OPEX-Limited 5G RAN Slicing: an Over-Dataset Constrained Deep Learning Approach

Hatim Chergui, and Christos Verikoukis
CTTC, Barcelona, Spain
Contact Emails: {hatim.chergui, cveri}@cttc.es

***Abstract*—In this paper, we investigate the concept of OPEX-limited resource provisioning as a key component in fifth generation (5G) radio access networks (RAN) slicing. The different RAN slices' tenants (i.e. logical operators) are dynamically allocated isolated portions of physical resource blocks (PRBs), baseband processing resources and backhaul capacity. To achieve this dynamic resource allocation, we rely on key performance indicators (KPIs) datasets stemming from a live cellular network endowed with traffic probes. These datasets are used to train a new class of deep neural networks (DNNs) models where OPEX requirements, formulated as non-convex non-differentiable violation rate constraints, are also dataset-dependent. The designed constrained DNNs are then optimized via a non-zero sum two-player game strategy. In this respect, we highlight the effect of the different hyperparameters on the respect of the OPEX limitations, while ensuring a dynamic RAN resource orchestration that follows the slices' traffics trends.**

***Index Terms*—5G, datasets, deep neural networks, dynamic slicing, game theory, non-convex optimization, OPEX, violation rate.**

I. Introduction

NETWORK slicing is a paramount feature in 5G cellular systems. It creates a number of logically isolated networks, or "slices", out of the same physical infrastructure shared by multiple over-the-top (OTT) tenants, offering thereby an increased statistical multiplexing [1]. This can lead to the reduction of the Capital Expenditures (CAPEX) significantly for these operators. Operational Expenditures (OPEX) can also be reduced thanks to the softwarization and virtualization technologies employed in network slicing [2], and that enable the end-to-end programmability of network functions, paving the way to a flexible and dynamic resource allocation for the slices, which allows to exploit the available physical resources in a more efficient way [3], [4]. In this context, machine learning (ML) techniques and in particular deep neural networks (DNNs) are expected to be the cornerstone in the automation of end-to-end resource provisioning. This includes standard DNNs that, based on the slices' traffics, enable the estimation of the required resources at each virtual network function (VNF) such as physical resource blocks at a transmission/reception points (TRP) and radio resource connected (RRC) users' licenses at a virtual baseband processing unit (vBBU).

From a financial point of view, imposing constraints on the OPEX of network resources is also required to let the slices' tenants properly control their expenditures while requesting the different resources from the physical operator. This implies that network slicing OPEX optimization should be conducted jointly with resource allocation, and requires the design of a new family of DNN models that take into account constraints while learning from datasets.

A. Related Work

In [5], the authors have studied elastic slice dimensioning with resource pricing as a Stackelberg pricing game, in which the network service provider (NSP) sells slices by pricing resources and network slice customers (NSCs) adjust their slice's resource demand on VNF capacity and bandwidth, while both are trying to maximize their profit.

In [6], the authors have deployed a cloud RAN environment and conducted two experiments on slicing resource management and cost optimization. In the first scenario, they have addressed slices' auto-scaling (Infrastructure Scale- Out/Scale-In) when free resources are available in the pool. In the second scenario, the authors have simulated slices' breathing (orchestration of resources) when the pool of resources is exhausted. Results show that leveraging cloud elasticity by auto-scaling resources saves costs by providing exactly "what-is-needed" "when-it-is- needed" in term of cloud computing. On the other hand, slices' breathing maximizes the usability by employing our "inhale-and-exhale" heuristic.

In [7], a novel methodology is proposed, in which a value chain in sliced networks has been presented. Based on the proposed value chain, the profits generated by different slices have been analyzed, and the task of network resource management has been modeled as a multi-objective optimization problem. Setting strong assumptions, this optimization problem has been analyzed starting from a simple ideal scenario. By removing the assumptions step-by-step, realistic but complex use cases have been approached. Through this progressive analysis, technical challenges in slice implementation and network optimization have been investigated under different scenarios.

