# DeepCog: Optimizing Resource Provisioning in Network Slicing with AI-based Capacity Forecasting

Dario Bega*†, Marco Gramaglia†, Marco Fiore‡ Albert Banchs*† and Xavier Costa-Perez§

*IMDEA Networks Institute, Madrid, Spain

†University Carlos III of Madrid, Madrid, Spain

‡CNR-IEIIT, Italy

§NEC Laboratories Europe, Heidelberg, Germany

**Abstract**

The dynamic management of network resources is both a critical and challenging task in upcoming multi-tenant mobile networks, which requires allocating capacity to individual network slices so as to accommodate future time-varying service demands. Such an anticipatory resource configuration process must be driven by suitable predictors that take into account the monetary cost associated to overprovisioning or underprovisioning of networking capacity, computational power, memory, or storage. Legacy models that aim at forecasting traffic demands fail to capture these key economic aspects of network operation. To close this gap, we present DeepCog, a deep neural network architecture inspired by advances in image processing and trained via a dedicated loss function. Unlike traditional traffic volume predictors, DeepCog returns a cost-aware *capacity forecast*, which can be directly used by operators to take short- and long-term reallocation decisions that maximize their revenues. Extensive performance evaluations with real-world measurement data collected in a metropolitan-scale operational mobile network demonstrate the effectiveness of our proposed solution, which can reduce resource management costs by over 50% in practical case studies.

**Index Terms**

Mobile networks, network slicing, 5G networks, artificial intelligence, deep learning.

## I. Introduction

Network slicing is an emerging paradigm that is expected to characterize 5G and beyond-5G mobile systems. Network slicing allows operators to tailor Virtual Network Functions (VNFs)

to the precise requirements of individual mobile services [1], and will be key in enabling the unprecedented heterogeneity of future mobile applications [2]. Managing sliced networks will represent a major challenge for operators, since this new paradigm represents a shift from the rather limited reconfiguration possibilities offered by current Operations and Business Support System (OSS/BSS) to a complex, software-defined control layer that must dynamically organize thousands of slices belonging to hundreds of tenants on the same infrastructure [3].

### *A. Network management and capacity forecast.*

To rise to the challenge, network operators must introduce substantial automation in the presently human-driven management and orchestration (MANO) processes, ultimately realizing the 5G principle of *cognitive network management* [4]. Achieving this objective requires advances in two complementary technologies: $(i)$ technical solutions that enable end-to-end Network Function Virtualization (NFV), providing the flexibility necessary for resource reallocation; and, $(ii)$ a network intelligence that automatically identifies and anticipates demand patterns from streaming network measurement data, and then takes decisions on how to configure VNF and allocate resources so as to maximize the system efficiency.

From a technical standpoint, solutions that implement NFV at different network levels are well established, and start to be tested and deployed. Examples include current MANO platforms architectures like ETSI NFV [5], and implementations of MANO controllers such as OSM [6] or ONAP [7] that support reconfiguring resources to VNFs on the fly. By contrast, the integration of intelligence in cognitive mobile networks is still at an early stage. Nowadays, resource assignment to VNFs is a reactive process, mostly based on hysteresis thresholding and aimed at self-healing or fault tolerance. There is a need for proactive, data-driven, automated solutions that enable cost-efficient network resource utilization, by anticipating future needs for capacity and timely reallocating resources just where and when they are required.

The aim of our work is precisely to contribute to the definition of a network intelligence that is adapted to a network slicing environment. More specifically, the focus of this paper is on the design of data analytics that enable the anticipatory allocation of resources in cognitive mobile networks. To this end, we investigate a machine learning solution that runs on traffic measurement data and provides operators with information about the capacity needed to accommodate future demands at each network slice – a critical knowledge for data-driven resource orchestration. We

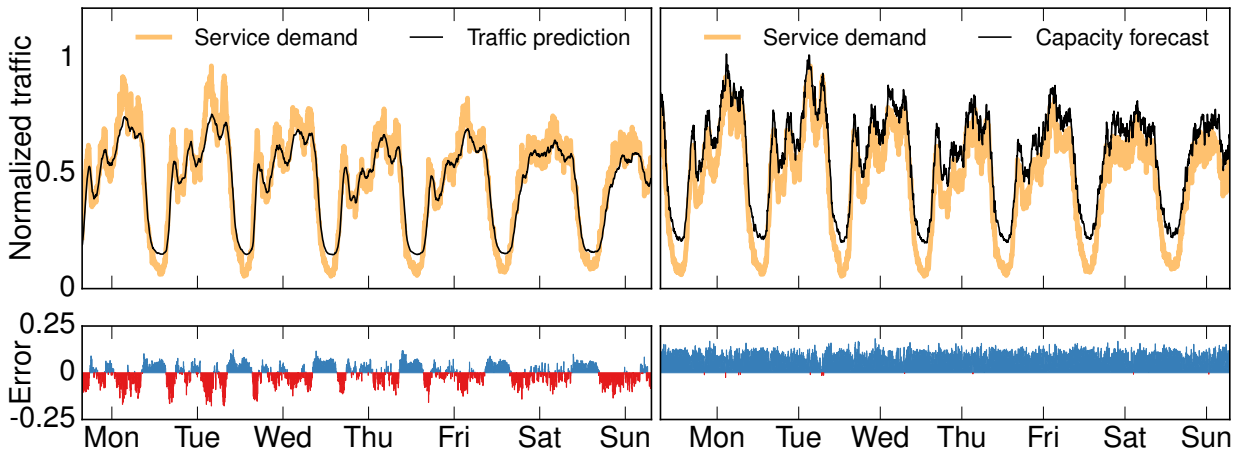
take a pragmatic approach, and duly account for the economic costs associated to the operation above.

Indeed, resource orchestration decisions have a direct monetary impact for the network operator in terms of operating expenses, which can be divided into two macroscopic categories of cost.

- *Overprovisioning* – when providing excess capacity with respect to the actual resource demand, the operator incurs a cost due to the fact that it is reserving more resources than those needed to a network entity (*e.g.*, a network slice, a network function, or a virtual machine). As resources are typically isolated across slices, this seizes the excess resources from other network entities that may have possibly used them. At a global system level, continued overprovisioning implies that the operator will have to deploy more resources than those required to accommodate the user demand, limiting the advantage of a virtualized infrastructure and of cognitive networking solutions in general.
- *SLA violation* – if insufficient resources are allocated to a network entity, users will suffer low Quality of Service (QoS), or even discontinued service. This has an indirect price for the operator, in terms of customer dissatisfaction and increased churning rates, which is not simple to quantify. However, in emerging contexts such as those promoted by network slicing, underprovisioning also entails a different, more direct and quantifiable economic penalty for the operator. Under slicing, operators will sign Service Level Agreements (SLAs) with the mobile service providers, which need to be strictly enforced. Underprovisioning means violating such SLAs, which results in substantial monetary fees for the network operator.

Clearly, the cost is not the same in the two cases, and it may also vary depending on the specific settings, including the nature of the concerned resources, the technologies deployed in the network infrastructure, or the market strategies of the operator. In all cases, we posit that, once suitably modeled, such costs shall be at the core of the orchestrating decisions.

Legacy techniques for the prediction of mobile network traffic, such as the one reviewed in Section II, fall short in this respect. Such models aim at perfectly matching the temporal behaviour of traffic, independently of whether the anticipated demand is above or below the target, and are thus agnostic of the aforementioned costs. As a result, they return forecasts as that depicted in Fig. 1a, which refers to a real-world case study of YouTube video streaming traffic at a core network datacenter. Note that no distinction is made between positive and negative erros, which leads to substantial SLA violations covering roughly half of the observation time. The operator




(a) Legacy traffic prediction

(b) DeepCog capacity forecast

Fig. 1: Top: actual and predicted weekly demands for YouTube at a datacenter controlling 470 4G eNodeBs. Bottom: levels of overprovisioning (blue) and capacity violations (red) over time. (a) Output of a recent deep learning predictor of mobile traffic [8]. (b) Output of DeepCog, tailored to anticipatory network resource allocation. Figure best viewed in colors.

may then attempt to apply overprovisioning to the output provided by such a traffic predictor. Unfortunately, legacy forecast models do not offer any insight on how large the excess resource allocated on top of the forecast demand should be. As we will demonstrate later in the paper, this makes such a strategy highly inefficient.

### *B. Paper contributions and data-driven setup*

In this paper, we present DeepCog, a new mobile traffic data analytics tool that is explicitly tailored to solve the capacity forecast problem exposed above. The design of DeepCog yields multiple novelties, summarized as follows:

- It hinges on a deep learning architecture inspired by recent advances in image and video processing, which exploits space- and time-independent correlations typical of mobile traffic and computes outputs at a datacenter level;
- It leverages a customized loss function that targets capacity forecast rather than plain mobile traffic prediction, letting the operator tune the balance between overprovisioning and demand violations;
- It provides long-term forecasts over configurable prediction horizons, operating on a per-service basis in accordance with network slicing requirements.

Overall, these design principles jointly solve the problem of capacity forecast in network slicing. This is illustrated by Fig.1b, which shows an example of the required capacity forecast by DeepCog in a real-world case study. We remark that DeepCog is one of the very first examples of rigorous integration of machine learning into a cognitive network management process, and marks a difference from the common practice of embedding vanilla deep learning structures into network operation [9]. Extensive performance evaluations with substantial measurement data collected in an operational metropolitan-scale mobile network demonstrate the superiority of our approach over a wide range of benchmarks based on traditional and state-of-the-art mobile traffic predictors.

The document is organized as follows. We first provide a review of related works, and highlight the novelty of our proposed method, in Section II. We then outline the overall framework of DeepCog in Section III, and detail the design of its most critical component, *i.e.*, the loss function, in Section IV. The quality of the solution is then assessed in realistic scenarios in Section V. Finally, we draw conclusions in Section VI.

## II. Related Work

Applications to networking problems of machine learning in general, and of deep learning in particular, are starting to become popular. Artificial intelligence can indeed be applied to solve many different problems that emerge in computer networks, as highlighted in recent comprehensive surveys on the topic [9], [10].

In the context of network management, emerging paradigms like slicing increase substantially the complexity of orchestrating network functions and resources, at all levels. For instance, intelligence is needed for the admission control of new slices: as resources are limited and slicing entails their strong isolation, this is critical to ensure that the system operates efficiently. With potentially hundreds of slices allocated simultaneously, and a need to anticipate highly profitable future requests, the decision space for admission control becomes so large that traditional approaches become impractical. Solutions based on deep learning architectures represent here a viable approach [11]. Similar considerations apply to other aspects of sliced network management, *e.g.*, the allocation of computational resources to slices at the radio access, based on transmission (*e.g.*, modulation and coding scheme, channel load) and environmental (*e.g.*, signal quality, hardware technology) conditions [12], or the anticipatory reservation of Physical Resource Blocks (PRBs) to user traffic to be served in target network slices [13].

