An efficient and lightweight load forecasting for proactive scaling in 5G mobile networks

Imad ALAWE*, Yassine HADJADJ-AOUL*, Adlen KSENTINI†,
Philippe BERTIN*, Cesar VIHO‡, and Davy DARCHE§
*Firstname.Lastname@b-com.com, IRT b<>com, Rennes, France
†adlen.ksentini@eurecom.fr, EURECOM, Sophia-Antipolis, France
‡Cesar.Viho@irisa.fr, IRISA, Univ Rennes, France
§davy.darche@tdf.fr, TDF, Paris, France

***Abstract*—The number of connected devices is increasing with the emergence of new services and trends. This phenomenon is leading to a traffic growth over both the control and the data planes of the mobile core network. It is expected that the traffic will increase more and more with the installation of the new generation of mobile networking (5G) as it offers more services that are intended to be connected over the same network, in addition to the legacy ones. Therefore, the 3GPP group has rethought the architecture of the New Generation Core (NGC) by defining its components as Virtualized Network Functions (VNF). However, scalability techniques should be envisioned in order to answer the needs, in term of resource provisioning, without degrading the Quality Of Service (QoS) already offered by hardware based core networks. Neural networks, and in particular deep learning, having shown their effectiveness in predicting time series, could be good candidates for predicting traffic evolution. In this paper, we propose a novel solution to generalize neural networks while accelerating the learning process by using K-means clustering, and a Monte-Carlo method. We benchmarked multiple types of deep neural networks using real operator's data in order to compare their efficiency in forecasting the upcoming network load for dynamic and proactive resources' provisioning. The proposed solution allows obtaining very good predictions of the traffic evolution while reducing by 50% the time needed for the learning phase.**

***Keywords*—5G, Scaling, Mobile Core Network, Prediction, Neural Network, Training**

I. Introduction

Mobile networks are facing a traffic load increase over both the data and the control planes due to the explosion of the number of connected devices and the emergence of new trends as the Internet of Things (IoT), and e-Health ...etc. In fact, the number of devices connected to 5G is expected to reach 28 billions by 2021, in front of 9 billions in 2016 [1].

In the new 5G mobile core architecture, the Access and Mobility Function (AMF), the new generation component of the Mobility Management Entity (MME), is likely to be congested since it is the single point of access for the control plane in the core network. Indeed, as shown in [2], the overhead causes an increase in latency for the Network Access Stratum (NAS) procedures, up to some requests' rejects when certain load thresholds are exceeded.

The 3GPP group [3] has rethought the architecture of the New Generation Mobile Core Network (NGC) [4], to tackle the issue listed above, by introducing more modular Network Functions (NF) to compose the control plane service. Those NF could also relies on network virtualization via Network Function Virtualization (NFV). Even though, this new architecture offers more flexibility and elasticity, the 3GPP group didn't tackle yet the strategies or the techniques to be used in order to dynamically provision the needed resources without degrading the Quality of Service (QoS).

Resource scaling, when required, is effective when accompanied by a traffic evolution prediction function to avoid not only wasted resources but also quality degradation. Neural networks, and in particular deep learning, having shown their effectiveness in predicting time series, could be good candidates for predicting traffic evolution. Indeed, as neural networks are able to predict the upcoming load combined with an orchestrator, it will offer dynamic and proactive resources provisioning. However, we still need to determine what kind of neural networks best fit the prediction of Mobile Networking load evolution and how historical data should be formatted in order to achieve the most accurate results.

In this paper, we analyze the most effective techniques for predicting the traffic evolution of a mobile network. We are particularly interested in the possibility of generalizing prediction techniques to allow them to effectively predict traffic evolution even when the scale of traffic changes. To that purpose, we introduce a novel technique allowing to generalize the neural network while accelerating significantly the training phase using K-means clustering and a Monte-Carlo method. We apply our technique to three types of neural networks: Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) cell; Convolutional Neural Network (CNN); and CNN combined with LSTM Neural Network. Finally a benchmark lists the effects of parameters variation (i.e. Number of Clusters, epochs, Number of boosting, etc.) on the prediction efficiency for each of the neural network types listed above. Simulation using TensorFlow library [5] are used to provide this benchmark.

