Machine Learning for Cognitive Network Management

Sara Ayoubi, Noura Limam, Mohammad A. Salahuddin, Nashid Shahriar, Raouf Boutaba, Felipe Estrada-Solano, and Oscar M. Caicedo

The complexity, heterogeneity, and scale of networks have grown far beyond the limits of manual administration. Furthermore, the main cause of outages in many network environments is human error. This has triggered a shift in the design philosophy of network management systems to minimize the role of humans in the control loop.

Abstract

Over the last decade, a significant amount of effort has been invested on architecting agile and adaptive management solutions in support of autonomic, self-managing networks. Autonomic networking calls for automated decisions for management actions. This can be realized through a set of pre-defined network management policies engineered from human expert knowledge. However, engineering sufficiently accurate knowledge considering the high complexity of today's networking environment is a difficult task. This has been a particularly limiting factor in the practical deployment of autonomic systems. ML is a powerful technique for extracting knowledge from data. However, there has been little evidence of its application in realizing practical management solutions for autonomic networks. Recent advances in network softwarization and programmability through SDN and NFV, the proliferation of new sources of data, and the availability of low-cost and seemingly infinite storage and compute resource from the cloud are paving the way for the adoption of ML to realize cognitive network management in support of autonomic networking. This article is intended to stimulate thought and foster discussion on how to defeat the bottlenecks that are limiting the wide deployment of autonomic systems, and the role that ML can play in this regard.

Introduction

The complexity, heterogeneity, and scale of networks have grown far beyond the limits of manual administration. Furthermore, the main cause of outages in many network environments is human error [1]. This has triggered a shift in the design philosophy of network management systems to minimize the role of humans in the control loop.

In 2001, IBM proposed the autonomic computing initiative. The vision was to have strategies for self-* (i.e., self-configuring, self-healing, self-optimizing, and self-protecting) IT systems, a goal also shared by HP's Adaptive Enterprise and Microsoft's Dynamic Systems. As part of this initiative, IBM proposed an architecture for autonomic computing [2], where autonomic managers maintain a Monitor-Analyze-Plan-Execute (MAPE) over shared knowledge control loop with the managed resources. Since then, several extensions to the MAPE control loop have been proposed, including Foundation-Observe-Compare-Act-Learn-rEason (FOCALE) [1] that is based on Observe-Orient-Decide-Act (OODA) [1]. Essentially, FOCALE offers extensions for knowledge use and learning with dynamic control loops, namely reactive, deliberative and reflective with increasing level of cognitive capabilities. However, IBM's MAPE remains the most widely adopted with many incarnations of its vision proposed for networking, such as *cognitive networks* [1, 3] and *knowledge-driven networking* [4, 5]. In essence, these initiatives advocate for incorporating intelligence and autonomy in network management.

Autonomic networks call for automated decisions for management actions. This can be achieved via policy-based management (PBM) through a set of pre-defined self-* policies engineered from human expert knowledge or derived from high-level policies provided by humans. Activating backup resources upon predicting a fault and steering traffic flows through deep packet inspection (DPI) based on a blacklist are examples of self-healing and self-protection policies, respectively. However, considering the high complexity of today's networking environments, engineering sufficiently accurate knowledge is a cumbersome task. This has prohibited the practical deployment of autonomic systems. In autonomic systems, possible actions should be *learned* from the operating environment, and *reasoned* and *adapted* to changes, while respecting operational goals and requirements.

Machine learning (ML) is a popular technique for extracting knowledge from data. In theory, ML can be used for automating network operations and management. However, there has been little evidence of its application in realizing autonomic networks. Prohibiting factors include the distributed control and vendor-specific nature of legacy network devices, lack of available data, and cost of compute and storage resources. Several technological advances have been made in the last decade to overcome these limitations. The advent of network softwarization and programmability through software-defined networking (SDN) and network functions virtualization (NFV) offers centralized control and alleviates vendor lock-in. The advances in ML along with the proliferation of new sources of data and big data analytics platforms provide abundant data and extract knowledge from them. For instance, recent breakthroughs in deep learning (DL) allow generation of models from raw data without the

Digital Object Identifier:
10.1109/MCOM.2018.1700560

Sara Ayoubi, Noura Limam, Mohammad A. Salahuddin, Nashid Shahriar, and Raouf Boutaba are with the University of Waterloo.
Felipe Estrada-Solano and Oscar M. Caicedo are with the University of Cauca.