Our specific problem relates to the orchestration of generic resources (*e.g.*, CPU time, memory, storage, spectrum) to slices at different network entities, which is tightly linked to mobile traffic prediction. The literature on forecasting network traffic is in fact vast [9], [14]. Solutions to anticipate future offered loads in mobile networks have employed a variety of tools, from autoregressive models [15]–[17] to information theoretical tools [18], passing by Markovian models [19] and deep learning [8], [11], [13], [20], [21]. However, we identify the following major limitations of current predictors when it comes to supporting resource orchestration in mobile networks.

First, predictors of mobile traffic invariably focus on providing forecasts of the future demands that minimize some absolute error [9], [14]. This approach leads to predicted time series that deviate as little as possible from the actual traffic time series, as exemplified in Fig. 1a for a real-world case study. While reasonable for many applications, such an output is not appropriate for network resource orchestration. As explained in Section I, the operator aims at provisioning sufficient capacity to accommodate the offered load at *all* times, since failing to do so implies high costs in terms of high subscribers' churn rates, as well as significant fees for violating SLAs signed with tenants. Yet, if an operator decided to allocate resources based on a legacy prediction like that in Fig. 1a, it would incur into capacity violations most of the time (as illustrated in the bottom subplot).

Second, with the adoption of network slicing, forecasts must occur at the slice level, *i.e.*, for specific mobile services in isolation. However, most traffic predictors are evaluated with demands aggregated over all services – an easier problem, since aggregate traffic yields smoother and more regular dynamics – and may not handle well the bursty, diversified traffic exhibited by each service. The only attempts at anticipating the demands generated by specific mobile services have been made by using multiple-input single-output (MISO) autoregressive models [22], and hybrid prediction methods that incorporate $\alpha$-stable models and sparsity with dictionary learning [18].

Third, existing machine learning predictors for mobile traffic typically operate at base station level [8], [21]. However, NFV operations mainly occur at datacenters controlling tens (*e.g.*, at the mobile edge) to thousands (*e.g.*, in the network core) of base stations. Here, prediction should be more efficient when performed on the aggregate traffic at each datacenter, where orchestration decisions are taken, rather than combining independent forecasts from each base station.

Our proposed solution, DeepCog, addresses all of the open problems above, by implementing a first-of-its-kind predictor that anticipates the minimum provisioned capacity needed to cut down

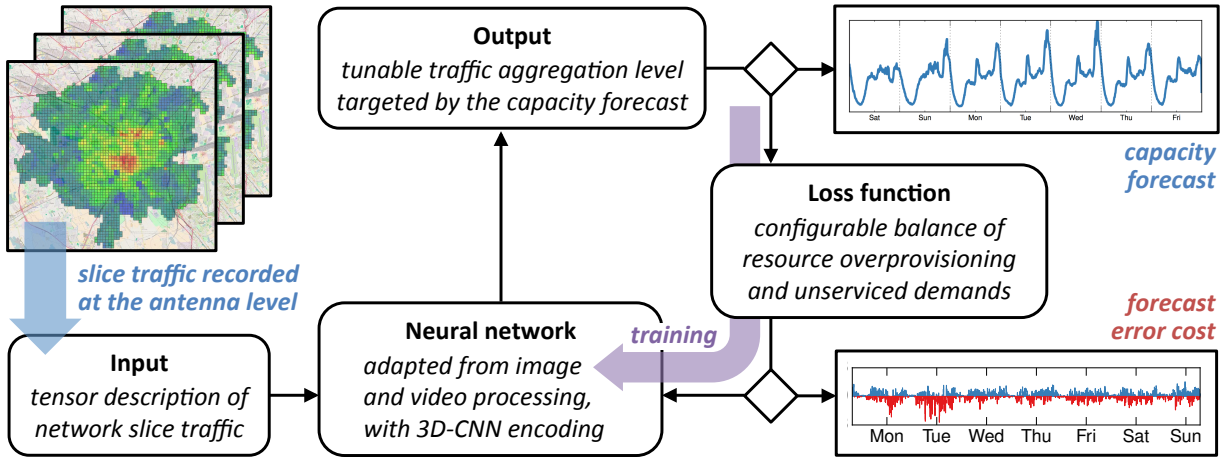



Fig. 2: Outline and interaction of the DeepCog components.

SLA violations. This closes the present gap between traffic prediction and practical orchestration, as it provides the operator with an explicit capacity forecast that mitigates underprovisioning in Fig. 1b while minimizing unnecessary resource reservation. We remark that early versions of the DeepCog framework were presented in [23] and [24]. Those preliminary variants of our solution could achieve short-term capacity forecasting over the next time step, whereas the complete version presented in this paper supports long-term capacity prediction over a configurable number of future time steps.

## III. A Deep Learning Framework for Resource Orchestration

The design of DeepCog is outlined in Fig. 2. Its organization is that typical of deep learning systems, and it stems from $(i)$ properly formatted *input* data used to build the forecast, which, in our case, represents the current and past traffic associated to a specific network slice as a tensor. Such input is fed to $(ii)$ a *deep neural network* architecture that extrapolates and processes input features to provide $(iii)$ an *output* value: the capacity forecast. During the training phase, the output is used to evaluate $(iv)$ a *loss function* that quantifies the error with respect to the ground truth, and, in DeepCog, accounts for the costs associated to resource overprovisioning and service request denial.

In our network model, we consider that time is divided in slots, which we denote by $t$. Let $\delta_s^i(t)$ be the traffic associated with slice $s$ that is observed at base station $i \in \mathcal{N}$ and time $t$. A *snapshot* of the demand of slice $s$ at time $t$ is given by a set $\boldsymbol{\delta}_s(t) = \{\delta_s^1(t), \ldots, \delta_s^N(t)\}$, and provides a

global view of the traffic for that slice at time $t$ across the whole network. We let $\mathcal{N}$ denote the set of $N$ base stations in the network, and $\mathcal{M}$ the set of $M < N$ datacenters. Base stations are associated to datacenters via a surjective mapping $f : \mathcal{N} \rightarrow \mathcal{M}$, such that a datacenter $j \in \mathcal{M}$ serves the aggregated load of all of the associated bases stations, *i.e.*, $d_s^j(t) = \sum_{i|f(i)=j} \delta_s^i(t)$ for slice $s$ at time $t$. The set of demands across all datacenters is then given by $\mathbf{d}_s(t) = \{d_s^1(t), \ldots, d_s^M(t)\}$. Let us denote the allocated capacity for slice $s$ at datacenter $j$ and time $t$ as $c_s^j(t)$, and the set of capacities at all $j \in \mathcal{M}$ as $\mathbf{c}_s(t) = \{c_s^1(t), \ldots, c_s^M(t)\}$. Then, the capacity forecast problem is that of computing a *constant* capacity $\bar{\mathbf{c}}_s(t, T_h) = \{\bar{c}_s^1(t, T_h), \ldots, \bar{c}_s^M(t, T_h)\}$ that is allocated in the network datacenters over a time horizon $T_h$, *i.e.*, through an interval between the present time $t$ and a future time $t + T_h$. In practice, this models the typical situation where the resource reconfiguration frequency is limited (*e.g.*, by the NFV technology), and the operator must decide in advance the amount of resources that will stay assigned to a slice until the next reallocation takes place. The time horizon $T_h$ thus corresponds to the reconfiguration period, and the allocated capacity is such that $c_s^j(t) = \bar{c}_s^j(t, T_h)\ \forall j \in \mathcal{M}, t \in [t, t + T_h]$.

The forecast builds on knowledge of the previous $T_p$ traffic snapshots $\boldsymbol{\delta}_s(t-1), \ldots, \boldsymbol{\delta}_s(t-T_p)$. The quality of the capacity forecast $\bar{\mathbf{c}}_s(t, T_h)$ is measured by means of a suitable loss function $\ell\left(\bar{\mathbf{c}}_s(t, T_h), \mathbf{d}_s(t), \ldots, \mathbf{d}_s(t + T_h)\right)$. This function $\ell(\cdot)$ determines the compound cost of over-provisioning and underprovisioning network resources at the target datacenters, as produced by allocating a constant capacity $\bar{\mathbf{c}}_s(t, T_h)$ when the actual time-varying demand is in fact $\mathbf{d}_s(t), \ldots, \mathbf{d}_s(t + T_h)$.

Below, we present each of the components of the framework, and discuss its mapping to the elements of a 5G network architecture running cognitive resource management.

### *A. The Neural Network*

DeepCog leverages a deep neural network structure composed of suitably designed encoding and decoding phases, performing a capacity forecasting prediction over a given time horizon. The structure is general enough that it can be trained to solve the capacity forecast problem for $(i)$ network slices dedicated to different services with significantly diverse demand patterns, $(ii)$ any datacenter configuration, and $(iii)$ any time horizon $T_h$.

