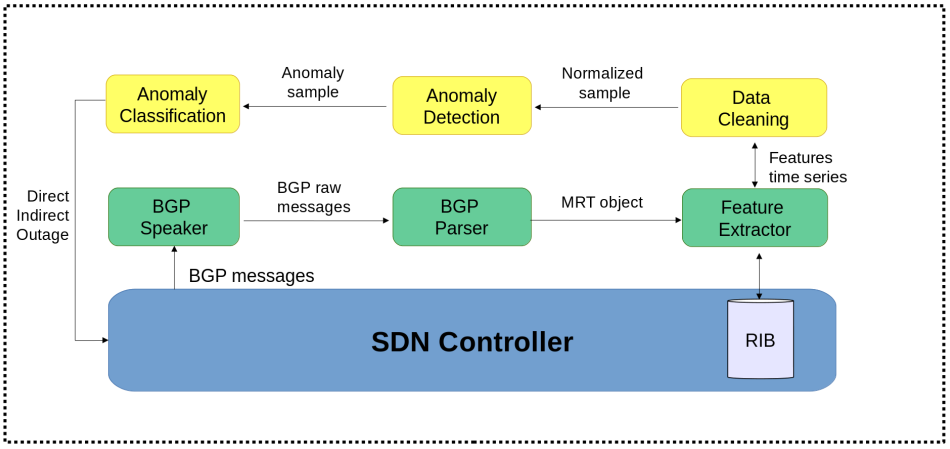


Fig. 5.10 Global view of the SDN scenario with anomaly detection

Chapter 6

Conclusion

This chapter concludes the thesis and present future work. The contributions are reinforced in section 6.1, future directions are discussed in section 6.2.

6.1 Conclusion

With almost two decades of research, BGP anomaly detection is an active and important area, with new approaches showing that there still room for improvement [124, 3]. However, the developed research lacks reproducibility as well as a way to compare different approaches directly, because each work uses its own anomaly dataset which can lead to significant disparities caused by different labeling and training. A public common dataset is important in areas such as handwritten digit recognition [32] and illegitimate traffic classification [143].

To the best of our knowledge, this work is the first to provide public labeled datasets of time series for the most used features and events for BGP anomaly detection. We expect that these datasets will allow researchers to focus on the BGP anomaly detection and classification models, instead of spending time extracting features and labeling data. We also made available our feature extraction tool, thus it can be extended with new features and events. Besides, the datasets are built in a format friendly for machine learning and deep learning approaches, which are increasingly popular in networking research, thus, such framework is timely and useful for the research community.

Based on the analysis of volume and AS path features behavior, we proposed novel features that represent the distribution of message types. These fields are useful to analyze BGP trends in a straight-forward manner. Our results also indicated that use of these features yields precision and recall equivalent or better than those found with volume and AS path features. For neural networks, an additional benefit of using distribution features is the reduction in network complexity, because there are less features than in the traditional Volume + AS path features.

For anomaly detection, the results indicate that our proposed customized neural network architecture demonstrated improvement in comparison with out-of-the-box methods, indicating that our model successfully exploited temporal and spatial properties of BGP anomalies. Since, the other methods [4] were not trained with the same datasets, it is hard to compare the approaches directly. We expect that our public datasets will allow such comparison in the future. The results obtained so far are encouraging and indicate to us that the addition of new events and point-of-views will improve the proposed model performance.

To the best of our knowledge, this is the first work to perform anomaly classification in an automated manner. Whereas previous methods only detected either a general class of anomalies or a specific class, our model performs a two-layered anomaly categorization. Such classification is the first step towards specialized mitigation techniques, endowing networks with the ability to perform closed-loop decisions in a timely manner.

