Behavioral analysis for Virtualized Network Functions: a SOM-based approach

Tommaso Cucinotta[1], Giacomo Lanciano[12], Antonio Ritacco[1], Marco Vannucci[1], Antonino Artale[3], Joao Barata[4], Enrica Sposato[3] and Luca Basili[3]

[1]*Scuola Superiore Sant'Anna, Pisa, Italy*

[2]*Scuola Normale Superiore, Pisa, Italy*

[3]*Vodafone, Milan, Italy*

[4]*Vodafone, Lisbon, Portugal*

Keywords: Self-Organizing Maps, Machine Learning, Network Function Virtualization

Abstract: In this paper, we tackle the problem of detecting anomalous behaviors in a virtualized infrastructure for network function virtualization, proposing to use self-organizing maps for analyzing historical data available through a data center. We propose a joint analysis of system-level metrics, mostly related to resource consumption patterns of the hosted virtual machines, as available through the virtualized infrastructure monitoring system, and the application-level metrics published by individual virtualized network functions through their own monitoring subsystems. Experimental results, obtained by processing real data from one of the NFV data centers of the Vodafone network operator, show that our technique is able to identify specific points in space and time of the recent evolution of the monitored infrastructure that are worth to be investigated by a human operator in order to keep the system running under expected conditions.

1 INTRODUCTION

In the context of network services provisioning, the novel *Network Function Virtualization (NFV)* paradigm (NFV Industry Specif. Group, 2012) has recently gained more and more traction due to the increasingly demanding requirements and complex scenarios faced by network operators.

Such approach has been developed with the purpose of replacing the traditional deployment of *specialized* physical appliances, typically sized for the peak-hour and very costly to maintain, in favor of *cloud computing* technologies, enabling on-demand access to a diverse set of virtualized resources (computing, storage, networking, etc.) that can be dynamically allocated to fit the needs of time-varying workloads. The high flexibility of this type of infrastructures is nowadays crucial for an organization operating in this area, as it allows for quickly adapting and effectively coping with the numerous challenges coming from the new connectivity scenarios of the future Internet.

Virtualized Network Functions (VNFs) are thus implemented as distributed software applications that can be deployed on a – private – cloud infrastructure managed by a network operator, enabling elastic and resilient services that can be easily reconfigured according to the requirements of highly dynamic workloads. For NFV data centers, the choice of private cloud infrastructures – as opposed to the use of public cloud services – is also corroborated by latency-related concerns. Indeed, since such service-chains are highly delay-sensitive (e.g. LTE, 4G), it is unpractical to rely on public cloud infrastructures, that are usually shared among multiple tenants and not necessarily deployed according to the network operator needs.

In order to guarantee scalability, robustness to failure, high availability, low latency, such systems are typically designed as large-scale distributed systems (Ostberg et al., 2017), often partitioned and/or replicated among many geographically dislocated data centers. The larger the scale, the more operations teams have to deal with complex interactions among the various components, such that diagnosis and troubleshooting of possible issues become incredibly difficult tasks (Gulenko et al., 2016a). Since many diverse kinds of Service Level Agreements (SLAs) – specifying which are the guaranteed quality of service (QoS) requirements – are often in place between network operators and their customers, it is crucial to effectively monitor the status of the data center through

an efficient distributed monitoring infrastructure that continuously gathers system-level metrics from all the different levels of the architecture, e.g., physical hosts metrics, virtual machines metrics, application-level key performance indicators (KPIs), event logs, etc.. Such data usually drives the decisions of human operators, for instance, in terms of which actions must be taken to restore the expected conditions of the system after an outage has occurred, or how the available components should be reconfigured to prevent possible SLA violations in case of an unexpected increase in the workload.

One of the major problems of data center operators is *anomaly detection*, i.e., pinpointing unexpected and/or suspect behaviors of the system whenever it significantly deviates from the normal conditions. Indeed, system outages are typically preceded by failures, performance degradation and similar anomalous behaviors that, if detected on time, can be acted upon by raising timely alerts and/or autonomously triggering suitable procedures *before* other components, or the end users, experience any actual issue. This is fundamental to the purpose of establishing – possibly automated – *proactive* strategies to minimize the risk of SLA violations (i.e., such that human experts can focus their efforts on the most critical activities), or at least to alert the staff to prepare the remediation/mitigation procedures in advance. The large availability of data produced in data centers like those dedicated to NFV allows for employing effective data-driven methods, such as those coming from the research field of Machine Learning (ML).

1.1 Paper contributions and organization

In this paper, we propose an approach based on *Self-organizing Maps (SOM)* to perform a pattern analysis of VM metrics aiming at providing a comprehensive overview of the major behavioral patterns and detecting possible anomalies in a data center for NFV. The technique can be used to perform a joint analysis of *system-level metrics* available from the infrastructure monitoring system and *application-level metrics* available from the individual VNFs. It aims at supporting data center operations and specifically capacity and performance monitoring, by providing insightful information on the behavioral patterns, in terms of resource consumption and exhibited performance, of the analyzed VNFs.