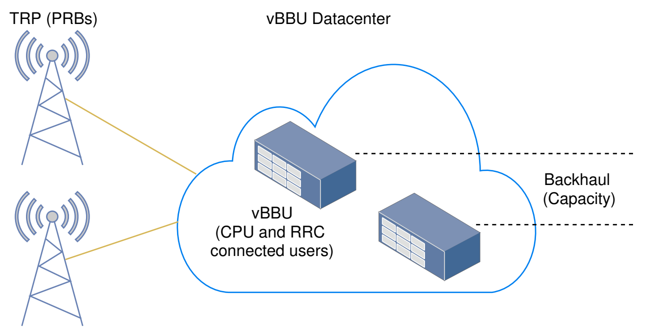


Figure 1: Cloud RAN architecture.

In [8], the authors have proposed a convolutional neural network (CNN) architecture to predict the traffic demand per slice while taking into account SLA violation cost. In this regard, we notice that this CNN strategy is of high complexity [9].

B. *Contributions*

In this paper, we investigate the following aspects:

- At each virtual function/interface of the cloud RAN, we build and train a joint multi-slice DNN model to estimate the resource provision based on the traffic per slice. In this regard, we invoke live network key performance indicators (KPIs) datasets involving end-to-end metrics such as traffic volume per slice, downlink (DL) physical resource blocks (PRBs), CPU load and RRC connected users' licenses at the virtual baseband units (vBBUs), and backhaul capacity.
- The designed models incorporate OPEX control through the integration of constraints on OPEX violation rate. Unlike existing online DNN optimization strategies, we introduce a new dataset-based training approach. It consists on imposing dataset-dependent custom non-convex constraints to the DNN output and using a two-player non-zero sum game strategy to solve the resulting offline optimization task. In this intent, the OPEX thresholds act as hyperparameters that can be fine-tuned by the infrastructure operator according to the SLAs with the slices' tenants. Note that we have adopted deep learning since it enables automatic discovery of important features from raw datasets, as well as yields generalized models, which is suitable for heterogeneous resources allocation.

II. Network Configuration and Datasets

A. *Network Configuration*

The collected KPIs correspond to an LTE-advanced (LTE-A) dense urban area, covered by 440 LTE-A eNodeBs (eNBs) and 3200 cells, including 800 MHz, 1800 MHz and 2.6 GHz bands. Table II summarizes the network configuration used throughout this paper, in particular to aggregate the traffic at the vBBU datacenter.

Table I: Network Configuration

Entity	Quantity
TRP	3200
eNB	440
BBU datacenters	10 uniformly distributed, with $\times 100$ CPU resources compared to a single 4G eNodeB

B. *Datasets*

Based on the architecture depicted in Fig. 1, the measured datasets are stemming from two network components. First, thanks to their deep inspection capabilities, dedicated probes—usually installed at the core network—are collecting and analyzing the traffic per OTT at a granularity of 1 hour for each TRP. The traffic is then aggregated at eNB and vBBU datacenter for each OTT. Once the slices are defined, the traffic of the underlying OTTs is summed to yield the traffic per slice. Second, the key performance indicators are collected by the operational support system (OSS) platform at TRP, eNB and vBBU levels. The KPIs have a granularity of 1 hour and are formatted as detailed in Table I. Note that we have used Huawei's PRS tool to export the OSS KPIs (e.g., PRB usage, CPU load...) and Netscout of Tektronix to get the probes' OTT KPIs.

III. End-to-End Resource Provisioning under OPEX Constraints

In this section, we build deep learning models to estimate the end-to-end required resources for each slice. Moreover, these models have to respect some OPEX constraints that are the subject of a contract between the infrastructure operator and the slices' tenants. In this regard, we consider for each slice n and RAN virtual network function $m \in \{\text{TRP}, \text{vBBU}, \text{Backhaul}\}$, a set of resources $r_{m,n,k}$ $(k = 1, \ldots, K)$. Examples of resources are the DL PRBs at TRP and the CPU load at the vBBU datacenter. For notation simplicity and without loss of generality, we adopt neural networks of similar depth L wherefore the input features, weights and biases are denoted by $\mathbf{s}_n$, $\mathbf{W}_n$ and $\mathbf{b}_n$, respectively, while $\ell(\cdot)$ and N_{B} stand for the squared error loss function and the batch size, respectively. In the sequel, we formulate the OPEX-constrained deep learning-based resource provisioning problem, and show how one can proceed to solve the underlying optimization problem.