As pointed in section 1.5, in the first part of this work, we developed a comprehensive survey [30] which identified main gaps in current research of fault management in SDN. However, due to the denial of access to data centers to collect data about failures the subject of this research had to be changed from fault management in software-defined networking to the current line of work. During this former phase, it was possible to realize the potential of SDN to simplify network management and to leverage the self-driving networks [42, 155, 75]. Our proposed SDN framework is a step towards the realization of this goal, with respect to BGP anomalous behaviors efforts.

6.2 Future Work

With the anomaly detection mechanism presented in this thesis and from knowledge obtained from the state-of-the-art review on BGP anomaly detection and classification, we open up new directions to be explored.

Regarding the BGP anomaly detection and classification learning models, we believe a myriad of new applications can be created. After classify the type of an detected anomaly, another step could be to extend the presented mechanism by inserting a new module to investigate the collected data and identify the root cause of the anomaly. This would be a natural direction for the BGP anomaly classifier, since the network operator or manager would obtain deeper understanding on the anomaly sources and reason about the correlation between the identified type of anomaly, the root cause and the observed effect on its network.

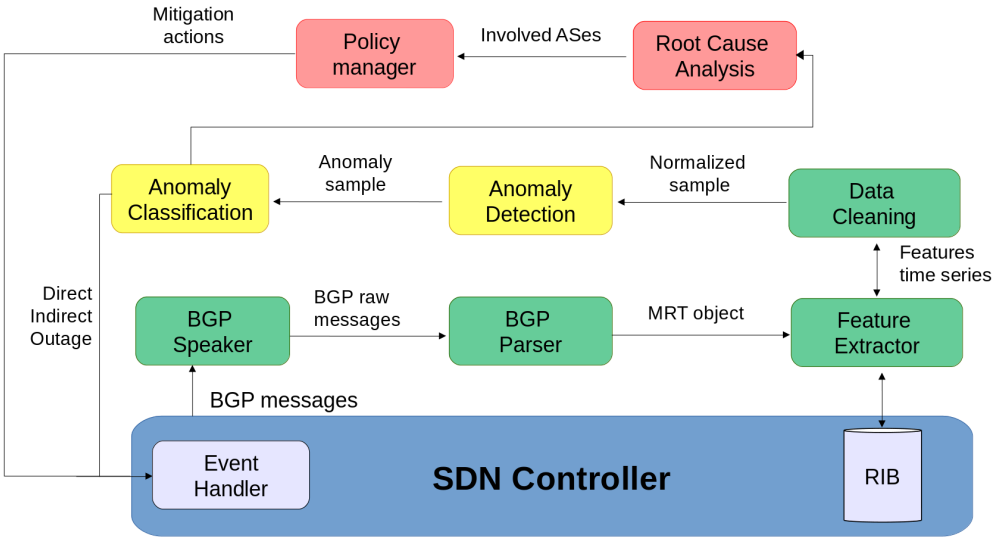


Fig. 6.1 SDN architecture with anomaly detection and mitigation

After the deployment of a BGP anomaly detection, classification and root cause analyzer, another layer of automation could be added to the mechanism by creating and adding an Anomaly Mitigation module. This module would free the network managers and operators from the responsibility to create actions to try to mitigate the cause of the anomaly. An interesting challenge happens when the identified cause of the anomaly originates from anther location outside the managed AS. This opens up further possibilities, such as cooperative

multi-agent systems could be proposed between different ASes, which could be supported by a Policy Manager module, since the policy interests and expressiveness from each AS must be preserved. A possible architecture including all these modules is depicted in Figure 6.1.

Another interesting and challenging future work is in the explainable AI research field. With the increasing in the datasets' size and with more complex deep-learning architectures, the output results from the models become every time more difficult to interpret and challenger to be improved. This lead us to the importance of two concepts: interpretability and explainability. The first give us a better understanding on the obtained results, and the latter helps to comprehend the mapping between the input and the output from the model [35].

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Aceto, G., Botta, A., Marchetta, P., Persico, V., and Pescapé, A. (2018). A comprehensive survey on internet outages. *Journal of Network and Computer Applications*, 113:36 – 63.

