Auto-scaling VNFs using Machine Learning to Improve QoS and Reduce Cost

Sabidur Rahman*, Tanjila Ahmed*, Minh Huynh†, Massimo Tornatore*‡, and Biswanath Mukherjee*
*University of California, Davis, USA †AT&T Labs, USA ‡Politecnico di Milano, Italy
Email: {krahman, tanahmed, mahuynh, mtornatore, bmukherjee}@ucdavis.edu

***Abstract*—Virtualization of network functions (as virtual routers, virtual firewalls, etc.) enables network owners to efficiently respond to the increasing dynamicity of network services. Virtual Network Functions (VNFs) are easy to deploy, update, monitor, and manage. The number of VNF instances, similar to generic computing resources in cloud, can be easily scaled based on load. Auto-scaling (of resources without human intervention) has been investigated in academia and industry. Prior studies on auto-scaling use measured network traffic load to dynamically react to traffic changes. In this study, we propose a proactive Machine Learning (ML) based approach to perform auto-scaling of VNFs in response to dynamic traffic changes. Our proposed ML classifier learns from past VNF scaling decisions and seasonal/spatial behavior of network traffic load to generate scaling decisions ahead of time. Compared to existing approaches for ML-based auto-scaling, our study explores how the properties (e.g., start-up time) of underlying virtualization technology impacts QoS and cost savings. We consider four different virtualization technologies: Xen and KVM, based on hypervisor virtualization, and Docker and LXC, based on container virtualization. Our results show promising accuracy of the ML classifier. We also demonstrate using realistic traffic load traces and optical backbone network that our ML method improves QoS and saves significant cost for network owners as well as leasers.**

***Index Terms*—Auto-scaling; virtual network functions; optical backbone network; machine learning; docker container; virtual machine; QoS; cost savings.**

I. Introduction

Network functions, such as those implemented in firewall, switch, router, Customer Premises Equipment (CPE), etc. have been traditionally deployed on proprietary hardware equipment, referred as middleboxes. This makes equipment upgrade, deployment of new features, and maintenance complex and time consuming for network owners. Virtualization of network functions can enable faster service deployment and flexible management [1]. Virtual Network Functions (VNFs) also allow us to use Commercial-Off-The-Shelf (COTS) hardware to replace costly vendor hardware. VNFs are usually hosted inside cloud data centers (DCs), or in smaller metro data centers (e.g., Central Office Re-architected as a Datacenter (CORD) [2]), or even inside core network nodes.

Auto-scaling is traditionally used in cloud computing, where amount of computational resources scales automatically based on load [3]. Auto-scaling is an important mechanism for VNF management and orchestration; and it can reduce the operational cost for network owners (e.g., telco-cloud operators, DC operators). Also network leasers (e.g., mobile virtual network operators, enterprise customers) can benefit from flexible usage and pay-per-use pricing models enabled by auto-scaling.

Traditional auto-scaling typically uses *reactive* threshold-based approaches, that are simple to implement. But thresholds are difficult to choose and unresponsive to handle dynamic traffic. Recent studies suggest *proactive* VNF scaling by combining traffic prediction and threshold-based methods. But using such methods for auto-scaling will lead to sub-optimal decisions. Hence, we propose a *proactive* Machine Learning (ML) classifier to produce scaling decisions (instead of traffic prediction) ahead of time.

Our proposal converts the VNF auto-scaling problem into a supervised ML classification problem. To train the supervised ML classifier, we use past VNF scaling decisions and measured network loads. The classification output is number of VNF instances required to serve future traffic without violating QoS requirements and deploying unnecessary VNF instances. Another contribution of this study is the analysis of the impact of underlying virtualization technology on auto-scaling performance. We compare four different virtualization technologies: Xen and KVM, based on hypervisor virtualization, and Docker and LXC, based on container virtualization. A virtual machine (VM) is an application/operating system (OS) environment that runs on host OS, but imitates dedicated hardware. In contrary, a container is a lightweight, portable, and stand-alone package of a software that can run on top of a host OS or host VM. Depending on the deployment scenario, critical properties of these VMs/containers such as start-up time can be significantly different.

Our study compares accuracy of the proposed ML classifier using realistic network traffic load traces collected from an ISP. We discuss the impact of additional features and more training data on classification accuracy; and quantify how the proposed methods improve QoS and reduce operational cost for network owners and leasers over a practical use-case example (Software Defined Wide Area Network (SD-WAN) with VNFs and optical backbone network).