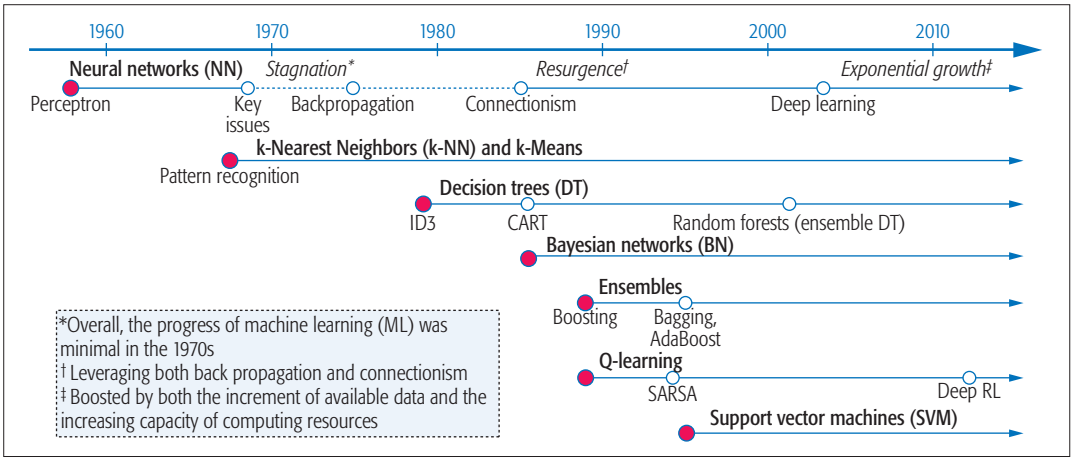

Figure 1. The evolution of machine learning algorithms with key milestones.

Fundamental to ML is feature extraction that determines the best discriminators for learning and inference. Note that learning splits the data into training and validation sets. This split can conform to the 80:20 or 70:30 ratio rule-of-thumb; or follow the k-fold cross validation technique.

need for data labeling. Furthermore, the availability of seemingly *infinite* storage and compute resources through the cloud overcomes the cost of resources. These together provide the environment to realize the vision of autonomic networks. The main contributions of this article are:

- We expose how ML can be used to realize autonomicity in each of the fault, configuration, accounting, performance, and security (FCAPS) [6] management areas.
- We show how ML can be leveraged to realize a cognitive MAPE control loop for network management.
- We discuss the opportunities and challenges pertaining to using ML for the management of autonomic networks.

The remainder of this article is organized as follows. We provide a high-level background on ML in the following section. After that we highlight how ML has been leveraged for network management. We showcase how ML can be used to realize a cognitive control loop, and present a use case illustrating its realization in practice using existing technologies and protocols. Then we present future research directions to facilitate a holistic cognitive management framework.

Machine Learning

ML goes beyond learning or extracting knowledge to utilizing it and improving it with experience. It has given rise to a plethora of algorithms, as shown in Fig. 1. Essentially, ML is applied to problems that can be solved using *inference* [7] and have large representative training *data*. Fundamental to ML is *feature extraction*, which determines the best discriminators for learning and inference [8]. Note that *learning* splits the data into training and validation sets. This *split* can conform to the 80:20 or 70:30 ratio rule of thumb, or follow the *k*-fold cross-validation technique.

ML is classified into three categories, based on how the learning is achieved [1].

Supervised Learning: uses labeled training datasets to create models that map inputs to corresponding outputs. Typically, this approach is used to solve *classification* and *regression* problems that pertain to predicting discrete or continuous valued outputs, respectively. For example, a classification problem can be to identify an attack as either denial of service (DoS), root-to-local (R2L), user-to-root (U2R), or probing. A regression problem can be to predict the time of future attacks.

Unsupervised Learning: uses unlabeled training datasets to create models that find dominating structure or patterns in the data. This approach is appropriate for *clustering*, *outliers detection*, and *density estimation* problems. For example, the clustering problem can pertain to grouping different instances of attacks based on their similarities.

Reinforcement Learning (RL): is an iterative process that uses the feedback from the environment to learn the correct sequence of actions to maximize a cumulative reward. Unlike other approaches that are myopic in nature, RL may sacrifice immediate gains for long-term rewards. Hence, RL is best suited for making cognitive choices, such as decision making, planning, and scheduling [9].

Different ML techniques (Fig. 1) belong to one or more of the aforementioned categories. Bayesian networks (BNs) and support vector machines (SVMs) are typically applied in supervised learning. *k*-nearest neighbors (k-NN), decision trees (DTs), and neural networks (NNs) have been used in both supervised and unsupervised learning. *k*-means operates only in unsupervised learning. Q-learning, the most prominent RL technique, has recently used NN and deep NN (DNN) to approximate its action-value function (i.e., deep RL).

Machine Learning for Network Management

Machine Learning for FCAPS: What Has Been Done?

Application of ML to automate network management and close the management loop is a nontrivial task. Table 1 highlights representative ML techniques employed in the literature for the FCAPS management areas to provide some degree of autonomy.

Fault Management: Failure in networks is a norm rather than an exception, and its impact can be quite costly [10]. The slow reaction time and poor accuracy of traditional fault management techniques further increase this cost. This has motivated efforts that leveraged ML for proactive fault prediction. Additional works considered the usage of ML for fault localization and automat-

ed mitigation to minimize downtime and human intervention.

Configuration Management: Operators must implement increasingly sophisticated network policies that have to be translated into constrained low-level configuration commands, and adjusted to changes in network conditions (e.g., intrusions, traffic shifts, performance degradation). As the network state is constantly changing, network managers find themselves constantly configuring the network to adapt to these changes, which is a cumbersome and error-prone process. ML can help automate this process by training models to identify optimal state-action pairs as the network behavior changes over time. A handful of works have showcased the benefits of ML for dynamic resource allocation and service configuration.