Note that the resource provisioning DNN models are multi-slice, i.e., jointly trained using the N slices' traffics. During the test, however, the resources allocated to a given slice n are obtained by keeping only the features related to that slice, and setting those corresponding to the remaining slices to zero.

Table II: Datasets Features

TRP Feature	Description
`OTT Traffics per TRP`	Includes the hourly traffic for the top OTTs: Apple, Facebook, Facebook Messages, Facebook Video, Instagram, NetFlix, HTTPS, QUIC, Whatsapp, and Youtube
`CQI`	Channel quality indicator reflecting the average quality of the radio link of the TRP
`MIMO Full-Rank`	Usage of MIMO full-rank spatial multiplexing in %
`DLPRB`	Number of occupied downlink physical resource blocks
vBBU Feature	**Description**
`OTT Traffics per eNB`	Aggregated OTT traffics per eNB
`CPU Load`	CPU resource consumption in %
`RRC Connected Users`	Number of RRC users licenses consumed per eNB
Backhaul Feature	**Description**
`OTT Traffics per BBU datacenter`	Aggregated OTT traffics per BBU datacenter
`Backhaul capacity`	Effective aggregated throughput per BBU datacenter

A. Resource Pricing Model

Inspired by cloud resource pricing strategies presented in [10] and without loss of generality, we model the RAN resource pricing function as

$$\pi\left(r_{m,n,k}^{(i)}\right) = \gamma_{m,n,k} r_{m,n,k}^{(i)}, \tag{1}$$

where γ_k is the unitary price per resource k. This corresponds to a pay-per-use strategy, where the tenant pays only for the used resources. A practical example of such a payment scheme is *Amazon Web Services/Elastic Compute Cloud (EC2),* which charges the customer on the hourly usage of RAM and CPU [10].

B. DNNs with Dataset-dependent Constraints

Deep neural networks with dataset-dependent constraints are a novel concept. It considers problems that minimize DNN loss function subject to data-dependent constraints, expressed in terms of expectations over a data distribution $\mathcal{D}$:

$$\min_{\mathbf{W}} \mathbf{E}_{\mathbf{x}\sim\mathcal{D}}\, \ell_0\left(\mathbf{x}, \mathbf{W}\right), \tag{2a}$$

$$s.t.\, \mathbf{E}_{\mathbf{x}\sim\mathcal{D}}\, \ell_i\left(\mathbf{x}, \mathbf{W}\right) \leq 0,\, i = 1, \ldots, m, \tag{2b}$$

where $\mathbf{W}$ are the weights of the DNN, $\mathbf{x}$ are the features, while ℓ_0 and ℓ_i stand for the DNN loss function and the m constraints, respectively.

C. Offline Violation Rate-Based OPEX Enforcement

In this section, we adopt a novel offline approach to train dataset-based DNN models. It consists on directly enforcing an upper bound on the OPEX violation rate. In this case, the deep learning training amounts to solving the datatset-based constrained optimization task expressed as,

$$\min \frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \ell\left(r_{m,n,k}^{(i)}, \hat{r}_{m,n,k}^{(i)}\left(\mathbf{W}_n, \mathbf{b}_n, \mathbf{s}_n\right)\right), \tag{3a}$$

$$\text{s.t.}\, \mathbf{W}_{l,n} \in \mathbb{R}^{N_{l-1}\times N_l}, l = 1, \ldots, L+1, \tag{3b}$$

$$\mathbf{b}_{l,n} \in \mathbb{R}^{N_l \times 1}, l = 1, \ldots, L+1, \tag{3c}$$

$$\frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \mathbb{1}\left(\pi\left(\hat{r}_{m,n,k}^{(i)}\right) < \alpha_{m,n,k}\right) \leq \rho_{m,n,k}, \tag{3d}$$