The remainder of this paper is organized as follows. Section II lists and analyses related work on scalability solutions for VNFs and more precisely for mobile core network components. Section III presents the different types of considered neural networks. Section IV describes the proposed solution process from data management to neural network testing.

Model evaluation and results analysis are presented in Section V. Finally Section VI concludes this paper and introduces our future work.

II. Related Work

The introduction of the NGC by the 3GPP group provides the flexibility to scale the infrastructure much more easily to respond to increasing traffic loads or even to free up resources in the event of a drop in needs. However, new strategies and techniques should be envisioned for the dynamic scaling of the core network in order to adapt dynamically the resources provisioning to the needs.

The most commonly used techniques in the literature are based on fixed thresholds. The idea behind this technique is simple: a scale-in process is launched whenever the actual load goes below a fixed ”scale in” threshold and a scale-up process is launched when the actual load bypass a fixed ”scale up” threshold. In [6], we proposed such a solution to manage the congestion that the AMF may undergo with the massive access of IoT devices. Besides, these kind of techniques are widely deployed in many commercial solutions and open-source cloud solutions [7][8].

This type of techniques is rather effective and very simple to implement but can present stability problems. The latter can be a source of degradation of the quality of service making it not suitable for mobile networks. Indeed, deploying, for example, a new VNF may take up to a dozen of minutes depending on the data-center configuration. Thus, the delay between a reactive decision and the scaling process will lead to an increase of the UE attach duration and some requests rejection.

In order to solve the threshold configuration problem, the authors, in [9], proposed a Reinforcement Learning (RL)-based strategy to adapt to dynamic environments and to improve the scaling policy before taking any action following a reward and a penalty process. The authors showed that it behaves better than threshold-based techniques. However as it is a reactive solution, it inherits the drawbacks listed above. In addition, RL may need a considerable time before converging and taking adequate scaling decisions. This is not acceptable in 5G networks due to the time constraint of certain type of applications and the low latency requirement fixed by the standards.

The authors, in [10], proposed a proactive strategy, which consists in forecasting the CPU load of the system based on a historical dataset. Even though this solution is proactive, it is not adapted for the mobile context as it considers only system level information, like CPU or memory usage. In addition to system level information, service level information should also be taken into consideration while taking scaling decisions as it reflects better the global load of the core network.

In this paper, we propose to forecast the evolution of traffic based on neural networks. The potential variability of telecom network traffic has led us to propose a method to further generalize neural networks so that they can respond correctly to an amplitude change. The relearning, in case of a significant deviation in the prediction, also led us to propose a solution accelerating the learning phase.

The authors, in [11], worked on accelerating neural networks training using clustering. The main objective of the paper was to use k-means clustering to reduce data size for the training phase while keeping almost the same accuracy with a shallow neural network. However, the analysis of the results did not go far enough to show the amount of samples needed to converge the classification problem under consideration. Indeed, no clear results was presented on how much the technique can accelerate the training process or even its efficiency. Moreover, in high dimensionality, k-means is not that efficient and the determination of the number of samples remain an open issue. On the other hand, reducing the number of samples results in a loss of information that could severely affect the quality of learning. This is one of the key points that we are improving in this paper by resorting to a Monte-Carlo method.

III. Architecture

As mentioned above, the 3GPP group introduced the New Generation Core Network (NGC) in order to answer the requirements of 5G. In what follows, we consider this NGC, in which prediction techniques will have to be introduced to better manage the scaling of resources.

Figure 1 depicts the architecture of the NGC. The 3GPP group decided to introduce the components of the core network as modular functions allowing to benefit from Network Function Virtualization (NFV) to scale in/out depending on the needs as it is now a matter of software deployment.

Even if the technique we are proposing can be applied to different components of the NGC, for the moment, our proposal targets especially the Access and Mobility function (AMF) as it is the single point of access of the core network. In fact, the control traffic in 5G is expected to increase significantly as the number of connected devices is increasing [2].