Accounting Management: Accounting is tightly coupled with business and control models. These models leverage accounting data in decision making, service planning, and delivery, and designing tariffs and pricing plans. Therefore, it is essential to ensure the integrity of accounting data by accurate collection of usage data and fraud detection. The use of ML for network accounting management is rather unexplored.

Performance Management: Today's networks typically run a variety of services with different performance requirements to serve an increasing number of users with distinct profiles. Guaranteeing performance is a daunting task. In fact, without the ability to accurately predict network behavior, how can we provide such guarantees? This realization has attracted numerous efforts that have leveraged ML for performance and traffic load prediction, and quality of experience/service (QoE/QoS) correlation for proactive and adaptive network performance management [11].

Security Management: The most commonly employed security approach consists of monitoring the network for patterns of well-known threats. However, this renders the network vulnerable to *zero-day* attacks. This vulnerability is critical as new attacks emerge daily [12]. The need for robust security measures is clear, and the role of ML toward this end has been investigated extensively [13]. Existing efforts have concentrated on using ML for misuse detection in order to learn complex attack patterns from historical data and generate generic rules that allow detecting variations of known attacks. Anomaly detection using ML has also been explored to detect zero-day attacks. This consists of learning patterns of normal behavior and detecting deviations from the norm.

The aforementioned efforts show promising results toward incorporating cognition in network management. However, leveraging ML for the different network management functions alone will not fulfill the vision of cognitive management. In fact, there is a need for a cognitive control loop, detailed below.

Management area	Management function	Machine learning techniques
Fault	Fault prediction	NN, k-NN, k-Means, DT, BN, SVM
	Fault localization	NN, k-NN, k-Means, DT
	Automated mitigation	BN, SVM
Configuration	Adaptive resource allocation	Q-Learning, Deep
	Adaptive service configuration	Q-Learning
Accounting	–	–
Performance	Traffic load and metrics prediction	(Ensemble) NN, BN, SVM,
	QoE-QoS correlation	DT, BN, SVM, Q-learning
Security	Misuse detection	NN, DT, BN, SVM
	Anomaly detection	(Ensemble) NN, DNN, *k*-NN, *k*-means, (Ensemble) DT, Ensemble BN, SVM

Table 1. Sample machine learning techniques used in FCAPS.

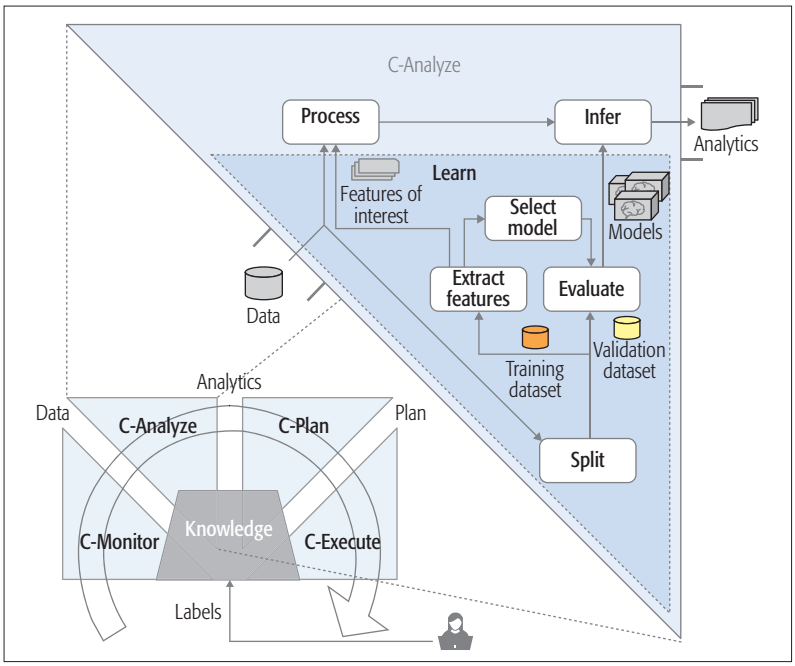

Figure 2. Cognitive control loop for network management.

C-MAPE: A Cognitive Control Loop

To date, IBM's architecture for autonomic computing [2] is the most influential reference model for autonomic systems and networks. It comprises several layers of autonomic managers. The behavior of each manager is governed by the MAPE control loop that consists of four functions: *monitor*, *analyze*, *plan*, and *execute*. As shown in Fig. 2, the *knowledge* source is orthogonal to every MAPE function. Functions can retrieve data from and/or log created knowledge to the *knowledge* source. For example, the *analyze* function obtains information about the historical behavior of a managed resource and stores the ML models and the analytics it generates in the *knowledge* source.