$$\frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \mathbb{1}\left(\pi\left(\hat{r}_{m,n,k}^{(i)}\right) > \beta_{m,n,k}\right) \leq \rho_{m,n,k}, \tag{3e}$$

where $\mathbb{1}(\cdot)$ stands for the indicator function, and the constraint (3d) is imposing an upper bound $\rho_{m,n,k}$ on the OPEX violation rate, i.e., the probability that the price of allocated resource $\hat{r}_{m,n,k}$ is outside the interval $[\alpha_{m,n,k}, \beta_{m,n,k}]$.

The loss function $\ell(\cdot)$ is not a well-behaved function of $\mathbf{W}_n$ because of the deep neural network structure, resulting in non-convex objective and constraint functions. Worse, the violation rate constraint is a linear combination of indicators, hence is not even subdifferentiable w.r.t. $\mathbf{W}_n$. Fixing this issue by replacing the constraints with differentiable surrogates introduces a new difficulty: solutions to the resulting problem will satisfy the surrogate constraints, rather than the actual ones. To sidestep this blocking point, let us consider the functions Φ_1 and Φ_2 defined as,

$$\Phi_1(\mathbf{W}_n) = \frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \mathbb{1}\left(\pi\left(\hat{r}_{m,n,k}^{(i)}\right) < \alpha_{m,n,k}\right) - \rho_{m,n,k}, \tag{4}$$

$$\Phi_2(\mathbf{W}_n) = \frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \mathbb{1}\left(\pi\left(\hat{r}_{m,n,k}^{(i)}\right) > \beta_{m,n,k}\right) - \rho_{m,n,k}, \tag{5}$$

and let Ψ_1 and Ψ_2 be sufficiently-smooth approximations of Φ [11] verifying

$$\Psi_1\left(\mathbf{W}_n\right) = \frac{1}{N_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{B}}} \sigma\left(\alpha_{m,n,k}^{(i)} - \pi\left(\hat{r}_{m,n,k}\right)\right) - \rho_{m,n,k} \leq 0, \tag{6}$$

$$\Psi_2\left(\mathbf{W}_n\right)=\frac{1}{N_{\mathrm{B}}}\sum_{i=1}^{N_{\mathrm{B}}}\sigma\left(\pi\left(\hat{r}_{m,n,k}\right)-\beta_{m,n,k}\right)-\rho_{m,n,k}\leq 0, \quad (7)$$

where σ stands for the sigmoid function. The problem (3) can then be solved by invoking the so-called *proxy Lagrangian* framework [12]. This starts by forming two Lagrangians as follows:

$$\begin{aligned}\mathcal{L}_{\mathbf{W}_n}=&\frac{1}{N_{\mathrm{B}}}\sum_{i=1}^{N_{\mathrm{B}}}\ell\left(\mathbf{r}_{m,n,k}^{(i)},\hat{\mathbf{r}}_{m,n,k}^{(i)}\left(\mathbf{W}_n,\mathbf{b}_n,\mathbf{s}_n\right)\right)\\&+\lambda_1\Psi_1(\mathbf{W}_n)+\lambda_2\Psi_2(\mathbf{W}_n),\end{aligned} \quad (8a)$$

$$\mathcal{L}_{\lambda}=\lambda_1\Phi_1\left(\mathbf{W}_n\right)+\lambda_2\Phi_2\left(\mathbf{W}_n\right), \quad (8b)$$

where their optimization can be viewed as a non-zero-sum two-player game in which the $\mathbf{W}_n$-player wishes to minimize $\mathcal{L}_{\mathbf{W}_n}$, while the λ-player wishes to maximize $\mathcal{L}_{\lambda}$. Intuitively, the λ-player chooses how much to weigh the proxy constraint function, but does so in such a way as to satisfy the original constraint, and reach a nearly-optimal nearly-feasible solution to the original constrained problem. Note that $\lambda\leq R$, where R represents the maximum radius of Lagrange multipliers; introduced as a hyperparameter controlling the dependency to the constraints. In practice, we implement the deep learning objective function, the constraints (3d)-(3e) and the proxy constraints (6) and (7) on top of Google's `constrained optimization package` [13] that uses two different approaches to optimize the Lagrangians: a Bayesian optimization oracle for $\mathcal{L}_{\mathbf{W}_n}$ and projected gradient ascent for $\mathcal{L}_{\lambda}$. A definition of the oracle is given as follows:

Definition 1 (Approximate Bayesian Optimization Oracle)**.** *A δ-approximate Bayesian optimization oracle is a routine $\mathcal{O}_\delta$ that given a loss function/Lagrangian $\mathcal{L}$, returns the quasi-optimal weights $\mathbf{W}_n$ such that*

$$\mathcal{L}\left(\mathcal{O}_\delta\left(\mathcal{L}\right)\right)\leq\inf_{\mathbf{W}_n^\star}\mathcal{L}\left(\mathbf{W}_n^\star\right)+\delta. \quad (9)$$

IV. Numerical Results

A. Slices Scenario

For the sake of simplicity and without loss of generality, we consider three slices defined as follows:

- **eMBB:** involves NetFlix, Youtube and Facebook Video,
- **Social Media:** includes Facebook, Facebook Messages, Whatsapp and Instagram,
- **Browsing**: encompasses Apple, HTTP and QUIC.

Note that the traffic of a slice is the aggregation of the unitary traffics of the corresponding OTTs.

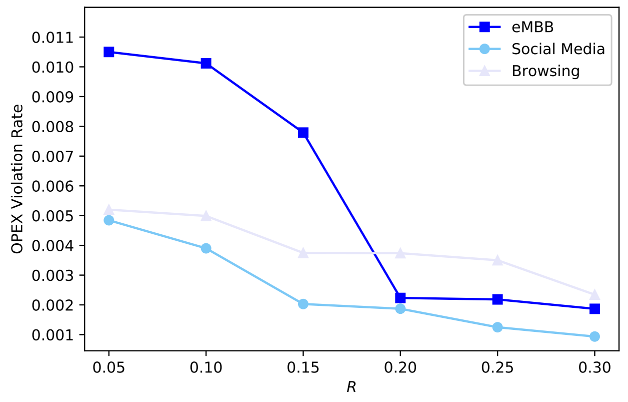


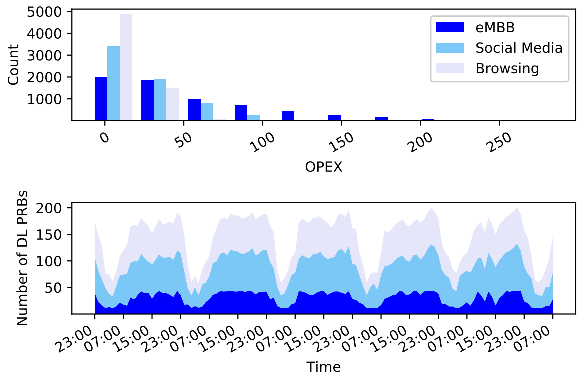
Figure 2: DL PRB OPEX violation rate vs. R with $\alpha=[0,0,0]$ and $\beta=[200,250,250]$ \$, $\gamma=[4,2,1]$, for target $\rho=0.005$.