[3] Al-Musawi, B. (2018). *Detecting BGP Anomalies Using Recurrence Quantification Analysis*. PhD thesis, Ph. D. dissertation, Swinburne University of Technology.

[4] Al-Musawi, B., Branch, P., and Armitage, G. (2017). Bgp anomaly detection techniques: A survey. *IEEE Communications Surveys and Tutorials*, 19(1):377–396.

[5] Aslahi-Shahri, B., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., and Ebrahimi, A. (2016). A hybrid method consisting of ga and svm for intrusion detection system. *Neural computing and applications*, 27(6):1669–1676.

[6] Atwal, K. S. and Bassiouni, M. (2018). Deepsdn: Connecting the dots towards self-driving networks. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE.

[7] Bates, T., Chandra, R., Katz, D., and Rekhter, Y. (2007). Multiprotocol extensions for bgp-4. RFC 4760, RFC Editor. http://www.rfc-editor.org/rfc/rfc4760.txt.

[8] Bates, T., Chen, E., and Chandra, R. (2006). Bgp route reflection: An alternative to full mesh internal bgp (ibgp). RFC 4456, RFC Editor. http://www.rfc-editor.org/rfc/rfc4456.txt.

[9] Ben Chaabane, C., Mellouli, D., Hamdani, T. M., Alimi, A. M., and Abraham, A. (2018). Wavelet convolutional neural networks for handwritten digits recognition. In Abraham, A., Muhuri, P. K., Muda, A. K., and Gandhi, N., editors, *Hybrid Intelligent Systems*, pages 305–310, Cham. Springer International Publishing.

[10] Bennesby, R., Mota, E., Fonseca, P., and Passito, A. (2014). Innovating on interdomain routing with an inter-sdn component. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 131–138. IEEE.

[11] Benson, T., Akella, A., and Maltz, D. A. (2009). Unraveling the Complexity of Network Management. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI '09, pages 335–348, Boston, MA, USA. USENIX Association.

[12] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[13] Braga, R., Mota, E., and Passito, A. (2010). Lightweight ddos flooding attack detection using nox/openflow. In *35th IEEE Conference on Local Computer Networks*, LCN '10, pages 408–415, Denver, CO, USA.

[14] Cai, X., Heidemann, J., Krishnamurthy, B., and Willinger, W. (2010). Towards an as-to-organization map. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 199–205, New York, NY, USA. ACM.

[15] Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

[16] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

[17] Chandra, R., Traina, P., and Li, T. (1996). Bgp communities attribute. RFC 1997, RFC Editor.

[18] Chandrashekhar, A. and Raghuveer, K. (2012). Performance evaluation of data clustering techniques using kdd cup-99 intrusion detection data set. *international journal of information and network security*, 1(4):294.

[19] Chaudet, C. and Haddad, Y. (2013). Wireless software defined networks: Challenges and opportunities. In *Microwaves, Communications, Antennas and Electronics Systems (COMCAS), 2013 IEEE International Conference on*, pages 1–5, Tel Aviv, Israel.

[20] Cheng, M., Xu, Q., Lv, J., Liu, W., Li, Q., and Wang, J. (2016). Ms-lstm: A multi-scale lstm model for bgp anomaly detection. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–6.

[21] Cheng, M., Xu, Q., Lv, J., Liu, W., Li, Q., and Wang, J. (2017). Mslstm. https://github.com/jayvischeng/MSLSTM.

[22] Choi, C. and Yi, M. H. (2018). The internet, r&d expenditure and economic growth. *Applied Economics Letters*, 25(4):264–267.

[23] Chollet, F. et al. (2015). Keras: The python deep learning library. https://keras.io.

[24] Chollet, F. et al. (2018). Optimizers in keras. http://faroit.com/keras-docs/0.2.0/optimizers/#adam.

[25] Comer, D. E. (2018). *The Internet book: everything you need to know about computer networking and how the Internet works*. Chapman and Hall/CRC.