This paper is organized as follows. After discussing the related literature in Section 2, we present the approach in Section 3, along with its implementation and the data processing workflow we designed to apply it to the massive data set available in the Vodafone infrastructure. In Section 4, we discuss some of the obtained experimental results to prove the practical relevance of the proposed approach. Section 5 concludes the paper with our final remarks and ideas for future directions of research in the area.

2 RELATED WORK

In this section, we briefly review some of the most related works that are found in the research literature on using ML, and SOMs in particular, for classification and anomaly detection in cloud and NFV data centers.

Anomaly detection can be framed as the problem of pinpointing unexpected and/or suspect behaviors of a system whenever it significantly deviates from the normal conditions. Similar problems can be found in other fields and applications such as, for instance: intrusion detection in cyber-security, machinery fault (Samrin and Vasumathi, 2017) and product quality issues detection (Van den Berg et al., 2018) in industrial contexts, or fraud detection in finance (Malini and Pushpa, 2017). It is important to stress that anomaly detection is, in general, an inherently imbalanced problem due to the scarcity of anomalous observations with respect to the ones related to the normal conditions of a system. In order to tackle this kind of challenges, a huge amount of solutions have been proposed that, depending from the scenario and the nature of the data to be processed, pose their foundations on well-established techniques coming, for instance, from the research fields of information theory and statistics.

In the recent years, ML techniques have been gaining more and more traction in the context of anomaly detection applications because of their proven effectiveness in many of the aforementioned scenarios. This is mainly due to the versatility of this kind of methods and the increasing availability of data from which they can learn from, in a continuous manner (Buczak and Guven, 2015). Most of the approaches to anomaly detection address the associated challenges by feeding ML models with counters like CPU utilization, memory contention and network-related metrics (Sauvanaud et al., 2018; Wallschläger et al., 2017; Gulenko et al., 2016a). Others include also system-level and/or application-level event logs in the analysis to increase the amount of features and facilitate the extraction of relevant patterns (Watanabe et al., 2012; Farshchi et al., 2018). Embedding textual information has been in fact made easier by the advancements in Natural Language Processing (NLP) research field (Bertero et al., 2017). Few ex-

isting works also consider the need of assisting human operators in conducting root-cause analysis to be a highly desirable feature of anomaly detection systems (Gupta et al., 2017; Pitakrat et al., 2018).

One of the major roadblocks that can be encountered when applying ML for solving a task is the scarcity, or the complete absence, of labelled data, a very common scenario in many practical applications. Such issues can be overcome by employing so-called *unsupervised* learning techniques that, as the definition suggests, are designed to operate without a ground truth (i.e., annotated data). It is worth noticing that this characteristic of such class of learning algorithms has the side-effect of increasing the amount of data that can be used for training a ML model. The principal application of unsupervised techniques is *clustering* that consists in the formation of groups (the *clusters*) of data samples that are similar, where similarity is defined according to the employed distance function.

A SOM is a particular kind of neural network that applies the so-called *competitive learning* for cluster formation (Haykin, 2007). In this context when a new sample is presented to the SOM during the training, the Best Matching Unit (BMU) – the closest neuron to the data sample according to the employed distance function – is selected and BMU and its neighbors are rewarded through a weight update that make them more similar to the selected sample. The iteration of this process leads to the formation of the clusters that are represented by the associate SOM neurons. SOMs are designed for mapping high-dimensional data into a lower-dimensional (typically 2-dimensional) space. One of the main characteristics of the obtained clustering is that it preserves the *topology* and *distribution* of training data, at clusters-level. In practice it means that more clusters will be located in the more dense regions of the original domain (distribution) and that similar data samples will be associate to the same cluster or to neighbor clusters (topology).

In the context of anomaly detection, such approaches usually operate by building, starting from training data, a set of clusters of samples that are representative of the expected – normal – conditions of a system. After training, such model can be exploited to compare new patterns to known behaviors according to a predefined distance metric, in order to infer whether the observations are anomalous or not. As a *neural* approach to clustering, SOMs have achieved remarkable results at processing industrial data (Díaz et al., 2008; Canetta et al., 2005) given their ability to yield a distribution of the clusters in the problem domain that faithfully reflects the observed phenomenon behavior.

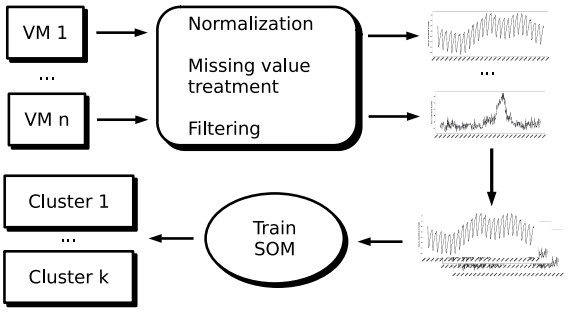


Figure 1: Overview of the SOM-based clustering workflow.

