Near Real-Time Anomaly Detection in NFV Infrastructures

Arman Derstepanians*, Marco Vannucci*, Tommaso Cucinotta*
* Scuola Superiore Sant'Anna, Pisa, Italy
* firstname.lastname@santannapisa.it

Avhad Kiran Sahebrao†, Sourav Lahiri†,
Antonino Artale§, Silvia Fichera§
†Vodafone Intelligent Solutions, Pune, India
§ Vodafone, Milan, Italy
† § firstname.lastname@vodafone.com

***Abstract*—This paper presents a scalable cloud-based architecture for near real-time anomaly detection in the Vodafone NFV infrastructure, spanning across multiple data centers in 11 European countries. Our solution aims at processing in real-time system-level data coming from the monitoring subsystem of the infrastructure, raising alerts to operators as soon as the incoming data presents anomalous patterns. A number of different anomaly detection techniques have been implemented for the proposed architecture, and results from their comparative evaluation are reported, based on real monitoring data coming from one of the monitored data centers, where a number of interesting anomalies have been manually identified. Part of this labelled data-set is also released under an open data license, for possible reuse by other researchers.**

***Index Terms*—Anomaly Detection, Network Function Virtualization, Machine Learning**

I. Introduction

Network operators are undergoing a steep change in their operational practices, in order to deal with the high-bandwidth, low-latency and high-reliability connectivity requirements in modern and future mobility scenarios [1]. The increasing dynamicity level in networking traffic conditions requires an infrastructure more flexible than it used to be in the past. This flexibility is being handled, among others, by recurring to network function virtualization (NFV) [2], [3] and software-defined networking (SDN) [4], [5]. These constitute fundamentally a paradigm shift by which network operators walk away from the traditional static allocation of physical dedicated network appliances sized for peak-hour operations, to switch to a much more flexible configuration where these telecom network functions become software components, a.k.a., virtualized network functions (VNFs). These are deployed and managed as virtual machines (VMs) or containers within a set of general-purpose servers, constituting a highly flexible network virtual infrastructure (NVI). The NVI reuses principles and lessons learned from the area of cloud computing [6], [7] to allow for automated and rapid provisioning of VNFs that are capable of dynamically tracking in real-time workload fluctuations and to adapt to the always-changing conditions of the networking traffic to handle [8].

In the context of NFV and cloud computing, the monitoring subsystem of the infrastructure is a critical component, leveraged at various levels of daily operational activities. Specifically, monitoring of networking [9] appliances as well as computational elements (both physical hosts and virtualized instances), becomes critical [10] to support daily activities such as realizing appropriate elasticity [11] and placement [12], [13] policies or anomaly detection techniques [14], [15], where metric forecasting can also play a key role [16], [17].

The monitoring subsystem of these NFV infrastructures ends up collecting a very large amount of data, generated every day in the form of many time-series per monitored instance. The monitored metrics may range from system-level metrics (e.g., CPU, network, disk utilization, power consumption), to application/service-level metrics (e.g., volumes of correctly served requests, experienced error conditions, etc.), reaching easily to up to dozens and dozens of metrics per monitored instance. Therefore, network operators are looking into appropriate techniques leveraging on scalable big-data analytics that are applicable for this purpose. From a network operator perspective, deploying big-data analytics on their virtualized infrastructures is becoming a critical task. Indeed, a comprehensive analysis of the data from the various components of the NVI helps the operator in different activities, e.g., workload forecast for infrastructure sizing, capacity planning, anomaly detection and alert management, fault management, troubleshooting and root-cause analysis (RCA), etc.

This paper presents the software architecture under development at Vodafone for near real-time (NRT) anomaly detection, based on monitoring data coming from the NVI of the operator, spanning across 12 different EU countries (OpCos) where Vodafone has NFV data centers. Our solution is based on Function-as-a-Service (FaaS) facilities of a public cloud provider, for scalable processing of monitoring data collected from the various NFV data centers. These include thousands of hosts in which tens of thousand VMs are deployed. For each VM and physical host dozens of metrics are potentially available for analysis, with the time granularity of one sample every 5 minutes, resulting into having GBs of data to analyze every month. This makes the design of the anomaly detection system particularly challenging due to the need for balancing precision for computational overheads. A number of different anomaly detection techniques have been designed for the architecture and evaluated using real data from Vodafone NFV data centers located in Italy and Czech Republic, where a number of interesting anomalies have been manually identified. The presented experimental results highlight benefits and shortcomings of these techniques. Part of the dataset used for the results shown in this paper is released with an open data license (see Section IV).

The paper is structured as follows: Section II presents briefly related research in the area, while Section III presents the reference architecture we have developed and deployed for NRT anomaly detection, along with an overview of algorithms that looked interesting for our use-case; Section IV presents experimental results from the application of these algorithms to a sample dataset from the Vodafone NVI, to identify anomalous behaviors; finally, Section V provides a few concluding remarks.