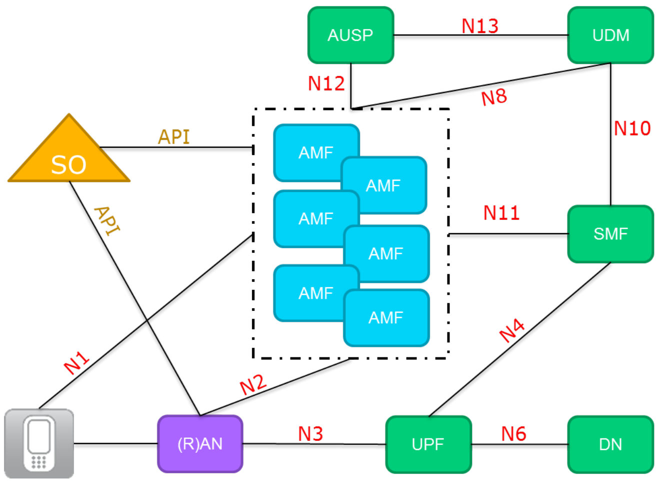


Fig. 1: Architecture proposal for 5G

Thanks to NFV, dynamic scaling issue can be addressed by using machine learning in order to forecast the upcoming load of the core network and thus, scale in/out resources in a proactive manner to avoid congestion while meeting the 5G requirements in terms of latency. In addition to NFV, dynamic scaling is now possible as the 3GPP group has split the Mobility and Management Entity (MME), the AMF version in 4G, into AMF and Unified Data Management (UDM) entities. Therefore, the AMF function is now stateless as all the User Equipments (UE) contexts and session contexts are hosted in the UDM, adding more flexibility and elasticity to the mobile core network [4]. Indeed, in that case, any user procedure can be handled by any AMF worker.

When a request is received by an AMF instance, the AMF brings the user context from the UDM, using the reference interfaces noted Nx and represented in Figure 1. The AMF processes the request and stores the information and status of the new user context in the UDM.

Based on ETSI's NFV reference architecture, the scalability of a VNF is a decision made by the NFV Orchestrator (NFVO), using information provided by both the Virtual Infrastructure Manager (VIM) and the VNF Manager (VNFM). Therefore, for the sake of simplicity, we have added in this work only one service orchestrator (SO) to the NGC architecture. As already mentioned, our approach aims to balance the burden between AMF instances and to scale in/out the AMF functions as required. So the intelligence needed to apply this strategy is located in the SO.

IV. Neural networks traffic load estimation for resources' scaling in 5G

The prediction of the evolution of a time series, whether periodic or not, has been an active research topic for several years [12]. Several studies, in this sense, have shown the potential of neural networks, and especially LSTM, to identify traffic patterns [13].

In this section, we analyze different types of neural networks for predicting load evolution in a network. We use traditional neural networks, CNN and LSTM, but we also suggest a combined architecture of the latter, which not only reduces learning time but also makes it more effective. In addition, we introduce a novel technique, based on the Monte-Carlo method, allowing to generalize the neural network to allow them to effectively predict traffic evolution even when the scale of traffic changes. The proposed technique also has the advantage of making learning much faster thanks to the use of K-means clustering technique on the data.

A. A combined CNN-LSTM Neural Networks

Before going deeper into the selected neural network architecture, let's start with a brief look at its different components.

1) Convolutional Neural Networks (CNN): A CNN, depicted in Figure 2 (CNN), comprises three main layers. The convolution layer is used to extract features from the input data, while preserving its spatial relationship. Then the non-linear operation Rectified Linear Unit (ReLu) is applied. Following the ReLU function, spatial pooling (sub-sampling) is applied in order to reduce the dimensionality of each feature map while keeping the most important information. Finally before the output a final fully connected layer is used. This final step will allow to classify the output and to learn non-linear combinations of the features (refer to [14] for more details).

2) Long Short Term Memory (LSTM): The second neural network used is RNN and more precisely an LSTM cell, which is another type of neural network. Compared to a conventional neural network, an LSTM cell can keep state memory of the last passed activation events in the network as temporal contextual information [15][16] (cf. Figure 2(LSTM)). Hence, the advantage of the LSTM cell is to allow data patterns to be stored without degradation over time.