For what concerns NFV applications, the existing literature reports that ML techniques have been effectively used to solve different problems. In particular, in (Gulenko et al., 2016b) a set of ML techniques are tested for an anomaly detection application. In this case, though, only supervised methods are considered and their performance is compared on data sets containing NFV features associated to different types of faults. Similarly, in (Miyazawa et al., 2015), a *supervised* SOM-based method is proposed for fault detection. Here, a SOM is used to cluster *labelled* data, annotated by human experts to state which clusters correspond to faulty conditions, related to NFV performance indicators. In (Niwa et al., 2015), SOM-based and other general clustering techniques are used for the same purpose in a small test-bed in the context of NFV. Likewise, in (Le et al., 2018), the popular K–means algorithm is used to cluster cells traffic data in order group cells with similar through–time behavior and allow resources optimization.

3 PROPOSED APPROACH

In this paper, we propose the use of self-organizing maps (SOMs) in order to perform a pattern analysis of the VMs behavior. Our approach focuses on the joint analysis of two classes of metrics that are normally collected and analyzed independently from one another: *system-level metrics*, mostly related to resource consumption of the hosted VMs, i.e., those related to the utilization of the underlying infrastructure, hereafter also referred to as *INFRA metrics*, which are usually available through the NFV infrastructure manager (e.g., the well-known VMWare vRealize Operations[1] or others); and *application-level metrics*, published by individual virtualized services through their own monitoring subsystems, which will be referred to as *VNF metrics*. This allows for gathering a comprehensive overview of the major behavioral pat-

[1]More information at: https://docs.vmware.com/en/vRealize-Operations/index.html.

terns that characterize VMs and possibly identifying suspect (anomalous) behaviors.

Our technique relies on SOMs because of their beneficial features which make them a useful method for clustering, such as the ability to preserve the topology in the projection, meaning that similar input patterns are captured by nearby neurons. A VM is observed trough its movement among best matching unit (BMU) during the time horizon under analysis, so that any changes in *"far"* BMU could be used to trigger an alarm.

3.1 Workflow

We realized a SOM-based clustering tool that is capable of applying clustering using jointly a list of input metrics. In our experimentation, we have been applying this technique over individual monthly data available with a 5-minutes granularity (288 samples per day per metric per monitored VM or physical host), amounting to several GBs of data per month, for a specific region. The overall workflow that we applied to transform the available input INFRA metrics is summarized in Figure 1. First, the raw data are preprocessed to address possible data-quality issues and to retain only the information related to the metrics relevant for the analysis. The input samples to the SOM are then constructed, for each VM, by dividing the time horizon under analysis according to a predefined period and consolidating the contributions of the individual metrics in a single vector. Then, such data are fed to the SOM, that outputs for each of them the neuron capturing the closest behavior, providing a clustering of the input samples.

The input data are filtered, on the k specified metrics, and partitioned to have a sample (i.e., a time-series) for each metric, VM and period (usually a day) of the time horizon under analysis. Before being fed to the SOM, samples are subject to a preprocessing phase, focusing on possible issues such as (i) missing values and (ii) significant differences in the magnitude of the values of different metrics.

On the one hand, to address (i), a data imputation strategy, consisting in a simple linear interpolation, is performed to mitigate the effect of the possible absence of data points within a sample, to retain as much data as possible for the analysis. Although, in order to preserve the quality of the data set, the interpolation step has been designed not to be *aggressive*, so that, if too many consecutive samples are missing for a given input time series, then the time series is discarded. Note that each time-series contains the evolution of one or more metrics throughout a specific day for a given VM (or host).

On the other hand, it is recommended to address (ii) when applying SOM since, due to the samples distance evaluation mechanism, metrics with significantly larger values (e.g. number of transmitted/received packets or bytes) tend to hide the contribution of other metrics which can only take on smaller values, being bounded by a predefined range that is much smaller (e.g., CPU utilization percentage). We have devised two possible strategies in order to have the SOM dealing with values included in similar ranges, for each of the metrics under analysis. The first strategy, referred to as *normalized*, consists in scaling each time-series by subtracting its mean and dividing by its standard deviation each data point. Notice that using such strategy hides any information regarding the magnitude of the original values and emphasizes the shapes. The second strategy, referred to as *non-normalized*, consists in scaling each time-series to a range of values between 0 and 1 considering, for each metric, the historical minima and maxima values observed for that metric only. Note that such strategy retains information regarding the magnitude of the original values while keeping the data bounded in a relatively small interval. However, this technique results in having different metric patterns with very similar shape but differing merely in their magnitude, being grouped into different SOM neurons at a certain distance from each other (in the SOM grid topology), whereas the normalized strategy would group them together within a single neuron (or a few very close ones). Depending on the chosen strategy, we obtain an analysis focusing on the shapes of the behavioral patterns only, or we can distinguish also among different absolute values of the average levels of the metrics. In general, in the latter case (non-normalized) one should expect more clusters to exist with respect to the former case (normalized), due to the fact that the system could have experienced very diverse levels of load during the time horizon under analysis. Hence, a non-normalized analysis needs generally an increase of the SOM grid size, in order to avoid over-population of neurons with too many (non-normalized) patterns crowding within the same (BMU) neuron, despite them being significantly distant from each other.