In [2], we observe that cognition has been restricted to the *analyze* function, which inhibits the ability to achieve closed-loop cognitive network management. In this article, we propose to incorporate cognition at every function in the loop. For example, the *monitor* function should be able to determine what, when, and where to monitor. ML can be leveraged to build this cognition in every function and allow each function to operate in full autonomy. Therefore, we extend IBM's MAPE control loop into a cognitive control loop we call *C-MAPE*. As illustrated in Fig. 2,

cognition is achieved by introducing learning and inference in every function.

C-Monitor Function: refers to the cognitive monitor that performs intelligent probing. For instance, when the network is overloaded, the C-Monitor function may decide to reduce the probing rate and instead perform regression for data prediction.

C-Analyze Function: is responsible for detecting or predicting changes in the network environment (e.g., faults, policy violations, frauds, performance degradation, and attacks). ML has been leveraged to address some of these challenges, as discussed previously.

C-Plan Function: can leverage ML to develop an intelligent automated planning (AP) engine that reacts to changes in the network by selecting or composing a change plan. In the last decade, AP systems have been applied to real-world problems and have been relying on ML (e.g., DT, RL) for automating the extraction and organization of knowledge (e.g., plans, execution traces), and for decision making [14].

C-Execute Function: can use ML to schedule the generated plans and determine the course of action should the execution of a plan fail. These tasks lend themselves naturally to RL where the C-Execute agent could *exploit* past successful experiences to generate optimal execution policies, and explore new actions in case the execution plan fails.

Closing the control loop is achieved by monitoring the state of the network to measure the impact of the change plan.

Use Case: A Cognitive Security Manager

We showcase how C-MAPE can be used for security anomaly detection and mitigation. We present a use case over a software-defined infrastructure (SDI) that can be realized in production. Figure 3 illustrates the resource orchestrator (e.g., OpenStack; https://www.openstack.org/, accessed 14 June2017) and the SDN controller (e.g., OpenDaylight; https://www.opendaylight.org/, accessed 14 June 2017) that directly communicate with the computing and networking resources in the SDI. The resource orchestrator administers the physical and virtual resources, while the SDN controller facilitates automated and flexible configuration of the network resources.

We assume that all information regarding the physical and virtual resources (e.g., topology changes), and data (e.g., flow statistics, links' states) are periodically stored in a central repository, by the resource orchestrator and the SDN controller, respectively. This repository supplements the *knowledge* source. The cognitive security manager (CSM) in Fig. 3 depicts the cognitive control loop for *C-MAPE* functions in security management. It communicates with the resource orchestrator, the SDN controller, and the repository via REST application programming interfaces (APIs) to perform control and management functions.

As illustrated in Fig. 3, the *C-Monitor* function pulls flow-level information and packet-level statistics for ingress traffic via the SDN controller from the SDN switch X. The flow-level information includes source IP, destination IP, source port, destination port, and protocol. The packet-level

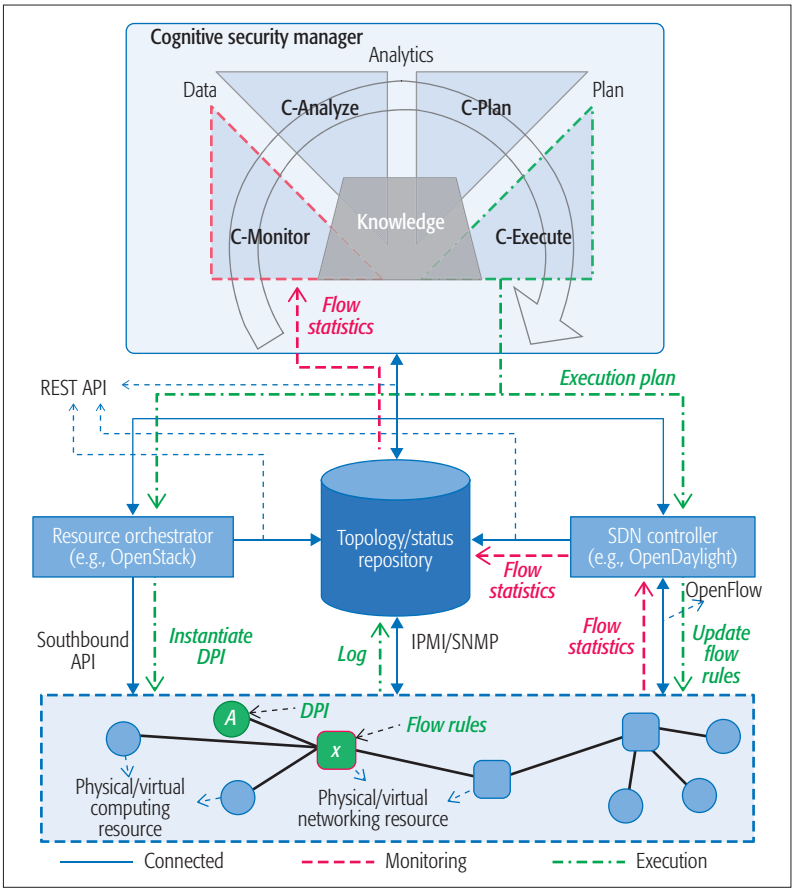

Figure 3. Cognitive security manager for anomaly inference and mitigation over a software-defined infrastructure.

statistics include packet inter-arrival time, average packet length, and bytes per packet. The controller augments the central repository with this information and statistics.