The rest of this study is organized as follows. Section II reviews prior work on virtualization technologies and auto-scaling. Section III provides a formal problem statement for VNF auto-scaling. Section IV describes the proposed ML classifier and methodology. Section V discusses the performance of ML classifier and illustrates numerical results on improvement of QoS and cost savings. Section VI concludes

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

the study and indicates directions for future works.

II. Background and Related Work

VNFs can be deployed based on hypervisor virtualization or container virtualization. Compared to virtual machines (VMs), containers are lightweight, consume less CPU/memory/power resources, and have significantly less start-up time. Ref. [4] compares hypervisor-based virtualization (Xen, KVM, etc.) and container-based virtualization (Docker, LXC, etc.) in terms of power consumption. Containers require much less power, and this has a significant impact on operational cost. Choice of virtualization technology also impacts QoS. Ref. [5] discusses start-up time of different virtualization technologies, and shows how, after scaling decision, spawning a new VM/container takes longer/shorter time depending on the virtualization technology.

Prior studies on auto-scaling can be classified in two groups: threshold-based vs. prediction-based (time series analysis, ML, etc.). Threshold-based approaches have been used by DC owners [3] for scaling computing resources. Static-threshold-based approaches [6] [7] [8] use predefined upper and lower thresholds for scaling, which is not a practical solution in a dynamic demand scenario. Improvements have been proposed using dynamic threshold-based approaches [9] [10] [11]. As for prediction-based approaches, prior studies have used Auto-Regression (AR) [12], Moving Average (MA) [13], and Auto-Regressive Moving Average (ARMA) [14] to predict future workload for auto-scaling.

Recent studies have explored scaling of VNFs in telco-cloud networks too. Ref. [15] proposes a deadline- and budget-constrained auto-scaling algorithm for 5G mobile networks. Here, VNFs are dynamically scaled based on number of current jobs in the system using a heuristic algorithm. Being reactive in nature, this method will yield sub-optimal results in a network with fluctuating traffic. Ref. [16] proposes a resource-efficient approach to auto-scale telco-cloud using reinforcement learning, and emphasizes that VNF auto-scaling is crucial for both QoS guarantee and cost management. Reinforcement learning runs without any knowledge of prior traffic pattern or scaling decisions. Hence, auto-scaling approaches in [15] [16] do not benefit from the historic traffic data and scaling decisions.

Ref. [17] covers auto-scaling from DevOps point of view. This study considers scaling data-plane resources as a function of traffic load, but also in this case, a reactive approach that does not benefit from historic data is used. Moreover, compared to [15]–[17], our study explores the benefits of auto-scaling also from the leasers point of view.

III. Problem Description

The research problem of auto-scaling VNFs can be summarized as follows. Given:

- Set of VNF deployments (V), where each deployment $v \in$ V has one or more instances of VNFs serving network traffic. Each deployment *v* has a minimum number of allowed VNF instances (*v.min*) and a maximum number of allowed VNF instances (*v.max*).
- Optical backbone network topology G(D,E), where D is set of core network nodes and E is set of optical network links connecting the VNF deployments.
- Set of historic traffic load measurement data (H(*v,t*)) that requires processing from the VNF deployment *v* at time *t*.
- Set of QoS requirement (Q(*v*)): For each VNF deployment *v*, network packets can tolerate some level of delay, loss, etc.[1]

Our objective is to develop a method which dynamically scales the VNFs to minimize QoS violation and cost of operation/leasing. The next section will show how we can convert this problem to a ML classification problem. We assume that the ML classifier will be part of the orchestration and management modules of the network owner (similar to AT&T's ECOMP [18]).

IV. Proposed ML Classifier and Methodology

In machine learning, an *instance* is a set of features/values representing a specific instance of the problem. For example, in our study, one *feature* of the problem instance is time of day. Another *feature* is the value of the measured traffic load at time of day. We can associate each instance (set of *features*) of the problem to a *class*, i.e., a classification decision. We convert the auto-scaling problem to a classification problem by training the classifier with a set of correctly-identified instances, called *training set*. In the training phase, the ML classifier learns a mapping between the *features* and *classes*. After the training phase, a classifier can be tested using a set of instances, called *test set*, which is not part of the *training set*.