The design of the neural network structure in DeepCog is inspired by recent breakthroughs [25] in deep learning for image and video processing. As summarized in Fig. 3, the network is composed of an encoder that receives an input representing the mobile traffic data $\boldsymbol{\delta}_s(t -$

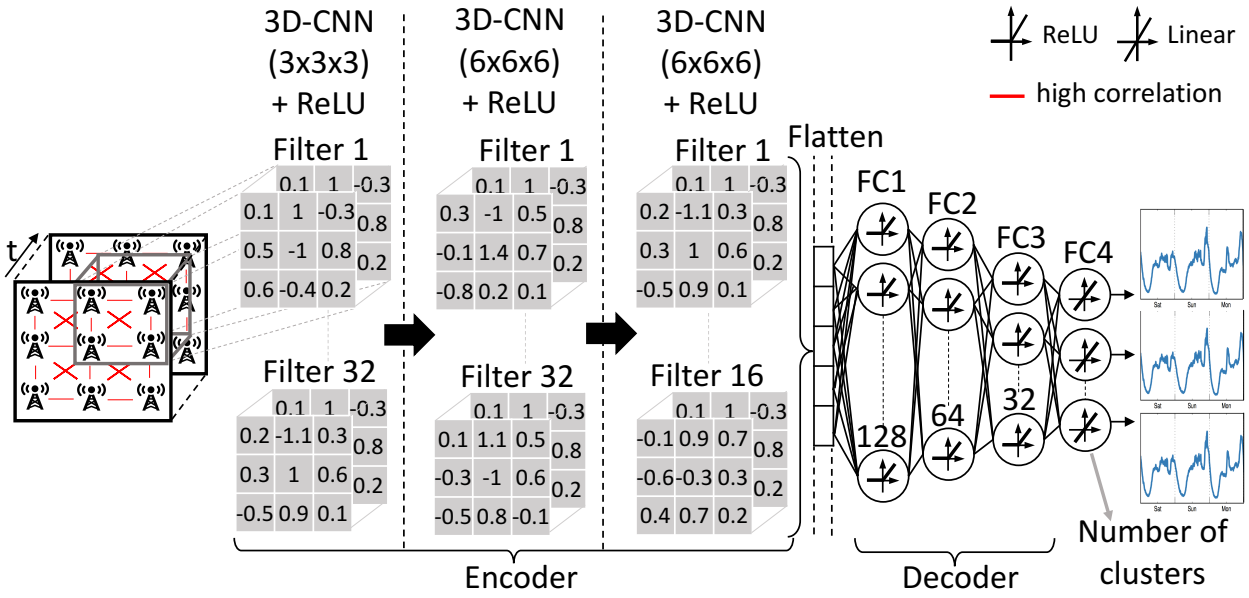



Fig. 3: DeepCog neural network encoder-decoder structure.

$1), \ldots, \boldsymbol{\delta}_s(t - Tp)$ and maps important spatial and temporal patterns in such data onto a low-dimensional representation. The result of the encoder undergoes a flattening process that converts the 3D (space and time) tensor data into a unidimensional vector format. This is the input format required by the fully connected layers that form the decoder, which then generates the final capacity forecast $\bar{\mathbf{c}}_s(t, T_h)$ at the target set of datacenters $\mathcal{M}$. Below, we detail the encoder and decoder implementations, and discuss the training procedure.

*1) The Encoder:* The encoder is composed by a stack of three three-dimensional Convolutional Neural Network (3D-CNN) layers [26]. Generic Convolutional Neural Networks (CNNs) are a specialized kind of deep learning structure that can infer local patterns in the feature space of a matrix input. In particular, two-dimensional CNNs (2D-CNNs) have been extensively utilized in image processing, where they can complete complex tasks on pixel matrices such as face recognition or image quality assessment [27]. 3D-CNNs extend 2D-CNNs to the case where the features to be learned are spatiotemporal in nature, which adds the time dimension to the problem and transforms the input into a 3D-tensor. Since mobile network traffic exhibits correlated patterns in both space and time, our encoder employs 3D-CNN layers.

Formally, the 3D-CNN layers receive a tensor input $\mathcal{T}\left(\boldsymbol{\delta}_s(t-1)\right), \ldots, \mathcal{T}\left(\boldsymbol{\delta}_s(t-T_p)\right)$, where $\mathcal{T}(\cdot)$ is a transformation of the argument snapshot into a matrix. This input is processed by three subsequent 3D-CNN layers. Each neuron of these layers runs a filter $\mathcal{H}\left(\sum_\tau \mathbf{I}(\tau) * \mathbf{K}(\tau) + \mathbf{b}\right)$

where $\mathbf{I}(\tau)$ is the input matrix passed to the neuron (*e.g.*, $\mathbf{I}(\tau) = \mathcal{T}(\boldsymbol{\delta}_s(\tau))$ at the very first layer, for slice $s$ and generic time $\tau$), $*$ denotes the 3D convolution operator, $\mathbf{K}(t)$ is the kernel of filters, $\mathcal{H}(\cdot)$ is a non-linear activation function, and $\mathbf{b}$ is a bias vector. We use two different kernel configurations $\mathbf{K}(\tau)$, as shown in Fig. 3: a $3 \times 3 \times 3$ kernel for the first 3D-CNN layer, and a $6 \times 6 \times 6$ kernel for the second and third layers. These settings allow limiting the *receptive field*, *i.e.*, the portion of input analyzed by each neuron, to small regions: in presence of strong local correlation of the input data, this approach is known to yield good performance with fairly limited training, in particular compared to Recurrent Neural Networks. As for the choice of the activation function, many different options have been proposed in the literature, spanning from linear functions to $\tanh$, sigmoid or Rectified Linear Unit (ReLU). Among these, we select ReLU, and set $\mathcal{H}(\mathbf{x}) = \max(0, \mathbf{x})$, which provides advantages in terms of discriminating performance and faster learning [28]. Finally, $\mathbf{b}$ is randomly set at the beginning of each training phase.

The second and third 3D-CNN layers are interleaved with Dropout layers: such layers regularize the neural network and reduce overfitting [28] by randomly setting to zero a number of output features from the preceding layer during the training phase. The *dropout rate* defines the probability with which output features undergo this effect. During training, we employ two Dropout layers with dropout rate equal to 0.3.

*2) The decoder:* The decoder uses Multi-Layer Perceptrons (MLPs) [29], a kind of fully-connected neural layers, where every neuron of one layer is connected to every neuron of the next layer. This provides the ability to solve complex function approximation problems. In particular, MLPs are able to learn global patterns in their input feature space [30]. In our structure, each layer performs an operation $\mathcal{H}'(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})$, where $\mathbf{x}$ is the MLP layer input vector, $\mathbf{W}$ a weight matrix related to the neurons of each layer, and $\mathbf{b}$ the bias vector. $\mathbf{W}$ plays a similar role to $\mathbf{K}(t)$ in the encoder part: its values drive the prediction through the layers of the decoding part.

As for the activation functions $\mathcal{H}$, we employ ReLU for all MLP layers except for the last one, where a linear activation function is used since the desired output takes real values. The last linear layer can be configured to produce multiple predictions in parallel, each matching the aggregate capacity required by a subset of base stations, thus allowing to forecast the needed capacity for different datacenters comprising a subset of base stations. Ultimately, this organization makes the DeepCog neural network capable of predicting per-slice capacity requirements at datacenter level, in a way that can adapt to any configuration of $\mathcal{M}$ and to any time horizon $T_h$.

*3) The training procedure:* We leverage the popular Adam optimizer, which is a Stochastic Gradient Descent (SGD) method that provides faster convergence compared to other techniques [31]. SGD trains the neural network model, evaluating at each iteration the loss function $\ell(\cdot)$ between the forecast and the ground truth, and tuning the model parameters in order to minimize $\ell(\cdot)$. For the configuration of the Adam optimizer, we use the default configuration with a learning rate of $5 \times 10^{-4}$.

An important element that concerns the training of the DeepCog architecture is that the encoder and the decoder described in Section III-A1 and Section III-A2 have independent roles. The encoder extracts the relevant features from the input traffic tensors $\boldsymbol{\delta}_s(t-1), \ldots, \boldsymbol{\delta}_s(t-Tp)$; the decoder leverage such features to generate a capacity forecast that is tailored to a given combination of slice, prediction time horizon, and datacenter class (*e.g.*, datacenters deployed close to the radio access versus datacenters co-located with the Internet gateways). Therefore, while the decoder heavily depends on the forecast specifications, the encoder does not, and is agnostic to the final usage of the extracted features. This fact allows adopting a *transfer learning* approach during training: instead of treating the two blocks as a whole (and performing the training over the full system for all the possible slices, datacenter classes and horizons), we can train them separately. Specifically, an horizon-independent encoder can be trained on past traffic tensors at maximum time granularity, and then reused in combination with dedicated decoders tailored to each $T_h$ value. Beside reducing the training time, this strategy reduces the need for neural-network-wide training to different settings of slice and datacenter only.

### *B. Arrangement of input data*

The input is composed by measurement data generated in a specific network slice, and recorded by dedicated probes deployed within the network infrastructure. Depending on the type and location of the probe, the nature of the measurement data may vary, describing the demands in terms of, *e.g.*, signal quality, occupied resource blocks, bytes of traffic, or computational load on VNFs. DeepCog leverages a set of transformations to map any type of slice traffic measurements into a tensor format that can be processed by the learning algorithm.

The 3D-CNN layer adopted as the first stage of the decoder requires a multidimensional tensor input. We thus need to define the transformation $\mathcal{T}(\cdot)$ of each traffic snapshot into a matrix. Note that 3D-CNN layers best perform in presence of a tensor input that features a high level of local correlation, so that neurons operate on similar values. In image processing, where close-by pixels

typically have high correlation, this is easily solved by treating the pixel grid as a matrix. In line with this strategy, the current common practice in mobile network traffic prediction is to leverage the geographical locations of the base stations, and assign them to the matrix elements so that their spatial proximity is preserved as much as possible [8], [9]. However, this approach does not consider that correlations in mobile service demands at a base station level do not to depend on space, rather on land use [32]: base stations exhibiting strongly correlated network slice traffic may be far apart, *e.g.*, covering the different train stations within a same large city. Thus, we aim at creating a tensor input whose neighboring elements correspond to base stations with strongly correlated mobile service demands. To this end, we construct the mapping of base stations into a matrix structure as follows.

- For each base station $i$, we define its historical time series of total traffic as $\boldsymbol{\tau}^i = \{\delta^i(1), \ldots, \delta^i(t-1)\}$, where $\delta^i(t) = \sum_s \delta^i_s(t)$. Then, for each pair $i$ and $j$, we determine the similarity of their recorded demands by computing $\text{SBD}^{ij} = f_{\text{SBD}}(\boldsymbol{\tau}^i, \boldsymbol{\tau}^j)$, where $f_{\text{SBD}}(\cdot)$ is the shape-based distance, a state-of-the-art similarity measure for time series [33]. All pairwise distances are then stored in a distance matrix $\mathbf{D} = (\text{SBD}^{ij}) \in \mathbb{R}^{M \times M}$.
- We compute virtual bidimensional coordinates $\mathbf{p}_i$ for each base station $i$ so that the values in the distance matrix $\mathbf{D}$ are respected as much as possible. Formally, this maps to an optimization problem whose objective is $\min_{x_1,\ldots,x_M} \sum_{i<j} (\|\mathbf{p}_i - \mathbf{p}_j\| - \text{SBD}^{ij})^2$, efficiently solved via Multi-Dimensional Scaling (MDS) [34].
- We match each point $\mathbf{p}_i$ to an element $e$ of the input matrix $\mathbf{I}$, again minimizing the total displacement. To this end, we: $(i)$ quantize the virtual surface encompassing all points $\mathbf{p}_i$ so that it results into a regular grid of $N$ cells; $(ii)$ assume that each cell is an element of the input matrix; $(iii)$ compute the cost $k_{ie}$ of assigning a point $\mathbf{p}_i$ to element $e$ as the Euclidean distance between the point and the cell corresponding to $e$. We then formalize an assignment problem with objective $\min_a \sum_{i \in \mathcal{N}} \sum_{e \in \mathbf{I}} k_{ie} x_{ie}$, where $x_{ie} \in [0, 1]$ is a decision variable that takes value 1 if point $\mathbf{p}_i$ is assigned to element $e$, and must fulfill $\sum_{i \in \mathcal{N}} x_{ie} = 1$ and $\sum_{e \in \mathbf{I}} x_{ie} = 1$. The problem is solved in polynomial time by the Hungarian algorithm [35].

The solution of the assignment problem is the transformation $\mathcal{T}(\cdot)$ of the original base stations into elements of the matrix $\mathbf{I}$. The mapping function $\mathcal{T}(\cdot)$ allows translating a traffic snapshot $\boldsymbol{\delta}_s(t)$ into matricial form. Applying this to snapshots at different times, $\boldsymbol{\delta}_s(t-1), \ldots, \boldsymbol{\delta}_s(t-T)$, we can thus build the tensor required by the entry encoder layer in Fig. 3.

### C. The Output function

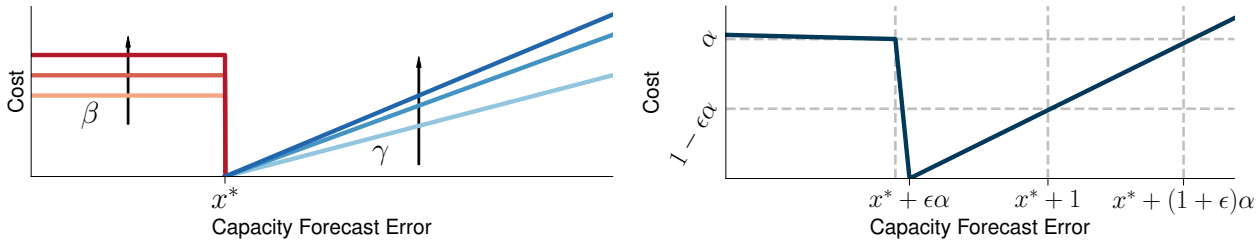
DeepCog is designed for flexibility, and can be used for different orchestration scenarios. This is achieved thanks to an adaptable last layer of the deep neural network, and a configurable loss function. In general, the learning algorithm returns a forecast of the capacity required to accommodate the future demands for services associated to a specific network slice. This generic definition of output can then be applied to different orchestration use cases that may differ in the traffic aggregation level at which the resource configuration takes place, and/or in the frequency at which resource reallocation can be realized.

For instance, the anticipatory assignment of baseband processing units to network slices in a Cloud Radio Access Network (C-RAN) datacenter requires a prediction of the capacity needed to accommodate the traffic of a few tens of base stations; instead, reserving memory resources for a specific network slice at a core network datacenter implies forecasting capacity for the data sessions of subscribers associated to hundreds of base stations. The output format of DeepCog can accommodate any datacenter layout, by tailoring the last linear layer of the neural networks to the specific requirements of the layout (as discussed in Section III-A2).

Also, as discussed previously, the time horizon over which the forecast is performed is another relevant system parameter, which depends on NFV technology limitations and current trends in *commoditization* of softwarized mobile network. When the technology limitations does not allow frequent reconfiguration opportunities, resources need to be allocated over long periods, e.g., of tens of minutes or even hours. In this case, forecasting over long-term horizons provides the operator with information on the constant capacity to be allocated during long intervals. To realize this, DeepCog operates on configurable time horizons, thanks to the flexible loss function that we will discuss next.

## IV. $\alpha$-OMC

One of the key components of the system proposed in the previous section is the *loss function*, denoted by $\ell(\cdot)$. This function determines the penalty incurred when making a prediction error. In this paper, we propose a novel loss function that is tailored to the specific requirements of the capacity forecast problem. Our design of $\ell(\cdot)$ accounts for the costs resulting from $(i)$ forecasting a lower value than the actual offered load, which leads to an *SLA violation* due to the provisioning of insufficient resources, $(ii)$ predicting a higher value than the actual one, which leads to *overprovisioning*, allocating more resources than those needed to meet the demand. In




(a) Ideal Model

(b) Actual Implementation

Fig. 4: Cost model $\ell'(c_s^j(t) - d_s^j(t))$. Left: ideal model. Right: actual implementation in (1).

order to ensure that we drive the system towards an optimal trade-off between overprovisioning and SLA violations, over a generic time horizon $T_h$, $\ell(\cdot)$ must account for the penalty inflicted in each case. In what follows, we describe the design of $\alpha$-OMC (Operator Monetary Cost), a loss functions that provides DeepCog with the capability of optimizing the overall running costs of the system.

### A. *Loss function design*

In DeepCog, the loss function steers the behaviour of the neural network by adjusting the weights of the neurons according to the error between the estimated value and the real one. To achieve the objective of minimizing the overall cost, a custom loss function for the capacity forecasting problem is composed by a term $f(x, x^*)$ that deals with the resource overprovisioning penalty, and a term $g(x, x^*)$ that models the cost of resource violations. The variable $x$ represents the allocated resources at a given time interval, while $x^*$ is the real demanded load for the same period. So the overall cost is due by the discrepancy between $x$ and $x^*$ in any time horizon.

The shape of overall cost function $f(x, x^*)+g(x, x^*)$ is depicted in Fig. 4a. A perfect algorithm (*i.e.*, an *oracle*) always keeps the system in the optimal operation point $x = x^*$ where no penalty is introduced, *i.e.*, $f(x^*, x^*) = g(x^*, x^*) = 0$. Of course, errors are inherent to predictions, and it is very unlikely that the forecast perfectly matches the real demand: hence, a penalty value is back-propagated depending on whether $x$ is above or below the target operation point $x^*$.

*1) $g(x, x^*)$, a reactive approach to SLA violations:* When the orchestrated resources are less than those needed in reality (*i.e.*, $x < x^*$) the network operator pays a monetary compensation to the tenant. We assume an SLA that guarantees a proportional compensation depending on the number of time intervals in which an operator fails to meet the requirements set by a tenant due

to insufficient capacity allocated to the slice. Thus, SLA violations determine a fixed cost for the operator at every time interval where the tenant demand is not satisfied. Accordingly, we let the system learn that the operation point $x^*$ is actually higher than the currently estimated one by applying a penalty $\beta$ as soon as the estimation falls below the real value. The parameter $\beta$ can be customized depending on the scenario: higher values may be used for cases where reliability is paramount (*e.g.*, an URLLC network slice), while lower values can be applied when KPIs are measured over longer time intervals. Higher $\beta$ values are likely to bring the system toward $x > x^*$, incurring hence in higher deployment costs, as discussed next.

*2) $f(x, x^*)$, a monotonically increasing cost for resource overprovisioning:* While SLA violations depend on the agreements between the tenants and the operator, the overprovisioning cost solely depends on the network operator, and more specifically on the deployment costs associated with excess allocated capacity. We assume that such a cost grows with the amount of unused capacity at each time interval, and model it as a positive monotonic function that is only applied when $x > x^*$: the higher the resource provisioning error, the more (unnecessarily) expensive is the deployment. The exact expression of $f(x, x^*)$ may vary, and one could consider, *e.g.*, linear (*i.e.*, $f(x, x^*) = \gamma x$), super-linear (*i.e.*, $f(x, x^*) = x^\gamma$), or exponential (*i.e.*, $f(x, x^*) = e^{\gamma x}$) variants. For DeepCog, we design $\alpha$-OMC to use a linear function, as in Fig. 4a. The parameter $\gamma$ is configurable by the operator, and represents the monetary cost of resource allocation: for instance, resources at the edge (*i.e.*, spectrum) are typically scarcer and more expensive to deploy than those in a network core datacenter. Therefore, a positive error $x$ in case of expensive (*i.e.*, high $\gamma$) resources will tend to bring the system to a lower allocation, with higher risks to hit the SLA violation zone.

*3) Balancing the two cost contributions.:* In general, $\beta$ and $\gamma$ are highly intertwined: the amount of resources that a network operator is willing to add depends on the cost that it has to pay when failing to meet the demands, given by $\beta$, but also on the cost associated with adding extra resources, given by $\gamma$. In the end, what matters is the relative value of the two parameters, rather than their absolute values. Therefore, in the following, we express the custom loss as a function of a single paramter $\alpha \doteq \frac{\beta}{\gamma}$. We underscore that $\alpha$ indicates the monetary costs of SLA violations with respect to the overprovisioning: failing to meet the slice requirements once costs as much as allocating $\alpha$ units of excess capacity. Clearly, a higher $\alpha$ implies relatively higher SLA violation costs. A mobile network operator can easily set this parameter based on its deployment costs, SLA fees, and market strategies.

Another important remark is that the SGD method used to train the neural network does not work with constant or step functions, and requires that the loss function be differentiable in all its domain. We solve this problem by introducing minimum slopes of very small intensity $\epsilon$ for $x < x^*$ and at $x = x^*$. We name the resulting loss function Operator Monetary Cost, which has a single configurable parameter $\alpha$. The final expression of $\alpha$-OMC is

$$\alpha\text{-OMC}(x, x^*) = \begin{cases} \alpha - \epsilon\,(x - x^*) & \text{if } x \leq x^* \\ \alpha - \frac{1}{\epsilon}\,(x - x^*) & \text{if } x^* < x \leq x^* + \epsilon\alpha \\ x - x^* - \epsilon\alpha & \text{if } x > x^* + \epsilon\alpha. \end{cases} \tag{1}$$

Fig. 4b provides a sample illustration of (1) above. This expression is employed to evaluate the quality of the forecast over the time horizon $T_h$ in the final loss function $\ell(\cdot)$, as follows:

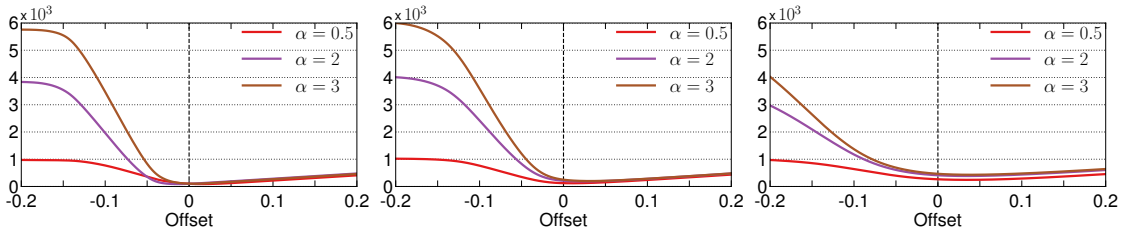
$$\ell\left(\bar{\mathbf{c}}_s(t, T_h), \mathbf{d}_s(t), \ldots, \mathbf{d}_s(t + T_h)\right) = \sum_{j \in \mathcal{M}} \sum_{\tau=0}^{T_h} \alpha\text{-OMC}\left(\bar{c}_s^j(t, T_h), d_s^j(t + \tau)\right). \tag{2}$$

### *B. Correctness and convergence*

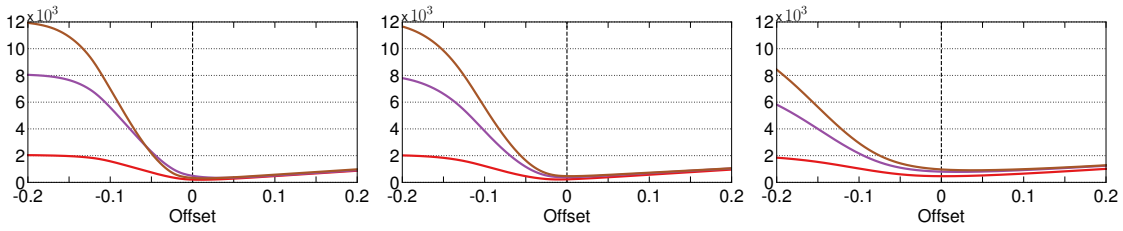
We now analyze the proposed loss function in terms of $(i)$ correctness, *i.e.*, its capability of achieving a performance that is close to the optimal, and $(ii)$ convergence, *i.e.*, the time it requires to learn such a correct strategy.

In Fig. 5, we run DeepCog in the representative network resource management case studies that are later detailed in Section V, where a slice is dedicated to one particular mobile service and runs in a specific class of network datacenter. For each case study, DeepCog forecasts a given level of capacity to be allocated which leads to an associated monetary cost. In order to investigate the correctness of the solution, we vary the provisioned capacity by adding to or subtracting a fixed offset from the capacity indicated by DeepCog.

The curves of Fig. 5 illustrate the variation of the monetary cost (in the y axis) as the offset is shifted (in the x axis), where increasingly positive (respectively, negative) values on the x axis correspond to a higher (respectively, lower) level of capacity provisioning with respect that suggested by our solution. The results prove that DeepCog always identifies the capacity allocation that minimizes the monetary cost for the operator under the inherently inaccurate prediction, as both a higher and a lower level of overprovisioning leads to a greater cost. This

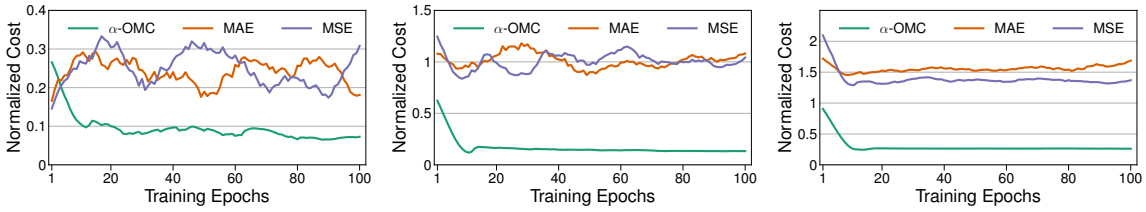



(a) Facebook, core datacenter (b) Snapchat, MEC datacenter (c) YouTube, C-RAN datacenter




(d) Facebook, core datacenter (e) Snapchat, MEC datacenter (f) YouTube, C-RAN datacenter

Fig. 5: Monetary cost (aggregated over time and normalized by the cost of one capacity unit) incurred when the overprovisioning level is shifted from that selected by DeepCog (at the abscissa origin). Each plot refers to one case study, *i.e.*, a combination of $(i)$ mobile service associated to a dedicated slice and $(ii)$ datacenter type. Top row: $T_h = 5$ minutes, bottow row: $T_h = 30$ minutes.




(a) Facebook, Core, $\alpha = 0.5$ (b) Snapchat, MEC, $\alpha = 2$ (c) YouTube, C-RAN, $\alpha = 3$

Fig. 6: Average cost versus the learning epochs, when the DeepCog neural network architecture is trained with $\alpha$-OMC, MSE and MAE loss functions.

holds under any combination[1] of target mobile service, datacenter class, and system settings $\alpha$ or $T_h$, which demonstrates the high consistency of our solution in balancing costs caused by SLA violations and overprovisioning.

[1]Fig. 5 shows results for $\alpha$ in the range $[0.5, 3]$, and two exemplary $T_h$ values, 5 and 30 minutes. Similar curves characterize all $\alpha$ values and prediction horizons (up to 8 hours) we tested.

We next assess the convergence properties of the loss function that drives DeepCog, by observing its behaviour over time. Specifically, we measure the normalized cost of the solution identified by our learning algorithm, and compare it against that returned by the same neural network trained with legacy loss functions.

Fig. 6 shows how the average normalized cost of network operation varies during the training phase for different $\alpha$, services and datacenter classes. While the $\alpha$-OMC loss function minimizes the monetary cost of the operator in less than $20$ epochs, both MAE and MSE converge to a fixed fee that grows as $\alpha$ increases. This confirms that classical loss functions are not effective when dealing with capacity forecasting, resulting in high penalties for operators. The results are consistent across all of the different configuration scenarios we tested.

## V. Performance Evaluation

We evaluate the performance of DeepCog in realistic case studies set in an operational mobile network providing coverage to a large metropolitan region of around 100 km$^2$. For our evaluation, we leverage real-world measurement data that describes the demands generated by several millions of users in the target region for individual mobile services. The traffic demands, expressed in bytes, are aggregated at the antenna sector level and over intervals of 5 minutes. They were collected by a major local operator by monitoring the GPRS Tunneling Protocol (GTP) via dedicated probes deployed at the network gateway.

We employ the measurement data to design three case studies combining several popular mobile services and different classes of network datacenters[2]. Each class is defined by the network location and number of served eNodeBs, ranging from centralized datacenters located in the core and serving many eNodeBs to more distributed ones located in the edge and serving a smaller number of eNodeBs. By selecting a diverse set of case studies, we can assess the DeepCog flexibility serving heterogeneous NFV scenarios, comprising different services and datacenter classes (C-RAN, MEC and core). In a first case study, we consider that a slice is instantiated for the incumbent video streaming service, *i.e.*, YouTube, at C-RAN datacenters in the target metropolitan area, each located in proximity of the radio access and performing baseband processing and scheduling for around ten eNodeBs. In the second case study, we look into Mobile

[2]The internal organization of the mobile network – hence the demand recorded at each datacenter – is inferred by adopting the methodology proposed in [36].

Edge Computing (MEC) datacenters that handle the traffic of around 70 eNodeBs each, where a dedicated slice accommodates the traffic generated by Snapchat, a favoured messaging app. The third case study focuses on a network slice dedicated to social network services provided by Facebook that are run at a core network datacenter controlling all 470 4G eNodeBs in the target metropolitan area. The three case studies cover applications with diverse requirements in terms of bandwidth and latency; also, they entail very different spatiotemporal dynamics of the mobile traffic, as the considered services feature different loads and activity peaks [37]. In addition, the datacenter classes we consider have dissimilar geographical coverage and aggregated traffic volumes, as they serve the demands associated to a variable number of antennas, from ten to several hundreds. Overall, the three case studies portray scenarios that jointly provide an adequate picture of the performance of DeepCog under the diversified network traffic and conditions that will characterize future mobile services.

As discussed in Section III, DeepCog outputs a capacity forecast within a variable time-horizon $T_h$. We measure this time in the number of steps it comprises, where each step corresponds to the 5 mins granularity of our measurement data. In our evaluation $T_h$ ranges from 5 minutes (which maps to a next-step prediction) to 8 hours (which corresponds to a forecast with a 96 time steps look-ahead). These are reasonable values in our context, since resource reallocation updates in the order of minutes are typical for computational and memory resources in architectures implementing NFV [38], and are in line with those supported by any state-of-the-art Virtual Infrastructure Manager (VIM) [39]. Conversely, larger intervals are more suitable for operations involving manual intervention, *e.g.*, spectrum leasing.

In all cases, we use the previous 30 minutes of traffic (*i.e.*, $T_p = 6$) as the DeepCog input, arranged in a $47 \times 10$ matrix as an input. This configuration proved to yield the best results when confronted to a number of other design strategies for the input that we explored, including longer, shorter, or non-continuous historical data time intervals. Capacity is predicted in terms of bytes of traffic, which is a reasonable metric to capture for resource utilization in actual virtual network functions [40], and is independent of the exact type of resources relevant for the mobile operator in each case study. We employ two months of mobile traffic data for training, two weeks of data for validation and another two for the actual experiments. This setting is also used for all benchmark approaches. All results are derived with a high level of confidence and low standard deviation.

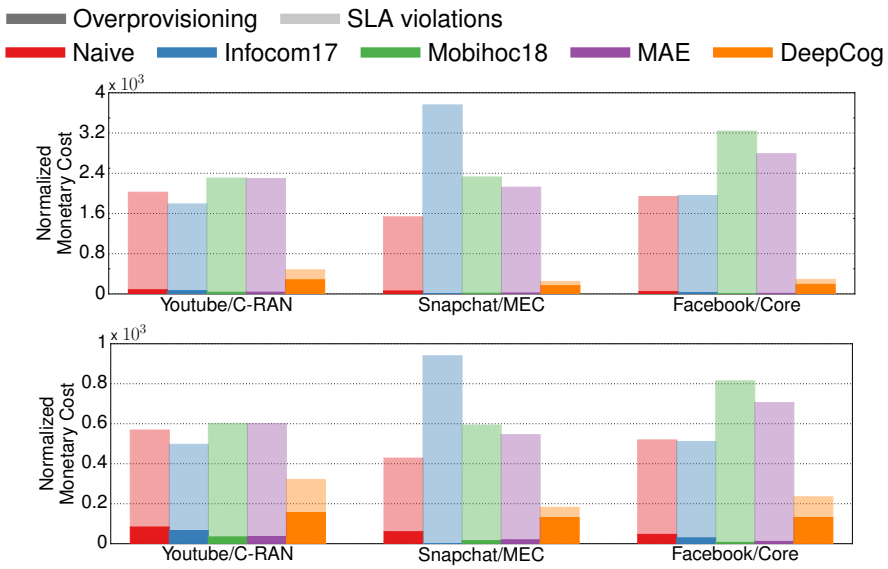



Fig. 7: Comparative evaluation of DeepCog with four benchmarks in three representative case studies. The monetary cost (normalized by the cost of one capacity unit) incurred by the operator is split into costs due to overprovisioning (dark) and SLA violations (light). Top: $\alpha = 2$. Bottom: $\alpha = 0.5$.

### A. *Gain over state-of-the-art traffic predictors*

We first focus on the particular case of next-step prediction, *i.e.*, $T_h = 5$ minutes, as this benchmark lets us compare our framework against state-of-the-art solutions that can only perform a forecast for the following time interval. We compare DeepCog against four benchmarks: ($i$) a `naive` technique that forecasts the future offered load by replicating the demand recorded at the same time during the previous week; ($ii$) the first approach proposed to predict mobile traffic based on a deep learning structure, referred to as `Infocom17` [8]; ($iii$) a recent solution for mobile network demand prediction that leverages a more complex deep neural network, referred to as `MobiHoc18` [21]; ($iv$) a reduced version of DeepCog, which replaces $\alpha$-OMC with a legacy Mean Absolute Error (MAE) loss function[3].

The results achieved in our three reference case studies by DeepCog and by the four benchmarks above are shown in Fig. 7. The plots report the normalized monetary cost for the operator, broken down into the expenses for unnecessary resource allocation (*i.e.*, overprovisioning) and

[3]We also experimented with other popular loss functions, *e.g.*, Mean Squared Error (MSE), with comparable results, omitted for space reasons.

fees for unserviced demands (*i.e.*, SLA violations). We observe that DeepCog yields substantially lower costs than all other solutions. Indeed, the cost incurred by DeepCog for $\alpha = 2$ ranges between 15% (Facebook/Core) and 27% (Youtube/C-RAN) of the cost provided by the *best* competitor, depending on the case study. `Infocom17`, as all other benchmarks, targets mobile network traffic prediction, whereas DeepCog aims at forecasting capacity. As a result, DeepCog balances overprovisioning and SLA violations so as to minimize operation expenses, while `Infocom17` is oblivious to such practical resource management considerations. In other words, legacy predictors follow as closely as possible the general trend of the time series and allocate resources based on their prediction, which leads to systematic SLA violations that are not acceptable from a market viewpoint and determine huge fees for the operator. Instead, DeepCog selects the appropriate level of overprovisioning that, by suitably overestimating the offered load, minimizes monetary penalties (see Fig. 5). Indeed, even when choosing a low value such as $\alpha = 0.5$, which inflicts a small penalty for a SLA violation, the cost incurred by DeepCog is 64% of that incurred by the best performing benchmark.

### *B. Comparison with overprovisioned traffic prediction*

In the light of the above results, a more reasonable approach to resource allocation could be to consider a traditional mobile traffic prediction as a basis, and adding some *overprovisioning offset* on top of it. In order to explore the effectiveness of such an approach, we design and implement several variants to `MAE`, as follows.

A first variant adds an *a-posteriori* constant overprovisioning offset to the `MAE` output. This strategy, referred to as `MAE-post`, requires selecting a value of the static offset, which is then added to the predicted traffic. We dimension the offset as a certain percentage of the peak traffic activity observed in the whole historical data, and set it at 5%, which we deem a reasonable value in presence of a decently accurate prediction. Alternatively, we also consider a best-case version of this solution, named `MAE-post-best`, where an a-posteriori overprovisioning is chosen by performing an exhaustive search over all possible offset values and selecting the one that minimizes the loss function $\ell(\cdot)$.

A second variant accounts for some level of overprovisioning in a *preemptive* fashion, by introducing the offset during the deep neural network training. To this end, the `MAE-pre` solution replaces the MAE loss function with a new loss function $\mathcal{O} + \frac{1}{\mathcal{M}} \sum_{j \in \mathcal{M}} |c_s^j(t) - d_s^j(t)|$, where $\mathcal{O}$ denotes the a-priori overprovisioning offset. Also in this case, we set $\mathcal{O}$ equal to 5% of the

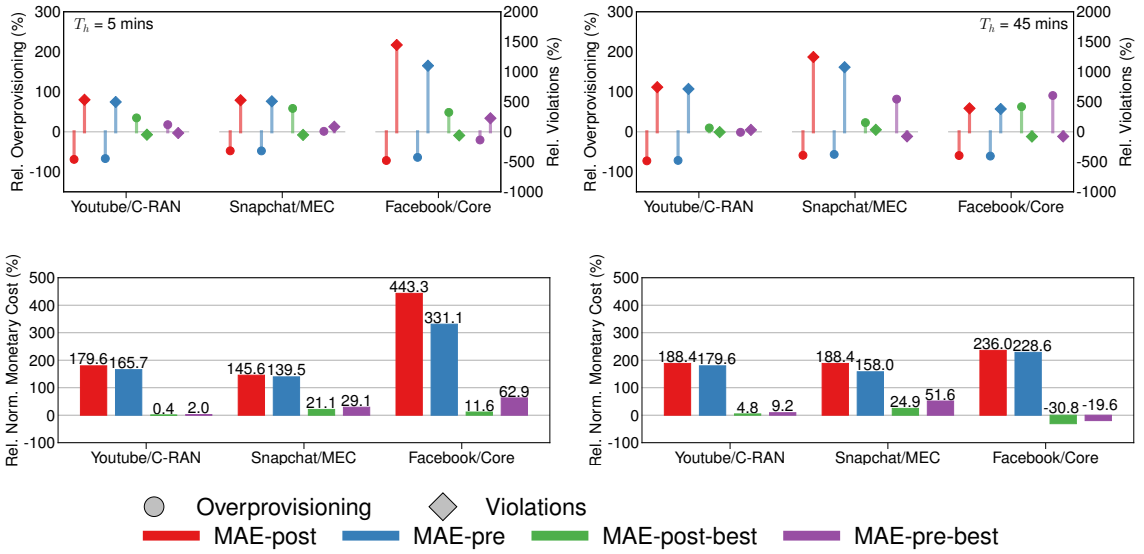



Fig. 8: Relative performance of overprovisioned traffic predictors, expressed as a percent of the cost attained by DeepCog. Top: relative overprovisioning and SLA violations. Bottom: relative monetary cost. Results refer to $\alpha = 2$ and prediction horizons of $5$ (left) and $45$ (right) minutes.

peak traffic in the historical data. To compare against the best possible operation of this scheme, we also consider a `MAE-pre-best` variant where $\mathcal{O}$ is set equal to the average overprovisioning level provided by DeepCog for the test period.

We remark that the `MAE-post-best` and `MAE-pre-best` approaches are oracles and not feasible in practice, since they require knowledge of the future to determine the best a-posteriori values for the offset and the value of $\mathcal{O}$, respectively. Yet, they provide a benchmark for comparing the performance of DeepCog against optimal solutions that rely on traditional mobile network traffic prediction.

Fig. 8 shows the relative performance of the four variants above with respect to that attained by DeepCog, for $T_h$ = 5 min (left) and $T_h$ = 45 min (right). The figure shows the oveprovisioned capacity, unserviced traffic, and total economic cost incurred by the operator relative to the performance offered by DeepCog (in percentage). For $T_h$ = 5 min, the results highlight how using a static overprovisioning in combination with a tradition traffic prediction is largely suboptimal, both when the additional offset is considered preemptively or a-posteriori. Indeed, the two practical solutions considered, *i.e.*, `MAE-post` and `MAE-pre`, cause SLA violations that are two- to three-fold more frequent than that incurred into by DeepCog, resulting in an economic cost that is 140% to 400% higher. Interestingly, even when parametrized with the best possible offsets,

the approaches based on legacy traffic prediction cannot match the performance of DeepCog: `MAE-post-best` and `MAE-pre-best` dramatically reduce the penalties of their viable counterparts, yet lead to monetary costs that are up to 60% higher than those of DeepCog.

The results for $T_h$ = 45 min[4], show that the above considerations hold across different values of the prediction horizon.The advantage over feasible overprovisioned traffic predictors such as `MAE-pre` or `MAE-post` is aligned with that observed under a next-step prediction, as such solutions increase the overall cost by 188% to 236%. When considering a long horizon of $45$ minutes, oracle methods based on overprovisioning like (*i.e.*, `MAE-pre-best` and `MAE-post-best`) can outperform DeepCog, further reducing the operator cost by 19% to 30%. This is due to the fact that, when prediction must be performed with significant time advance, the accuracy of DeepCog cannot be as high as an oracle that knows future demand and has hence a significant advantage. However, even under such conditions DeepCog performs almost as good as the oracles or better.

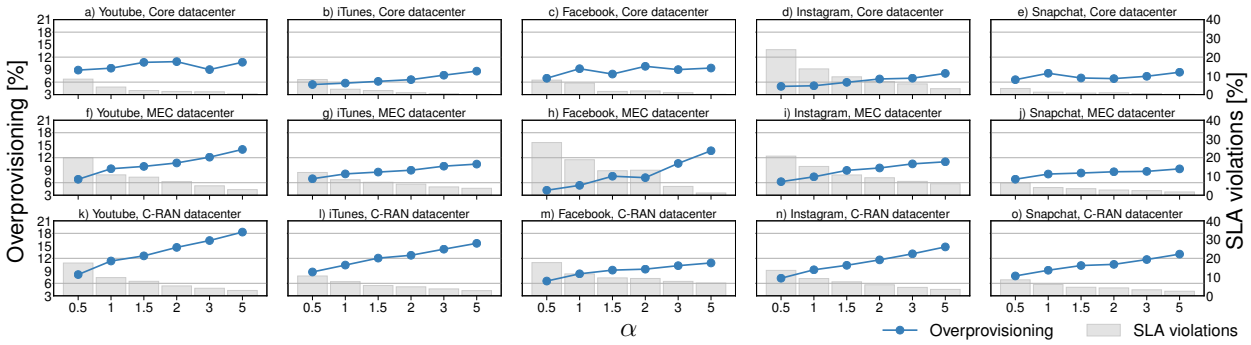
We conclude that traffic predictors – no matter how they are enhanced – are not appropriate for the capacity forecast problem, for the simple reason that they are designed for a different purpose. Indeed, they ignore the economic penalties incurred by SLA violations, and this limits drastically their ability to address this problem. Strategies that rely on integrating such costs into the solution after the traffic prediction is performed are largely suboptimal.

### *C. Controlling resource allocation trade-offs with the $\alpha$ parameter of DeepCog*

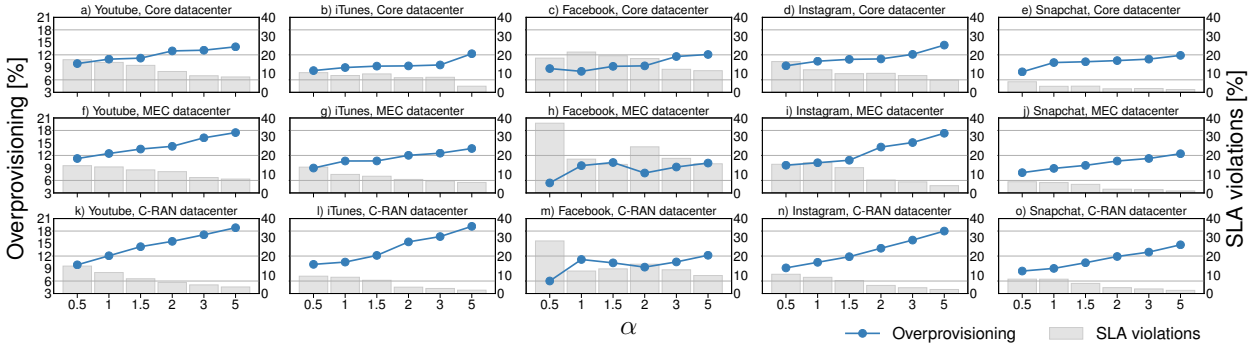
As discussed in Section IV, DeepCog addresses a fundamental trade-off between overprovisioning and SLA violations, aiming to find the best possible compromise between the two. An operator is given the flexibility of choosing the desired operation point within this trade-off, by suitably setting the $\alpha$ parameter. In the following, we carry out an extensive analysis of the trade-off between overprovisioning of resources and failing to meet service demands. This study is conducted for a large number of practical scenarios that extend the original three case studies considered in the comparative analysis. Specifically, we select five different network slices, dedicated to the same number of popular mobile services: the three we already studied, *i.e.*, YouTube, Facebook, and Snapchat, plus iTunes and Instagram. We then investigate the performance of DeepCog when such slices are deployed at the three classes of datacenters

[4]Note that, in order to perform a fair comparison, we had to extended the MAE policy to compute the average absolute error on each time slot in the $[t, \ldots, t + T_h]$ interval.

introduced before, *i.e.*, at the C-RAN, MEC, and network core. Overall, this leads to 15 distinct scenarios.




(a) $T_h = 5$ minutes




(b) $T_h = 120$ minutes

Fig. 9: Tradeoff between resource overprovisioning (expressed as a percentage of the actual demand) and SLA violation (expressed as a percentage of time slots), as a function of the $\alpha$ parameter. Results refer to 15 different scenarios, and two values of the prediction horizon $T_h$, *i.e.*, 5 minutes (a) and 120 minutes (b).

Fig. 9a shows results in all of the above settings under different economic strategies that are reflected by the $\alpha$ parameter of the loss function $\ell(\cdot)$. Configurations range from policies that prioritize minimizing overprovisioning over avoiding SLA violations ($\alpha = 0.5$) to others that strictly enforce the SLAs at the price of allocating additional resources ($\alpha = 5$). The plots tell apart the contribution of the two components that contribute to the total monetary cost: overprovisioning is expressed as a percentage of the actual demand, and SLA violations are measured as a percentage of the time slots in the test period. As expected, higher $\alpha$ values reduce the number SLA violations, as they become increasingly expensive; this occurs at the cost of provisioning additional capacity, which becomes instead cheaper in proportion. The trend is

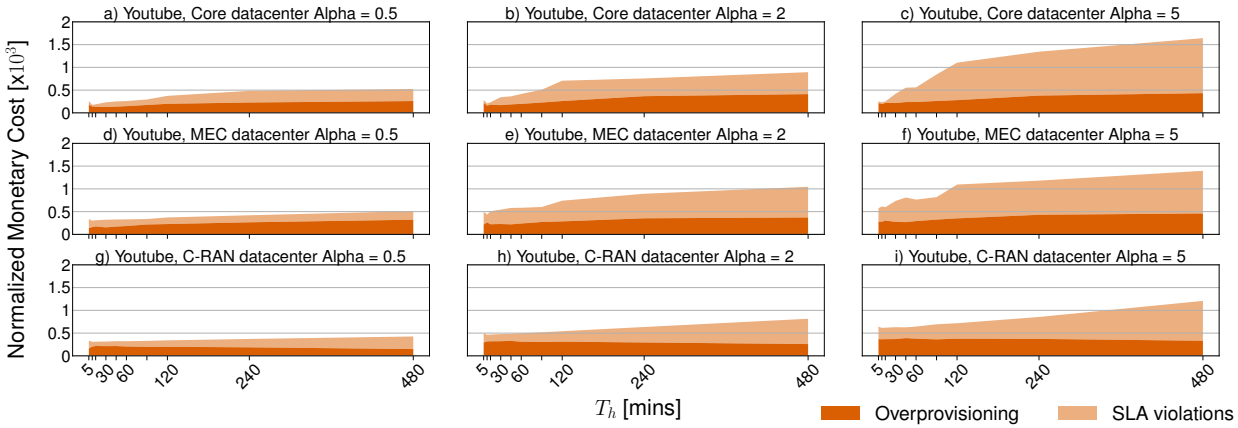



Fig. 10: Monetary cost (normalized by the cost of one capacity unit) incurred by the operator, versus the prediction time horizon $T_h$. The plots refer to different combinations of datacenter class and economic strategies modelled by $\alpha$, for a slice dedicated to the YouTube mobile service.

consistent across all scenarios, confirming that $\alpha$ effectively drives resource orchestration towards the desired operation point.

Our analysis also reveals that the level of overprovisioning grows as one moves from datacenters in the network core outwards. The phenomenon is observed for all studied slices, and is due to the fact that more centralized datacenters serve an increasingly aggregate traffic that is less noisy and easier to predict. Under such conditions, DeepCog needs a lower level of additional capacity to limit unserviced demands; indeed, the amount of SLA violations is typically lower at core datacenters.

Fig. 9a refers to a short-term prediction for $T_h = 5$ minutes, however the same trends discussed above are confirmed for larger prediction horizons. For instance, Fig. 9b reports the same results for $T_h = 120$ minutes. The only remarkable difference is that overprovisioning and SLA violations are higher than in the case of a 5-minute prediction, as forecasting on larger time horizons is obviously harder. Yet, the impact of $\alpha$ is equivalent to that observed for $T_h = 5$ minutes. We analyze in more detail the performance of DeepCog as a function of $T_h$ in the next section.

### *D. Long-term capacity prediction with DeepCog*

DeepCog aims at forecasting the (constant) capacity that should be allocated over a long-term horizon, so as to minimize the monetary cost incurred by the operator. As discussed in Section III, this is particularly useful in practical settings where the NFV technology imposes

limits on the frequency upon which resources can be reallocated. In this section, we thoroughly study how the performance of DeepCog varies with the prediction horizon.

Fig. 10 summarizes the overall trend of the monetary cost incurred by DeepCog, as the periodicity of the reconfiguration opportunities ranges from $5$ minutes to $8$ hours. The plots outline a diversity of scenarios, combining different datacenter classes (C-RAN, MEC, and core) and relative expenses of overprovisioning and SLA violations ($\alpha$ equal to $0.5$, $2$, and $5$). The results correspond to the case where one slice is dedicated to the traffic generated by YouTube, but equivalent behaviors were observed for the other services. In all settings, the cost grows with the prediction horizon, which, as already mentioned, is largely expected. What is less expected, however, is the quasi-linear relationship between the cost and $T_h$. This is a very important result, as it shows that even if we increase the intervals for resource reallocation (*i.e.*, the time horizon), the economic expenses of the operator remain bounded and do not skyrocket (as they would if the growth was, *e.g.*, exponential). The result thus demonstrates the efficiency of DeepCog in limiting the unavoidable increased penalty associated to forecasting long-term capacity: as an indicative figure, the cost is roughly increased by two when moving from a $5$-minute prediction to one that spans the following $8$ hours which is a very reasonable factor.

The impact of the other system parameters is in line with our previous analysis: higher monetary fees for SLA violations (*i.e.*, higher $\alpha$ values) lead to increased costs, whereas the performance is comparable across resource allocations over different classes of datacenter (C-RAN, MEC and core), each corresponding to different traffic volumes. It is nonetheless interesting to note that the property of a linear growth of the cost over $T_h$ is preserved under any combination of such parameters.

Fig. 10 also offers a breakdown of the overall monetary costs into the two contributions (overprovisioning and SLA violations). Violations of SLAs yield substantially higher absolute costs and dominate the increase of total cost with $T_h$; the effect is clearly stronger for higher values of $\alpha$. A more detailed view that highlights the exact evolution of the two cost components as a function of $T_h$ is provided in Fig. 11, showing that both contribute to increasing costs over longer-term forecasts. However, and interestingly, the dynamics of the two components with $T_h$ are diverse depending on the system settings such as the datacenter class and the value of $\alpha$. The common trend here is that the penalty associated with both overprovisioning and SLA violations is fairly stable when the horizon is increased from $5$ minutes up to two hours. For forecasts beyond two hours, however, these fees (one of the two or both) tend to increase substantially