We assume that *C-Analyze* in CSM has already augmented the *knowledge* source with an outliers detection model using an ML algorithm (e.g., *k*-means, *k*-NN) for anomaly inference. It does so by leveraging the historical data from the *knowledge* source to train and validate the model. In real time, the *C-Analyze* function passes the data collected by *C-Monitor* through the trained model and infers a security anomaly associated with a sequence of flows pertaining to the same source IP.

The generated analytics are then used by the *C-Plan* function that employs RL to choose an optimal change plan based on the criticality of the anomaly. This plan entails installing a DPI virtual network function along with updating flow rules to route packets from the suspected source IP to the DPI.

Based on the chosen plan from the *C-Plan* function, the *C-Execute* function directs the resource orchestrator to instantiate a DPI VM on computing resource A in Fig. 3. The DPI VM is pre-configured to log DPI results in the repository. The *C-Execute* function also directs the SDN controller to install flow rules in the SDN switch X to route packets from the suspected IP to the DPI A for further investigation.

The illustrated use case ends here; however, the results from the DPI could further be used

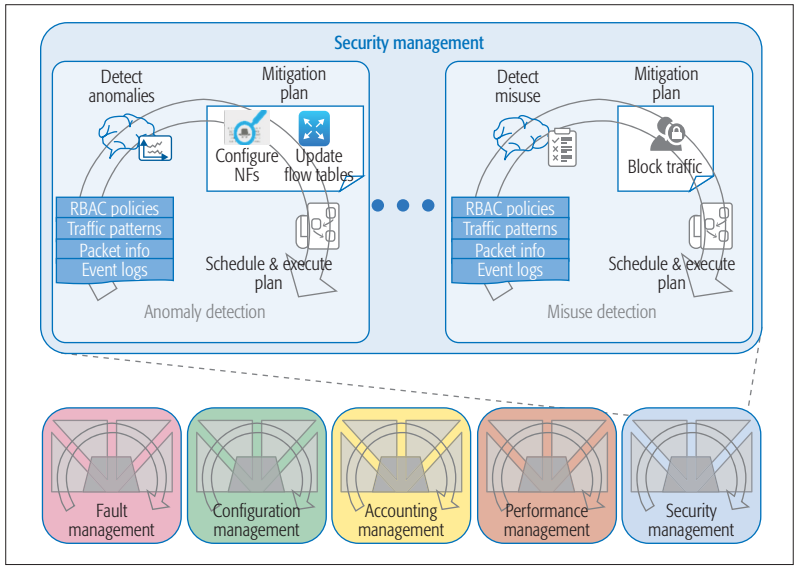

Figure 4. C-MAPE in each function of FCAPS management area.

to install a firewall and configure it to drop packets from the suspicious IP if deemed malicious. Although this use case focuses on *C-MAPE* for security management, *C-MAPE* can play a vital role in all areas (e.g., fault, configuration) to offer holistic cognitive network management.

Future Research Directions

Machine Learning for FCAPS: What Can Be Done?

Above, we highlighted the previous efforts toward employing ML in FCAPS. These efforts are indeed fundamental toward realizing autonomic network management. As illustrated in Fig. 4, *C-MAPE* will reside in every management area, and within different functions of every area. However, as we survey previous and ongoing efforts, we observe that the automation of many management tasks has not been explored yet. As a call for action, we identify further research opportunities, where ML can be applied with respect to FCAPS.

Fault Management:

Failure Prevention: ML has been used for proactive failure prediction, leaving the localization and mitigation steps mostly as reactive. However, proactive mitigation combined with fault prediction can help prevent upcoming failures. Since a proactive mitigation approach requires a set of actions to be taken, RL can be a prospective candidate. To select the appropriate mitigation step, the root cause of the predicted fault has to be precisely identified. Existing ML-based localization approaches suffer from poor scalability when analyzing the high-dimensional device log attributes in moderate-size networks. Dimensionality reduction can be leveraged to improve the scalability of fault localization techniques without sacrificing accuracy.

Fault Management in Cloud and Virtualized Environments: The efforts on fault management discussed above focus on single tenant networks. The advent of new technologies, such as multi-tenancy in cloud and virtualization of network functions, magnify the complexity and dimensions of the fault space in a network. For instance, any failure in the underlying physical resource can propagate to the hosted virtual resources. However, the reverse is not always true. To predict and locate faults in such networks, we can use DNNs, which can model complex multi-dimensional state spaces. A reliable virtual network (VN) embedding algorithm can leverage these predictions to set up a VN. Furthermore, any automated mitigation within a VN should not affect the operations of other coexisting VNs. Here, RL combined with DNNs can learn to optimize mitigation steps.

Configuration Management:

Mapping High-Level Requirements to Low-Level Configurations: Networks are configured to satisfy certain requirements in terms of performance, connectivity, fault tolerance, security, and so on. The gap between high-level requirements and low-level configurations (e.g., resources to be provisioned) is difficult to bridge. RL techniques can be leveraged in this context. The reward for selecting a configuration setting of a given network element can be thought of as the utility of that particular setting in delivering the high-level requirement under a given network condition.

Configuration Verification: Configuration changes (e.g., access control lists, routing tables) should not conflict with high-level requirements; nor should they adversely affect the expected behavior of the network. Formal methods have been used to analyze and verify network configurations [15]. However, these methods are found to be highly complex and are difficult to scale. A growing interest has been shown in applying DL-aided verification, code correction, and theorem proving. Considering the scale and complexity of the configuration parameters in a network, a DL-based verification approach is worth exploring.

Configuration Rollback: After verification, a decision is made to either accept the configuration changes or to revert (some of) them back to a previous stable state. To avoid service disruption and performance degradation, it is essential that the stable state is reached with minimum delay. Assuming that snapshots of stable configurations are logged, the system still has to decide in what order the changes should be applied, while minimizing both disruption and delay. Here, RL can be used to find the optimal rollback strategy, assuming, for instance, that rollbacks are assigned rewards that are inversely related to the incurred disruption and time-to-stable state.

Accounting Management:

Making Accounting Smart: Collecting accurate customer usage yields opportunities for increasing customer experience and resource utilization, and reducing cost of operations. The collected customer usage data can be leveraged by supervised and unsupervised ML models to deduce norms and predict customers' usage habits. These models identify deviations from the norm, triggering misuse detection and root cause analysis for fraudulent activities. Such prediction models can also help service providers to convene smart pricing schemes and smart forecasted bills, provide incentives, and bundle services. Furthermore, these models can facilitate opportunistic, dynamic, and proactive provisioning of resources to improve QoS.

Performance Management:

Adaptive Probing: Obtaining measurements from the network is required for monitoring network behavior. However, the large number of devices in the network, the variety of parameters to measure, and the small time intervals to log data exponentially increase the amount of traffic overhead, resulting in network performance degradation. Regression, mostly based on time series data, can predict the value of the measured parameters to optimize probing. The goal is to set probing rates that keep traffic overhead within a target value, while minimizing performance degradation and providing high prediction accuracy.

Detecting Patterns of Degradation: Poor QoS can be addressed by monitoring network metrics that indicate performance degradation. The challenge is to detect the characteristic patterns of degradation before the quality drops below an acceptable level. This information can be reported to the network administrator or used for autonomic tuning of network parameters to achieve optimum performance. Here, elastic resource allocation can be leveraged to dynamically accommodate user demands for achieving optimum performance while maximizing resource utilization. As aforementioned, supervised learning has been used to predict the value of network performance metrics. However, employing performance prediction for autonomic tuning of the network behavior remains an open challenge.

Security Management:

Reducing Classification Errors: While ML for anomaly detection has received significant interest in the literature, it has not yet been employed in practice. This is mainly due to their high false positive rate that wastes expensive network analyst time. Hence, further efforts are needed to reduce classification errors in anomaly detection. Some promising research directions include the use of alarm post-processing (e.g., correlation, filtering, prioritization) and the correlation of host-based and network-based data traces. The latter allows the detection of threats that fail to be detected at the network level but exhibit anomalous behavior on the host and vice versa. Ensemble learning has also shown encouraging results in terms of its ability to overcome data skew.

Security of Management: An important aspect of management is ensuring that the management interface itself is secured. This is primarily achieved by limiting access to authorized users. However, it cannot guarantee safety against malicious authorized users. Indeed, an authorized user with malicious intent can create havoc. To the best of our knowledge, there has been no work tailored to address the security of management, leaving it as an open research direction.

Autonomic Security Management: Anomaly detection has been extensively explored for its ability to detect new attacks. This is achieved by learning normal behavior and raising an alarm when a deviation from the norm is detected. However, this does not provide any intuition of the attack taking place. Identifying the nature of the attack is fundamental to determine the criticality of the situation and take appropriate mitigation actions. This is indeed vital to achieve autonomic security management and must be realized in real time to avoid detrimental consequences.

Challenges in Using Machine Learning

Representative Datasets: ML is inherently data-driven. Hence, the quality of the data used for training and validation is critical. Non-representative datasets can have a severe impact on the accuracy of the models. Gaining access to representative data is not an easy task mainly due to its sensitive and confidential nature. One possible direction is to encourage sharing of data in the research community. Toward fulfilling this need, we have launched a website (https://sites.google.com/site/cnetmag/; accessed 14 June 2017) for the research community to share datasets, tools, and platforms.

Speed vs. Accuracy: Achieving high accuracy often comes at the cost of high computation time for training the model. This is particularly a challenge when dealing with online models. A promising direction to explore to overcome this challenge is the use of ensemble learning and hybrid techniques.

Ground Truth: A key challenge in applying ML is the need for ground truth. For supervised learning, ground truth provides the labels for training. For unsupervised learning it allows the accuracy of the model to be evaluated. However, to obtain the ground truth, one must either manually label the data or synthetically generate it. While the former allows the use of real data traces, the manual labeling process can be highly cumbersome and error-prone. On the other hand, the latter can render unrealistic traffic traces. An interesting research direction worth exploring is the application of active learning [16] to facilitate labeling.

ML Techniques for Networks: Another key challenge with the application of ML in networking is the lack of a "Theory of Networks." This concern was raised by David Meyer during his talk at Internet Engineering Task Force 97 (IETF97) [17] on machine intelligence and networking. Indeed, without a unified theory, each network has to be learned separately. This could truly hinder the speed of adoption of ML in networking. Furthermore, the currently employed ML techniques in networking have been designed with other applications in mind. An open research direction in this realm is to design ML algorithms tailored for networks [5].

Incremental Learning: Due to the high dynamicity of networks, the ML models have to be constantly re-trained to ensure their validity over time. Re-training a model from scratch is computationally expensive and time consuming, particularly for online applications. An interesting research direction is to achieve fast incremental learning, where the model is re-trained with only the new data.

Security of Machine Learning: ML is prone to adversarial attacks [18], also known as mimicry attacks, that aim to confuse learning. For instance, when employing ML for intrusion detection, an adversarial attack can trick the model into misclassifying malicious events as benign by poisoning the training set. In [18], the authors performed a proof of concept to showcase how a learning algorithm (outlier detection) can be manipulat-

Achieving high-accuracy often comes at the cost of high computation time for training the model. This is particularly a challenge when dealing with online models. Some promising directions to overcome this challenge is exploring the use of ensemble learning and hybrid techniques.

The network has come a long way since then with the increasing adoption of SDN, NFV, and Cloud Computing. These technological advances have rendered the infrastructure more agile and compute and storage resources more abundant than ever before.

ed. They assumed complete knowledge of the ML algorithm in use and its decision boundaries. However, without such knowledge, the vulnerability of ML models remain an open research question. That is, it is unclear what it would take an outsider (without inside information) to pull off an adversarial attack in a blackbox setting. An interesting initiative worth mentioning is Cleverhans (https://github.com/tensorflow/cleverhans; accessed 14 June 2017), a useful library that allows to craft adversarial examples. It provides training datasets that can be used to build robust ML models capable of distinguishing legitimate datasets from poisoned ones.

Challenges in Autonomic Network Management

Orchestration of Cognitive Management Functions: One key challenge in autonomic network management is how to orchestrate *C-MAPE* functions within a management area or across management areas. Such coordination is fundamental to attain high-level network policies that ensure the correctness of the system and the stability of the *C-MAPE* loop. Clearly, an orchestration layer is needed that will sit atop the management areas. Such a layer will need to address three key issues:

- Define valid operating regions of functions and ensure that a function remains within its boundary.
- Enable synchronization among functions so that *C-MAPE* can converge to a steady state.
- Resolve conflicting policies posed by different functions to ensure that the system behaves correctly.

However, to the best of our knowledge such a cognitive management orchestrator does not exist, and thus remains an open research direction.

Technological Barriers: To attain full autonomy, the management functions must be able to easily interact with the managed resources. Without standardized open interfaces, such autonomous interactions cannot be achieved. With the advent of SDN and NFV, many of these complexities have been alleviated. Although SDN enjoys the advantage of a well defined southbound API between the controller and network devices, the northbound API between the controller and the management applications has yet to mature. In addition, while NFV offers the flexibility to instantiate network functions on the fly, there are no standard APIs to configure their states. These challenges and more demand further efforts toward facilitating and standardizing network configuration and control.

Conclusion

More than a decade has passed since the vision of autonomic computing was initially proposed. The gap between the vision's demands and the network capabilities have inhibited the former from being effectuated. However, the network has come a long way since then with the increasing adoption of SDN, NFV, and cloud computing. These technological advances have rendered the infrastructure more agile, and compute and storage resources more abundant than ever before. Motivated by this evolution, coupled with the growing need for enhanced management, this article presents our preliminary effort to realize a cognitive network management framework using ML. We motivate the major role that ML can play in realizing cognitive network management, and highlight existing efforts that have leveraged ML for performing various management tasks. We follow this discussion with an elucidation of how ML extends the MAPE control loop to realize cognitive network management. We present a use case of a cognitive security manager that can be implemented in practice, and conclude with open research directions to realize a holistic cognitive network management framework.

Acknowledgments

This work was supported in part by the NSERC Discovery Grants Program (Canada), the Quebec FRQNT postdoctoral research fellowship (Canada), the ELAP scholarship (Canada), and the COLCIENCIAS Scholarship Program No. 647-2014 (Colombia).

References

[1] Q. Mahmoud, *Cognitive Networks: Towards Self-Aware Networks*, Wiley-Interscience, 2007.
[2] S. R. White *et al.*, "An Architectural Approach to Autonomic Computing," *Proc. Int'l. Conf. Autonomic Computing*, 2004, May 2004, pp. 2–9.
[3] L. Xu *et al.*, "Cognet: A Network Management Architecture Featuring Cognitive Capabilities," *Proc. Euro. Conf. Networks and Commun.*, June 2016, pp. 325–29.
[4] D. D. Clark *et al.*, "A Knowledge Plane for the Internet," *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Commun.*, 2003, pp. 3–10.
[5] A. Mestres *et al.*, "Knowledge-Defined Networking," *ACM SIGCOMM Computer Commun. Review*, vol. 47, no. 3, 2017, pp. 2–10.
[6] A. Clemm, *Network Management Fundamentals*, Cisco Press, 2006.
[7] P. Langley and H. A. Simon, "Applications of Machine Learning and Rule Induction," *ACM Commun.*, vol. 38, no. 11, Nov. 1995, pp. 54–64.
[8] E. Brill et al., "Data-Intensive Question Answering," *TREC*, 2001.
[9] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," *IEEE Internet Computing*, vol. 11, no. 1, Jan. 2007, pp. 22–30.
[10] J. Stanganelli, "The High Price of IT Downtime," *Computer Networking*, 2016; online, accessed 14 June 2017.
[11] N. Bui *et al.*, "A Survey of Anticipatory Mobile Networking: Context-Based Classification, Prediction Methodologies, and Optimization Techniques," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 3, 3rd qtr. 2017, pp. 1790–1821.
[12] V. Harrison and J. Pagliery,"Nearly 1 Million New Malware Threats Released Every Day," *CNN Tech.*, 2015; online, accessed 14 June 2017.
[13] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 2, 2016, pp. 1153–76.
[14] T. Zimmerman and S. Kambhampati, "Learning-Assisted Automated Planning: Looking Back, Taking Stock, Going Forward," *AI Mag.*, vol. 24, no. 2, 2003, pp. 73–96.
[15] H. Yang and S. S. Lam, "Real-Time Verification of Network Properties Using Atomic Predicates," *IEEE/ACM Trans. Networking*, vol. 24, no. 2, Apr. 2016, pp. 887–900.
[16] I. Zliobaite *et al.*, "Active Learning with Drifting Streaming Data," *IEEE Trans. Neural Networks and Learning Systems*, vol. 25, no. 1, 2014, pp. 27–39.
[17] D. Meyer, "Machine Intelligence and Networks," IETF97, 2016; online, accessed 14 June 2017.
[18] M. Barreno *et al.*, "Can Machine Learning Be Secure?" *Proc. ACM Symp. Info., Computer and Commun. Security*, 2006, pp. 16–25.

Biographies

Sara Ayoubi received her M.Sc. in 2012 from the Lebanese American University and her Ph.D. in 2016 from the Concordia Institute for Information and Systems Engineering. She is currently a postdoctoral fellow at the Cheriton School of Computer

Science at the University of Waterloo. She is a co-founder of the Montreal Operations Research Student Chapter. Her research interests are in the fields of operations research, networks, and computer systems.

NOURA LIMAM received her M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris VI, in 2002 and 2007, respectively. She is currently a research assistant professor of computer science at the University of Waterloo. She is on the Technical Program Committees and Organization Committees of several IEEE conferences. Her contributions are in the area of network and service management. Her current research interests are in network softwarization and cognitive network management.

MOHAMMAD A. SALAHUDDIN is a postdoctoral fellow at the Cheriton School of Computer Science, University of Waterloo. He received his Ph.D. in computer science from Western Michigan University in 2014. His current research interests include the Internet of Things, content delivery networks, network softwarization, cloud computing, and cognitive network management. He serves as a TPC member for international conferences and is a reviewer for various journals and magazines.

NASHID SHAHRIAR is a Ph.D. candidate at the Cheriton School of Computer Science, University of Waterloo. He received his M.Sc. and B.Sc. degrees in computer science and engineering from Bangladesh University of Engineering and Technology in 2011 and 2009, respectively. His research interests include future network architectures, network virtualization, and optical networks. He is a recipient of a David R. Cheriton Graduate Scholarship at the University of Waterloo.

RAOUF BOUTABA received his M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is a professor in the Cheriton School of Computer Science and Associate Dean, Research of the Faculty of Mathematics at the University of Waterloo, and holds an INRIA International Chair at INRIA Nancy. His research interests include network and service management, cloud computing, network virtualization, and network softwarization.

FELIPE ESTRADA-SOLANO is a Ph.D. student of telematics engineering at the University of Cauca, Colombia, and an international visiting student at the Cheriton School of Computer Science at the University of Waterloo. He received his Master's degree in telematics engineering (2016) and his Bachelor's degree in electronics and telecommunications engineering (2010) from the University of Cauca. His topics of interest include network and service management, network virtualization, software-defined networking, machine learning, and big data.

OSCAR M. CAICEDO is a full professor at the Universidad of Cauca, Colombia, where he is a member of the Telematics Engineering Group. He received his Ph.D. degree in computer science (2015) from the Federal University of Rio Grande do Sul, Brazil, and his M.Sc. in telematics engineering (2006) and his degree in electronics and telecommunications engineering (2001) from the University of Cauca. His research interests include network and service management, network virtualization, and software-defined networking.