B. Neural Network Settings

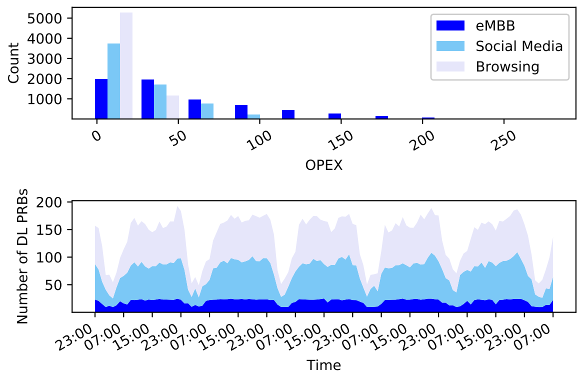
Throughout this paper, we consider deep neural networks of $L=2$ hidden layers with $N_1=16$ and $N_2=8$ neurons, respectively. We set the training epochs to 300 and the optimizer to `Adam` with learning rate 0.1. These parameters are set following extensive experiments and turn out to yield the best results. The training dataset size varies from one network function to another. Hence, at TRPs and vBBUs levels, $N_{\mathrm{TR}}=21417$ and $N_{\mathrm{TR}}=9681$ samples, respectively, with batch size $N_{\mathrm{B}}=100$. On the other hand, the test dataset at each network function consists of the hourly traffics of OTTs for a period of 5 days. The slices' traffics are obtained by aggregating the corresponding OTTs' traffics. In this work, we consider three slices, namely, enhanced mobile broadband (eMBB), Social Media and Browsing. In both training and test datasets, features normalization is activated. For the sake of simplicity, we drop the indexes m,n,k, and use vectors α, β and γ and scalar ρ instead. These vectors encompass the resource bounds corresponding to the different slices at a given network function, and can be easily understood from the context.

C. Effect of Lagrange Multiplier Radius on the Violation Rate

Lagrange multiplier radius $0<R<1$ controls the dependency to the constraints (3d) and (3e) and proxy constraints (6) and (7). With very low R, the deep learning becomes unconstrained, while with high R, the constraints are prioritized during the deep learning offline optimization over the dataset. As depicted in Fig. 2, we study the variation of the effective DL PRBs OPEX violation rate versus parameter R. The tolerated hourly OPEX of DL PRBs resources in \$ for eMBB, Social Media and Browsing slices are given by the upper-bound vector $\beta=[200,250,250]$ with unitary prices $\gamma=[4,2,1]$. As expected, the achieved violation rate is a decreasing function of R, and with values



(a) $R = 0.01$



(b) $R = 0.2$

Figure 3: DL PRBs evolution and OPEX distribution per slice, with $\alpha = [0, 0, 0]$ and $\beta = [200, 250, 250]$ \$, $\gamma = [4, 2, 1]$, $\rho = 0.005$.

of R as low as 0.3, we are able to reach a violation rate around 0.002, which is a good performance indicator to enable an efficient OPEX control between the slices tenants and the operator. To achieve the target violation rate $\rho = 0.005$ for the three considered slices, one should set $R = 0.2$. The violation rate presents the same behavior with respect to R for other RAN resources, i.e., vBBU CPU usage, vBBU RRC connected users and backhaul capacity. We therefore settle for Fig. 3 results for the sake of brevity. We set $R = 0.2$ in all the subsequent scenarios to achieve a violation rate as low as 0.005 for all slices.

D. Resource Evolution and OPEX Distribution

The DNN models, fed by the traffics of the three slices, yield an estimation of the required resources at each RAN entity. By imposing upper and lower bounds on the OPEX, and constraining their violations, the DNN model learns from the dataset while ensuring the respect of the OPEX constraints. This translates into a control of the allocated resources per slice, since the DNN model automatically increases or the decreases the amount of resources according to both the input traffic trend and the OPEX constraints. Note that the considered vBBUs can scale up to accommodate more traffic.

DL PRBs: In Fig. 3, we show the hourly evolution of the allocated DL PRBs over a period of 5 days, and the corresponding OPEX distribution. With $R = 0.2$, we remark for instance that the eMBB OPEX hourly upper-bound of 200\$ is respected as depicted in the histogram. This translates into a reduction of allocated PRBs compared to the case of $R = 0.01$ where the constraints are not fully respected and the OPEX violation rate is higher. On the other hand, we notice that the DL PRBs variation over time is induced by the trend of hourly traffics per slice that are fed to the DNN model. We also remark that most of PRB resources are dedicated to Social Media and Browsing slices, since they are viewed as massive access services.