II. Related Work

Anomaly detection has been widely studied in the context of NFV and cloud operations, often with the help of fault injection techniques. For example, in [18], researchers have built a data set injecting faults in a Kubernetes cluster, and evaluating different techniques for anomaly detection based on supervised machine learning (ML), including support vector machines (SVMs), nearest neighbor, naive Bayes and random forests. SVMs have also been used in [19] for on-line detection of anomalies in data from transmissions in Wireless Sensor Networks. In order to deal with transients of the time-series, here an SVM with a Gaussian kernel has been applied to data fitted with a least-squares regressor over a sliding window on the raw data to process.

An evaluation of several supervised ML techniques for off-line anomaly detection in NFV can be found in [20]. Authors compared 13 different techniques, including various types of decision trees, random forests, Bayesian networks and SVMs, on host monitoring data obtained by injecting anomalies in a test set-up running components from the ClearWater IMS system within KVM VMs deployed in an OpenStack environment. Another interesting survey can be found in [14], where authors discuss the risk of facing anomalies when switching to a NFV/cloud model, mostly due to virtualization and resource over-commitment issues causing temporal interferences among co-hosted VNFs.

Unsupervised techniques for anomaly detection have also been proposed in literature. For example, a technique based on Hierarchical Temporal Memory (HTM) has been proposed in [21] for analyzing streaming data in real-time. However, the technique was evaluated on a benchmark using a single-variate data. Self-Organizing Maps (SOMs) have been proposed specifically for anomaly detection in NFV data centers [22], with a multi-variate analysis method that identifies clusters of similar daily patterns in multiple metrics of VMs of one or more VNFs, so that changes in the classified behavior is marked as a possible anomaly. The technique was coupled with a heuristic for removing false positives, as often occurring over transitions between working and weekend days. However, the proposal focused on off-line analysis of daily behavioral patterns as observed in a recent time horizon.

In [23], a digital twin approach is used for RCA of anomalies in NFV infrastructures, formulating the problem as a dynamic set-covering problem, using also hidden Markov models and transfer learning.

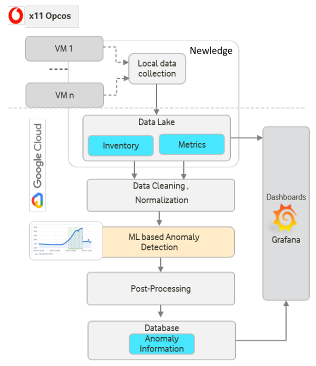


Fig. 1: Anomaly Detection System Architecture

Compared to the techniques recalled above, this paper aims at constituting an industrial experience report, providing information on how a variety of ML/AI algorithms for anomaly detection have been evaluated and compared, many of which having also been used in the above referenced papers. This paper explains how these techniques have been integrated with a paradigm for scalable big-data processing in cloud environments, combining them into the architecture that is currently under development at Vodafone to help in operating its NFV infrastructure. We hope this paper may help other industrial practicioners with the task of designing processing pipelines for anomaly detection in big infrastructures.

III. Proposed Architecture and Algorithms

Anomaly detection techniques are used in NFV and cloud management frameworks to detect problems occurring within the infrastructure, analyzing the plethora of data available within its monitoring subsystem. Real-time, or NRT, anomaly detection aims at performing this task in a timely manner, just as soon as new data is collected at run-time. This allows for attracting operators' attention to promptly resolve possible issues causing the spotted anomalies within the monitored time series, minimizing or even preventing possible impact on the hosted services. In our case, we are building a system to analyze metrics in NRT from all the NFV data centers spread across 12 EU countries. Therefore, we need a scalable architecture design, as detailed below.

A. *Proposed System Architecture*

Fig. 1 shows our design for a NRT anomaly detection architecture. The goal of the proposed system is to detect anomalous points in the temporal evolution of metrics related to VMs/VNFs, focusing on their resource consumption metrics, i.e., related to the utilization of the underlying infrastructure (INFRA metrics), as well as the application-level metrics (VNF metrics). More information about these metrics and a visualization of their typical patterns can be found in a prior work of ours [22]. Data related to VMs are collected from proprietary management platforms by the local data collection component and stored in a *Data Lake* within a Google Cloud Platform (GCP) environment[1], where we use the Cloud Big Table service[2] as a reliable NoSQL storage for the gathered time-series. We also use a separate SQL database to store meta-data about all active VMs in the Vodafone NVI, including their unique identifiers and timestamps of creation, termination or other relevant events. Raw metrics data is extracted, in the form of time-series with a data point every 5 minutes, for each VM in the Data Lake, for a given period and merged into one single vector. Then, data is cleaned so to be ready to be processed with different algorithms for anomaly detection. The data cleaning phase performs an initial check on the data, and interpolates possible missing values if detected, in addition to marking the corresponding timestamps as possible anomalies. Interpolation is necessary due to some anomaly detection techniques that do not tolerate the presence of missing values in the input data. Also, min-max scaling is applied on each time-series individually for faster convergence.