The Time vs. Traffic (bits) graph in Fig. 1 explains the input and output of the ML classifier. Fig. 1 shows different timestamps (*a, b, c,* etc.) and auto-scaling decisions $s(n-1), s(n), s(n+1)$ etc., where $s(n)$ indicates the n-th auto-scaling decision. Let network traffic load for timestamp *x* be given by $\lambda(x)$ and timestamp of auto-scaling step *n* be given by $\tau(n)$.

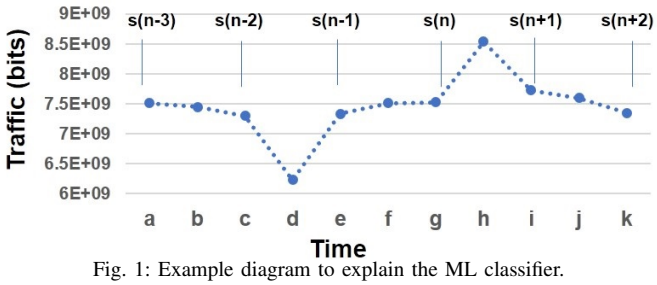

Fig. 1: Example diagram to explain the ML classifier.

A. Feature Selection (Input)

Feature selection is the first important step towards defining the ML classifier. All our *features* are of numeric value. Referring to Fig. 1, for auto-scaling decision $s(n+2)$, the proposed ML classifier maps recent traffic trends to a proactive scaling decision (*class*). In this study, we convert the network traffic load measurements into following features:

- Time of the day (k)
- Measured traffic at time *k* $(\lambda(k))$

[1]Prior studies [15] [16] assume that this QoS requirement can be converted to a number of VNFs that can serve the packets without violating QoS requirements. For example, one VNF instance can serve up to 1 Gbps line-rate traffic without violating QoS.

- Traffic change from time *j* to *k* ($\lambda(k) - \lambda(j)$)
- Measured traffic at time *j* ($\lambda(j)$)
- Traffic change from time *i* to *j* ($\lambda(j) - \lambda(i)$)
- Measured traffic at time *i* ($\lambda(i)$)
- Traffic change from time *h* to *i* ($\lambda(i) - \lambda(h)$)
- Measured traffic at time *h* ($\lambda(h)$)
- Traffic change from time *g* to *h* ($\lambda(h) - \lambda(g)$)
- Measured traffic at time *g* ($\lambda(g)$)
- Traffic change from time *f* to *g* ($\lambda(g) - \lambda(f)$)
- Measured traffic at time *f* ($\lambda(f)$)
- Traffic change from time *e* to *f* ($\lambda(f) - \lambda(e)$)
- Measured traffic at time *e* ($\lambda(e)$)
- Traffic change from time *d* to *e* ($\lambda(e) - \lambda(d)$)
- Measured traffic at time *d* ($\lambda(d)$)
- Traffic change from time *c* to *d* ($\lambda(d) - \lambda(c)$)
- Measured traffic at time *c* ($\lambda(c)$)
- Traffic change from time *b* to *c* ($\lambda(c) - \lambda(b)$)
- Measured traffic at time *b* ($\lambda(b)$)
- Traffic change from time *a* to *b* ($\lambda(b) - \lambda(a)$)
- Measured traffic at time *a* ($\lambda(a)$)

We consider up to 22 features containing information of measured traffic load and traffic load change from recent past. Feature selection can be improved by removing less significant *features* using attribute selection algorithms, Principal Component Analysis (PCA), learning curve analysis, etc. In Section V, we explain how we determine the number of features that provides the best accuracy for the classifier.

B. Class Definition (Output)

Next step is to define the output of the ML classifier, i.e., set of target *classes* that the classifier tries to classify. In our study, *class* depicts number of VNF instances, which is an integer value between *v.min* and *v.max*. To generate the labeled training set (*instances* with known *class* labels), we ensure that the scaling decision taken at step *n* allocates enough resources to serve traffic until the next decision-making step $n + 1$. To define how we generate the *class* value, we propose two different approaches:

1) QoS-prioritized ML (QML): In VNF auto-scaling, there is a trade-off between QoS and cost saving. We need to allocate more resources to guarantee QoS, but allocating more resources reduces cost saving. QML gives priority to QoS over cost saving. To guarantee QoS, auto-scaling decision at step n (present) considers future traffic changes until next auto-scaling step $n + 1$. QML generates the *class* value as follows:

$$s(n) = max(qos(\lambda(t))), \forall t \epsilon \{\tau(n), , \tau(n+1)\} \qquad (1)$$

where *t* is timestamp with traffic data between steps n and $n+1$ (including $\tau(n)$ and $\tau(n + 1)$).