vBBU CPU Usage: In Fig. 4, the vBBU CPU usage evolution and OPEX distribution are depicted. With $R = 0.2$, the OPEX bounds are respected as shown in the histogram. While the three slices are presenting quite similar CPU resource usage, they differ in the incurred hourly OPEX due to the difference in the unitary price γ. On the other hand, we notice that while eMBB slice generally presents the lowest traffic, it consumes similar CPU resources as the other slices. This is justified by the large processing requirements of eMBB service compared to Social Media or Browsing services. We also remark that during the quiet time (generally overnight, at e.g., $5 : 00$ h), the CPU usage does not decrease significantly for the three slices. This is due to the enforced OPEX lower bounds $\alpha = [30, 0, 0]$. Indeed, imposing a lower bound might be seen as ensuring an isolation between the different slices, where even during low traffic periods a slice is allocated a minimum number of resources. Note that the presented CPU consumption is with respect to a single vBBU instance that is processing the data of one eNB.

vBBU RRC Connected Users: As depicted in Fig. 5, the OPEX distribution of RRC connected users licenses corresponding to a single vBBU instance respect the imposed hourly upper-bounds $\beta = [400, 200, 100]$ \$ for the three slices. Due to the high unitary price of eMBB slice, the DNN model reduces its allocated licenses in order to keep the incurred OPEX violation rate below the target value $\rho = 0.005$. On the other hand, since Social Media slice is a massive access service, we notice that it presents the highest number of assigned RRC connected users licenses.

Backhaul Capacity: Fig. 6 shows the backhaul capacity evolution and OPEX distribution. With $R = 0.2$, the DNN model constraints are active, and the enforced OPEX upper-bounds $\beta = [2000, 1000, 500]$ \$ are respected for the different slices. We remark that while eMBB service is presenting the lowest number of users, it requires a backhaul capacity comparable to the other

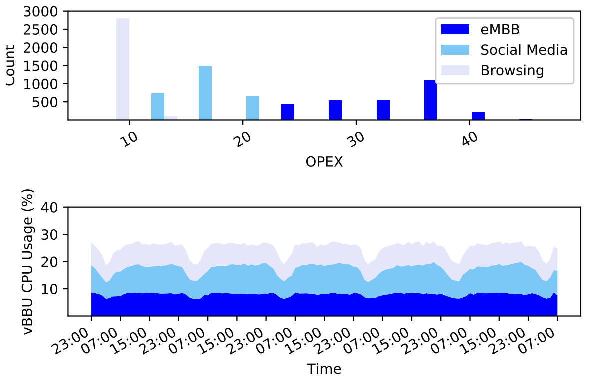


Figure 4: vBBU CPU usage and OPEX distribution per slice, with $\alpha = [30, 0, 0]$ and $\beta = [50, 50, 50]$ \$, $\gamma = [4, 2, 1]$, $\rho = 0.005$ and $R = 0.2$.

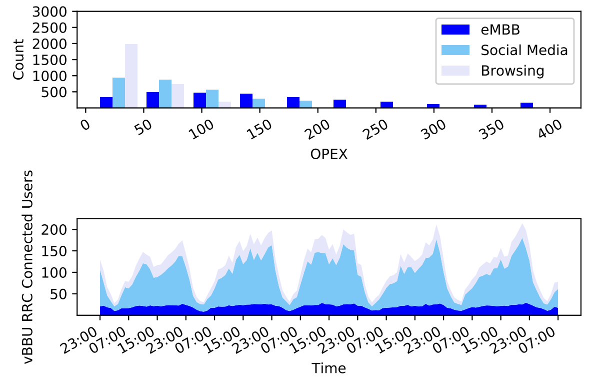


Figure 5: vBBU RRC connected users licenses and OPEX distribution per slice, with $\alpha = [0, 0, 0]$ and $\beta = [400, 200, 100]$ \$, $\gamma = [4, 2, 1]$, $\rho = 0.005$ and $R = 0.2$.

slices. This is due to the nature of the eMBB service that involves high throughput-demanding applications.