[26] Cosovic, M., Obradovic, S., and Junuz, E. (2017). Deep learning for detection of bgp anomalies. In *International Work-Conference on Time Series Analysis*, pages 95–113. Springer.

[27] Ćosović, M., Obradović, S., and Trajković, L. (2015). Performance evaluation of bgp anomaly classifiers. In *2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, pages 115–120. IEEE.

[28] Cowie, J., Ogielski, A., Premore, B., and Yuan, Y. (2001). Global routing instabilities triggered by code red ii and nimda worm attacks. Technical report, Tech. Rep., Renesys Corporation.

[29] Cui, L., Yu, F. R., and Yan, Q. (2016). When big data meets software-defined networking: Sdn for big data and big data for sdn. *IEEE Network*, 30(1):58–65.

[30] d. R. Fonseca, P. C. and Mota, E. S. (2017). A survey on fault management in software-defined networks. *IEEE Communications Surveys Tutorials*, 19(4):2284–2321.

[31] Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005.

[32] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

[33] Deshpande, S., Thottan, M., Ho, T. K., and Sikdar, B. (2009). An online mechanism for bgp instability detection and analysis. *IEEE Transactions on Computers*, 58(11):1470–1484.

[34] Ding, Q., Li, Z., Batta, P., and Trajković, L. (2016). Detecting bgp anomalies using machine learning techniques. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003352–003355.

[35] Doran, D., Schulz, S., and Besold, T. R. (2017). What does explainable AI really mean? A new conceptualization of perspectives. *CoRR*, abs/1710.00794.

[36] Doria, A., Salim, J. H., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and Halpern, J. (2010). Forwarding and control element separation (forces) protocol specification. RFC 5810, RFC Editor. http://www.rfc-editor.org/rfc/rfc5810.txt.

[37] Duan, Z., Chandrashekar, J., Krasky, J., Xu, K., and Zhang, Z.-L. (2007). Damping bgp route flaps. *Journal of Communications and Networks*, 9(4):490–498.

[38] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[39] Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[40] El Khayat, I., Geurts, P., and Leduc, G. (2005). Improving tcp in wireless networks with an adaptive machine-learnt classifier of packet loss causes. In Boutaba, R., Almeroth, K., Puigjaner, R., Shen, S., and Black, J. P., editors, *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 549–560, Berlin, Heidelberg. Springer Berlin Heidelberg.

[41] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P. (2018). Deep learning for time series classification: a review. *CoRR*, abs/1809.04356.

[42] Feamster, N. and Rexford, J. (2017). Why (and how) networks should run themselves. *arXiv preprint arXiv:1710.11583*.

[43] Feamster, N., Rexford, J., and Zegura, E. (2013). The Road to SDN. *Queue*, 11(12):20:20–20:40.

[44] Flach, T., Katz-Bassett, E., and Govindan, R. (2012). Quantifying violations of destination-based forwarding on the internet. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 265–272, New York, NY, USA. ACM.

[45] Fonseca, P. (2019). BGP anomaly datasets. https://github.com/ufam-lia/bgp-feature-extractor/tree/master/datasets.

[46] Fonseca, P., Bennesby, R., Mota, E., and Passito, A. (2019). Bgp dataset generation and feature extraction for anomaly detection. In *2019 IEEE Symposium on Computers and Communications*.

[47] for Applied Internet Data Analysis (CAIDA), C. (2007). Archipelago (Ark) Measurement Infrastructure. http://www.caida.org/projects/ark/. [Online; accessed 25-August-2018].

[48] Fossaceca, J. M., Mazzuchi, T. A., and Sarkani, S. (2015). Mark-elm: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Systems with Applications*, 42(8):4062–4080.

[49] Foundation, O. N. (2010). OpenFlow Switch Specification Version 1.1.0. http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf. Accessed: 2016-06-30.

[50] Foundation, O. N. (2012). Software-defined networking: The new norm for networks. ONF White Paper, Open Networking Foundation.