Function qos(.) takes the traffic load measured at time *t* as input, and outputs the number of VNFs required to serve the measured traffic in line rate, without violating QoS.

2) Cost-prioritized ML (CML): In some cases, network owner/leaser may choose to ignore short-lived bursty traffic between steps n and $n + 1$ to save cost by avoiding over-provisioning of VNFs and accepting short-lived degradations. Auto-scaling decision for CML considers measured traffic load only at step n (present) and at next auto-scaling step $n + 1$ (future). CML generates the *class* value as follows:

$$s(n) = max(qos(\lambda(\tau(n))), qos(\lambda(\tau(n+1)))) \qquad (2)$$

where $\tau(n)$ is the time at which step n takes place and $\tau(n+1)$ is the time when step $n + 1$ occurs.

C. Data Generation

After defining the input and output of the classifier, the next step is dataset generation. For training-set and test-set generation, we assume that realistic traffic-load measurement data H(v, t) is available at the network node where auto-scaling is performed. Table I shows an example of training instance for QML and CML for scaling decision at steps n and $n + 1$ (Fig. 1) where f, g, h, i, etc. are time values.

TABLE I: Instances with Known Labels

Case	Fea. 1	Fea. 2	Fea. 3	Fea. 4	...	QML Class	CML Class
1	g	$\lambda(g)$	$\lambda(g)-\lambda(f)$	$\lambda(f)$		$qos(\lambda(h))$	$qos(\lambda(\tau(n+1)))$
2	i	$\lambda(i)$	$\lambda(i)-\lambda(g)$	$\lambda(g)$		$qos(\lambda(i))$	$qos(\lambda(\tau(n+1)))$

D. Machine Learning Algorithms

Selecting the right ML algorithm is the next step towards training the classifier. We have explored different categories of algorithms available in the ML suite WEKA [19], including decision-tree-based algorithms, rule-based algorithms, artificial neural networks, and Bayesian-network-based algorithms. Below is a brief introduction to the algorithms in our study:

(a) Random Tree: Random tree is the decision-tree implementation of WEKA. "Random" refers to a decision tree built on a random subset of columns. Decision trees classify instances by sorting them based on feature values. Each node in a decision tree is a feature, and each branch represents a value that the node uses to classify instances.

(b) J48: *C4.5* algorithm, proposed by Ross Quinlan is one of the most well-known algorithms to build decision trees [20]. J48 is WEKA's implementation of C4.5.

(c) REPTree: Reduced Error Prunning (REP) tree is another improved version of decision-tree algorithm. REPTree benefits from information gain and minimization of error arising from variance.

(d) Random Forest: Random forest improves the tree classifier by averaging/voting between several tree classifiers. Instead of building multiple trees on the same data, random forest adds randomness by builting each tree on slightly-different rows and randomly-selected subset of columns.

(e) Decision Table: Similar to tree-based algorithms, rule-based algorithms such as decision table try to infer decisions by learning the "rule" hidden inside the training instances. Each decision corresponds to a variable or relationship whose possible values cover the decisions.

(f) Multi-Layer Perceptron (MLP): MLP is a class of feed-forward artificial neural networks, which are powerful classification and learning algorithms.

(g) Bayesian Network (BayesNet): Bayesian network is a probabilistic model that represents a set of random variables and their conditional dependencies.

E. Performance Evaluation

A test dataset is used to evaluate the performance of the trained classifier. Given a trained ML classifier and a test set, the test outcome is divided into four groups: i) True Positive (TP): positive instances correctly classified; ii) False Positive (FP): negative samples incorrectly classified as positive; iii) True Negative (TN): negative samples correctly classified; iv) False Negative (FN): positive samples wrongly classified as negative.

We consider three different performance metrics for our study:

(a) Precision (%): Precision corresponds to the fraction of predicted positives which are in fact positive. Precision is a strong indication of accuracy for the ML classifier. Precision is given by percentage of: $TP/(TP+FP)$.
(b) False Positive (%): FP is an important indication of the ML classifiers as lower FP indicates less classification mistakes.
(c) ROC area (%): Receiver Operating Characteristic (ROC) curve is a graphical plot in which true positive rate $(TP/((TP+FN))$ is plotted as function of the false positive rate $(FP/(FP+TN))$. ROC area is a robust metric for ML classifier performance evaluation.