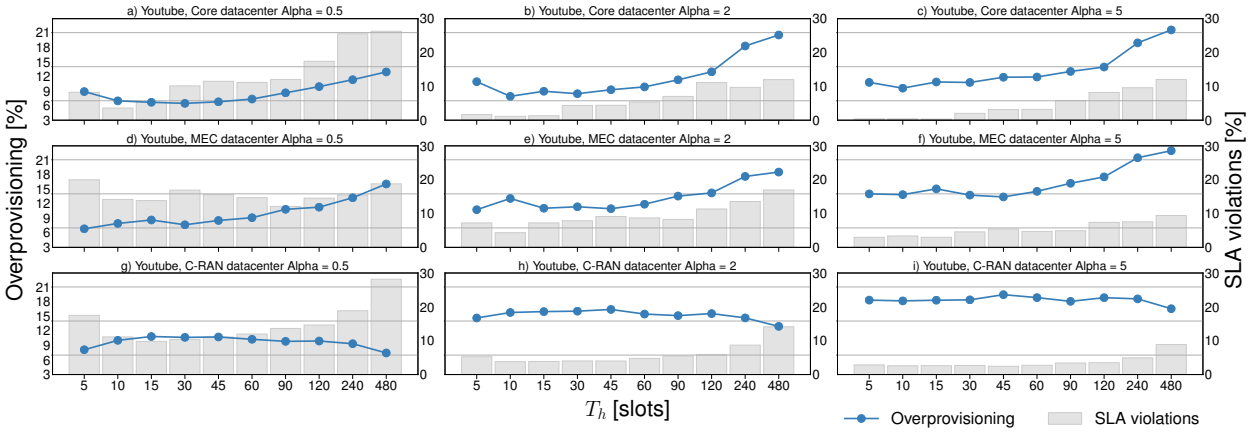
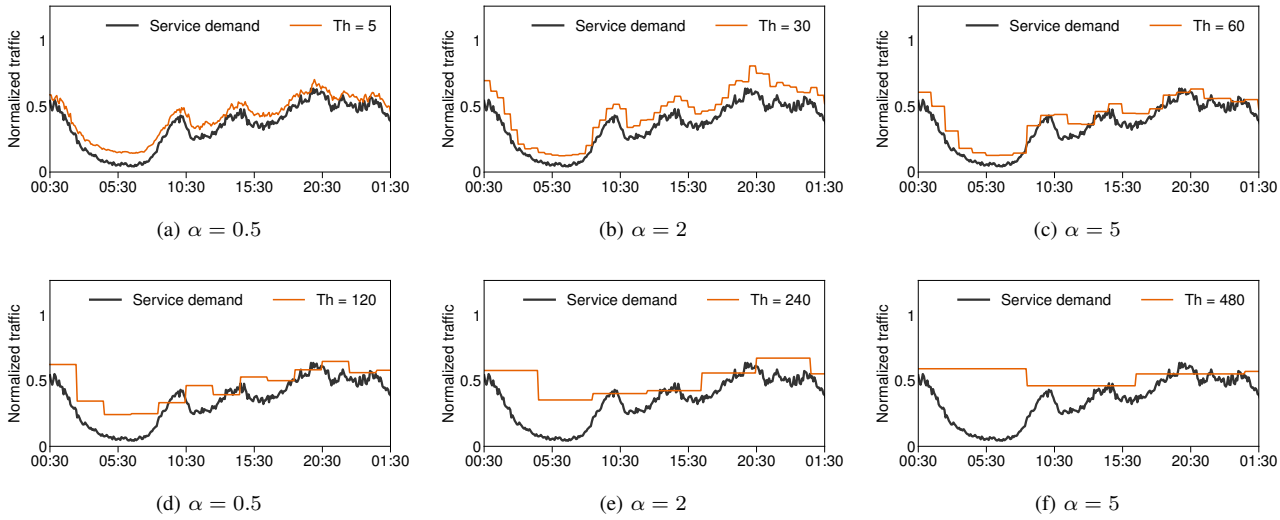



Fig. 11: Breakdown of monetary costs into two contributions: ($i$) overprovisioning (expressed as a percentage of the actual demand) and ($ii$) SLA violations (expressed as a percentage of time slots), in the scenarios of Fig. 10.




(a) $\alpha = 0.5$ (b) $\alpha = 2$ (c) $\alpha = 5$

(d) $\alpha = 0.5$ (e) $\alpha = 2$ (f) $\alpha = 5$

Fig. 12: Illustrative examples of the capacity forecast returned by DeepCog behaviour under different prediction time horizons. he scenario refers to a network slice dedicated to the YouTube mobile service that is deployed at a core datacenter, under $\alpha = 2$.

with $T_h$.

We ascribe these behaviors to ($i$) the relationship between $T_h$ and the timescale of temporal fluctuations in the input demand, and ($ii$) the way DeepCog reacts to the problem of devising a capacity forecast, which becomes harder for larger $T_h$ values. The first point relates to the characteristics of the input data (see Fig. 12 for illustrative examples of the temporal oscillation of the service demand). For very large $T_h$ (above 120 minutes) the prediction task performed

by DeepCog resorts to an "envelope" of the demand that accommodates the peak over the $T_h$ period. This means that for those times where demand is below the peak, we incur a high level of overprovisioning that increases the resulting cost. In contrast, smaller $T_h$ allow to adapt the capacity forecasting to the actual demand at each point in time, providing an advantage in terms of cost. The second point relates to the behavior and performance of DeepCog under large $T_h$ values. DeepCog aims at providing a similar level of overprovisioning over time, as exemplified by the top three plots of Fig. 12. For large $T_h$ values this yields increased SLA violations, since the oscillations make it more likely that the constant capacity falls below the demand curve at some point during $T_h$. Additionally, larger $T_h$ values make the prediction task inherently harder, which further contributes to increasing the SLA violations costs.

## VI. Conclusions

In this paper we presented and evaluated DeepCog, an original data analytics tool for the cognitive management of resources in sliced 5G networks. DeepCog tackles the novel problem of capacity forecasting, whose solution is key to the sustainable operation of future multi-tenant mobile networks. Inspired by recent advances in deep learning for image and video processing, DeepCog hinges upon a deep neural network structure, which analyzes antenna-level demand snapshots for different services in order to provide a prediction of the resources that the operator has to allocate to accommodate the future load. The operation is performed for individual mobile services separately, and over a configurable time horizon. At the core of DeepCog there is $\alpha$-OMC, a new and customized loss function that drives the deep neural network training so as to minimize the monetary cost contributed by two main deployment fees, *i.e.*, overprovisioning and SLA violation. Ours is, to the best of our knowledge, the only work to date where a deep learning architecture is explicitly tailored to the problem of anticipatory resource orchestration in mobile networks. The solution presented in this paper thus represents a first attempt to integrate data analytics based on machine learning into an overall cognitive management framework. Thorough empirical evaluations with real-world metropolitan-scale data show the substantial advantages granted by DeepCog over state-of-the-art predictors and other automated orchestration strategies, providing a first analysis of the practical costs of heterogeneous network slice management across a variety of case studies.

# REFERENCES


[1] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network Slicing to Enable Scalability and Flexibility in 5g Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.

[2] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5g: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[3] A. de la Oliva, X. Li, X. Costa-Perez, C. J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. Casetti, J. E. Gonzalez, and A. Azcorra, "5g-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 78–84, Aug. 2018.

[4] 5G-PPP, "The 5G Infrastructure Association. Pre-structuring Model, version 2.0," Nov. 2017.

[5] ETSI, NFVGS, "Network Function Virtualization (NFV) Management and Orchestration," *NFV-MAN*, vol. 1, Dec. 2014.

[6] D. Lopez, "OpenMANO: The dataplane ready open source NFV MANO stack," in *IETF Meeting, Dallas, Texas, USA*, Mar. 2015.

[7] The Linux Foundation, "ONAP, Open Network Automation Framework." [Online]. Available: https://www.onap.org

[8] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[9] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *arXiv:1803.04311 [cs.NI]*, Mar. 2018.

[10] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, March 2018.

[11] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: ACM, 2018, pp. 353–365. [Online]. Available: http://doi.acm.org/10.1145/3281411.3281435

[12] J. J. Ayala, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrain: A deep learning approach tailoring computing and radio resources in virtualized rans," in *Submitted*, 2019.

[13] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "Ran resource usage prediction for a 5g slice broker," in *Proceedings of the 20th International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MOBIHOC19. New York, NY, USA: ACM, 2019.

[14] M. Joshi and T. H. Hadi, "A Review of Network Traffic Analysis and Prediction Techniques," *arXiv:1507.05722 [cs.NI]*, Jul. 2015.

[15] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 796–805, Sep. 2016.

[16] M. Zhang, H. Fu, Y. Li, and S. Chen, "Understanding Urban Dynamics From Massive Mobile Traffic Data," *IEEE Transactions on Big Data*, pp. 1–1, 2017.

[17] S. T. Au, G.-Q. Ma, and S.-N. Yeung, "Automatic forecasting of double seasonal time series with applications on mobility network traffic prediction," *JSM Proceedings, Business and Economic Statistics Section*, Jul. 2011.

[18] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 234–240, Jun. 2014.

[19] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *ACM SIGMETRICS*, San Jose, California, USA, Jun. 2011, p. 305.

[20] A. Y. Nikravesh, S. A. Ajila, C.-H. Lung, and W. Ding, "An Experimental Investigation of Mobile Network Traffic Prediction Accuracy," *Services Transactions on Big Data*, vol. 3, no. 1, pp. 1–16, Jan. 2016.

[21] C. Zhang and P. Patras, "Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks," in *ACM MobiHoc*, Los Angeles, CA, USA, Jun. 2018, pp. 231–240.

[22] S. Ntalampiras and M. Fiore, "Forecasting mobile service demands for anticipatory MEC," in *IEEE WoWMoM*, Chania, Greece, Jun. 2018, pp. 14–19.

[23] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *IEEE INFOCOM*, Paris, France, May 2019, pp. 1–9.

[24] ——, "Alfa-OMC: cost-aware deep learning for mobile network resource orchestration," in *IEEE INFOCOM NI Workshop*, Paris, France, May 2019, pp. 1–9.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, Sep. 2014.

[26] Y. LeCun, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, Jun. 1989.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE CVPR*, Jun. 2015, pp. 1–9.

[28] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *IEEE ICASSP*, May 2013.

[29] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron) - A review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, Aug. 1998.

[30] F. Chollet, *Deep learning with Python*. Manning Publications Co, 2018.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, Dec. 2014.

[32] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda, "A Tale of Ten Cities: Characterizing Signatures of Mobile Traffic in Urban Areas," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2682–2696, Oct. 2017.

[33] J. Paparrizos and L. Gravano, "k-Shape: Efficient and Accurate Clustering of Time Series," in *ACM SIGMOD*, Jun. 2015, pp. 1855–1870.

[34] I. Borg and P. Groenen, "Modern Multidimensional Scaling: Theory and Applications," *Journal of Educational Measurement*, vol. 40, no. 3, pp. 277–280, Sep. 2003.

[35] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955.

[36] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How Should I Slice My Network?: A Multi-Service Empirical Evaluation of Resource Sharing Efficiency." New Delhi, India: ACM MobiCom, Nov. 2018, pp. 191–206.

[37] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki, and Z. Smoreda, "Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: ACM, 2017, pp. 180–186. [Online]. Available: http://doi.acm.org/10.1145/3143361.3143369

[38] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[39] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[40] J.-J. Kuo, S.-H. Shen, H.-Y. Kang, D.-N. Yang, M.-J. Tsai, and W.-T. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," in *IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.