Anomaly Detection for 5G Softwarized Infrastructures with Federated Learning

Salah Bin Ruba, Nour-El-Houda Yellas, Stefano Secci

Anomaly Detection for 5G Softwarized Infrastructures with Federated Learning

Salah Bin Ruba, Nour El-houda Yellas, Stefano Secci
Cnam, Paris, France. {first-name.last-name}@cnam.fr

***Abstract*—We present how to distribute an anomaly detection framework at the state of the art, called SYRROCA (SYstem Radiography and ROot Cause Analysis), for edge computing and 5G environment, using federated learning. The goal is to leverage on the distributed nature of federated learning to support data locality and local training of artificial intelligence modules, such as anomaly detection modules needed for closed-loop automation systems. We describe how the different functional modules interact and can be demonstrated.**

***Index Terms*—Federated Learning, anomaly detection, autoencoders**

I. Introduction

Closed-loop automation systems are expected to be more deeply integrated in 6G systems than they are in the 5G system. Indeed, while the 5G system specification does plan for the integration of artificial intelligence modules, they mostly refer to the monitoring of 5G network functions or the acceleration of radio access functions. A hot topic in current research is finding a feasible and scalable way to support cross-domain, cross-technology, inter-domain and hierarchical closed-loop systems such that no or little human intervention is needed in the reconfiguration of a highly softwarized network infrastructure stack.

The Monitor-Analyze-Plan-Execute (MAPE) approach to closed-loop automation [1] is often taken as a reference in the design of the different subsystems needed for network automation. The first two steps are particularly challenging in that a high number of features to observe and analyze exist for a geographically distributed and highly composite system as the 5G one, and its ongoing evolution with edge computing penetration. Anomaly detection is in particular considered as an important module of the analysis subsystems, in that it triggers the decision-making plans to reconfigure the infrastructure stack and make it resilient against the diverse set of impairments that can take place.

At the state of the art, the SYRROCA (SYstem Radiography and ROot Cause Analysis) framework was recently proposed as an anomaly detection system for softwarized infrastructures [2]. It has a machine learning engine to monitor the infrastructure, making use of LSTM (Long-Short Term Memory) autoencoders for spotting anomalies from a high number of features time-series collected at multiple network and system subsystems, in a centralized way; in [2] it is applied to the virtualized IP-Multimedia Subsystem use-case. Its application to various use-cases is possible thanks to a certain modularity of the autoencoder core. Nonetheless, its actual implementability to 5G and edge computing environments, which can be highly distributed and require stringent latency and resource efficiency guarantees, does call for a form of decentralization that we describe in this paper.

The paper is organized as follows. We introduce the 5G-customized anomaly detection framework in Section II. Its distribution using federated learning is presented in Section III. Section IV introduces the experimental setting. Finally, we present demonstration plans and perspectives in this area.

II. Reference anomaly detection framework

We describe how we customized the SYRROCA framework [2] for the 5G environment.

We leverage on a multi-layer LSTM Neural Network (NN) because such systems are optimized to learn both short and long-time correlations and seasonality in time-series, hence allowing to capture the behaviour of complex multivariate sequences. More precisely, LSTM NNs use encoders and decoders [3] in order to form a deep autoencoder to train metrics. An autoencoder is composed of two blocks:

- Encoder: it compresses the original high-dimension input into the low-dimension latent space;
- Decoder: it reconstructs input from the latent space, likely with larger output layers.

In fact, LSTM autoencoders are trained to reconstruct the normal working conditions of a given system, i.e., based on a representation of the normal operation of the system [4], with minimum errors. When they fail in reconstructing these nominal conditions, the error increases. This occurrence can be translated as an anomaly.

To adapt the SYRROCA framework to the 5G environment, as depicted in Figure 1, we group metrics from a reference 5G infrastructure into component-specific groups, i.e. CPU, Memory, Network, Radio and Disk, at different infrastructure levels, namely physical level, virtual level and Radio Access Network (RAN) level.

We exploit a dataset of few thousands of time-series samples related to an emulated end-to-end 5G system using available open-source software stacks for the different systems, as described in [4].

Although the SYRROCA framework proves its algorithmic capacity to spot anomalies and support fine-grained root cause analysis as shown in [2], it cannot be directly applied to perform real-time anomaly detection scenarios of distributed low-latency infrastructures because of the following practical limitations: (i) it cannot scale with the increasing size of data

collected at different levels and for several groups of resources, (ii) it cannot ensure low-latency training and inference because of the high data volume and its impact on link and node utilization, for high-frequency collection rates, and (iii) it cannot guarantee data privacy, since the collected data needs to be centralized at one location. To cope with these limitations, we propose its distribution using federated learning.