[51] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*. Number 10 in 1. Springer series in statistics New York.

[52] Fujieda, S., Takayama, K., and Hachisuka, T. (2017). Wavelet convolutional neural networks for texture classification. *CoRR*, abs/1707.07394.

[53] Gibb, G., Zeng, H., and McKeown, N. (2012). Outsourcing network functionality. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 73–78, New York, NY, USA. ACM.

[54] Giotsas, V., Dietzel, C., Smaragdakis, G., Feldmann, A., Berger, A., and Aben, E. (2017). Detecting peering infrastructure outages in the wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 446–459, New York, NY, USA. ACM.

[55] Giotsas, V. and Zhou, S. (2012). Valley-free violation in internet routing — analysis based on bgp community data. In *2012 IEEE International Conference on Communications (ICC)*, pages 1193–1197.

[56] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[57] Goodfellow, I., Mirza, M., Courville, A., and Bengio, Y. (2013a). Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 548–556.

[58] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2013b). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*.

[59] Gudivada, V., Apon, A., and Ding, J. (2017). Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10:1–20.

[60] Gulli, A. and Pal, S. (2017). *Deep Learning with Keras*. Packt Publishing Ltd.

[61] Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2014). SDX: A Software Defined Internet Exchange. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 551–562, New York, NY, USA. ACM.

[62] Hariri, B. and Sadati, N. (2007). Nn-red: an aqm mechanism based on neural networks. *Electronics Letters*, 43(19):1053–1055.

[63] Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

[64] Heaton, J. (2008). *Introduction to neural networks with Java*. Heaton Research, Inc.

[65] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[66] Huang, Y., Feamster, N., Lakhina, A., and Xu, J. J. (2007). Diagnosing network disruptions with network-wide analysis. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '07, pages 61–72, New York, NY, USA. ACM.

[67] Hunt, K., Sbarbaro, D., Żbikowski, R., and Gawthrop, P. (1992). Neural networks for control systems—a survey. *Automatica*, 28(6):1083 – 1112.

[68] Hurvich, C. M. and Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307.

[69] Inc., M. N. (1995). Internet routing registry: Irr. [Online]. Available: http://www.irr.net/. [Online; accessed 16-December-2019].

[70] Jacobs, A. S., Pfitscher, R. J., Ferreira, R. A., and Granville, L. Z. (2018). Refining network intents for self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 15–21. ACM.

[71] Jain, R. and Paul, S. (2013). Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. *IEEE Communications Magazine*, 51(11):24–31.

[72] Jarraya, Y., Madi, T., and Debbabi, M. (2014). A Survey and a Layered Taxonomy of Software-Defined Networking. *IEEE Communications Surveys & Tutorials*, 16(4):1955–1980.

[73] Jenkins, Q. (2013). Answers about recent DDoS attack on Spamhaus. https://www.spamhaus.org/news/article/695/answers-about-recent-ddos-attack-on-spamhaus. [Online; accessed 25-August-2018].

[74] Jolliffe, I. (2011). *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg.

[75] Kalmbach, P., Zerwas, J., Babarczi, P., Blenk, A., Kellerer, W., and Schmid, S. (2018). Empowering self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 8–14. ACM.

[76] Khan, H. (2018). Waveletdeconv library. https://github.com/haidark/WaveletDeconv.

[77] Khan, H. and Yener, B. (2018). Learning filter widths of spectral decompositions with wavelets. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 4601–4612. Curran Associates, Inc.

[78] Kim, T.-Y. and Cho, S.-B. (2018). Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications*, 106:66–76.

[79] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[80] Kompella, K. (2019). Self-driving networks. In *Emerging Automation Techniques for the Future Internet*, pages 21–44. IGI Global.

[81] Kotronis, V., Gämperli, A., and Dimitropoulos, X. (2015). Routing centralization across domains via SDN: A model and emulation framework for BGP evolution. *Computer Networks*, 92:227 – 239. Software Defined Networks and Virtualization.