Each input vector to the SOM is constituted by the concatenation of k vectors, related to the preprocessed time-series of the k metrics, for each considered VM and period. Notice that, since INFRA metrics have been provided with a 5-minutes collection granularity, if a period of a day is considered, we typically have 288 data points of each metric for each VM, in each day. In order to train the SOM, a few hyperparameters must be provided:

- **SOM dimensions**: the map is usually defined as a finite two-dimensional region where neurons are arranged in a rectangular grid. A higher amount of neurons typically leads to a lower quantization error, at the cost of map interpretability.
- **learning rate**: this parameter takes on values in a range between 0 and 1 (inclusive) and controls how much each training sample contributes to updating SOM weight vectors. The higher the learning rate, the more a neuron is influenced by the observed training samples and tends to cluster more diverse behaviors. It is possible to adaptively decrease this parameters as the number of epochs increases.
- **neighborhood radius** (σ): this parameter takes on values in a range between 0 and 1 (inclusive) and refers to the coefficient of the Gaussian neighborhood function. The higher the value of σ, the more neighbor neurons are affected by the weights update of an individual neuron in each training step. It is possible to increase this parameters as the number of epochs increases.
- **number of epochs**: an epoch consists of computing weights update in a *full-batch* fashion (i.e., the update is computed upon seeing the whole training data set). A training process usually consists in multiple training epochs.

After the training phase, the SOM can be used to infer the BMU for each input sample, i.e., the neuron that exhibits the least quantization error when compared with the considered input sample. At this stage, the output of the analysis can be used by, e.g., a data center operator to visually inspect the behaviors captured by the trained SOM neurons, in order to spot possible suspect/anomalous ones and check which VMs are associated with them. Furthermore, since the individual input samples are related to the behavior of a specific VM at specific point in time, it is also possible to analyze the evolution of the VMs throughout the time horizon, to possibly detect patterns in their behavioral changes. In this way, an operator is able to focus the analysis on a restrained set of VMs (an their hosts) and to possibly trigger further, more specific, analysis that could be too time-consuming, or even unfeasible, to conduct on the whole infrastructure.

Additionally, we provide a mechanism to automatically detect possible suspect behaviors without the need for a human operator to inspect the status of the SOM at the end of the training. It consists in a rather simple threshold-based alert that is triggered whenever, during the inference phase, an input sample is associated to a neuron with a quantization error that is greater than the specified threshold. Because of the considerable distance from the BMU (i.e., the *closest* neuron), such samples are likely to depict an uncommon behavior and, thus, are marked as *misclassified*. Besides the aforementioned support that such a tool can give to data center operators in their manual activity, this feature in particular enables the possibility to deploy a fully automated anomaly detection system. Indeed, assuming for instance that a SOM is trained on carefully selected input samples that only depict the behaviors that considered as expected from the NFV infrastructure, the resulting model can be used just for inference on an unforeseen data set, whose misclassified samples can be regarded as suspect/anomalous patterns and should be further inspected. Another interesting use of the misclassification mechanism is its capability to notify immediately operators of possible misconfigurations where a too little SOM grid size has been chosen for the data set under analysis, leading to an excessive number of misclassified time series.

3.2 Grouping SOM neurons

An interesting aspect that came out during the use of the above mentioned SOM-based classification, is that, whenever using relatively big SOM networks, the training phase ended up with many closeby SOM neurons catching behaviors that were very similar to each other.

This is in line with the topology-preservation property of the SOMs, i.e., closeby input vectors in the input space are mapped to closeby neurons in the SOM grid. This phenomenon can be controlled to some extent using various neighborhood radiuses. However, from the viewpoint of data center operators, a set of closeby neurons with relatively similar weight vectors needs to be considered as a single behavioral cluster/group.

This has been achieved adding, after the SOM processing stage, a simple clustering strategy based on aggregating (transitively) neurons having weight-vectors at a distance lower than a given threshold, into the same group. Therefore, our technique offers the possibility to collapse, according to the distances among the representative vectors of the SOM neurons, similar clusters in order to decrease the possibility to raise an alarm when it is not needed (e.g., consider very frequent movements, of a VM over time, between two similar neurons) and to facilitate the human operators in interpreting the results.

Indeed, as we will show in Section 4.4, this led to the overall technique outputting a reduced and more comprehensible number of behavioral clusters.

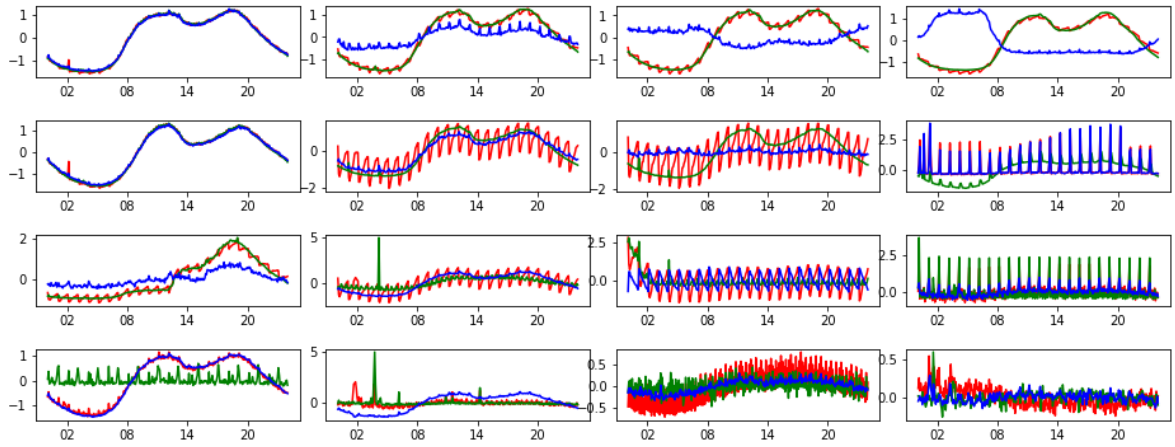


Figure 2: Example of INFRA resource consumption clusters identified with the multi-metric SOM analysis. The red, green and blue curves in each plot correspond to the `usage_average` metrics for the `cpu|usage_average`, `net|usage_average` and `cpu|capacity_contentionPct` metrics, respectively

.

3.3 SOM implementation

To implement our anomaly detection tool we leveraged a very efficient open-source SOM implementation, namely `somoclu`[2], which has been designed to employ multi-core acceleration, as well as GPGPU hardware acceleration, to perform massively parallel computations (Wittek et al., 2017). Such accelerations have been proved to be necessary in order to reach a satisfactory performance when tackling the massive data set provided by Vodafone.

4 EXPERIMENTAL RESULTS

In this section, we provide an overview of the results that can be obtained using the approach proposed in Section 3. For the analysis, we have relied on the experience of domain experts and focused our attention over a limited set of metrics that are considered the most relevant in this context, i.e., the ones related to the computational, networking and storage activity of VMs and VNFs of interest. Specifically, in the following, we highlight results obtained analyzing the following metrics: `cpu|capacity_contentionPct`, `cpu|usage_average`, `net|usage_average`.

4.1 Multi-metric analysis results

The set of clusters highlighted in Figure 2 is a clear example of the results that can be obtained through the multi-metric SOM-based analysis presented in Section 3, applied over few months worth of system-level (INFRA) metrics, using the normalized strategy. The trained SOM network is visually represented in terms of the weights of its neurons. Indeed, each subplot reports the VMs daily behavior that the specific neuron specialized into. In order to simplify the representation, the weight vectors – jointly computed over the three metrics `cpu|usage_average`, `net|usage_average` and `cpu|capacity_contentionPct` – are overlapped but in different colors.

For instance, one of the most recurrent patterns, occurring in 35.6% of the observations and depicted in Figure 3, is the one identified by the top-left neuron. Because of the standard data normalization – performed during the preprocessing phase to discard the magnitude information in favor of enhancing the behavioral information of the input samples – the values on the Y axis can be negative. This means that VMs have been clustered based on the joint shape of their daily resource consumption patterns, not their absolute values. Notice that in this example we can observe a quite suspect output, since the `cpu|capacity_contentionPct` figure follows closely the daily traffic pattern on the involved VMs. In a normal condition of a healthy system, i.e., when VMs are provided with appropriate computational resources, we would have expected this metric to stay close to zero, or at least experience a slight increase only during the peak hours.

A significantly different pattern is the one reported in Figure 4, corresponding to the top-right neuron in Figure 2. Such behavior represents the 7.84% of the observed daily patterns in the time period under analysis. As evident from the picture, there is a higher

[2]https://github.com/peterwittek/somoclu

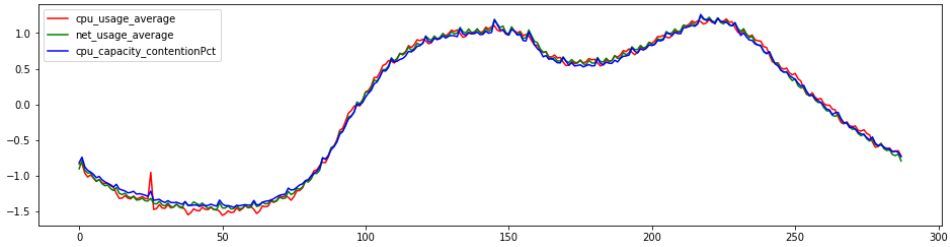


Figure 3: Most recurrent VM cluster identified by the multi-metric SOM-based analysis

.

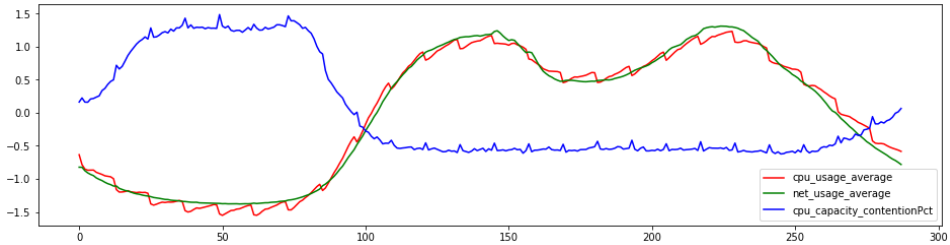


Figure 4: A singular VM pattern captured by the top-right neuron in the multi-metric SOM-based analysis

.

Table 1: The hyper-parameters values used for grid search.

