DELMU: A Deep Learning Approach to Maximising the Utility of Virtualised Millimetre-Wave Backhauls

Rui Li, Chaoyun Zhang, Pan Cao, Paul Patras, and John S. Thompson

***Abstract*—Advances in network programmability enable operators to 'slice' the physical infrastructure into independent logical networks. By this approach, each network slice aims to accommodate the demands of increasingly diverse services. However, precise allocation of resources to slices across future 5G millimetre-wave backhaul networks, to optimise the total network utility, is challenging. This is because the performance of different services often depends on conflicting requirements, including bandwidth, sensitivity to delay, or the monetary value of the traffic incurred. In this paper, we put forward a general rate utility framework for slicing mm-wave backhaul links, encompassing all known types of service utilities, i.e. logarithmic, sigmoid, polynomial, and linear. We then introduce DELMU, a deep learning solution that tackles the complexity of optimising non-convex objective functions built upon arbitrary combinations of such utilities. Specifically, by employing a stack of convolutional blocks, DELMU can learn correlations between traffic demands and achievable optimal rate assignments. We further regulate the inferences made by the neural network through a simple 'sanity check' routine, which guarantees both flow rate admissibility within the network's capacity region and minimum service levels. The proposed method can be trained within minutes, following which it computes rate allocations that match those obtained with state-of-the-art global optimisation algorithms, yet orders of magnitude faster. This confirms the applicability of DELMU to highly dynamic traffic regimes and we demonstrate up to 62% network utility gains over a baseline greedy approach.**

I. INTRODUCTION

The 5th generation mobile networks (5G) embrace a new wave of applications with distinct performance requirements [1]. For example, ultra-high definition video streaming and immersive applications (AR/VR) typically demand very high data throughput. Autonomous vehicles and remote medical care are stringently delay-sensitive, belonging to a new class of Ultra-Reliable Low-Latency Communications (URLCC) services [2]. In contrast, Internet of Things (IoT) applications, including smart metering and precision agriculture, can be satisfied with a best-effort service. In order to simultaneously meet such diverse performance requirements, while enabling new verticals, mobile network architectures are adopting a *virtually sliced* paradigm [3]. The core idea of slicing is to partition physical network infrastructure into a number of logically isolated networks, i.e. slices. Each slice corresponds to a specific service type, which may potentially belong to a certain tenant operator.

R. Li, C. Zhang, and P. Patras are with School of Informatics, University of Edinburgh, UK. Email: {rui.li, chaoyun.zhang, paul.patras}@ed.ac.uk.

P. Cao is with School of Engineering and Technology, University of Hertfordshire, UK. Email: p.cao@herts.ac.uk

J. S. Thompson is with School of Engineering, University of Edinburgh, UK. Email: John.Thompson@ed.ac.uk

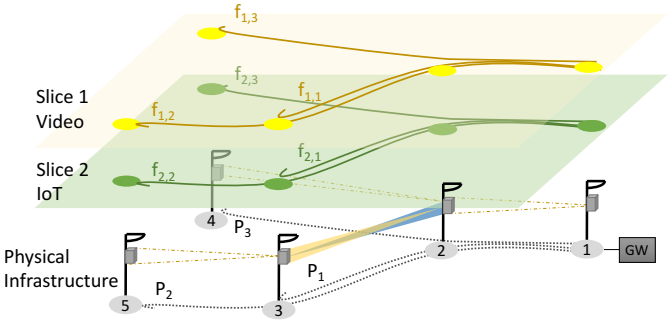

Fig. 1: Example of sliced backhaul over physical lamppost based mm-wave infrastructure. Slice 1 accommodates video streaming flows with sigmoid utilities and Slice 2 carries traffic from IoT applications, which have logarithmic utility.

At the same time, cellular and Wi-Fi base stations (BSs) are deployed massively, in order to increase network capacity and signal coverage. Millimetre wave (mm-wave) technology is becoming a tangible backhauling solution to connect these BSs to the Internet in a wireless fashion at multi-Gbps speeds [4]. In particular, advances in narrow beamforming and multiple-input multiple-output (MIMO) communications mitigate the severe signal attenuation characteristic to mm-wave frequencies and respectively multiply achievable link capacities [5].

Partitioning sliced mm-wave backhauls, and in general backhauls that employ any other communications technology, among traffic with different requirements, as in the example shown in Fig. 1, is essential for mobile infrastructure providers (MIPs). By and large, MIPs aim to extract as much value as possible from network resources, yet achieving this in sliced backhauls is not straightforward. In this example, five BSs are inter-connected via mm-wave directional links, forming a shared backhaul. The notion of rate utility is widely used to quantify the worth of an allocation of resources to multiple flows. The question is: *what type of utility is suitable to such multi-service scenarios?* Logarithmic utility as proposed in [6] has been adopted for elastic services and remains suitable for best-effort IoT traffic. On the other hand, applications such as video streaming typically throttle below a threshold, whilst an increase in service level is mostly imperceptible by users when the allocated rate grows beyond that threshold. Hence, the utility of such traffic can be modelled as a step-like sigmoid [7]. Had there been real-time applications to accommodate, their utility is typically formulated through

polynomial functions [8], [9]. Further, in the case of traffic for which the MIP allocates resources solely based on monetary considerations, a linear utility function can be employed. However, as the application scenarios diversify, a single type of utility cannot capture the distinct features of different service types. Therefore, we argue that a mixed utility must be considered. Unfortunately, combining all these utility functions may lead to non-concave expressions and computing in a timely manner the optimal rate allocation that maximises their value becomes a challenging task. Global search metaheuristics explore the feasible solution space intelligently to find global maxima [10], yet often involve unacceptably long computational times. Thus they fail to meet 5G specific delay requirements in highly dynamic environments, where application demands change frequently. Greedy approaches can be used to overcome the runtime burden, though these will likely settle on sub-optimal solutions.

Contributions: In this paper, we first put forward a general utility framework for sliced backhaul networks, which incorporates all known utility functions. We show that finding solutions to the network utility maximisation (NUM) problem when arbitrarily combining different utility functions is NP-hard. Inspired by recent advances in deep learning, we tackle complexity by proposing DELMU, a deep neural network model that learns the relations between traffic demands and optimal flow rate allocations. Augmented with a simple post-processing algorithm that ensures minimum service levels and admissibility within the network's capacity, we show that DELMU makes close-to-optimal inferences while consuming substantially shorter time as compared to state-of-the-art global search and a baseline greedy algorithm. In view of the current technological trends, we particularly focus on backhauls that operate in mm-wave bands. However, our utility framework and deep learning approach can be applied to other systems that operate in microwave or sub-gigahertz bands.