[82] Kotronis, V., Klöti, R., Rost, M., Georgopoulos, P., Ager, B., Schmid, S., and Dimitropoulos, X. (2016). Stitching inter-domain paths over ixps. In *Proceedings of the Symposium on SDN Research*, SOSR '16, pages 17:1–17:12, New York, NY, USA. ACM.

[83] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[84] Kumar, Y., Farooq, H., and Imran, A. (2017). Fault prediction and reliability analysis in a real cellular network. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1090–1095.

[85] Kurose, J. F. and Ross, K. W. (2012). *Computer Networking: A Top-Down Approach*. Pearson, 6th edition.

[86] Labovitz, C., Malan, G. R., and Jahanian, F. (1997). Internet routing instability. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, pages 115–126, New York, NY, USA. ACM.

[87] Labovitz, C., Malan, G. R., and Jahanian, F. (1999). Origins of internet routing instability. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 218–226 vol.1.

[88] Lad, M., Massey, D., Pei, D., Wu, Y., Zhang, B., and Zhang, L. (2006). Phas: A prefix hijack alert system. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA. USENIX Association.

[89] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

[90] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256.

[91] Lee, C.-Y., Gallagher, P. W., and Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pages 464–472.

[92] Li, J., Dou, D., Wu, Z., Kim, S., and Agarwal, V. (2005). An internet routing forensics framework for discovering rules of abnormal bgp events. *SIGCOMM Comput. Commun. Rev.*, 35(5):55–66.

[93] Li, J., Guidero, M., Wu, Z., Purpus, E., and Ehrenkranz, T. (2007). Bgp routing dynamics revisited. *SIGCOMM Comput. Commun. Rev.*, 37(2):5–16.

[94] Li, L. E., Mao, Z. M., and Rexford, J. (2012). Toward Software-Defined Cellular Networks. In *2012 European Workshop on Software Defined Networking*, pages 7–12, Hague, Netherlands.

[95] Li, Y., Liu, H., Yang, W., Hu, D., and Xu, W. (2016). Inter-data-center network traffic prediction with elephant flows. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 206–213.

[96] Li, Y., Xing, H., Hua, Q., Wang, X., Batta, P., Haeri, S., and Trajković, L. (2014). Classification of bgp anomalies using decision trees and fuzzy rough sets. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1312–1317.

[97] Liang, M. and Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375.

[98] Lin, P., Hart, J., Krishnaswamy, U., Murakami, T., Kobayashi, M., Al-Shabibi, A., Wang, K.-C., and Bi, J. (2013). Seamless interworking of sdn and ip. *ACM SIGCOMM computer communication review*, 43(4):475–476.

[99] Liu, Y., Luo, X., Chang, R. K., and Su, J. (2012). Characterizing inter-domain rerouting after japan earthquake. In *International Conference on Research in Networking*, pages 124–135. Springer.

[100] Lo, O., Buchanan, W. J., Griffiths, P., and Macfarlane, R. (2018). Distance measurement methods for improved insider threat detection. *Security and Communication Networks*, 2018.

[101] Luckie, M. and Beverly, R. (2017). The impact of router outages on the as-level internet. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 488–501, New York, NY, USA. ACM.

[102] Lumezanu, C., Baden, R., Spring, N., and Bhattacharjee, B. (2009). Triangle inequality and routing policy violations in the internet. In *Proceedings of the 10th International Conference on Passive and Active Network Measurement*, PAM '09, pages 45–54, Berlin, Heidelberg. Springer-Verlag.

[103] Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

[104] Machie, A., Roculan, J., Russell, R., and Velzen, M. (2001). Nimda worm analysis. Technical report, Tech. Rep., Incident Analysis, SecurityFocus.

[105] Mai, J., Yuan, L., and Chuah, C. (2008). Detecting bgp anomalies with wavelet. In *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, pages 465–472.