Hyper-parameter	**Space**
SOM dimensions	8×8, 12×12, 16×16, 24×24, 32×32, 48×48
learning rate	$0.1, 0.2, \ldots, 0.9, 1.0$
neighborhood radius (σ)	$0.1, 0.2, \ldots, 0.9, 1.0$
epochs	$5, 10, 20$

CPU contention during night, when the VM has lower traffic, than during the day.

An additional remark regarding the possible presence of anomalies can be done considering the fact that the VMs included in the analysis are guaranteed to have the same role in the corresponding VNFs, i.e., they manage traffic in load sharing-mode. While it was expected to obtain an identical output for all them, instead, the SOM-based analysis has pointed out that a subset of such VMs exhibits daily patterns very different to the expected ones. This could be considered as a warning by human operators, that shall be monitor and further analyze the involved components of the infrastructure. In addition, it is worth noticing that asynchronous changes among the metrics included in such analysis could be indications of anomalous behavior of the NFV environment, and not necessarily of the VNF itself.

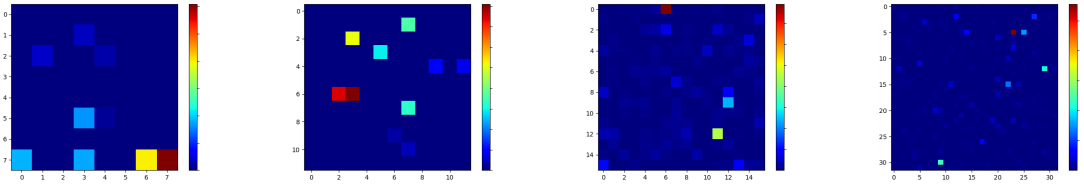
4.2 Grid search on hyper-parameters

As mentioned in Section 3.1, different hyper-parameters lead to very different clusters after training. An extensive grid search has been conducted over the search space summarized in Table 1. A total of 1600 different configurations has been tested monitoring quantization error and readability of results. Figure 5 shows the effect of using a low σ value (0.1) in different map sizes. Using a low σ with a low learning rate gives the worst results with very few BMUs that capture more than 95% of data, resulting in higher quantization errors.

SOM maps greater than 12×12 require very high σ (> 0.8) and very low learning rate (< 0.3) in order to have low quantization errors, but in these cases the results tend to become unreadable due to the fact that too many neurons specialize on similar patterns. In Figure 6, the SOM maps reported in Figures 6a and 6b are trained using high σ and low learning rate, while the ones reported in Figures 6c and 6d are trained using high σ and high learning rate. Therefore, for our analysis the best combination of hyper-parameters are high values of σ (¿0.6) and low values of learning rate (¡0.6) with results that are better both in terms of quantization error and readability.

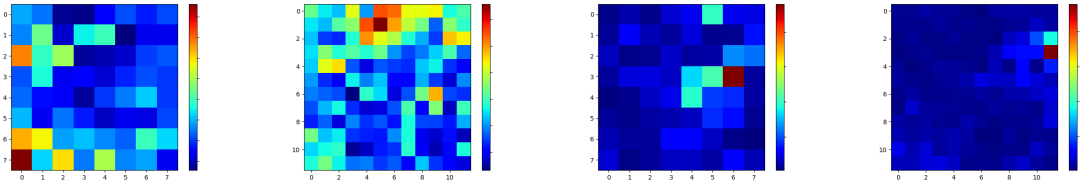
4.3 Per-VNF SOM-based analysis

Another interesting characterization we could perform applying the SOM-based analysis, is a study of



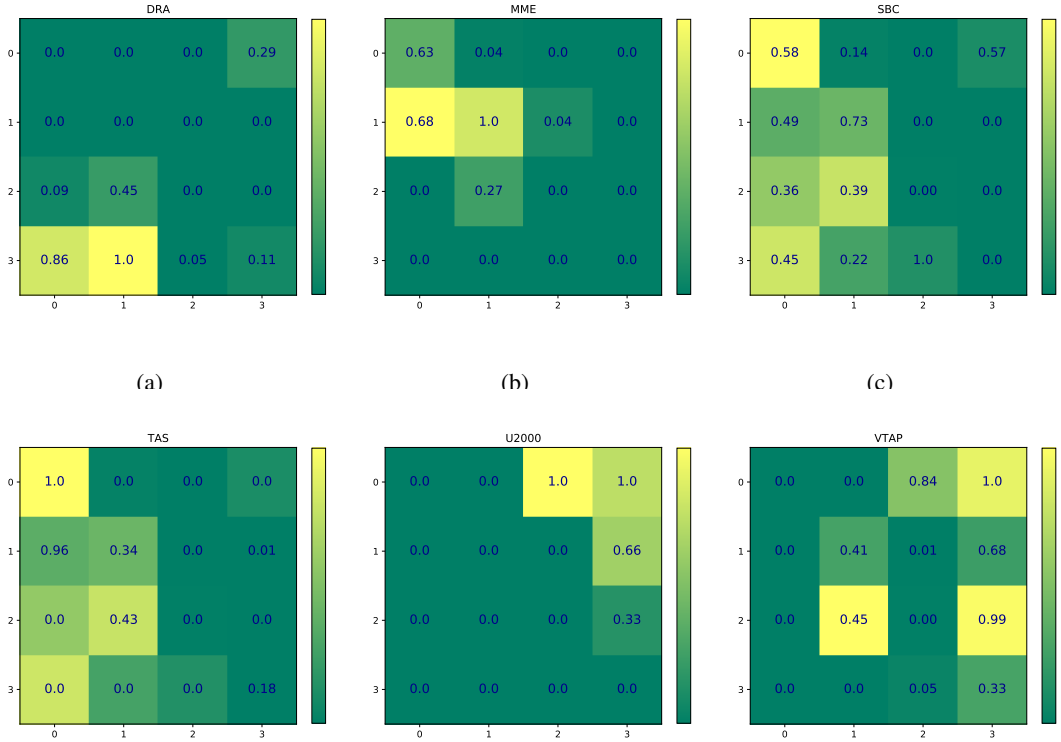
(a) (b) (c) (d)