The remainder of the paper is structured as follows. In Sec. II we discuss the system model and in Sec. III we formulate the general NUM problem in the context of sliced mm-wave backhauls. We present the proposed deep learning approach to solving NUM in Sec. IV and show its performance in Sec. V. We review relevant related work in Sec. VI. Finally, in Sec. VII we conclude the paper.

II. SYSTEM MODEL

We consider a backhaul network deployment with $\mathcal{B}$ base stations (BSs) inter-connected via mm-wave links.[1] Each BS is equipped with a pair of transceivers, hence is able to transmit and receive simultaneously, while keeping the footprint small to suit dense deployment. To meet carrier-grade requirements and ensure precise TX/RX beam coordination, the network operates with a time division multiple access (TDMA) scheme. We assume carefully planned deployments where BSs have a certain elevation, e.g. on lampposts, hence interference is minimal and blockage events occur rarely.

We focus on settings where the backhaul network is managed by a single MIP and is partitioned into I logical slices to decouple different services (e.g. as specified in [3]). $\mathcal{F}$ user flows traverse the network and are grouped by traffic type i corresponding to a specific slice, i.e. $\mathcal{F} = \cup_{i\in\{1,\dots,I\}}\mathcal{F}_i$. The MIP's goal is to adjust the flow rates according to corresponding demands, in order to maximise the overall utility of the backhaul network. Flow demands are defined by upper and lower bounds. Lower bounds guarantee minimum flow rates, so as to ensure service availability, whilst upper bounds eliminate network resources wastage. We assume a controller (e.g. 'network slice broker' [11]) has complete network knowledge, periodically collects measurements of flow demands from BSs, solves NUM instances, and distributes the flow rate configurations corresponding to the solutions obtained.

Link Capacity: To combat the severe path loss experienced at mm-wave frequencies and boost capacity, BSs employ multiple input multiple output (MIMO) antenna arrays. We consider K array elements deployed at each base station for TX/RX. In backhaul settings, the stations' locations are fixed and the channel coherence time is typically long; hence it is reasonable to assume full knowledge of the channel state information is available at both transmitter and receiver sides. Given the channel matrix $\mathbf{H_{m,n}}$ from BS m to BS n, the received signal at BS n can be computed as

$$\mathbf{y_n} = \mathbf{H_{m,n}}\mathbf{x_m} + \mathbf{n_{m,n}}, \tag{1}$$

where $\mathbf{x_m}$ is an K-dimensional signal transmitted by BS m, and $\mathbf{y_n}$ are the received symbols at BS n. The singular value decomposition (SVD) of $\mathbf{H_{m,n}}$ is:

$$\mathbf{H_{m,n}} = \mathbf{U_n}\mathbf{\Sigma}\mathbf{V_m^H}, \tag{2}$$

where $\mathbf{U_n}$ and $\mathbf{V_m}$ are $K\times K$ unitary matrices, i.e. $\mathbf{U_n}\mathbf{U_n^H} = \mathbf{I}$ and $\mathbf{V_m}\mathbf{V_m^H} = \mathbf{I}$, and $\mathbf{\Sigma}$ is an $K\times K$ non-negative diagonal matrix containing the singular values of $\mathbf{H_{m,n}}$. The k-th diagonal entries of $\mathbf{\Sigma}$, i.e. σ_k, represents the k-th channel gain, and is also the k-th non-negative square root of the eigenvalues of matrix $\mathbf{H_{m,n}}\mathbf{H_{m,n}^H}$.

The parallel channel decomposition can be implemented efficiently for mm-wave systems as follows [12]. The transmitter precoding performs a linear transformation on the input vector $\tilde{\mathbf{x}}_\mathbf{m}$, i.e. $\mathbf{x_m} = \mathbf{V_m}\tilde{\mathbf{x}}_\mathbf{m}$, and the received signal $\mathbf{y_n}$ is linearly decoded by $\mathbf{U_n^H}$, i.e. $\tilde{\mathbf{y}}_\mathbf{n} = \mathbf{U_n^H}\mathbf{y_n}$. Therefore, the link capacity $c_{m,n}$ between base station m and n can be computed as:

$$c_{m,n} := \max_{\substack{\mathbf{Q_m}:\\ Tr(\mathbf{Q_m})\le P_{\max}}} B\log_2\det(\mathbf{I} + \mathbf{H_{m,n}}\mathbf{Q_m}\mathbf{H_{m,n}^H}), \tag{3}$$

where $\mathbf{Q_m} = \mathbf{V_m}\mathbf{V_m^H}$ is the transmission covariance matrix, B is the channel bandwidth, and $P_{\max}$ is the maximum transmit power. Without loss of generality, we assume that all BSs have the same maximum transmit power budget.

[1] Although we primarily focus on mm-wave backhauls, due to their potential to support high-speed and low latency communications, the optimisation framework and deep learning solution we present next are generally applicable to other technology.

For a channel known at the transmitter, the optimal capacity can be achieved by the well-known channel diagonalisation and the water-filling power allocation method [13]. For all BS m, by employing the optimal transmit pre-coding matrix $\mathbf{V_m} = \mathbf{X_{m,n}}$, where $\mathbf{X_{m,n}}$ denotes the eigenvector matrix of $\mathbf{H^H_{m,n}} * \mathbf{H_{m,n}}$, the MIMO channel capacity maximisation can be reformulated as:

$$c_{m,n} := \max B \sum_{k=1}^{K} \left(\log \left(1 + \frac{\lambda_k p_m^k}{\epsilon^2} \right) \right), \quad (4)$$

$$\text{s.t. } 0 \leq \sum_{k=1}^{K} p_m^k \leq P_{\max}, \quad (5)$$

$$p_m^k \geq 0, \forall k, m \quad (6)$$

where $\lambda_k = \sigma_k^2$, and ϵ^2 denotes the noise power. If the power allocated on the k-th sub-channel is p_m^k at BS m, then (5) specifies the total transmit power constraint. The optimal water-filling power allocation yields $p_m^k = \max\{0, \mu - \epsilon^2/\lambda_k\}$, where $\mu > 0$ is the water-filling level such that $\sum_{k=1}^{K} p_m^k = P_{\max}, \forall m$ [13].

III. Problem Formulation

Our objective is to find the optimal end-to-end flow rates that maximise the utility of sliced multi-service mm-wave backhaul networks. We first introduce a general network utility framework, based on which we formulate the NUM problem, showing that in general settings this is NP-hard.

A. Utility Framework

Recall that network utility refers to the value obtained from exploiting the network, which can be monetary, resource utilisation, or level of user satisfaction. For any flow f we consider four possible types of utility functions of flow rate r, depending on which slice $\mathcal{F}_i$ that flow belongs to. The utilities considered are parameterised by α_i and β_i, whose values have practical implications, such as the amount billed by the MIP for a service. Given an allocated rate r, we distinguish the following types of services that can be mapped onto slices, whose utilities we incorporate in our framework:

1) Services for which the MIP aims to maximise solely the attainable **revenue**. Denoting $\mathcal{F}_1$ the set of flows in this class, their utility is formulated as a linear function [14]:

$$U_{\text{lnr}}(r) = \alpha_1 r + \beta_1, \quad \forall f \in \mathcal{F}_1. \quad (7)$$

We note that $U_{\text{lnr}}(r)$ is both concave and convex.

2) Flows $f \in \mathcal{F}_2$ generated by applications that require certain level of **quality of service**, e.g. video streaming, and whose corresponding utility is thus formulated as a sigmoid function [7]:

$$U_{\text{sig}}(r) = \frac{1}{1 + e^{-\alpha_2(r-\beta_2)}}, \quad \forall f \in \mathcal{F}_2. \quad (8)$$

Observe that $U_{\text{sig}}(r)$ is convex in $[0, \beta_2)$ and concave in (β_2, ∞), therefore non-concave over the entire domain.

3) **Delay sensitive** flows, $f \in \mathcal{F}_3$, whose utility is modelled as a polynomial function [8]:

$$U_{\text{ply}}(r) = \alpha_3(r^{\beta_3}), \quad \forall f \in \mathcal{F}_3, \quad (9)$$

where β_3 is in the range $(0, 1]$, for which the above expression is concave.

4) **Best-effort** traffic, $f \in \mathcal{F}_4$, that does not belong in any of the previous classes, and whose utility is commonly expressed through a logarithmic function [6]:

$$U_{\log}(r) = \log(\alpha_4 r + \beta_4), \quad \forall f \in \mathcal{F}_4. \quad (10)$$

It is easy to verify that $U_{\log}(r)$ is also concave.

Our general utility framework encompasses all the four types of traffic discussed above (which may be parametrised differently for distinct tenants), therefore we express the overall utility of the sliced backhaul network as

$$\mathcal{U} := \sum_{f \in \mathcal{F}} U(r) = \sum_{f_1 \in \mathcal{F}_1} U_{\text{lnr}}(r_1) + \sum_{f_2 \in \mathcal{F}_2} U_{\text{sig}}(r_2) + \sum_{f_3 \in \mathcal{F}_3} U_{\text{ply}}(r_3) + \sum_{f_4 \in \mathcal{F}_4} U_{\log}(r_4). \quad (11)$$

Arbitrary combinations of both concave and non-concave utility functions may result in non-concave expressions $\mathcal{U}$, as exemplified in Fig. 2. In this figure, we show the total utility when combining 4 flows with different utility functions, two of them sigmoidal and two polynomial, each with different parameters. We assume the rates of each type of flow increase in tandem. Observe that even in a simple setting like this one, the network utility is highly non-concave and finding the optimal allocation that maximises it is non-trivial. We next formalise this problem with practical mm-wave capacity constraints, following which we discuss its complexity.

B. Network Utility Maximisation

Consider a set of flows that follow predefined paths, $P_j, j \in \{1, 2, ..., J\}$, to/from the local gateway, where the number of possible routes in the network is J. We denote $f_{i,j}$ a flow on slice i that traverses path P_j, which is allocated a rate $r_{i,j}$. By contract, $r_{i,j}$ shall satisfy $\delta_{i,j} \leq r_{i,j} \leq d_{i,j}$, where $\delta_{i,j}$ is the minimum rate that guarantees service availability, and $d_{i,j}$ is the upper bound beyond which the service quality cannot be improved. $d_{i,j}$ is no less than $\delta_{i,j}$ by default. Furthermore, each path P_j consists of a number of mm-wave links, and the link between BSs m and n is subject to a link capacity $c_{m,n}$. We use $\tau^s_{j,m} \in \{0, 1\}, s \in \{Tx, Rx\}$, to indicate whether node m transmits or receives data of flows traversing path P_j. The total network utility in (11) can be rewritten as:

$$\sum_{f \in \mathcal{F}} U(r) = \sum_{i=0}^{I} \sum_{j=0}^{J} U_i(r_{i,j}). \quad (12)$$

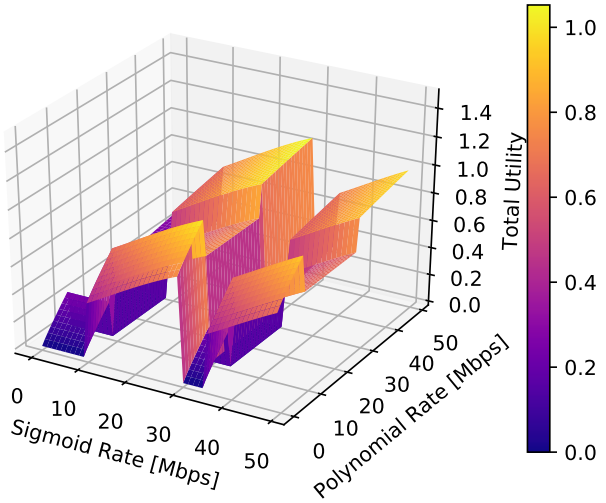

Fig. 2: Total utility when combining four flows with different utility functions; namely, two have sigmoid utility parametrised by $\alpha_2 = 0.08$, $\beta_2 = 15$ in the $[10-30]$ Mbps range, and respectively $\alpha_2 = 0.08$, $\beta_2 = 40$ in the $[35-50]$ Mbps range; the other two flows have polynomial utility with $\alpha_3 = 0.03651$, $\beta_3 = 0.9$ between $[0-10]$ Mbps, and $\alpha_3 = 0.03$, $\beta_3 = 0.6$ in $[30-50]$ Mbps. Rates increased in tandem for each type of flow.

Finding the flow rate allocation vector $\mathbf{r_{i,j}}, \forall i, j$, that maximises this utility requires to periodically solve the following optimisation problem:

$$\max \sum_{i=0}^{I} \sum_{j=0}^{J} U_i(r_{i,j}) \tag{13}$$

$$\text{s.t. } \delta_{i,j} \leq r_{i,j} \leq d_{i,j}, \forall i, j; \tag{14}$$

$$\sum_{i=0}^{I} \sum_{j=0}^{J} \tau_{j,m}^{s} \frac{r_{i,j}}{c_{m,n}} \leq 1, \{m,n\} \in P_j, s \in \{Tx, Rx\}. \tag{15}$$

In the formulation above, (13) is the overall objective function and (14) specifies the demand constraints. Each BS can transmit and receive to/from one and only one BS simultaneously, and the total time allocated at a single node for all flow Tx/Rx should not exceed 1, which is captured by (15). Here $r_{i,j}/c_{m,n}$ denotes the time fraction allocated to flow $f_{i,j}$ on link $l_{m,n}$.

C. Complexity

In what follows we briefly show that the network utility optimisation problem formulated above, where the objective function is a linear combination of linear, sigmoid, polynomial, and logarithmic functions, is NP-hard. By Udell and Boyd [15] any continuous function can be approximated arbitrarily well by a suitably large linear combination of sigmoidal functions [15]. Thus $\sum U(r)$ can be regarded as a sum of sigmoids and a larger number of other sigmoidal functions. Following the approach in [15], we can reduce an integer program

$$\text{find } \mathbf{r}$$
$$\text{s.t. } A\mathbf{r} \leq Z;\ \mathbf{r} \in \{0,1\}^n,$$

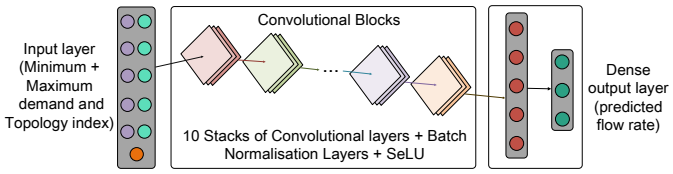

Fig. 3: Proposed Convolutional Neural Network with 10 hidden layers, which takes traffic demand and topology index as input, and infers the optimal flow rate allocations.

to an instance of a sigmoidal program

$$\max \sum_i g(r_i) = \sum_i r_i(r_i - 1)$$
$$\text{s.t. } A\mathbf{r} \leq Z;\ 0 \leq r_i \leq 1.$$

Here $g(r_i)$ enforces a penalty on non-integral solutions, i.e. the solution to the sigmoidal program is 0 if and only if there exists an integral solution to $A\mathbf{r} = Z$. Since the integer program above is known to be NP-hard [16], the reduced sigmoid program is also NP-hard, and therefore the NUM problem we cast in (13)–(15) is also NP-hard.

IV. DELMU: A DEEP LEARNING APPROACH TO NUM

To tackle the complexity of the optimisation problem formulated in the previous section and compute solutions in a timely manner, we propose DELMU, a deep learning approach specifically designed for sliced mm-wave backhauls and also applicable to other technologies. In essence, our proposal learns correlations between traffic demands and allocated flow rates, to make inferences about optimal rate assignments. We show that, with sufficient training data, our deep neural network finds solutions close to those obtained by global search, while requiring substantially less runtime.

A. Convolutional Neural Network

We propose to use a Convolutional Neural Network (CNN) to imitate the behaviour of global search. We train the CNN by minimising the difference between ground-truth flow rates allocations (obtained with global search) and those inferred by the neural network. In general CNNs preform weight sharing across different feature channels [17]. This significantly reduces the number of model parameters as compared to traditional neural networks, while preserving remarkable performance. At the same time, our approach aims to work well with a limited amount of training data, which makes CNNs particularly suitable for our problem. Therefore, we design a 12-layer CNN to infer the optimal flow rate and illustrate its structure in Fig. 3. The choice is motivated by recent results that confirm neural network architectures with 10 hidden layers, like ours, can be trained relatively fast and perform excellent hierarchical feature extraction [18].

The minimum and maximum traffic demand, and topology information are concatenated into a single vector, which will be subsequently fed to a sequence of convolutional blocks. Each block consists of a one-dimensional convolutional layer

and a Scaled Exponential Linear Unit (SELU) [19], which takes the following form:

$$\mathrm{SELU}(x) = \omega \begin{cases} x & x > 0 \\ \eta e^x - \eta & x \leqslant 0. \end{cases} \tag{16}$$

Here $\omega = 1.0507$ and $\eta = 1.6733$ by default. Employing SELU functions aims at improving the model representability, while enabling self-normalisation without requiring external techniques (e.g. batch normalisation). This enhances the robustness of the model and eventually yields faster convergence. Features of traffic demands are hierarchically extracted by convolutional blocks, and they are sent to fully-connected layers for inference. We train the CNN using a stochastic gradient descent (SGD) based method named Adam [20], by minimising the following mean square error:

$$L_e = \frac{1}{Q \times I \times J} \sum_{q=0}^{Q} \sum_{i=0}^{I} \sum_{j=0}^{J} (r_{q,i,j} - r'_{q,i,j})^2. \tag{17}$$

Q denotes the number of training data points, $r_{q,i,j}$ denotes the allocated rate allocated to flow j on slice i, with demand instance q, as suggested by global search. $r'_{q,i,j}$ is the corresponding rate inferred by the neural network. We train the CNN with 500 epochs, with an initial learning rate of 0.0001.

B. Post-Processing Algorithm

The output of the CNN on its own occasionally violates the constraints (14) and (15), because the model is only fed with traffic demands without embedding of constraints. We address this issue by designing a post-processing algorithm that adjusts the CNN solutions to fall within feasible domains, while maintaining minimum utility degradation and very short computation times. The idea is to first decrease recursively with a large step-length the rate of flows that breach the constraints, then increase repeatedly with a smaller step-length the rate of flows that can achieve the largest utility gains.

Algorithm 1 CNN Post-Processing Algorithm

1: Compute the time between each pair of nodes $t^s_{m,n}$
2: Compute the utility of each flow $u_{i,j} = U_i(r_{i,j})$
3: **while** Any $t^s_{m,n} > 1$ **do**
4: Find the link $l_{m,n}$ with the maximum $t^s_{m,n}$
5: deStepLen $= \min\{10, r_{i,j} - \delta_{i,j}\}$
6: **for** Flows satisfying $\tau^s_{j,m} == 1$ or $\tau^s_{j,n} == 1$ for $l_{m,n}$ **do**
7: Potential utility decrease $u'_{i,j} = U_i(r_{i,j} - \text{deStepLen})$
8: **end for**
9: Find the $f_{i,j}$ with the minimum non-zero $\Delta u_{i,j} = u_{i,j} - u'_{i,j}$
10: Decrease rate of $f_{i,j}$, i.e. $r_{i,j} = r_{i,j} - \text{deStepLen}$
11: Update $t^s_{m,n}$ and $u_{i,j}$
12: **end while**
13: **while** Any flow rate can be increased **do**
14: inStepLen $= \min\{1, d_{i,j} - r_{i,j}\}$
15: Potential utility increase $u''_{i,j} = U_i(r_{i,j} + \text{inStepLen}), \forall f_{i,j}$
16: Find the $f_{i,j}$ with the maximum $\Delta u_{i,j} = u''_{i,j} - u_{i,j}$
17: Increase rate of $f_{i,j}$, i.e. $r_{i,j} = r_{i,j} + \text{inStepLen}$
18: Update $t^s_{m,n}$ and $u_{i,j}$
19: **end while**

Algorithm 1 shows the pseudo-code of this procedure. The routine starts by computing the total time on each link for all traversing flows, i.e. $t^s_{m,n} = \sum_i \sum_j \tau^s_{j,m} r_{i,j} / c_{m,n}$ (line 1) and the utility of each individual flow based on the rate allocation returned by CNN (line 2). Then it searches recursively for a flow to decrease (lines 3–12). At each step, Algorithm 1 selects the link with the highest total time (line 4) and reduces the rate of the flow traversing the link with minimum possible utility loss (lines 5–10). Then the total link time and the flow utilities are updated (line 11). The process (lines 4–11) is repeated until the time for all links comply with the time constraints. Next, we increase iteratively a flow that yields the maximum potential utility gain, while ensuring that all constraints are satisfied (lines 13–19). This is done by tentatively increasing each flow, with a step-length that complies with the demand constraint (line 14), computing the corresponding utility increment (line 15), then finding the flow with maximum possible utility increase (line 16), and confirming the rate increment for that flow (line 17). Before the next round of increasing the rates, Algorithm 1 recomputes the total time on all links to verify that further rate increases are possible, and updates the utility of each flow (line 18).

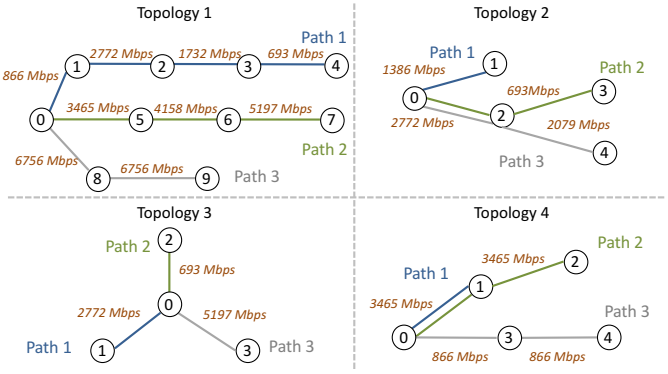

Fig. 4: The four network topologies used for evaluation. Circles represent the BSs, flow paths are shown with lines of different colour, and link capacities are labelled.

V. Performance Evaluation

We evaluate the proposed DELMU solution, which encompasses the CNN structure and the post processing algorithm, on different backhaul topologies under a range of conditions. Specifically, we use four different topologies as illustrated in Fig. 4, where the number of BSs varies from 4 to 10, and link rates range from 693 Mbps to 6.8 Gbps. Each path carries up to four types of flows, i.e. with individual sigmoid, linear, polynomial, and logarithmic utilities. For each topology, we generate randomly 10,000 combinations of flow demands in the range $[0 - 750]$ Mbps in increments of 50 Mbps. The corresponding minimum service rates are generated uniformly at random in the range $[0 - 100]$ Mbps as integer values, and are capped by the maximum flow demand. The parameters shown in Table I are used to model utility.

To train and subsequently test the neural network, we run a global search (GS) algorithm, the optimality of which is proven in [10], on each of the 10,000 network settings described above. We use 80% of the results obtained to construct

Utility Type	Linear	Sigmoid	Polynomial	Logarithmic
α_i	0.00133	0.08000	0.03651	0.00229
β_i	0	350	0.5	1

TABLE I: α_i and β_i parameters for the utility functions used in the evaluation.

a synthetic dataset that we use in the training process, which effectively seeks to minimise the mean square error expression defined in (17), by means of SGD. We use the remaining 20% of cases for as ground truth for testing the accuracy of the optimal rate allocation inferences that DELMU makes. More precisely, we compare the performance of DELMU in terms of total network utility and computational time, against the solutions obtained with GS and those computed with a baseline greedy approach that we devise. We discuss both benchmarks in more detail in the following subsection.

To compute solutions with the GS and greedy algorithms, and make inferences with the proposed CNN, we use a workstation with an Intel Xeon E3-1271 CPU @ 3.60GHz and 16GB of RAM. The CNN is trained on a NVIDIA TITAN X GPU using the open-source Python libraries TensorFlow [21] and TensorLayer [22]. We implement the greedy solution in Python and employ the GS solver of MATLAB®.

A. Benchmarks

The GS method works by starting from multiple points within the feasible space and searching for local optima in their vicinity, then concluding on the global optimum from the set of local optima obtained [10]. With default settings, which we employ in our evaluation, the GS generates 1,000 starting points using the scatter search algorithm [23], then eliminates those starting points that are not promising (judging by the corresponding value of the objective function and constraints). It then repeatedly executes a constrained nonlinear optimisation solver, i.e. `fmincon`, to search for local maxima around the remaining start points. Eventually the largest of all local maxima is taken as the global maximum, if one exists. We let the local optimisation routine work with the default Interior Point algorithm, which satisfies bounds at all iterations and can recover from non-numeric results. We note that simpler approximations such as semidefinite programming are constrained to convex optimisation problems, thus inappropriate for our task.

We also engineer a baseline greedy algorithm for the purpose of evaluation, with the goal of finding reasonably good solutions *fast*. The greedy approach starts by setting all flow rates to the minimum demand and then recursively chooses a flow to increase its rate, with the aim of achieving maximum utility gain at the current step, as long as the constraints (14)–(15) are respected. A solution is found when there are no remaining flows whose rates can be further increased. For fair comparison, the greedy approach takes exactly the same flow demands and the corresponding minimum service rates as used by GS and DELMU. A step size of 1 Mbps is employed.

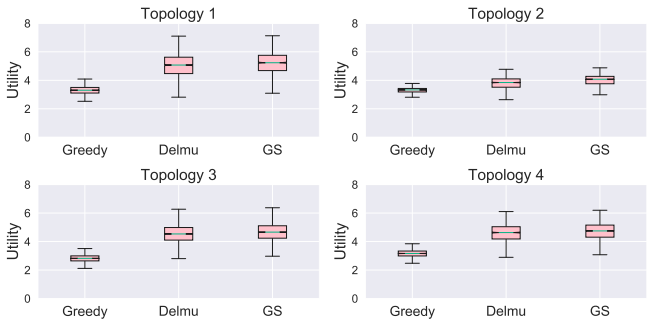

Fig. 5: Distribution of the total utility attained by the proposed DELMU, and the benchmark GS and greedy algorithms, for the four topologies shown in Fig. 4. Numerical results.

B. Total Utility

We first examine the overall utility performance of the proposed DELMU, in comparison with that of the greedy and the GS solutions. Fig. 5 illustrates the distributions of the total network utility for the 12 flows traversing the network, over the 2,000 instances tested. We observe that, among the 4 topologies used, the distribution of the total utility obtained by DELMU is almost the same as that of the optimal solution obtained with GS, as confirmed by the similar median values, the distance between the first and third quartiles, as well as the whiskers (minima and maxima). Specifically, the median values of the total utility attained by GS in Topologies 1–4 are 5.23, 4.07, 4.66, and 4.75, while those achieved by the proposed DELMU are 5.09, 3.88, 4.56, and 4.64. In sharp contrast to the DELMU's close-to-optimal performance, the greedy solution attains the medians of 3.30, 3.32, 2.81, and 3.16 utility units in the 4 topologies considered. Among these, for the case of Topology 3, DELMU obtains a 62% total utility gain over the greedy approach. It is also worth remarking that, although a greedy approach can perform within well-defined bounds from the optimum when working on submodular objective functions [24], this is clearly suboptimal in the case of general utility functions as addressed herein.

C. Decomposing Performance Gains

To understand how DELMU achieves close-to-optimal utility, and why the benchmark greedy solution performs more poorly, we examine one single instance for each topology, and dissect the utility values into the components corresponding to each type of traffic (i.e. slice). Fig. 6 illustrates the sum of utilities for each type of traffic, attained with the greedy, CNN, and GS approaches. We note that the greedy solution tends to allocate more resources to traffic with logarithmic utility (in all topologies) and respectively polynomial utility (in Topologies 2, 3, and 4). In contrast, the CNN allocates higher rates to traffic subject to sigmoid utility in all the scenarios studied, which results in higher overall utility. This is because the greedy approach gives more resources to the flows that yield utility gains in the first steps of the algorithm's execution and fails to capture the inflection point of the traffic with sigmoid utility, which can contribute to a higher overall utility, under limited resource constraints. Furthermore, the

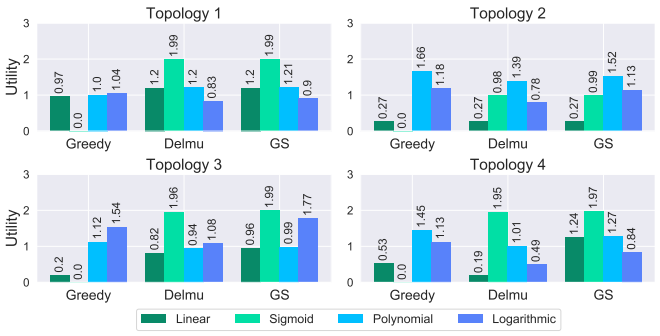

Fig. 6: An example instance of the utility corresponding to each traffic type in each topology. Bars represents the sum utility of flows in the same slice. Numerical results.

allocations of rates to different traffic types by DELMU show close resemblance to the GS behaviour, which confirms the fact that DELMU achieves overall close to optimal utility allocations, at a lower computational cost, as we will see next.

We delve deeper into the utility attained by each flow on each slice, along different paths, and in Fig. 7 compare the performance of our approach and the benchmarks considered in the case of Topology 1. Flows corresponding to slices that have linear, sigmoid, polynomial, and respectively logarithmic utility are indexed from 1 to 4. Again, observe that the greedy approach assigns zero utility to traffic subject to sigmoid utility, in stark contrast with the GS method. While DELMU obtains the highest gains from traffic with linear and sigmoid utility on paths 2 and 3, greedy dedicates most of the network resources to traffic with logarithmic and exponential utility, without obtaining significantly more utility from these types of flows. DELMU achieves accurate inference, as the performance is nearly the same with that of GS for all flows.

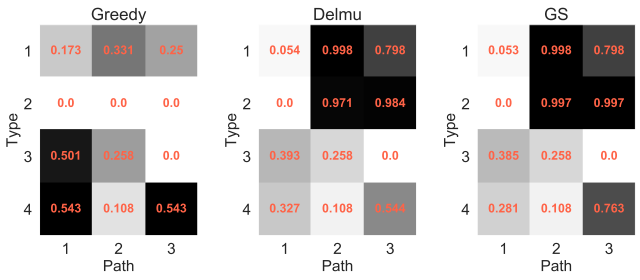

Fig. 7: Utility of all data flows (on different slices and over different paths) attained by greedy, DELMU, and GS in one demand instance in Topology 1. In each subfigure, darker shades represent higher utility and the actual values are labelled. Numerical results.

D. Real-time Inference

To shed light on the runtime performance of the proposed DELMU solution, we first examine the average time required for inferring a single solution throughout the performance analysis presented in Section V-B. We compare these computation times with those of the greedy and GS approaches over 2,000 instances and list the obtained results in Table II. Note that the values for DELMU include the post-processing

Topology Index	1	2	3	4
GS	8.4339s	4.6075s	3.4492s	4.8311s
Greedy	0.1500s	0.1590s	0.1178s	0.1345s
DELMU	**0.0036s**	**0.0035s**	**0.0025s**	**0.0026s**

TABLE II: Average computation time required to obtain a single solution to the NUM problem in Topologies 1–4 using GS, greedy, and the proposed CNN mechanism.

time. Observe that GS takes seconds to find a solution, while the greedy approach, although inferior in terms of utility performance, has runtimes in the order of hundreds of milliseconds for a single instance. In contrast, our CNN makes and adjusts inferences within a few milliseconds. That is, as compared to the greedy algorithm, CNN generally requires two orders of magnitude smaller computation time. On the other hand, the GS algorithm, although working optimality, has three orders of magnitude higher runtimes as compared to DELMU. Lastly, note that the CNN inference itself requires $\sim$1.5ms per instance, and hence the post-processing dominates the overall execution time in the first two topologies. We conclude that the proposed DELMU is suitable for highly dynamic backhauls.

We complete this analysis by investigating the ability of the proposed DELMU solution to handle network dynamics in sliced mm-wave backhaul settings, including changes in traffic demand due to e.g. on/off behaviour of user applications and variations in capacity triggered e.g. by occasional blockage on the mm-wave links. We consider Topology 3 in Fig. 4, transporting a mix of flows with linear, polynomial, and logarithmic utility and different lifetimes, considering a 10 Mbps minimum level of service, in all cases when a flow is active. Precisely, in Fig. 8 we examine the time evolution of the throughput DELMU allocates to flows on each slice, according to a sequence of events. In particular, flows subject to sigmoid utility start with 0 Mbps demands, whilst all flows of the other types on all path have each an initial demand of 200 Mbps. After 100 ms, a flow with sigmoid utility on path 2 (i.e. $f_{2,2}$) becomes active, adding a 400 Mbps demand to the network. At time 200 ms, partial link blockage occurs on the link between BS 0 and BS 1, causing the corresponding capacity $c_{0,1}$ to drop from 2,772 Mbps to 693 Mbps. $f_{2,2}$ finishes 100 ms later.

Observe in the figure that the DELMU performs a correct allocation as soon as a change occurs and, given the millisecond scale inference times, the transition is almost instantaneous even at the 100 ms granularity. For instance, when $f_{2,2}$ joins, the allocation of network resources is immediately rearranged, so that the request of $f_{2,2}$ is mostly satisfied, whereas the rest of flows receive reduced rates. In this case, all the flows with linear utility are reduced to close to the minimum level of service, i.e. each to 11 Mbps rate. The drop in $c_{0,1}$ capacity at 200 ms leads to a significant degradation of the rates assigned to flows with polynomial and logarithmic utility, while the linear and sigmoid flows remain unaffected. Eventually, at 300 ms, when flow $f_{2,2}$ finishes, the rate of the flows with polynomial and logarithmic utility are increased, yet remain

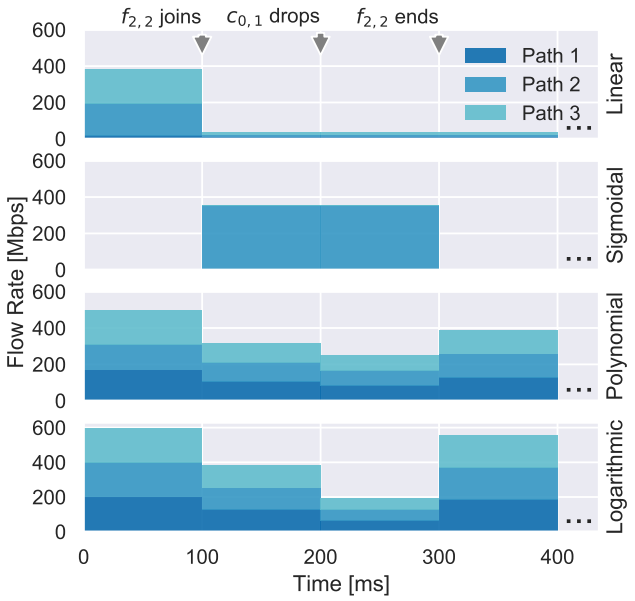

Fig. 8: Rate allocations performed by DELMU for flows of different slices and paths over time in Topology 3 (see Fig. 4), as a sequence of demand and capacity changes occur as labeled at the top of the figure. Numerical results.

below the values assigned initially, due to the inferior $c_{0,1}$ capacity. Hence, the proposed DELMU is suitable for highly dynamic backhaul environments, as it makes close to optimal inferences fast and is able to adapt to sudden changes.

VI. Related Work

In this section, we review previous work most closely related to our contribution, which touches upon network slicing, mm-wave backhauling, utility optimisation, and deep learning in networking.

Network slicing. Major 5G standardisation efforts put emphasis on the evolution towards sliced network architectures [3], [25], and recent research highlights the benefits of sharing mobile infrastructure among virtual operators [11], [26], [27]. In [11], a slice broker concept that enables MIPs to manage dynamically the shared network resources is proposed. Based on this concept, a machine learning approach that addresses admission control in sliced networks is given in [26]. An online slice brokering solution is studied in [27] with the goal of maximising the multiplexing gain in shared infrastructure. However, existing efforts do not address the diverse service requirements of different application scenarios.

Mm-wave backhauling. Mm-wave technology is recognised as a key enabler of multi-Gbps connectivity. Dehos *et al.* study the feasibility of employing mm-wave bands in access and backhaul networks, and highlight the significant throughput gain achievable at mm-wave frequencies as compared with microwave bands [4]. Hur *et al.* propose a beam alignment scheme specifically targeting mm-wave backhauling scenarios and study the wind effect on the performance of backhaul links [5]. Sim *et al.* propose a decentralised learning based medium access protocol for multi-hop mm-wave networks [28]. In [29], the authors advocate a max-min fair flow rate and airtime allocation scheme for mm-wave backhaul networks. These efforts however do not consider network utility and disregard sliced multi-service settings.

Network utility maximisation (NUM). With growing popularity of inelastic traffic, optimising a mix of both concave and non-concave utilities has been studied [8], [30], [31]. Fazel *et al.* propose a sum-of-square method to solve non-concave NUM problems that tackle primarily polynomial utility [8]. Hande *et al.* study the sufficient conditions for the standard price-based (sub-gradient based dual) approach to converge to global optima with zero duality gap, which relies on capacity provisioning [30]. Chen *et al.* consider NUM with mixed elastics and inelastic traffic, and develop a heuristic method to approximate the optimal [31]. Recent work investigates convex relaxation of polynomial NUM and employs distributed heuristics to approximate the global optimal [9], and Udell and Boyd define a general class of non-convex problems as sigmoidal programming and propose an approximation algorithm [15]. The limitation of these heuristics lies within their convergence times that is in the order of seconds, which can hardly meet the latency requirements of 5G networks. In contrast, our deep learning approach infers close to optimal rate allocations within milliseconds.

Deep learning in networking. With the increase in computational power and data sets availability, a range of deep learning applications in the computer and communications networking domain are emerging [32]. A fully-connected neural network is used in [33] to find optimal routes in wired/wireless heterogeneous networks. Zhang *et al.* employ a dedicated CNNs to infer fine-grained mobile traffic consumption from coarse traffic aggregates [34], improving measurement resolution by up to 100× while maintaining hight accuracy. CNNs have also employed been in [35], where the authors incorporate a 3D-CNN structure into a spatio-temporal neural network, to perform long-term mobile traffic forecasting. To the best of our knowledge, our work is the first that uses deep learning to solve utility optimisation problems in sliced backhauls.

VII. Conclusions

In this paper we tackled utility optimisation in sliced mm-wave networks by proposing DELMU, a deep learning approach that learns correlations between traffic demands and optimal rate allocations. We specifically deal with scenarios where traffic is subject to conflicting requirements and maximise non-concave utility functions that reconcile all services, while overcoming the inherent complexity of the problems posed. We demonstrated that the proposed convolutional neural network attains up to 62% utility gains over a greedy approach, infers close to optimal allocation solutions within orders of magnitude shorter runtimes as compared to global search, and responds quickly to network dynamics.

ACKNOWLEDGEMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] NGMN. 5G White Paper. *Next generation mobile networks*, 2015.

[2] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch. Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, Feb 2017.

[3] 3GPP. Technical Specification Group Services and System Aspects; System Architecture for the 5G System. *3GPP TS 23.501*, Dec. 2017.

[4] Cedric Dehos, Jose Luis González, Antonio De Domenico, Dimitri Ktenas, and Laurent Dussopt. Millimeter-wave access and backhauling: The solution to the exponential data traffic increase in 5G mobile communications systems? *IEEE Communications Magazine*, 52(9):88–95, 2014.

[5] Sooyoung Hur, Taejoon Kim, David J Love, James V Krogmeier, Timothy A Thomas, Amitava Ghosh, et al. Millimeter Wave Beamforming for Wireless Backhaul and Access in Small Cell Networks. *IEEE Trans. Communications*, 61(10):4391–4403, 2013.

[6] Frank Kelly. Charging and rate control for elastic traffic. *Trans. Emerging Telecommunications Technologies*, 8(1):33–37, 1997.

[7] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *Proc. ACM SIGCOMM*, pages 325–338, 2015.

[8] Maryam Fazel and Mung Chiang. Network utility maximization with nonconcave utilities using sum-of-squares method. *Proc. IEEE CDC-ECC*, 2005(1):1867–1874, 2005.

[9] Jingyao Wang, Mahmoud Ashour, Constantino Lagoa, Necdet Aybat, Hao Che, and Zhisheng Duan. Non-Concave Network Utility Maximization in Connectionless Networks: A Fully Distributed Traffic Allocation Algorithm. In *Proc. IEEE ACC*, pages 3980–3985, 2017.

[10] Zsolt Ugray, Leon Lasdon, John Plummer, Fred Glover, James Kelly, and Rafael Martí. Scatter search and local NLP solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3):328–340, 2007.

[11] Konstantinos Samdanis, Xavier Costa-Perez, and Vincenzo Sciancalepore. From network sharing to multi-tenancy: The 5G network slice broker. *IEEE Communications Magazine*, 54(7):32–39, 2016.

[12] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath. Spatially Sparse Precoding in Millimeter Wave MIMO Systems. *IEEE Trans. Wireless Communications*, 13(3):1499–1513, March 2014.

[13] G. G. Raleigh and J. M. Cioffi. Spatio-temporal coding for wireless communication. *IEEE Trans. Communications*, 46(3):357–366, Mar 1998.

[14] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[15] Madeleine Udell and Stephen Boyd. Maximizing a Sum of Sigmoids. *Optimization and Engineering*, pages 1–25, 2013.

[16] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[18] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2377–2385, 2015.

[19] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proc. NIPS*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2015.

[21] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *Proc. OSDI*, volume 16, pages 265–283, 2016.

[22] Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. TensorLayer: A versatile library for efficient deep learning development. In *Proc. ACM Multimedia Conference*, 2017.

[23] Fred Glover. A template for scatter search and path relinking. *Lecture Notes in Computer Science*, 1363:13–54, 1998.

[24] Kyuho Son, Eunsung Oh, and Bhaskar Krishnamachari. Energy-efficient design of heterogeneous cellular networks from deployment to operation. *Computer Networks*, 78:95–106, 2015.

[25] 3GPP. Technical Specification Group Services and System Aspects; Study on Architecture for Next Generation System. *3GPP TS 23.799*, Dec. 2016.

[26] Vincenzo Sciancalepore, Konstantinos Samdanis, Xavier Costa-Perez, Dario Bega, Marco Gramaglia, and Albert Banchs. Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization. *Proc. IEEE INFOCOM*, 2017.

[27] Vincenzo Sciancalepore, Lanfranco Zanzi, Xavier Costa-Perez, and Antonio Capone. ONETS: Online Network Slice Broker From Theory to Practice. *arXiv preprint arXiv:1801.03484*, 2018.

[28] Gek Hong Sim, Rui Li, Cristina Cano, David Malone, Paul Patras, and Joerg Widmer. Learning from experience: Efficient decentralized scheduling for 60GHz mesh networks. In *Proc. IEEE WoWMoM*, 2016.

[29] Rui Li and Paul Patras. WiHaul: Max-Min Fair Wireless Backhauling over Multi-Hop Millimetre-Wave Links. In *Proc. ACM Workshop HotWireless*, pages 56–60, 2016.

[30] Prashanth Hande, Shengyu Zhang, and Mung Chiang. Distributed rate allocation for inelastic flows. *IEEE/ACM Trans. Networking*, 15(6):1240–1253, 2007.

[31] Li Chen, Bin Wang, Li Chen, Xin Zhang, and Yang Dacheng. Utility-based resource allocation for mixed traffic in wireless networks. *Proc. IEEE INFOCOM Workshops)*, pages 91–96, 2011.

[32] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep Learning in Mobile and Wireless Networking: A Survey. *arXiv preprint arXiv:1803.04311*, 2018.

[33] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani. The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective. *IEEE Wireless Communications*, 24(3):146–153, Jun 2017.

[34] Chaoyun Zhang, Xi Ouyang, and Paul Patras. ZipNet-GAN: Inferring Fine-grained Mobile Traffic Patterns via a Generative Adversarial Neural Network. In *Proc. ACM CoNEXT*, pages 363–375.

[35] Chaoyun Zhang and Paul Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proc. ACM MobiHoc*, pages 231–240, 2018.