[106] MathWorks (2019). Wavelet denoising and nonparametric function estimation. [Online]. Available: https://ww2.mathworks.cn/help/wavelet/ug/wavelet-denoising-and-nonparametric-function-estimation.html. [Online; accessed 16-December-2019].

[107] McGlynn, K., Acharya, H., and Kwon, M. (2019). Detecting bgp route anomalies with deep learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1039–1040. IEEE.

[108] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

[109] Mckeown, N., Anderson, T., Balakrishnan, H., Parulkar, G. M., Peterson, L. L., Rexford, J., Shenker, S., Turner, J. S., and Louis, S. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38:69–74.

[110] Mitchell, T. (1999). *Machine Learning*. McGraw-Hill.

[111] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. (2003). Inside the slammer worm. *IEEE Security & Privacy*, 1(4):33–39.

[112] Moore, D., Shannon, C., et al. (2002). Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 273–284. ACM.

[113] Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann.

[114] Moustapha, A. I. and Selmic, R. R. (2008). Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. *IEEE Transactions on Instrumentation and Measurement*, 57(5):981–988.

[115] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., and Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634.

[116] Orsini, C., King, A., Giordano, D., Giotsas, V., and Dainotti, A. (2016). Bgpstream: a software framework for live and historical bgp data analysis. In *Proceedings of the 2016 Internet Measurement Conference*, pages 429–444. ACM.

[117] P. Fonseca, R. Bennesby, E. Mota and A. Passito (2019). BGP feature extraction and dataset generation tool. https://github.com/ufam-lia/bgp-feature-extractor/.

[118] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[119] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238.

[120] Percival, D. B. and Walden, A. T. (2006). *Wavelet methods for time series analysis*, volume 4. Cambridge university press.

[121] Popescu, A. C., Premore, B. J., and Underwood, T. (2005). The anatomy of a leak: As9121. *Renesys Corp., http://www. renesys. com/tech/presentations/pdf/renesys-nanog34. pdf*.

[122] Poupart, P., Chen, Z., Jaini, P., Fung, F., Susanto, H., Geng, Y., Chen, L., Chen, K., and Jin, H. (2016). Online flow size prediction for improved network routing. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–6.

[123] Prakash, B. A., Valler, N., Andersen, D., Faloutsos, M., and Faloutsos, C. (2009). Bgp-lens: Patterns and anomalies in internet routing updates. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1315–1324, New York, NY, USA. ACM.

[124] Putina, A., Barth, S., Bifet, A., Pletcher, D., Precup, C., Nivaggioli, P., and Rossi, D. (2018). Unsupervised real-time detection of bgp anomalies leveraging high-rate and fine-grained telemetry data. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE.

[125] Rawat, D. B. and Reddy, S. R. (2017). Software defined networking architecture, security and energy efficiency: A survey. *IEEE Communications Surveys & Tutorials*, 19(1):325–346.

[126] Rekhter, Y., Li, T., and Hares, S. (2006). A border gateway protocol 4 (bgp-4). RFC 4271, RFC Editor. http://www.rfc-editor.org/rfc/rfc4271.txt.

[127] Remy, P. (2018). Stateful lstm in keras. https://github.com/haidark/WaveletDeconv.

[128] RIPE (1999). Ripe network coordination centre. [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris. org. [Online; accessed 25-August-2018].

[129] Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Corrêa, C. N. A., Cunha de Lucena, S., and Raszuk, R. (2012). Revisiting Routing Control Platforms with the Eyes and Muscles of Software-defined Networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 13–18, New York, NY, USA. ACM.

[130] RouteViews, O. (2013). University of oregon routeviews project. Eugene, OR.[Online]. Available: http://www.routeviews.org. [Online; accessed 25-August-2018].

[131] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[132] Salahuddin, M. and Gow, J. (2016). The effects of internet usage, financial development and trade openness on economic growth in south africa: A time series analysis. *Telematics and Informatics*, 33(4):1141–1154.