The system has been designed with modularity in mind, with the ability to configure a number of different ML/AI techniques for anomaly detection, which comply to a simple interface, taking as input a vector of n values for each VM and metric separately. These are deployed as Google Cloud Functions[3], a scalable Function-as-a-Service (FaaS) framework available in GCP to design scalable serverless data processing pipelines. The execution of the ML-based anomaly detection Functions is triggered periodically exploiting the Google Tasks service[4]. The final output of the ML-based anomaly detection pipeline is an anomaly score for each new data point that was injected into the data processing pipeline since the last activation.

In the post-processing phase, isolated single anomalous points of the monitored metrics followed by non-anomalous points are actually ignored by the system and discarded, as the system likely already recovered from the issue. On the other hand, sequences of 3 or more timestamps marked as anomaly are brought forward in the processing pipeline, towards a persistent storage into a database. From there, operators may access the identified anomalies at any time and visualize them using an instance of the well-known Grafana framework[5].

The proposed system is capable of detecting anomalies in multiple input metrics. In our experimentation, we have applied this methodology over individual VMs metric data available with a 5-minutes granularity (288 observations per day, per metric, per monitored VM).

B. ML-based Anomaly Detection Algorithms

In this section, we describe the algorithms we tested to perform anomaly detection in the described context. Specifically, we tested the following well-known anomaly detection ML algorithms: an ensemble-based model i.e., Isolation Forest; a linear classification model, i.e., One-Class SVM; a model based on neighbors, namely Local Outlier Factor (LOF). A brief discussion of the main features of these techniques follows below for the sake of completeness, reminding the reader to ML/AI textbooks for a more detailed and comprehensive description of them.

a) Isolation Forest: (IF) [24] is an unsupervised approach to anomaly detection that can be applied to both univariate and multi-variate domains. IF is based on the intuition that, when partitioning a dataset using decision trees, isolated points correspond to leaves located in the upper part of the tree. Starting from this idea, IF builds a forest of random decision trees, where each tree is computed by: partitioning the training dataset by randomly selecting a feature and then randomly selecting the split value for the feature space within its minimum and maximum values. This operation is then iterated on both sides of the split, which are attached to the current node as children, until either a single point is isolated in a partition that becomes a leaf of the tree, or the depth of the current node/partition grows beyond a sufficiently large threshold. Now, an 'anomaly score' is assigned to each of the data points according to the depth of the tree required to arrive at that point. This score is an aggregation of the depth calculated from each tree within the forest. The points with an anomaly score lower than a pre-defined threshold (called contamination in the IF slang) are considered anomalous.

b) Local Outlier Factor: (LOF) [25] belongs to the family of density based methods for anomaly detection and is commonly used for outliers detection purpose. LOF compares the spatial density of single data points within a dataset with the density around its local neighbors [25], [26] according to an arbitrary distance metric. The LOF technique defines a local outlier score considering the relative density of the data point. By comparing the local density of a point to the local densities of its neighbors, one can identify regions of similar density, and points that have a substantially lower density than their neighbors. A normal data point has a LOF between 1 and 1.5 whereas anomalous observations will have a much higher LOF. The higher the LOF, the more likely it is an outlier.

c) One-Class SVM: is a variant of classic SVM devoted to the detection of rare patterns (among which outliers and anomalies). Differently from standard SVM that use hyper-planes to partition the feature space, One-Class SVM uses hyper-spheres to incorporate all the instances from the feature space [27], [28]. The resulting hyper-sphere is characterized by its center and radius. Those points that lie inside or *close* to the hyper-sphere are classified as normal data points, whilst others are classified as anomalous.

C. Predictive Anomaly Detectors

In this subsection we introduce two threshold-based anomaly detectors based on comparing the real values with the predictions performed with either Long-Short Term Memory (LSTM) autoencoders or a predictor based on the statistical median, where an anomaly is declared once the actual value deviates from its prediction for more than a threshold.

[1]More information is available at: https://cloud.google.com/.
[2]More information is available at: https://cloud.google.com/bigtable.
[3]More information is available at: https://cloud.google.com/functions.
[4]More information at: https://cloud.google.com/tasks/docs/tutorial-gcf.
[5]More information is available at: https://grafana.com/.

a) LSTM autoencoder: is one of the well-known Neural Network (NN) based approaches for anomaly detection. We consider a monthly data of the target metric during training, normalizing the data before the analysis using a *Robust Scaler*.

As a baseline for our approach, we have used two different LSTM autoencoders; one with 36 and the other with 6 LSTM neurons for both the encoding and the decoding layers, in both cases using 24H of samples as the NN input, corresponding to 288 samples. A per-VM cluster model is trained on a month of data on a random VM among those belonging to the same behavioral cluster. These clusters are identified by a SOM-based technique described in a prior work of ours [22]. Clustering is applied to avoid having to train an LSTM model for each VM, considering that in our experimentation each model training process takes about 3 hours of processing time on a single processor (multi-core and/or GPU acceleration might be conveniently used to speed-up the process). Every 30 minutes, the detector is fed with additional 6 samples, corresponding to 30 minutes, which are compared to the corresponding predictions made by the LSTM using as input the 288 samples just preceding the first sample under analysis.

Anomalies are defined as the points for which the *Mean Absolute Error* (MAE) of the predicted values and the actual ones are above a certain threshold, which has been chosen as any of the following values, in our evaluations below:

$$T_1 = 0.20\,, \quad T_2 = 0.24\,, \quad T_3 = 0.28$$

These values can be obtained by studying the statistical features of the prediction errors after normalizing the data with *Robust Scaler*. Namely, they are obtained by calculating the median of the prediction errors for different VMs plus their median absolute deviation which is multiplied by a coefficient.

b) Simple Median predictor: is a model used to forecast a metric behaviour for a given timestamp of a day, based on an "averaged" behavior of the same metric from the same timestamp on the previous few days. To obtain this average behaviour we perform the following steps:

- we classify the data into the three categories of Saturdays, Sundays and weekdays since the daily behavior of the metric is strictly dependent on whether it belongs to a weekday or weekend;
- the final goal is to predict anomalies in 6 data points, as the anomaly detector is activated every 30 minutes;
- the "target day" is defined as the previous 24H of the most recent timestamp;
- the "averaged" behavior of the data is then defined as the median among the 5 values obtained for the same timestamp, in the 5 days prior to the "target day".

Here it should be mentioned that, for the weekends, since the previous 5 days will go too much back to the past, we just consider the previous three days, e.g., for the upcoming Saturday, we pick the previous three Saturdays. The threshold in this approach, in contrast to the LSTM approach, is not set to a constant number. Instead, it is defined as follows:

$$Med(\mathbb{E}) + m \times MAD(\mathbb{E}) \tag{1}$$

where Med is the median of the prediction error $\mathbb{E}$ obtained by calculating the MAE of the *target* and the *averaged behavior*, and the MAD is the median absolute deviation. The coefficient m is a free parameter we can adjust depending on the general behavior of the dataset under consideration.

The next step, is to set both upper and lower limits for the usage of the above formula. Indeed, since Eq.(1) strictly depends on the distribution of the errors, there will be two different scenarios where it might fail to give us correct predictions. First, when the numerical values of *target* and *averaged behavior* are very close to each other: then, even a small deviation will be spotted as an anomaly. Thus, for such kind of cases the above formula will produce a lot of *false positives*. Second, when the ratio between numerical values of *target* and *averaged behavior* is folded many times, e.g., doubled, tripled: then, since Eq.(1) calculates the median of differences between the inputs and outputs, the threshold would not be successfully detecting the anomalies. Thus, in these situations we would have *false negatives* (missed anomalies). In other words, we would have a very large threshold which would include all points within.

In order to overcome the above mentioned problems, we propose upper and lower limits as follows:

- By setting $m = 4$ in the Eq.(1) we calculate the threshold.
  - If the obtained value is less than 0.24 then we set the threshold as
$$T = 0.24 \tag{2}$$
- Otherwise we calculate the $Med(\mathbb{E})$.
  - If, $0.36 < Med(\mathbb{E}) < 0.96$, then we use Eq.(1) with $m = 1$,
$$T_1 = Med(\mathbb{E}) + MAD(\mathbb{E}), \tag{3}$$
  - and the threshold is set as
$$T = min(T_1, 0.72). \tag{4}$$
  - Otherwise, we compute
$$T_2 = Med(\mathbb{E}) + 2 \times MAD(\mathbb{E}),$$
  - and the threshold is set as
$$\begin{aligned} T_{max} &= max(0.24, T_2) \\ T &= min(T_{max}, 0.72). \end{aligned} \tag{5}$$

The numerical value 0.24 in Eq.(2) is obtained by calculating Eq.(1) with $m = 2$ for the distribution of the error of different samples of VMs with normalized data (using a *Robust Scaler*). Accordingly, the rest of those numerical values in the above threshold are obtained as multiples of 0.24.

IV. Experimental results

A. *ML-based anomaly detection:*

All the ML-based anomaly detection models mentioned above are tested on a portion of the dataset gathered from 25 randomly selected virtual machines (VMs) for the metric *cpu|demandPct*. The models are tested for the following day's detections after being trained on the 30 days of prior data. The anomaly detector, which was employed in our current preliminary evaluation, activates every 30 minutes for each VM/metric, trained on the previous 30 days of data, and used to look for potential anomalies in the most recent six newly collected data points. The performance of each model is demonstrated with an example shown in Fig. 2. The plots in the figure show the typical behaviour from earlier days for the metric *cpu|demandPct* of one VM and the abnormal behaviour occurred on February 2nd, 2022, where all the detected anomalous points are depicted in red, for each of 3 considered predictors, as detailed below.

a) Isolation Forest: We used the open-source implementation of IF as available in the well-known scikit-learn software[6], tuned with 100 estimators, 1.0 max feature and contamination of 0.015 as percentage of the tree depth from the root, up to which the looked-up points are to be considered anomalous. With these settings, it nearly takes 2ms to train and predict anomalies for each VM/metric combination, analyzing a month of single-metric data. IF works best in case of outliers neatly falling outside of the main sample distribution observed in the training data set.

Fig. 2(a) shows that IF is able to identify most of the anomalous cases(depicted in red), generating fewer false positives.

b) Local Outlier Factor: We made our evaluation using the open-source implementation of LOF as available within the scikit-learn software, considering a number of neighbors equal to 100 for density estimation, the “Minkowski” setting for distance computations among neighbors, and a contamination factor of 0.015 as the percentage of outliers in the data set. LOF trains faster and produces average results with more false positives than IF.

Fig. 2(b) shows that LOF is not able to detect anomalous cases at the beginning of the dataset.

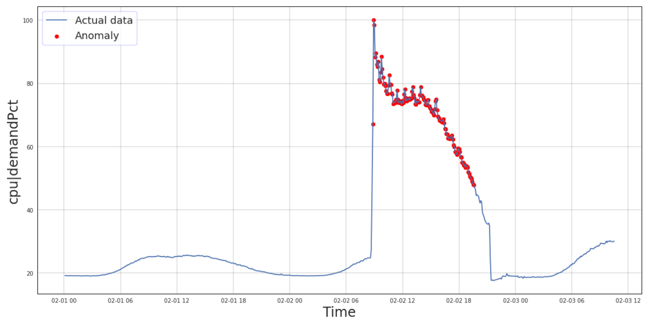
c) One-Class SVM: We trained a One-Class SVM model with an *rbf* kernel and 0.0005 $gamma$ value. One-Class SVM gives significant results with correct detection of downside anomalies as well as sudden peaks. Fig. 2(c) shows that One-Class SVM is able to detect maximum anomalous points, but it is giving more false positive cases as well (rightmost points).

Comparing the results obtained from the three anomaly detection algorithms just mentioned, IF performs better than other techniques with greater number of true positive detections. LOF is able to detect fewer actual anomalies in our data set, with many false positives. One-Class SVM performs better for actual anomalous points detection, but it has detected normal points as anomalous in the lower band of the dataset. To sum up the comparison, we have also obtained the confusion matrix for all detectors. Accordingly, the Accuracy, Precision, Recall and F1 score of this dataset are reported in Table I.

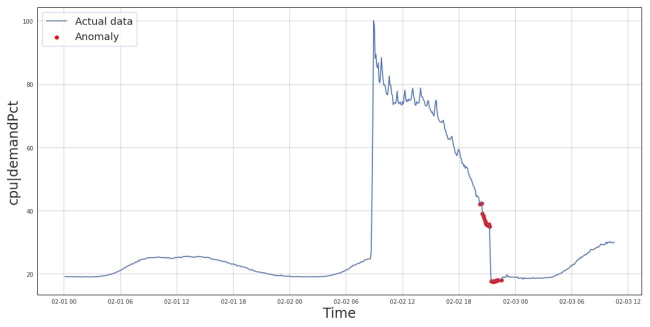
B. *Prediction-based anomaly detection*

In what follows, we show results obtained with the other two techniques based on a predictor followed by a threshold-based

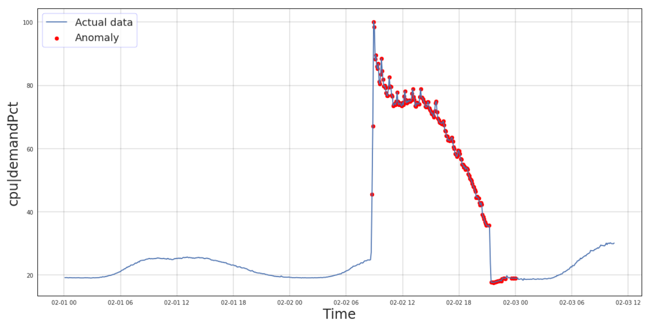
[6]More information is available at: https://scikit-learn.org/.



(a) Isolation Forest



(b) Local Outlier Factor



(c) One-Class SVM

Fig. 2: Anomalies detected by different ML algorithms for cpu|demandPct for 2nd of February 2022.

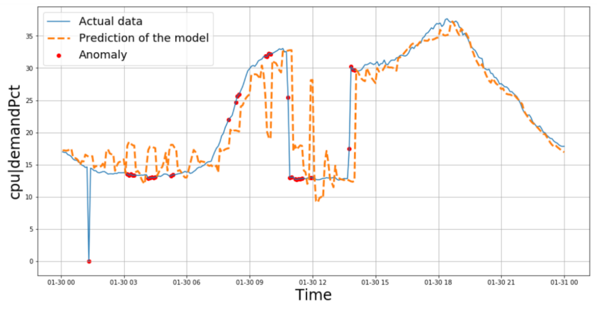
TABLE I: Scores of the ML based A/D models

Predictor	Accuracy	Precision	Recall	F1 score
IForest	0.75	0.71	0.76	0.74
LOF	0.51	0.55	0.51	0.53
OC-SVM	0.67	0.56	0.75	0.64

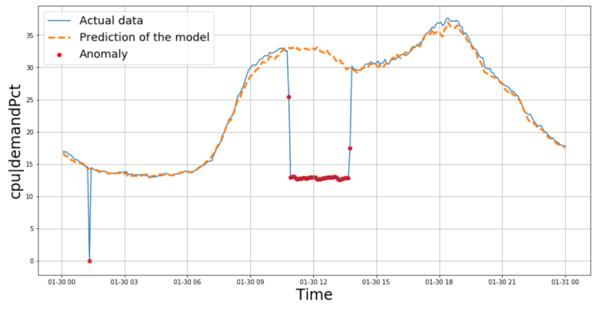
identification of anomalies, namely the LSTM-based and the SM based anomaly detectors.

A new test dataset is chosen which is a subsample of 25 VMs containing numerous normal days as well as anomalous data points, including cases with an anomalous day preceeded by either regular days only, or by one or more anomalous days, and sudden upward and downward hiccups. See [22] for more information on these behavioral patterns. This dataset is released with an open data license, and it is accessible at: http://retis.sssup.it/~tommaso/papers/nfvsdn22.php.

Comparing the anomaly detectors based on LSTM autoen-



(a) LSTM with 36 neurons

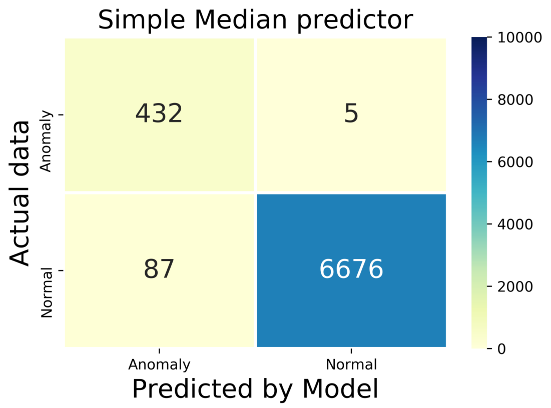


(b) SM

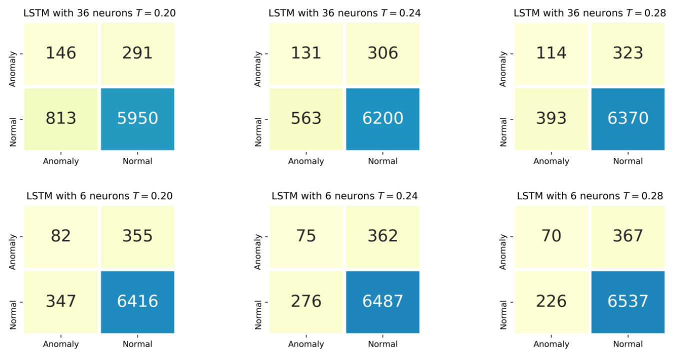
Fig. 3: Predicted anomalies of cpu|demandPct for MISBC41-SE2900-HRU1-10 on January 30th, 2020, according to LSTM autoencoder with 36 neurons ($T = 0.24$) and SM.

coders with the one based on the "Simple Median" (SM) approach, it turns out that the latter has by far a better power of identifying anomalies. In particular, when LSTM autoencoders are used, a lot of false positive and false negatives are obtained. As an example, let us look at the data of "MISBC41-SE2900-HRU1-10" on January 30th 2020. The predicted anomalies according to the LSTM autoencoder with 36 neurons (the threshold is set to 0.24) are shown in Fig. 3(a) which indicates that the predictor is not able to fully predict the anomalous interval starting from 10:49 till 13:44. In addition, between the 3:00 to 10:00 some points have been spotted as anomalies. However, in contrast to LSTM autoencoders, the SM predictor (see Fig. 3(b)) can correctly detect both the isolated anomaly at 1:19 (which is anyway discarded during post-processing, as explained above) as well as the whole anomalous interval without spotting false positives.

To complete the comparison, we also show the confusion matrix obtained for all predictors, where we show the performance obtained by LSTM with 3 different threshold values. The results have been illustrated in Figures 4(a) and 4(b). Accordingly, the *Accuracy*, *Precision*, *Recall* and *F1 score* have been shown in Table II. The results indicate that for LSTM autoencoders, as the baseline of our models, increasing the complexity of the architecture (increasing the number of LSTM units from 6 to 36) can increase the power of predictor to detect anomalies. Also, the increase of threshold from $T = 0.20$ to $T = 0.28$ improves the $F1$ scores. Nevertheless, their obtained $F1$ scores is several times lower than SM.



(a) The SM predictor



(b) LSTM autoencdorers

Fig. 4: The confusion matrix for different predictors i.e. the SM as well as the LSTM autoencoders with corresponding number of LSTM units and different thresholds.

TABLE II: Scores of each Model

Predictor	Accuracy	Precision	Recall	F1 score
LSTM(6) T=0.20	0.90	0.19	0.18	0.18
LSTM(6) T=0.24	0.91	0.21	0.17	0.19
LSTM(6) T=0.28	0.91	0.23	0.16	0.19
LSTM(36) T=0.20	0.84	0.15	0.33	0.20
LSTM(36) T=0.24	0.87	0.18	0.29	0.23
LSTM(36) T=0.28	0.90	0.22	0.26	0.24
SM	0.98	0.83	0.98	0.90

At the end it is essential to outline the main difference among the ML-based algorithms and the predictive models. Indeed, as mentioned above, based on their mechanism for detecting anomalies, the ML-based algorithms perform well for those cases where the values of anomalous points are considerably larger (or smaller) than the rest of the points e.g, Fig. 2(a). However, for those cases such as the anomalous interval in Fig. 3(b), where the values of anomalous points are in the range of local minimum and maximum of normal daily behavior, usually those anomalies cannot be found.

V. Conclusions

In this paper, we tackled the problem of Near Real-Time Anomaly Detection in NFV infrastructures, proposing a modular and scalable architecture as it is being engineered for the Vodafone European NFV Infrastructure, a.k.a., NVI. First, we explained the general architecture designed to handle the whole process in a scalable way. Then, we introduced different approaches, based on ML algorithms, Neural Network models and a SM predictor, showing the advantages and drawbacks for each of them.

The ML-based detection models are properly identifying anomalies with less computational effort. Despite the nature of the data, the detection results show that Isolation Forest and One-Class SVM are doing effectively. The isolation forest's nature is to separate points from dense points, therefore depending on the contamination parameter, it may occasionally identify more erroneous points. The isolation forest and LOF models do not perform well as values fall, in contrast to One-Class SVM.

For approaches based on a threshold-based comparison of the time-series samples with the values output by a predictor, tuning the threshold becomes fundamental for spotting anomalies correctly. We experimented with a LSTM-based predictor, and with a Simple Median (SM) one. A key advantage of the SM predictor, is that it is only based on the mathematical and statistical relations among the values of the dataset, so it does not need heavyweight training/fitting, like LSTM does.

The results show that, due to the specific nature of our data, i.e., daily periodicity, the SM predictor can perform well despite its simplicity, compared to standard approaches using ML/AI algorithms, and, in some cases, even outperform them. More than everything, the SM approach presents a much reduced computational burden, which is of paramount importance, given the amount of data to process in the industrial use-case under consideration.

References

[1] M. M. Erbati and G. Schiele, "Application- and reliability-aware service function chaining to support low-latency applications in an NFV-enabled network," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2021, pp. 120–123.

[2] ETSI, "Network Functions Virtualisation," SDN and Openflow World Congress, Darmstadt, Germany, White Paper 1, 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] ——, "Network Functions Virtualisation," SDN and Openflow World Congress, Dusseldorf, Germany, White Paper 3, 2014. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper3.pdf

[4] Open Network Foundation (ONF), "ONF SDN Evolution," ONF, White Paper, 2016. [Online]. Available: http://www.opennetworking.org/wp-content/uploads/2013/05/TR-535_ONF_SDN_Evolution.pdf

[5] H. Woesner and D. Verbeiren, "SDN and NFV in telecommunication network migration," in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, Sep. 2015.