Figure 5: SOMs with low σ values: (a) 8×8, σ: 0.1, lr: 0.2; (b) 12×12, σ: 0.1, lr: 0.2; (c) 16×16, σ: 0.1, lr: 0.9; (d) 32×32, σ: 0.1, lr: 0.8 For confidentiality reasons, the scale has been omitted.



(a) (b) (c) (d)

Figure 6: SOMs with high σ values: (a) 8×8, σ: 0.6, lr: 0.2; (b) 12×12, σ: 0.6, lr: 0.2; (c) 8×8, σ: 0.6, lr: 0.9; (d) 12×12, σ: 0.6, lr: 0.9 For confidentiality reasons, the scale has been omitted.



(a) (b) (c)

(d) (e) (f)

Figure 7: SOM clusters and corresponding per-VNF hitmaps identified by the multi-metric SOM-based analysis. For confidentiality reasons, the total number of hits in the hitmap cells has been rescaled to 1, to avoid disclosing the actual figures.

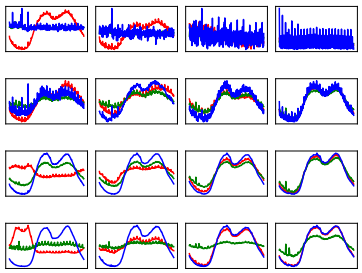
how different VNFs behave in terms of their daily resource consumption patterns.

In this case, we produced hitmaps highlighting how many daily patterns of VMs of each given VNF map onto each SOM neuron. The result can be visualized as in Figure 7, using the same data set used in Figure 2.

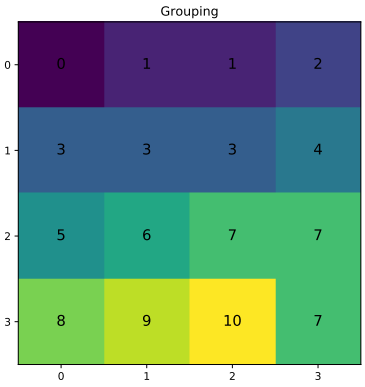
For example, the plot highlights that both the SBC and the TAS VNFs have mostly the usual "nightly/daily" pattern characterized by a low workload over nightly hours and a high workload over daily hours, with peaks around noon and 6pm. These are caught by the top-left neuron with coordinates $(0,0)$. On the other hand, the DRA VNF captured by neurons $(3,0)$ and $(3,1)$ exhibit the classical nightly/daily pattern for the `cpu|capacity_contentionPct` metric, and periodic peaks every 30 minutes for the other two metrics. Moreover, a consistent number of VTAP VMs are captured by neuron $(2,3)$ that is characterized by hourly periodic peaks.

4.4 Grouping of SOM neurons

In this section, we report the output of the grouping/clustering technique described in Section 3.2, starting from another month of data, with respect to the experiments shown above. We obtained the SOM neurons whose weights are highlighted in Figure 8a. After applying the distance-based grouping, with a group-distance threshold of 0.007, we obtained a reduced number of groups, as visualized in Figure 8b, where each cell corresponding to a neuron has been labelled with the associated group identifier. These reflect better the different behaviors we have in the resource consumption patterns.



(a)



(b)

Figure 8: Distance-based grouping of similar neurons using a 0.007 threshold.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we focused on the problem of analysis and classification of the behavioral patterns of VM metrics in a NFV data center. We described the technique we realized, based on self-organizing maps, that is being used across the data centers of the Vodafone network operator. We described some of the initial results we obtained from its application, highlighting the capability of our technique to identify interesting points in space and time (i.e., precise VMs and hosts within the infrastructure, and precise days within the analyzed time range) with potentially anomalous behaviors, thus deserving further attention and investigations by data center operators.

In our experimentation, we identified a number of open questions that still need additional investigations. First, the proposed technique has a number of hyper-parameters (SOM grid size and parameters, and various thresholds as described in Section 3) that have to be decided. A grid search can be used for such purpose, but it requires a non-negligible processing time as configurations to try can easily grow in the range of tens or hundreds. In order to avoid manual and tedious evaluations by operators, the various analysis runs should be compared with one another using an automated and quantitative assessment method. This cannot be simply done based on the SOM quantization error, as it would decrease increasing the SOM size, driving the choice towards excessively large networks. For example, we plan to use the average silhouette width to such purpose (Rousseeuw, 1987).

Another interesting path we plan to explore is the one to combine our approach with the use of Deep Learning (DL) for time series classification (Malhotra et al., 2017; Cui et al., 2016; Ismail Fawaz et al., 2019; Kashiparekh et al., 2019), in order to build more effective anomaly detection models.

REFERENCES

Bertero, C., Roy, M., Sauvanaud, C., and Tredan, G. (2017). Experience Report: Log Mining Using Natural Language Processing and Application to Anomaly Detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 351–360. IEEE.

Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176.

Canetta, L., Cheikhrouhou, N., and Glardon, R. (2005). Applying two-stage SOM-based clustering approaches to industrial data analysis. *Production Planning & Control*, 16(8):774–784.

Cui, Z., Chen, W., and Chen, Y. (2016). Multi-Scale Convolutional Neural Networks for Time Series Classification.

Díaz, I., Domínguez, M., Cuadrado, A. A., and Fuertes, J. J. (2008). A new approach to exploratory analysis of system dynamics using som. applications to industrial processes. *Expert Systems with Applications*, 34(4):2953 – 2965.

Farshchi, M., Schneider, J.-G., Weber, I., and Grundy, J. (2018). Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, 137:531–549.

Gulenko, A., Wallschlager, M., Schmidt, F., Kao, O., and Liu, F. (2016a). Evaluating machine learning algorithms for anomaly detection in clouds. In *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pages 2716–2721. IEEE.

Gulenko, A., Wallschläger, M., Schmidt, F., Kao, O., and Liu, F. (2016b). A system architecture for real-time anomaly detection in large-scale nfv systems. *Procedia Computer Science*, 94:491–496. The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.

Gupta, L., Samaka, M., Jain, R., Erbad, A., Bhamare, D., and Chan, H. A. (2017). Fault and performance management in multi-cloud based NFV using shallow and deep predictive structures. *Journal of Reliable Intelligent Environments*, 3(4):221–231.

Haykin, S. (2007). *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Inc., USA.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963.

Kashiparekh, K., Narwariya, J., Malhotra, P., Vig, L., and Shroff, G. (2019). ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time Series Classification.

Le, L., Sinh, D., Lin, B. P., and Tung, L. (2018). Applying big data, machine learning, and sdn/nfv to 5g traffic clustering, forecasting, and management. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 168–176.

Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification.

Malini, N. and Pushpa, M. (2017). Analysis on credit card fraud identification techniques based on knn and outlier detection. In *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, pages 255–258.

Miyazawa, M., Hayashi, M., and Stadler, R. (2015). vnmf: Distributed fault detection using clustering approach for network function virtualization. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 640–645.

NFV Industry Specif. Group (2012). Network Functions Virtualisation. Introductory White Paper.

Niwa, T., Miyazawa, M., Hayashi, M., and Stadler, R. (2015). Universal fault detection for nfv using som-based clustering. In *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 315–320.

Ostberg, P. O., Byrne, J., Casari, P., Eardley, P., Anta, A. F., Forsman, J., Kennedy, J., Le Duc, T., Marino, M. N., Loomba, R., Pena, M. A. L., Veiga, J. L., Lynn, T., Mancuso, V., Svorobej, S., Torneus, A., Wesner, S., Willis, P., and Domaschka, J. (2017). Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *EuCNC 2017 - European Conference on Networks and Communications*, pages 1–6. IEEE.

Pitakrat, T., Okanović, D., van Hoorn, A., and Grunske, L. (2018). Hora: Architecture-aware online failure prediction. *Journal of Systems and Software*, 137:669–685.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65.

Samrin, R. and Vasumathi, D. (2017). Review on anomaly based network intrusion detection system. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 141–147.

Sauvanaud, C., Kaâniche, M., Kanoun, K., Lazri, K., and Da Silva Silvestre, G. (2018). Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned. *Journal of Systems and Software*, 139:84–106.

Van den Berg, F. D., Kok, P., Yang, H., Aarnts, M., Meilland, P., Kebe, T., Stolzenberg, M., Krix, D., Zhu, W., Peyton, A., et al. (2018). Product uniformity control-a research collaboration of european steel industries to non-destructive evaluation of microstructure and mechanical properties. In *Electromagnetic Non-Destructive Evaluation (XXI). 6 September 2017 through 8 September 2017*, pages 120–129.

Wallschläger, M., Gulenko, A., Schmidt, F., Kao, O., and Liu, F. (2017). Automated Anomaly Detection in Virtualized Services Using Deep Packet Inspection. In

Procedia Computer Science, volume 110, pages 510–515. Elsevier.

Watanabe, Y., Otsuka, H., Sonoda, M., Kikuchi, S., and Matsumoto, Y. (2012). Online failure prediction in cloud datacenters by real-time message pattern learning. In *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, pages 504–511. IEEE.

Wittek, P., Gao, S. C., Lim, I. S., and Zhao, L. (2017). somoclu : An Efficient Parallel Library for Self-Organizing Maps. *Journal of Statistical Software*, 78(9).