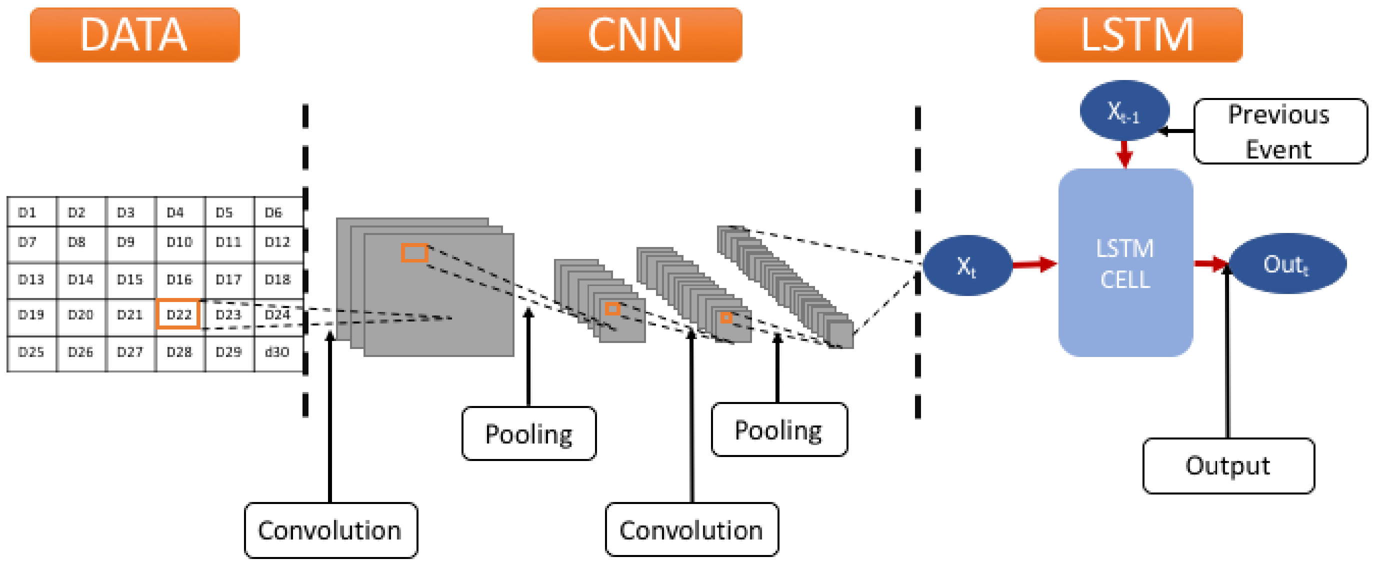


Fig. 2: CNN-LSTM Architecture

For more details on RNN and LSTM structures and functions, interested readers may refer to [17].

3) A combined CNN–LSTM architecture: In addition to the two types of neural networks explained above, we added another one by combining both types used before. This third type will allow getting improved results by benefiting from the strengths of both the LSTM and the CNN. Indeed, as an entry we use the CNN in order to extract features (i.e. traffic pattern) and simplify them by using the pooling, which will allow retaining the main part of the pattern. Hence, beside using a fully connected layer as an output layer, we use the LSTM cell in order to forecast the upcoming traffic load.

B. Generalizing neural nets and accelerating the training

The generalization of neural networks during learning is an important characteristic, since it makes it possible to avoid the phenomenon of over-fitting, which leads to poor performance with new data.

Various techniques have been introduced in the literature to avoid the over-fitting problem. In this paper, we propose a novel approach that has the advantage of accelerating significantly the speed of learning. This technique is particularly effective when predicting time series due to redundancy in the data. The main idea consists first in grouping the data using the K-means clustering technique.

Existing approaches proposed to reduce the size of the initial data after the clustering phase. This speeds up learning but reduces the accuracy of prediction, as only a sub-set of the initial data is used for the training. In contrast with these approaches, we propose to exploit all this data using a Monte-Carlo method.

The proposed solution consists in getting randomly, from each cluster, an element for the training phase, using the function "getRandom()". This process is repeated at each epoch, which allows covering the whole dataset and thus not loosing any precision and accuracy while accelerating significantly the training.

Algorithm 1 Monte Carlo-based training procedure

```
1: Inputs: $epochs$, $dataset$, $nClusters$
2: $Clusters[] = K$-means( $nClusters$,$dataset$)
3: for $ep$ in $epochs$ do
4:     $trainset = []$
5:     for $clst$ in $Clusters$ do
6:         $trainset$.add(getRandom($clst$))
7:     Train the neural network with $trainset$
```

Note that the obtained performance is directly related to the number of clusters *nClusters*. Indeed, this number must be related to the percentage of data to be considered in each period. The more similarities we have in the input data, the smallest is the number *nClusters*.