V. Conclusion

In this paper, we have presented new RAN slicing resource allocation schemes under OPEX limitations. By invoking key performance indicators datasets stemming from a live cellular network, the problem has been modeled using a new class of deep neural networks, where OPEX requirements have been formulated as dataset-dependent non-convex non-differentiable violation rate constraints. The designed constrained DNNs have then been optimized via a non-zero sum two-player game strategy. In this respect, we have highlighted the effect of the different hyperparameters on the respect of the OPEX limitations, while guaranteeing a dynamic RAN resource orchestration that follows the slices' traffics trends.

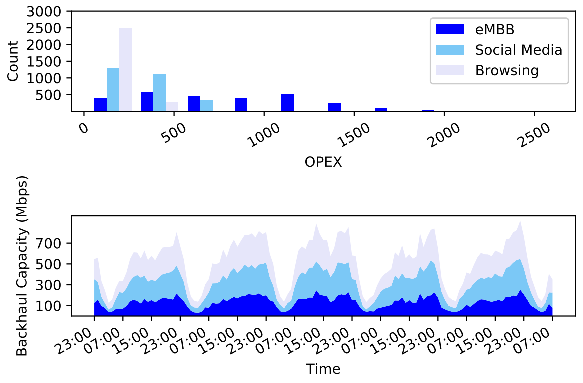


Figure 6: Backhaul capacity and OPEX distribution per slice, with $\alpha = [0, 0, 0]$ and $\beta = [2000, 1000, 500]$ \$, $\gamma = [5, 2, 1]$, $\rho = 0.005$ and $R = 0.2$.

VI. Acknowledgment

This work has been supported in part by the research projects CONNECT (737434), 5G-SOLUTIONS (856691), AGAUR (2017-SGR-891) and SPOT5G (TEC2017-87456-P).

References

[1] NGMN Alliance, "Description of network slicing concept," [Online]. Available: https://www.ngmn.org, accessed Mar. 2019.
[2] K. Samdanis, X. Costa-Perez and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," in *IEEE Communications Magazine,* vol. 54, no. 7, pp. 32-39, Jul. 2016.
[3] Y. Zaki *et al.,*"LTE wireless virtualization and spectrum management," in *Wireless and Mobile Networking Conference (WMNC), 2010,* Third Joint IFIP, pp. 1â6, Oct 2010.
[4] M. Jiang *et al.,* "Network slicing management and prioritization in 5G mobile systems," in *European Wireless EW 2016*, Oulu, Finland, 2016.
[5] G. Wang *et al.,* "Optimizing Network Slice Dimensioning via Resource Pricing," in *IEEE Access,* vol. 7, pp. 30331-30343, 2019.
[6] N. Salhab *et al.,* "Optimization of the implementation of network slicing in 5G RAN," in *IEEE Middle East and North Africa Communications Conference (MENACOMM),* Jounieh, 2018, pp. 1-6.
[7] B. Han, S. Tayade and H. D. Schotten, "Modeling profit of sliced 5G networks for advanced network resource management and slice implementation," in *IEEE Symposium on Computers and Communications (ISCC),* Heraklion, 2017, pp. 576-581.
[8] D. Bega *et al.,* "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *IEEE INFOCOM'2019,* Paris, France, May 2019.
[9] K. He and J. Sun, "Convolutional neural networks at constrained time cost," [Online]. Available: https://arxiv.org/pdf/1412.1710.pdf
[10] A. Mazrekaj, I. Shabani and B. Sejdiu, "Pricing schemes in cloud computing: An overview," in *International Journal of Advanced Computer Research,* vol. 7, no. 2, Feb. 2016.
[11] J.A.K. Suykens *et al.*, *Advances in Learning Theory: Methods, Models and Applications,* IOS Press, May 2003.
[12] A. Cotter *et al.,* "Training well-generalizing classifiers for fairness metrics and other data-dependent constraints" [Online]. Available: arxiv.org/abs/1807.00028.
[13] A. Cotter *et al.,* Constrained Optimization (TFCO). [Online]. Available: https://codeload.github.com/google-research/tensorflow_constrained_optimization/zip/master