F. Conversion of Number of VNFs to Optical Backbone Network Capacity

In Section V, we explore leasing cost of a SD-WAN use-case with VNFs and optical backbone network. We use the following equation to convert the output from ML classifier (number of VNFs required) to the required amount of bandwidth (C) over the transport network:

$$C = s(n) * Q' \tag{3}$$

where $s(n)$ is number of required VNFs at step n, and Q' is the conversion unit for line-rate traffic processed by each VNF (e.g., one VNF can process upto 1 Gbps traffic).

V. Illustrative Neumerical Examples

This section shows the performance of the proposed ML classifier for auto-scaling. Our results show how the proposed approaches improve QoS and reduce cost for both network owners and leasers.

A. Simulation Setup

We consider that VNFs are hosted inside virtualized instances on top of physical servers, similar to generic VMs in DC deployments [5]. We assume that the maximum traffic load that requires processing at such a deployment is equivalent to 10 Gbps, and each VNF can handle maximum 1 Gbps traffic without degrading QoS. The acceptable range for number of VNFs is minimum (*v.min*) one and maximum (*v.max*) ten. For VNF auto-scaling, we consider "horizontal" scaling, i.e., removing or adding new VNF instances every time we scale in/out. When load increases, the network owner deploys additional VNF instances to satisfy QoS requirement. In case of decreasing load, the network owner removes some VNF instances to save operational cost (e.g., electricity usage).

We use realistic traffic load traces from [21]. Traffic load data (in bits) was collected every five-minute interval over a 1.5-month period from a private ISP and on a transatlantic link [22]. We use the traffic data to generate "features" and "classes" for training and testing both QML and CML. As traffic load traces are at every 5-minute interval, we assume auto-scaling decisions are made every 10 minutes.

B. Auto-scaling Accurcy of Propsoed ML Classifier

In Fig. 2, we compare the auto-scaling accuracy (precision) of the proposed ML classifier. We compare QML (QoS-prioritized), CML (cost-prioritized) approach with prior study "MA" [13]. Prior study "MA" proposed a time-series-based prediction approach using Moving Average (MA) for proactive auto-scaling. For Fig. 2, we use forty days of data ($40 * 144 = 5760$ instances) for training and two days of data ($2 * 144 = 288$ instances) for testing. First, Fig. 2 shows that the ML classifier takes auto-scaling decisions with higher accuracy for both QML (96%) and CML (93%) compared to MA (71%) for the same test data. Second, Fig. 2 shows that, among different ML algorithms used to train the classifier, "Random Forest" has higher precision for both QML and CML approaches. Note that QML uses more data points to generate the "class" (Eqn. (1)) than CML (Eqn. (2)). Difference in prediction accuracy is due to different "class" generation results.

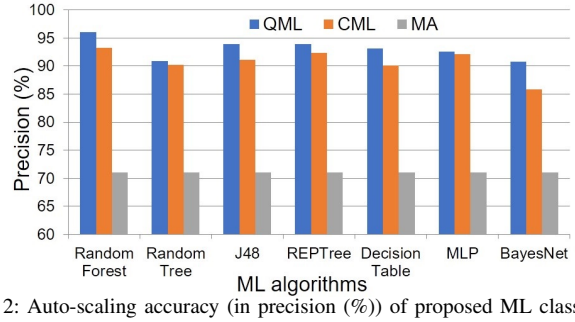

Fig. 2: Auto-scaling accuracy (in precision (%)) of proposed ML classifier.

Decision-tree-based algorithms perform better for the scaling decision compared to neural-network-based and Bayesian-network-based algorithms. Among decision-tree-based algorithms, "Random Forest" leads with highest precision (96%), highest ROC Area (99.6%), and lowest false positives (1.1%). Clearly, the pattern of the data and feature set favors decision-tree-based algorithms to learn better than other ML techniques. But "Random Forest" further improves the performance by averaging/voting between several tree-classifier instances. For the rest of the numerical evaluations, we use the results from "Random Forest" to explore learning curves, QoS degradation, and leasing cost.

C. Impact of Features and Training Dataset Size on Auto-scaling Accuracy

Figs. 3-4 provide auto-scaling learning curve of the proposed ML classifier. Learning curve analysis helps us to understand the following two important aspects of training a classifier: "How many features generate best results?" and "Does more data help or not?"