A boost process, in which all the dataset is injected, can also be considered for a number of epochs at the end of the training phase. This phase is efficient only when the number of clusters is too small.

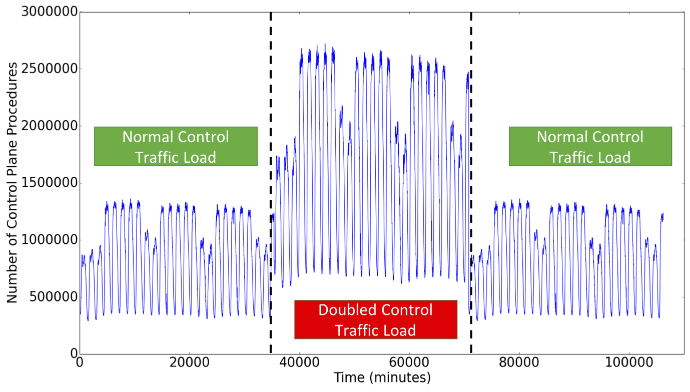


Fig. 3: Testing data used to test the Neural Networks

V. Models Evaluation and Results Analysis

A. Dataset

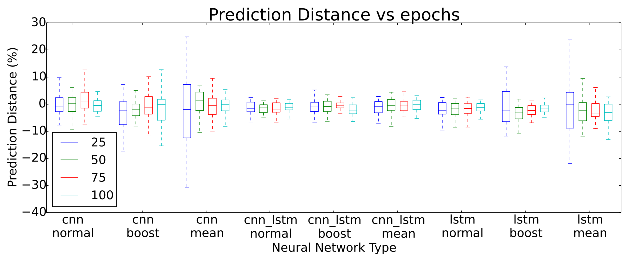
The dataset used in our work contains real mobile data collected by Telecom Italia Big Data Challenge [18] and hosted on Harvard Dataverse [19]. This dataset is rich of information. It contains two months of data collected on different Cells of the mobile core network of Mobile Italia in Milano, with a period of 10 minutes. First, the data is sorted following the timestamps and not the cell ids. In other words, for each period of time (10 minutes), we regrouped together the data of all the cells in Milano city. Then, we split the obtained dataset into two datasets: training dataset; and testing dataset. The training dataset will be constituted of 60% of the data and the testing dataset will be filled with the 40% left.

Finally, we decided to add for the testing dataset: (i) a copy of the 40% but doubled, and (ii) another copy of the 40%. Therefore the testing dataset, illustrated in figure 3, is split into 3 phases: Normal control traffic; an increased (doubled) control traffic; and return to a normal control traffic. By this, in the simulation section we will be testing the precision of the prediction of the neural networks on arrivals where the neural network is not trained for and hence testing it adaptation to traffic load amplitude change.

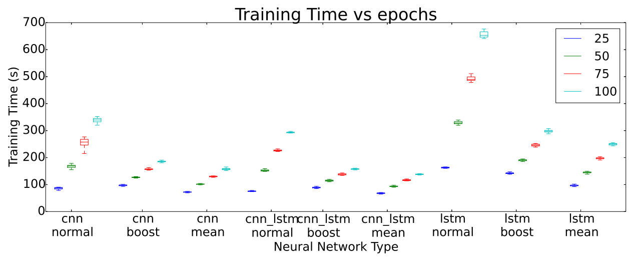
B. Scenario and simulations

In this section we explain the procedure and the simulation launched in order to validate the performance of the proposed training strategy. We built the three types of neural networks listed above using the Tensorflow library [5].

In our simulations we decided to test the effects of the number of epochs, the number of clusters (i.e. percentage of the data to consider), and the effects of the boost metric. The boost technique consists in adding, in some number of epochs at the end of the training phase, the whole training dataset and not only the clustered data. This test will allow to show if the K-means clustering based strategy is enough to get the same precision as if we inject the whole data or still a complete re-injection of the training data is needed.



(a)



(b)

Fig. 4: Evolution of prediction distance (a) and training phase duration (b) vs number of epochs in training phase

For each neural network three different types of simulations are launched. In each one we will vary one metric to see its effect on the prediction precision of the neural network and the time needed for the training phase. It worths mentioning that each simulation is repeated 30 times in order to get accurate and trustful results.