[133] Sapankevych, N. I. and Sankar, R. (2009). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38.

[134] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

[135] Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43.

[136] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). FlowVisor: A Network Virtualization Layer. Tech. Rep. OPENFLOW-TR-2009-1, Deutsche Telekom Inc. R&D Lab, Stanford, Nicira Networks.

[137] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2010). Can the Production Network Be the Testbed? *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 365–378.

[138] Shi, X., Xiang, Y., Wang, Z., Yin, X., and Wu, J. (2012). Detecting prefix hijackings in the internet with argus. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 15–28, New York, NY, USA. ACM.

[139] Siddiqui, M. K. and Naahid, S. (2013). Analysis of kdd cup 99 dataset using clustering based data mining. *International Journal of Database Theory and Application*, 6(5):23–34.

[140] Smith, M., Dvorkin, M., Laribi, Y., Pandey, V., Garg, P., and Weidenbacher, N. (2014). OpFlex Control Protocol. Internet-Draft draft-smith-opflex-00, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-smith-opflex-00.txt.

[141] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[142] Tangmunarunkit, H., Govindan, R., Shenker, S., and Estrin, D. (2001). The impact of routing policy on internet paths. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 2, pages 736–742 vol.2.

[143] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6. IEEE.

[144] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

[145] Toonk, A. (2013). Looking at the spamhouse DDoS from a BGP perspective. https://bgpmon.net/looking-at-the-spamhouse-ddos-from-a-bgp-perspective/. [Online; accessed 25-August-2018].

[146] van der Merwe, J. E., Rooney, S., Leslie, L., and Crosby, S. (1998). The Tempest-a practical framework for network programmability. *IEEE Network*, 12(3):20–28.

[147] Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

[148] Vissicchio, S., Vanbever, L., and Bonaventure, O. (2014). Opportunities and research challenges of hybrid software defined networks. *SIGCOMM Computer Communication Review*, 44(2):70–75.

[149] Wang, G., Ng, T. E., and Shaikh, A. (2012). Programming Your Network at Run-time for Big Data Applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 103–108, Helsinki, Finland. ACM.

[150] Wang, L., Zhao, X., Pei, D., Bush, R., Massey, D., Mankin, A., Wu, S. F., and Zhang, L. (2002). Observation and analysis of bgp behavior under stress. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 183–195, New York, NY, USA. ACM.

[151] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[152] Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. (2015). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51.

[153] Xia, W., Zhao, P., Wen, Y., and Xie, H. (2017). A survey on data center networking (dcn): Infrastructure and operations. *IEEE Communications Surveys & Tutorials*, 19(1):640–656.

[154] Yang, L., Dantu, R., Anderson, T., and Gopal, R. (2004). Forwarding and Control Element Separation (ForCES) Framework. RFC 3746, RFC Editor.

[155] Yaqoob, T., Usama, M., Qadir, J., and Tyson, G. (2018). On analyzing self-driving networks: A systems thinking approach. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 1–7. ACM.

[156] Zakka, K. (2019). Applying Deep Learning. https://kevinzakka.github.io/2016/09/26/applying-deep-learning/.

[157] Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

[158] Zhang, J., Rexford, J., and Feigenbaum, J. (2005). Learning-based anomaly detection in bgp updates. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, MineNet '05, pages 219–220, New York, NY, USA. ACM.

[159] Zhang, J., Walter, G. G., Miao, Y., and Lee, W. N. W. (1995). Wavelet neural networks for function learning. *IEEE Transactions on Signal Processing*, 43(6):1485–1497.

[160] Zhang, Z. (2007). Learning algorithm of wavelet network based on sampling theory. *Neurocomputing*, 71(1):244 – 269. Dedicated Hardware Architectures for Intelligent Systems Advances on Neural Networks for Speech and Audio Processing.

[161] Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer.

[162] Ćosović, M., Obradović, S., and Trajković, L. (2016). Classifying anomalous events in bgp datasets. In *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4.