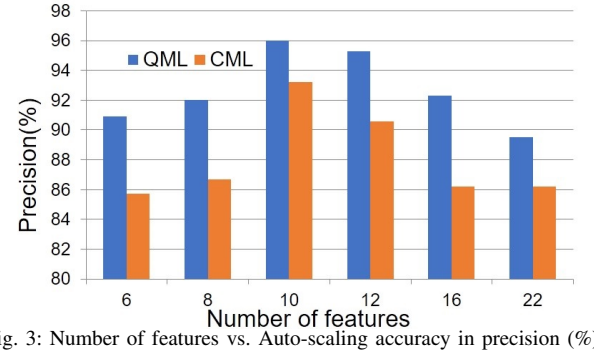

Fig. 3: Number of features vs. Auto-scaling accuracy in precision (%).

Fig. 3 shows the impact of number of features in the accuracy of auto-scaling decisions made by the ML classifier. For example, number of features "6" means we are using only the first six features from Section IV.A. As shown in Fig. 3, the accuracy of the ML classifier increases with the number of features. But, after number of features goes higher than "10", the accuracy decreases. This means that, if we keep adding more features by moving away from the time of scaling decision, the additional features impact the accuracy negatively.

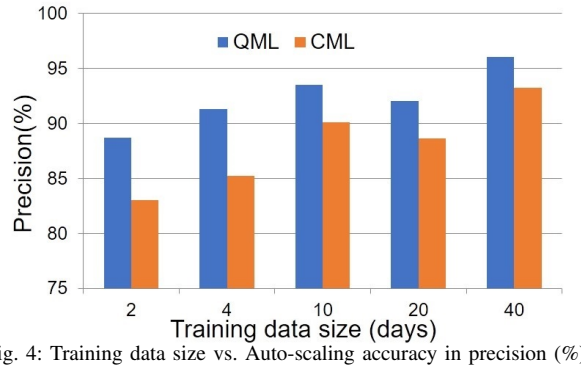

Fig. 4: Training data size vs. Auto-scaling accuracy in precision (%).

Fig. 4 shows the impact of training dataset size on the auto-scaling accuracy (precision) of the ML classifier. Although general intuition is that, with more data, the ML classifier should perform better, we observe that the precision steadily increases with additional training instances, with an exception at 20 days (2880 training instances). There can be two explanations of this phenomenon. First, the testing data for 20-day scenario is much different than the training data. Second, the seasonal pattern and periodicity of the traffic load introduced new pattern which was not present in training data. In other words, a classifier trained with 20 days of training instances fails to capture the changes of traffic over the month. But the classifier trained with 40 days of training instances overcomes limitations of the seasonal pattern, and utilizes the learned pattern to improve the precision of learning. Future studies should explore impact of such seasonal (weekly, monthly, quarterly, yearly, etc.) behavior of network traffic load data on ML classifier and auto-scaling. For results in Fig. 2, and rest of the study, we have considered 10 features and 40-day training data.

D. Impact of Auto-scaling on QoS

To explore the impact of auto-scaling (using the proposed ML classifier) and virtualization technology on QoS-guarantee, Fig. 5 compares QML, CML, and MA in terms of "Degraded QoS". We define "Degraded QoS" as the cumulative time (minutes) a VNF deployment spends in QoS violation. We explore two reasons behind "Degraded QoS": i) Low provisioning: The proactive method failed to allocate enough resources to accommodate future traffic; and ii) Start-up time: Even if the auto-scaling method decides to turn on more VNFs, spawning a new VM takes time. The impact on start-up time is expected to be different from different virtualization techniques. Hypervisor-based virtual machines (VMs) and containers are compared in Fig. 5. The results are taken from the two-day test output from ML classifier using "Random Forest" algorithm.

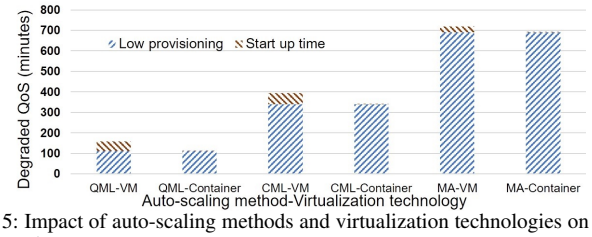

Fig. 5: Impact of auto-scaling methods and virtualization technologies on QoS (minutes).

Fig. 5 shows that QML spends significantly less time in "Degraded QoS" compared to CML and MA. We also observe that choice of virtualization technology can make a difference in terms of QoS as containers have much faster start time (0.4 seconds) than hypervisor-based VMs (100 seconds) [5], and this helps containers to improve QoS significantly.

E. Impact of Auto-scaling on Leasing Cost

We assume now an SD-WAN use-case, where a leaser leases VNFs and connectivity over an optical backbone transport network. Fig. 6 shows the backbone network topology and an enterprise customer network with one headquarter and three branch offices connected over the optical backbone network. For each of the four offices, VNFs are deployed on servers. For our simulation scenario, we assume that three VNF services (namely, firewalls, routers, and private branch exchanges (PBXs)) are deployed in all the sites. We also assume that each of these offices experiences traffic load same as the two-day test dataset. We have used Eqn. (3) to convert the output (auto-scaling decisions) from ML classifier to allocate the required connectivity over the optical backbone network. For example, if ML classifier outputs the number of VNFs required to serve the traffic to be two, and each VNF can serve up to 1 Gbps line-rate traffic, then the backbone network requires 2 Gbps capacity to carry the traffic without violating QoS.

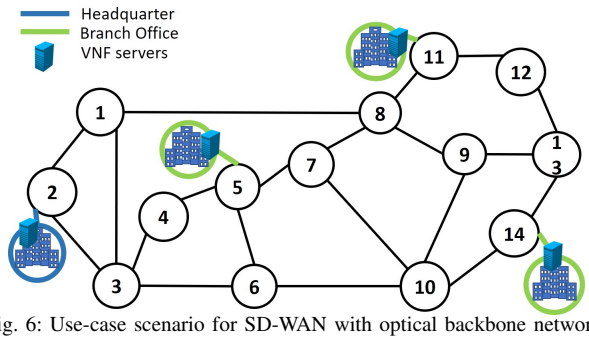

Fig. 6: Use-case scenario for SD-WAN with optical backbone network.

Fig. 7 compares the cost of VNFs, backbone network, and service degradation for different auto-scaling approaches. For VNF leasing cost, we consider the leasing cost of Google cloud [23]. For backbone-network leasing cost, we consider leasing cost from Google fiber [24]. For degraded-service cost, we assume the enterprise customer loses $1 for every 10 minutes of degraded service. Leasing cost shown in Fig. 7

is derived from testing on two days of data using "Random Forest" algorithm.

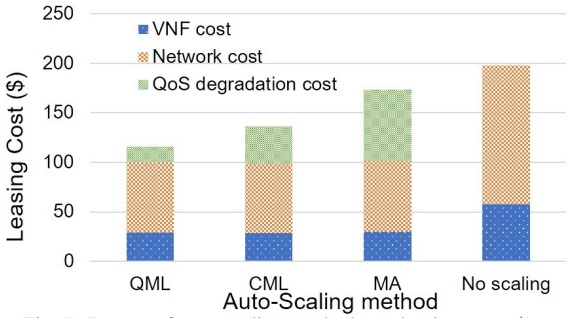

Fig. 7: Impact of auto-scaling methods on leasing cost ($).

Fig. 7 shows that QML ensures the lowest leasing cost. However, by considering just VNF cost and network cost, CML ensures lower cost than QML and MA. This means enterprises which are interested to reduce operational cost by sacrificing QoS will be benefited from CML. On the other hand, enterprises which cannot afford degradation of service (i.e., QoS violation is costly) will benefit from our QML method.

VI. Conclusion

Our study proposed a Machine Learning method to perform VNF auto-scaling. The ML classifier learns from historic VNF auto-scaling decisions and shows promising accuracy rate (96%). Illustrative results show the impact of number of features and training data size on the proposed ML classifier. We also studied the impact of start-up time of four different virtualization technologies on QoS. We explored a practical SD-WAN use-case with optical backbone network showing that our proposed method yields lower leasing cost for network leasers compared to prior works. Future studies should consider exploring detailed analysis of operational and leasing costs for more such use-cases. In our study, we explored only horizontal scaling of VNFs. Vertical scaling (i.e., adding/removing CPU/memory resources to the same virtual instance) for VNFs is an important direction yet to be explored. As network services are often deployed as service chains, future studies should explore auto-scaling methods for such scenarios. Future studies should also explore operational costs and cost model [25] exploring electricity usage, VNF life-cycle management, degradation of service, and other operational activities.

References

[1] ETSI, "Network functions virtualisation: Introductory white paper," 2012.

[2] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," *IEEE Commun. Magazine*, vol. 54, no. 10, pp. 96-101, Oct. 2016.

[3] "Amazon Web Sevices - Auto-scaling", Amazon, [Online]. Available: https://aws.amazon.com/autoscaling/. [Accessed: April 18, 2017]

[4] R. Morabito, "Power Consumption of Virtualization Technologies: an Empirical Investigation," *8th IEEE/ACM Intl. Conf. on Utility and Cloud Computing*, pp. 522-527, 2015.

[5] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "A Framework and Algorithm for Energy Efficient Container Consolidation in Cloud Data Centers," *IEEE World Congress on Services*, 2015.

[6] K. Kanagala and K. C. Sekaran, "An approach for dynamic scaling of resource in enterprise cloud," *IEEE Intl. Conf. on Cloud Computing Tech. and Sci.*, pp. 345-348, 2013.

[7] M. M. Murthy, H. A. Sanjay, and J. Anand, "Threshold based auto scaling of virtual machines in cloud environment," *Intl. Conf. on Network and Parallel Computing*, pp. 247256, 2014.

[8] C. Hung, Y. Hu, and K. Li, "Auto-scaling model for computing system," *Intl. Journal of Hybrid Info. Tech.*, vol. 5, no. 2, pp. 181-186, April 2012.

[9] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559-592, 2014.

[10] A. Beloglazov and R. Buyya. "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," *Proc., 8th Intl. Wksp on Middleware for Grids, Clouds & e-Science,* 2010.

[11] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities," *Proc., 1st workshop on Automated Control for Datacenters and Clouds,* pp. 13-18, 2009.

[12] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," *Proc., 11th Intl. Conf. on Quality of Service,* pp. 381398., 2003.

[13] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," *Proc., IEEE Intl. Conf. on Services Computing,* 2010.

[14] W. Fang, Z. Lu, J. Wu, and Z. Cao, "RPPS: a novel resource prediction and provisioning scheme in cloud data center," *IEEE 9th Intl. Conf. on Services Computing,* pp. 609616, 2012.

[15] T. Phung-Duc, Y. Ren, J. C. Chen, and Z. W. Yu, "Design and Analysis of Deadline and Budget Constrained Autoscaling (DBCA) Algorithm for 5G Mobile Networks," *arXiv preprint arXiv:1609.09368,* Sep. 2016.

[16] P. Tang, F. Li, W. Zhou, W. Hu, and L. Yang, "Efficient Auto-Scaling Approach in the Telco Cloud Using Self-Learning Algorithm," *Proc., IEEE GLOBECOM,* Dec. 2015.

[17] S. V. Rossem, X. Caiy, I. Cerratoz, and P. Danielssonx, "NFV Service Dynamicity with a DevOps Approach: Insights from a Use-case Realization," *IEEE Intl. Symp. on Integrated Network Management,* 2017.

[18] "ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper," AT&T Inc., [Online]. Available: http://about.att.com/content/dam/snrdocs/ecomp.pdf. [Accessed: April 18, 2017]

[19] E. Frank, M. A. Hall, and I. H. Witten, "The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques," Morgan Kaufmann, Fourth Edition, 2016.

[20] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, pp. 3-24, 2007.

[21] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Computer network traffic prediction: a comparison between traditional and deep learning neural networks," *Int. J. Big Data Intelligence*, vol. 3, no. 1, pp. 28-37, 2016.

[22] "Internet traffic data", DataMarket, [Online]. Available: https://datamarket.com. [Accessed: Mar. 15, 2017].

[23] "Google cloud pricing", Google, [Online]. Available: https://cloud.google.com/compute/pricing. [Accessed: Sep 20, 2017].

[24] "Google fiber pricing", Google, [Online]. Available: https://fiber.google.com/cities/kansascity/plans/. [Accessed: Sep 20].

[25] S. Rahman, A. Gupta, M. Tornatore, and B. Mukherjee, "Dynamic Workload Migration over Optical Backbone Network to Minimize Data Center Electricity Cost," *Proc., IEEE Intl. Conf. on Commun.*, May 2017.