The simulations parameters are listed in the following:

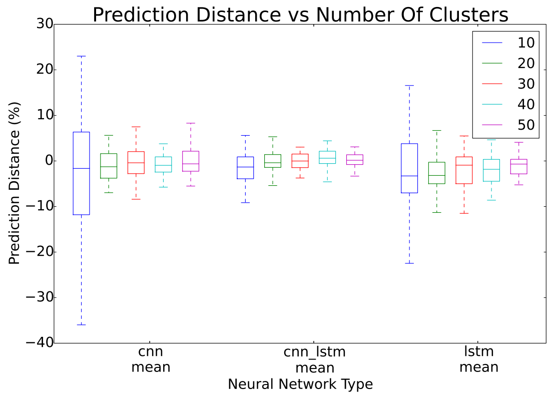
- Each entry of the neural network is filled with the last 20 past data in order to predict the 21st one.
- The number of clusters is fixed to 30% of the size of training dataset
- The number of epochs is fixed to 100 epochs by training phase.
- The number of boost epochs is fixed to 10.

Finally, those metrics will be fixed in each simulation type where only the concerned metric of the simulation will vary in order to understand its effect on the training phase, and hence the precision of the prediction and the duration of the training phase.

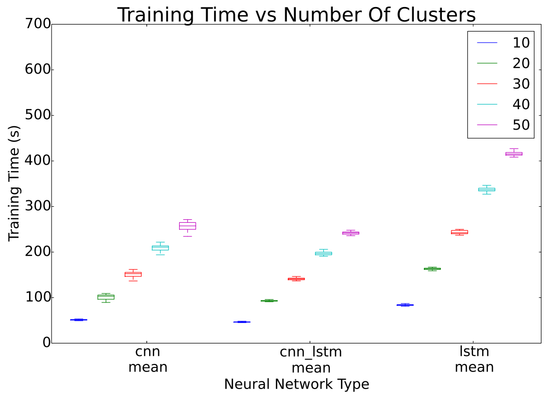
C. *Results and Analysis*

From Figure 4 (a), we can see the effects of the number of epochs in the training phase for the different types of neural networks as with K-means clustering or not and with boost session or not. For all the simulations and the neural networks, it is evident that 25 epochs is not enough to get precise prediction. On the other hand we can see that with clustering technique the precision for each neural network is kind of similar as if there was no data clustering. Thus, it shows that applying K-means clustering with random injection at each epochs allows to get the same precision. Also from Figure 4 (b), we can notice that the training time with K-means clustering is divided by two comparing to the normal injection of the whole dataset. Therefore K-means clustering allows to reduce training duration while keeping the same precision as if we inject the whole training dataset. It worths mentioning also that the boost at the end of an accelerated training phase is not so efficient as it does not add any precision to the output. So K-means clustering alone is a very interesting solution in order to accelerate the training phase of the neural network.

In the same spirit Figure 5 (a) shows the effect of the variation of the number of clusters used to classify the training dataset, on the output precision. As illustrated in the Figure, if the number of clusters is around 10% of the training dataset size, the results are not so good. On the other hand from 20 % we start to get interesting results in term of output precision. Also figure 5 (b) shows the duration needed to train the neural network depending on the number of clusters. It shows that

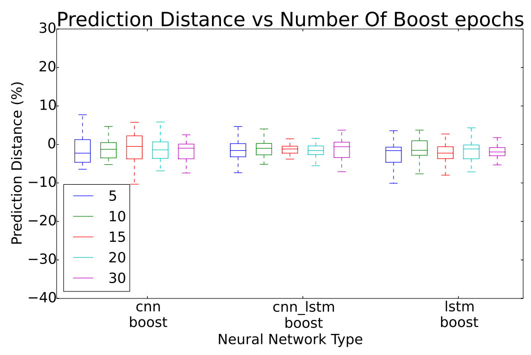


(a)

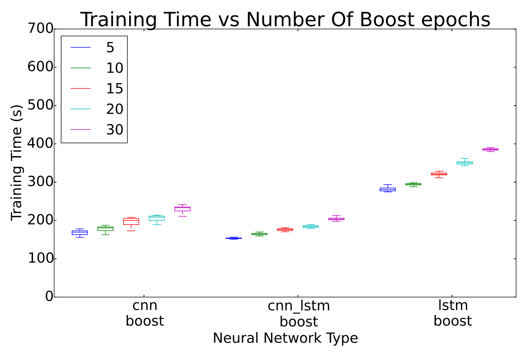


(b)

Fig. 5: Evolution of prediction distance (a) and training phase duration (b) vs number of clusters in training phase



(a)



(b)

Fig. 6: Evolution of prediction distance (a) and training phase duration (b) vs number of boost in training phase

the duration increases with the number of clusters chosen, and this is the expected behavior. But still worth mentioning that the duration needed to train the neural network even when the number of clusters is 50% of the training dataset size, is less than the training duration of a normal neural network with no generalization technique.