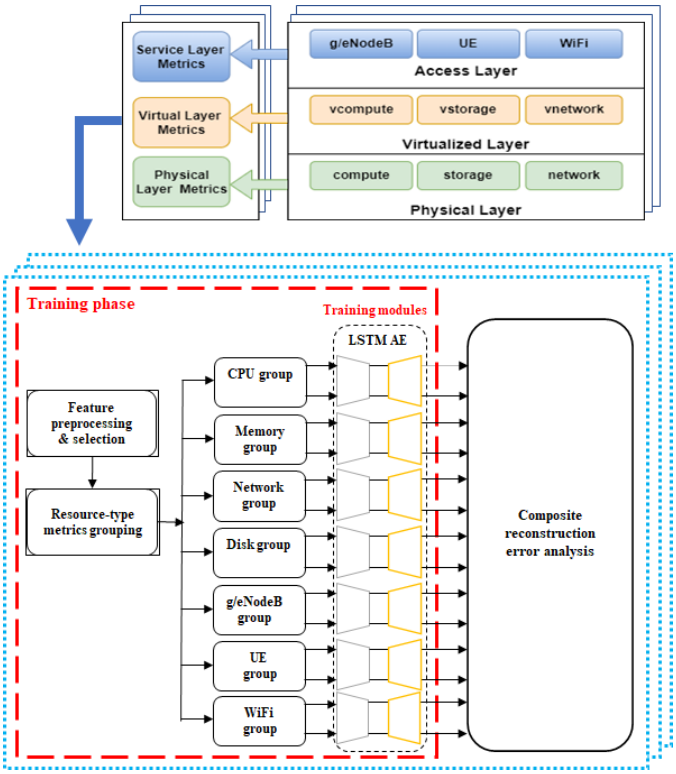

Fig. 1: Anomaly detection framework.

III. Federated Learning Deployment of the Anomaly Detection Framework

Federated Learning (FL) is a distributed ML technique, where the training is carried out on multiple network nodes. The whole process is governed by a central server, which is responsible for the initialization and assignment of hyper-parameters [5].

In order to achieve a satisfactory level of accuracy, global models, resulting from the aggregation of local model parameters at the aggregation servers, are exchanged iteratively between the FL server and the distributed FL clients. The so-updated model is used for inference, while allowing to have a global view of the system thanks to the aggregation step.

Our FL framework is implemented using Tensorflow Keras library for building and training the ML models at the clients side. For what concerns the aggregation algorithm at the server side, we use Federated Averaging approach (also called FedAvg) [6]; FedAvg, besides being computationally efficient, is sufficient for our data that is not highly heterogeneous (high heterogeneity may require a more sophisticated aggregation algorithm). Nonetheless, other aggregation algorithms may be further considered.

We use the Remote Procedure Calls (RPC) Python library called gRPC [7] to implement the communication between the FL server and clients, as gRPC has the ability to run procedures and code routines on remote machines by serializing and marshalling objects. In this sense, the exchanged training functions, ML models and initialization parameters between the central server and the participating nodes, are treated as objects and transmitted as function parameters.

We adapt the general FL process to the anomaly detection framework along the following steps:

- *Client registration*: each client registers with the FL server to be able to participate in the training task;
- *Parameter initialization*: consists of initializing the hyper-parameters and the LSTM autoencoders at the server side;
- *Parameter sharing*: during this step, the server sends the LSTM autoencoder model, the hyper-parameters and the process configuration to the clients;
- *Model training*: each client trains the autoencoder model with its own available data;
- *Parameter aggregation*: after each round of training, the model parameters are exchanged with the server and the global model is updated, either for inference or for a new round of training.

We stop the FL training after a fixed number of rounds, depending on the desired accuracy of the inference anomaly detection model.

As depicted in Figure 2, in the reference framework, both encoder and decoder layers are composed of two LSTM cells and one dropout regularization level to prevent over-fitting that could make the autoencoder input and output similar (i.e., no learning happens). Other factors that can effect the performance of the training are the hyper-parameters, which must be carefully selected. In general, the batch size and the number of epochs are selected to be proportional to the number of training samples.

In order to build an effective ML model, the autoencoder has to be designed in a such way that it is light, fast to train and able to produce a real representation on the trained metrics.

A. Data preprocessing and LSTM AE training

To perform realistic experiments, we use the 5G3E (5G End-to-End Emulation) dataset [4] to train and test our anomaly detection framework. The dataset is collected in a duration of 15 days, 14 days where the system runs under normal conditions - used for the training phase - and 1 day where different components of the infrastructure undergoe injected abnormal functional conditions. We feed the 5G3E data to the LSTM autoencoders to learn the normal 5G system state.

During the training phase, data is fed as input to the autoencoders in order to learn how to map the input metrics to a compact representation through the latent space of the autoencoder. As already mentioned, each autoencoder is responsible of learning the nominal state of the system based

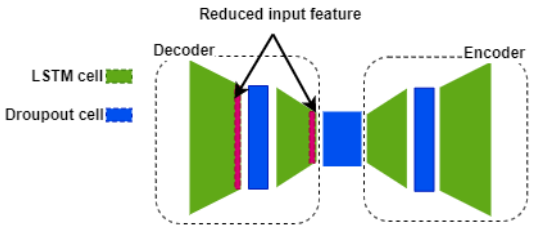

Fig. 2: Autoencoder architecture.

on one type of resource. To do so, a preprocessing step of the dataset is needed.

In fact, data preprocessing consists of (i) grouping data by type of resource; we split the overall dataset into sub-datasets having similar type of features (e.g. a sub-dataset could be for CPU-related features); (ii) reshaping the input data for LSTM networks into $number\ of\ samples \times number\ of\ steps \times number\ of\ features$, where the $number\ of\ steps$ represents a sequence of past observations given as input in order to map to the output observation.

When data is preprocessed, features are grouped according to the resource type. Each sub-dataset is fed to a dedicated autoencoder, where all autoencoders are sharing the same architecture. This reduces the complexity of the architecture and the training time.

The final goal is to minimize the reconstruction error, for which we use the Mean Square error (MSE). In fact, high MSE values, compared to those obtained during the training, indicate whether the metrics significantly deviate from the learned latent space representation during run-time. This is considered as an anomalous system state.

The anomaly behavior injection is done with different intensities in order to test the sensitivity of our model to anomaly severity, as explained in [4].

B. Hyper-parameters setting

We train our LSTM autoencoder while varying the number of distributed FL nodes in order to demonstrate the effect of the distributed architecture on the efficiency of both the training time and the quality of learning.

We use Kubernetes [8] to deploy the FL architecture. In fact, each FL entity is deployed using one pod, on which the LSTM autoencoder model at the clients side and the aggregation algorithm at the server side run. In addition, it is worth-mentioning that for each entity, we implement as much LSTM autoencoder models as resources and for each component-level data, where these models are supposed to run in parallel, allowing to detect anomalies at multiple layers of the system.

Once the nodes finish training, a threshold value is recomputed for each metric group; in our case, as for SYRROCA, we use the 99th-quantile value from the input dataset. At the test phase, this threshold is utilized to spot anomalies. In fact, all samples whose MSE exceeds the threshold value are considered as anomalies. It is worth-mentioning that the threshold value should be set rigorously since it plays an important role in distinguishing between normal and abnormal values. In Table I, we summarize the hyper-parameters used for the experimental evaluation of the proposed framework.

Optimizer	Adam
Loss function	MSE
Dropout rate	0.2%
Learning rate	0.01
Batch size	64
Number of epochs	50 (early stopping activated)
Number of rounds	5
LSTM Activation function	elu
LSTM recurrent activation function	sigmoid
FL node sampling	100%
Number of participating FL	1, 2, 4, 8

TABLE I: Hyper-parameters for FL anomaly detection

C. Data load-balancing strategy

As deployment strategy of the federated learning framework, we adopt an over-the-top edge-computing friendly approach as described in Figure 3, were FL nodes are deployed as servers possibly detached from the actual sources of data, yet supposed to be in strong proximity with the sources. This implies the adoption of a data pipe-lining system able to distribute data or make data available at FL nodes in runtime.

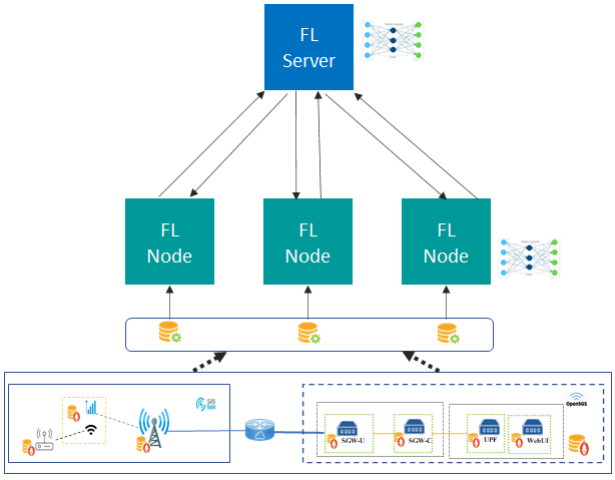

Fig. 3: Flat load-balancing approach.

IV. Technical Demonstration

We precise in the following additional details related to the technical demonstration of the framework.

A. Anomaly detection setting

In order to evaluate the resulted FL model, we use the 5G3E [4] testset, produced by injecting different anomalies; the anomaly injection is done with different intensities in order to test the sensitivity of our model to anomaly severity. The main goal is to be able to detect anomalous events which can affect system performance, regardless of the anomaly behavior

type. In the technical demonstration we show how the injected anomalies affect the following system performance:

- CPU load: to test the effect of CPU overload on the system.
- Bandwidth limitation: to show what effect a limited bandwidth has on the running state of system resources.
- Packet delivery: different packet loss rates are considered.
- Link restoration: like failure between different components of the infrastructure.

B. Data pipe-lining emulation

We are in the process of evaluating different possible data pip-lining able to make a high number of time-series available over a network.

In the meantime, in order to emulate the FL system, we dispatch the data from the 5G3E dataset to the training nodes. To emulate data arrival at FL nodes, we construct a dedicated layer for data collection with a single repository to save collected metrics, and mechanisms for data batching and integration. This layer provides the data-lake needed for The FL nodes by sharing a single data repository.

To demonstrate a real world case, the framework is trained under time constraints; i.e. the data is not completely fed into the system, instead only a specific number of samples is batched to the training process. This number is selected to respect the stringent latency induced by the infrastructure. In our case the maximum number of samples does not exceed 1000 per round. During training, each node fetches a subset of data, where the whole data in the data-lake is consumed by the participating nodes in rounds.

Each node has a different view of the system performance on which it relies to train the AE. The aggregation of parameters at the central server to constitute the global model allows to have a full representation of the system state.

Once finished local training, the nodes transmit there locally trained models to the FL server. The trained models learn a subset of the normal running state of the infrastructure. At the FL server, these fragments of knowledge are aggregated to reflect the entirety of the infrastructure. As the training advances in consuming more samples, the framework gradually builds the knowledge on the system.

Eventually the FL server produces a trained model on the whole representation of the infrastructure state. This model is then used for inference.

C. System state characterization

To characterize and understand the anomalies detected by the framework, we leverage on state graph and radiography visualization techniques at the state of the art:

- Radiography visualization [9], consists of a compact representation to exhibit the final results to system users and operators. It is obtained combining the MSE with a service metric to get a 2D density plot made with Kernel Density Estimation (KDE).
- Anomalous state graph [2] where directed state graphs are used to visualize the evolution of the running system across the nominal and the degraded states, indicating the most deviated resource group(s) for a given anomaly.

V. Perspectives

We synthetically presented a federated learning framework extending and adapting a centralized framework at the state of the art. We explain how LSTM encoders are used to calculate the MSEs that determine the deviation of the system from the nominal state. Different experiment methodologies mimic real world situations that occur in 5G systems and beyond-5G systems where the penetration of AI is expected to increase.

Further work will have to address a number of challenges of the proposed system. First, data pipe-lining and the impact of data transmission and propagation delays are aspects that need to be studied and evaluated. The heterogeneity and the fluctuation of FL nodes computing power are also factors expected to affect training precision. This also stands for the number of FL nodes that are deployed: the higher their number, the lower the individual training time but the lower the detection accuracy. These trade-offs need to be studied and lead to adequate system design choices and assessment.

Acknowledgement

This work was funded by the H2020 AI@EDGE (https://aiatedge.eu; grant nb. 101015922), the IA/AMI-5G INFLUENCE project, and the ANR PARFAIT project (grant nb: ANR-21-CE25-0013).

References

[1] Autonomic. Computing. “An architectural blueprint for autonomic computing.” In: *IBM White Paper* (2006), pp. 1–6.

[2] Alessio Diamanti, Jos Manuel Snchez Vlchez, and Stefano Secci. “An AI-empowered framework for cross-layer softwarized infrastructure state assessment”. In: *IEEE Transactions on Network and Service Management* (2022), pp. 1–1. DOI: 10.1109/TNSM.2022.3161872.

[3] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[4] C Phung et al. “An Open Dataset for Beyond-5G Data-driven Network Automation Experiments.” In: *Hal.* hal-03698732 (2022).

[5] HB McMahan et al. “Federated learning of deep networks using model averaging. CoRR abs/1602.05629”. In: *arXiv preprint arXiv:1602.05629* (2016).

[6] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[7] *gRPC*. https://github.com/grpc.

[8] *Kubernetes*. URL: http://kubernetes.io.

[9] Alessio Diamanti, José Manuel Sanchez Vilchez, and Stefano Secci. “LSTM-based radiography for anomaly detection in softwarized infrastructures”. In: *2020 32nd International Teletraffic Congress (ITC 32)*. IEEE. 2020, pp. 28–36.