[6] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," 2011-09-28 2011.

[7] R. Buyya, C. Vecchiola, and S. T. Selvi, "Chapter 1 - introduction," in *Mastering Cloud Computing*, R. Buyya, C. Vecchiola, and S. T. Selvi, Eds. Boston: Morgan Kaufmann, 2013, pp. 3–27.

[8] M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, B. Martini, F. Paganelli, W. Cerroni, P. Cappanera, and P. Castoldi, "Demonstration of Latency-Aware and Self-Adaptive Service Chaining in 5G/SDN/NFV infrastructures," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2018, pp. 1–2.

[9] G. Liu and T. Wood, "Cloud-Scale Application Performance Monitoring with SDN and NFV," in *Proceedings of the 2015 IEEE International Conference on Cloud Engineering*, ser. IC2E '15. USA: IEEE Computer Society, 2015, p. 440–445.

[10] J. Son, T. He, and R. Buyya, "Cloudsimsdn-nfv: Modeling and simulation of network function virtualization and service function chaining in edge computing environments," *Software: Practice and Experience*, vol. 49, no. 12, pp. 1748–1764, 2019.

[11] G. Lanciano, F. Galli, T. Cucinotta, D. Bacciu, and A. Passarella, "Predictive auto-scaling with OpenStack monasca," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing*. ACM, Dec. 2021.

[12] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, and A. Erbad, "Multi-cloud Distribution of Virtual Functions and Dynamic Service Deployment: Open ADN Perspective," in *2015 IEEE International Conference on Cloud Engineering*, March 2015, pp. 299–304.

[13] T. Cucinotta, L. Pannocchi, F. Galli, S. Fichera, S. Lahiri, and A. Artale, "Optimum VM Placement for NFV Infrastructures," in *Proceedings of the 10th IEEE International Conference on Cloud Engineering (IC2E)*, Pacific Grove, California, USA, September 2022.

[14] M. Zoure, T. Ahmed, and L. Réveillére, "Network Services Anomalies in NFV: Survey, Taxonomy, and Verification Methods," *IEEE Trans. on Network and Service Management*, pp. 1–1, 2022.

[15] T. Cucinotta., G. Lanciano., A. Ritacco., M. Vannucci., A. Artale., J. Barata., E. Sposato., and L. Basili., "Behavioral analysis for virtualized network functions: A som-based approach," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science - CLOSER,*, INSTICC. SciTePress, 2020, pp. 150–160.

[16] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[17] T. Cucinotta, G. Lanciano, A. Ritacco, F. Brau, F. Galli, V. Iannino, M. Vannucci, A. Artale, J. Barata, and E. Sposato, "Forecasting Operation Metrics for Virtualized Network Functions," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2021, pp. 596–605.

[18] Q. Du, Y. He, T. Xie, K. Yin, and J. Qiu, "An approach of collecting performance anomaly dataset for nfv infrastructure," in *Algorithms and Architectures for Parallel Processing*, J. Vaidya and J. Li, Eds. Cham: Springer International Publishing, 2018, pp. 59–71.

[19] H. Martins, L. Palma, A. Cardoso, and P. Gil, "A support vector machine based technique for online detection of outliers in transient time series," in *10th Asian Control Conference (ASCC)*, May 2015, pp. 1–6.

[20] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, and F. Liu, "Evaluating machine learning algorithms for anomaly detection in clouds," in *IEEE International Conference on Big Data*, Dec 2016, pp. 2716–2721.

[21] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017, online Real-Time Learning Strategies for Data Streams.

[22] G. Lanciano, A. Ritacco, F. Brau, T. Cucinotta, M. Vannucci, A. Artale, J. Barata, and E. Sposato, "Using Self-Organizing Maps for the Behavioral Analysis of Virtualized Network Functions," in *Cloud Computing and Services Science*, D. Ferguson, C. Pahl, and M. Helfert, Eds. Cham: Springer International Publishing, 2021, pp. 153–177.

[23] W. Wang, L. Tang, C. Wang, and Q. Chen, "Real-Time Analysis of Multiple Root Causes for Anomalies assisted by Digital Twin in NFV Environment," *IEEE Trans. on Netw. and Serv. Manag.*, pp. 1–1, 2022.

[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[25] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2003, pp. 25–36.

[26] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection," *Data Min. Knowl. Discov.*, vol. 28, no. 1, p. 190–237, Jan 2014.

[27] P. Oliveri, "Class-modelling in food analytical chemistry: Development, sampling, optimisation and validation issues - a tutorial," *Analytica Chimica Acta*, vol. 982, pp. 9–19, 2017.

[28] N. Japkowicz, "Supervised versus unsupervised binary-learning by feed-forward neural networks."