Figure 6 (a), shows the effect of adding some boost epochs at the end of a accelerated (using the proposed solution) training session. We can notice that the boost does not add precision. Some time even adds some confusion to the neural network as the distances goes beyond or beneath zero. In the same aspect boost epochs adds some latency to the duration of a training phase as it adds more data for the last epochs. Thus boost epochs are not efficient in those use case and using K-means clustering alone is better as it gives the desired accuracy while decreasing the training phase duration.

On the other hand from the Figures 4 (a), 5 (a) and 6 (a), we notice that the proposed model including CNN and LSTM is behaving very well and even better than the version of CNN standalone and the LSTM standalone. This is explained by the fact that the CNN is very good in extracting features from the input data and the LSTM is behaving so good in classifying the output.

VI. Conclusion

In this paper, we suggested combining a CNN and an LSTM neural networks to predict network loads' evolution. We proposed a novel solution to generalize neural networks based on K-means clustering and Monte-Carlo method. We used real Operator's data in order to evaluate the efficiency of the proposed technique in predicting the upcoming network load for dynamic and proactive resource provisioning. The simulations showed that our solution allows reducing by 50% the duration of the training phase of a Neural Network. In addition, the proposed combined neural network showed clearly its efficiency in predicting the evolution of the considered time series as it performs better than a standalone CNN or a standalone LSTM.

References

[1] A. Nordrum, "Popular internet of things forecast of 50 billion devices by 2020 is outdated," IEEE Spectrum. [Online]. Available: https://goo.gl/5TS9s5

[2] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin, and A. Kerbellec, "On evaluating different trends for virtualized and sdn-ready mobile network," in *CloudNet*. IEEE, 2017, pp. 1–6.

[3] "3rd generation partnership project (3GPP)." [Online]. Available: http://www.3gpp.org

[4] 3rd Generation Partnership Project (3GPP), "System architecture for the 5g system." [Online]. Available: https://tinyurl.com/yd9z3no9

[5] TensorFlow, "An open-source machine learning framework for everyone." [Online]. Available: https://www.tensorflow.org/

[6] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5g core network: the amf case," in *IEEE CCNC*, 2018.

[7] Amazon, "Web services auto scaling." [Online]. Available: https://aws.amazon.com/autoscaling/

[8] A. CloudStack, "Open source cloud computing." [Online]. Available: http://cloudstack.apache.org/

[9] C. H. T. Arteaga, F. Rissoi, and O. M. C. Rendon, "An adaptive scaling mechanism for managing performance variations in network functions virtualization: A case study in an nfv-based epc," in *CNSM*. IEEE, 2017, pp. 1–7.

[10] A. Bilal, T. Tarik, A. Vajda, and B. Miloud, "Dynamic cloud resource scheduling in virtualized 5g mobile systems," in *GLOBECOM*. IEEE, 2016, pp. 1–6.

[11] K. M. Faraoun and A. Boukelif, "Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions," *International Journal of Computational Intelligence*, vol. 5, no. 3, pp. 161–168, 2007.

[12] S. Basterrech, G. Rubino, and V. Snasel, "Sensitivity analysis of echo state networks for forecasting pseudo-periodic time series," in *SoCPaR*, Nov 2015, pp. 328–333.

[13] F. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," *Springer Perspectives in Neural Computing*, vol. Neural Nets WIRN Vietri-01, pp. 193–20, 2002.

[14] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, 2014.

[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[18] T. Italia, "Telecom italia big data challenge." [Online]. Available: http://www.telecomitalia.com/tit/en/bigdatachallenge/contest.html

[19] ——, "A multi-source dataset of urban life in the city of milan and the province of trentino dataverse." [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV