Deep Neural Network Model for Dynamic Functional Split Management in Beyond 5G

David Campoy[1], Jordi Pérez-Romero[1], Oriol Sallent[1], Antoni Gelonch[1], Xavier Gelabert[2], Bleron Klaiqi[2]

[1]*Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*
[2]*Huawei Technologies Sweden AB, Kista, Sweden*
david.campoy.garcia@upc.edu, jordi.perez-romero@upc.edu, sallent@tsc.upc.edu, antoni.gelonch@upc.edu, xavier.gelabert@huawei.com, bleron.klaiqi@huawei.com

***Abstract*—Radio Access Network (RAN) disaggregation consists in splitting the baseband functionalities of a base station between centralized processing centers and distributed locations close to the radio units. This is becoming a key trend in the roadmap from fifth to sixth generation (6G) systems because it can lead to a more efficient utilization of computational, storage and power resources. To achieve these improvements, a proper functional split selection is required, balancing the benefits of centralization with the costs of transporting data over the fronthaul (FH). In this context, this paper proposes a Deep Neural Network (DNN)-based solution to select the functional split on a per-cell basis with the objective of minimizing energy consumption. In order to find an optimal model, a genetic-based optimization procedure is used to adjust the weights and biases of the DNN given the non-differentiable nature of the loss function. Lastly, this work analyzes the optimality of DNN output split combinations compared to an exhaustive search approach under given problem constraints.**

***Keywords*— Beyond 5G, RAN disaggregation, functional split selection, Deep Neural Network.**

I. Introduction

As we transition to future mobile communications systems beyond the fifth generation (B5G) and towards the sixth generation (6G), the Radio Access Network (RAN) is expected to adopt a more flexible, intelligent, and disaggregated design, as envisaged for example by the Open RAN (O-RAN) Alliance [1]. The introduction of Artificial Intelligence (AI)-based applications into network management drives this transformation, enabling autonomous data processing, informed decision-making for network elements, and more efficient RAN operations. Containerized AI-based applications can be embedded within the near real-time RAN Intelligent Controller (near RT-RIC) and non-real-time RAN Intelligent Controller (non-RT-RIC) described in the O-RAN architecture. Furthermore, flexible and disaggregated designs are also facilitated by the availability of computing and storage resources across different parts of the network infrastructure, forming a compute continuum that ranges from the cell sites at the network edge up to centralized cloud data centers.

A Base Station (BS) includes a Remote Radio Head (RRH) for Radio Frequency (RF) processing and a Baseband Unit (BBU) for signal processing functions and upper layers of the radio interface protocol stack. While earlier generations collocated RRHs and BBUs at cell sites, the current trend to achieve a disaggregated and softwarized RAN is to centralize some baseband (BB) functions in data processing centers connected to cell sites via fronthaul (FH) links. This approach enables resource pooling, statistical multiplexing, and collaborative techniques for functions such as interference coordination. Moreover, this architecture efficiently handles the complex requirements of advanced features in B5G systems, like massive Multiple Input Multiple Output (MIMO).

The split of RAN functionalities (e.g. orthogonal frequency division multiple access (OFDMA) processing, channel coding, digital modulation, etc.) between the cell site and a central location is referred to as a *functional split*. The ability to dynamically change this split based on network conditions (e.g. traffic demands, FH bandwidth constraints, or computing capabilities) is known as flexible functional split management. This concept was first introduced by the iJOIN project in [2] through the so-called RAN as a Service (RANaaS) concept. Many surveys have analyzed possible split options, considering requirements, complexities, latencies, advantages, and disadvantages [3][4][5]. Our recent work [6] assessed the computational costs of the different BB functions at the physical layer and presented a model for characterizing computational and FH bandwidth requirements in both uplink (UL) and downlink (DL) transmissions with different functional splits. For each split option, the aggregated requirements at both the cell sites and central location were evaluated. Results showed that deciding the functional split embraces a trade-off between the benefits of centralization and the costs associated with FH data transmission between the cell sites and the centralized processing centers.

The problem of determining the optimal functional split under certain constraints has been addressed in recent literature. In [7], heuristic algorithms are proposed to solve a zero-one packing problem using a pure integer non-linear programming model. Similarly, authors in [8] approached the problem from a delay perspective, employing a minimum delay policy and numerical methods within a nonlinear program. Additionally, [9] presented a migration strategy aimed at reducing FH overload in split transitions. The authors in [10][11] propose heuristics to select the optimal functional split among four possibilities solving a mixed integer non-linear problem. Our prior work [12] provided an architectural framework for solving the functional split selection under varying load conditions and derived an initial optimization analysis subject to FH constraints using exhaustive search. However, as network configurations become more complex, conventional heuristic algorithms or exhaustive search optimization may struggle to efficiently explore the vast search space. In this direction, machine learning has emerged as a promising approach to solve such complex problems. Reinforcement Learning (RL) is commonly used in functional split selection to address dynamic discrete optimization problems. For instance,

[13] and [14] use Q-learning and SARSA policies to select the optimum functional split to minimize the system power consumption. Overprovisioning, virtual resource instantiation, reconfiguration, routing, and computing costs are also considered in [15], which builds an RL solution based on Q-learning and a regression-based neural network. In [16], an actor-critic framework is employed to solve a combinatorial problem for optimizing the energy and delay of central and distributed platforms subject to Quality of Service (QoS) constraints.

With the increasing options for defining the functional split associated with the disaggregation and softwarization of the B5G RAN together with the extensive use of highly computational demanding features such as mMIMO, the problem of determining the optimal functional split becomes increasingly challenging and has to deal with a huge number of possible network configurations. With multiple split options, cells, sectors and sites, traditional optimization techniques like heuristic algorithms can take considerable time to find a feasible solution, and RL algorithms are well-known to present significant challenges for large discrete action spaces [17]. Moreover, potential solutions based on supervised learning would require labeled training data that may not be readily available. In these cases, getting the labels with the optimum solutions for each training data sample would involve an exhaustive search optimization process, which can be computationally prohibitive for large datasets and large network architectures.

To address this limitation, the contribution of this paper is to propose a versatile deep learning solution relying on unsupervised training for optimal functional split selection, aiming to minimize energy consumption. Building on the work in [6] and [12], the proposed solution integrates a detailed characterization of the computational and FH costs into the training process to reduce the energy consumption of processing platforms. The use of a Deep Neural Network (DNN) with unsupervised training enables learning from unlabeled data, making it adaptable to a wide range of network architectures without the need for specific labelled training datasets. In addition, due to the non-differentiable nature of the problem's loss function, this paper also proposes the use of an evolutionary algorithm for optimizing the DNN weights, as an alternative to traditional gradient-based methods.

The rest of the paper is organized as follows: Section II presents the considered scenario and the optimization problem. Section III develops the proposed deep learning solution and the unsupervised training process. Finally, Sections IV and V present the results and conclusions, respectively.

II. Problem Statement

The considered scenario depicted in Fig. 1 assumes a 5G RAN composed of K_S sites numbered as k=1,...,K_S. Each site has K_{sec} sectors numbered as j=1, ..., K_{sec}, each one supporting a single 5G NR cell over a certain frequency and bandwidth. The BB computing resources are distributed between a central location, which includes the Baseband High (BBH) resources, and individual sites, which include the Baseband Low (BBL) resources. The BBL resources host the physical (PHY) layer functionalities of the site's sectors based on the selected functional split. Similarly, the BBH resources host the PHY layer functionalities of multiple sectors/sites, again determined by the selected functional split. The FH network interconnects sites with the central location. The specific FH topology may vary depending on the use case, but it generally involves a unit that multiplexes the data from the different sectors of a site, referred to as the Site Switch Unit (SXU) in Fig. 1, and a unit that multiplexes the data coming from different sites, referred to as the Central Switch Unit (CXU) in Fig. 1. An SXU and a CXU are interconnected via a FH link with maximum capacity $R_{FH,max}$ (Gb/s) in each direction, i.e. UL (from SXU to CXU) and DL (from CXU to SXU).

Let us denote as $\mathcal{S}$ the set of possible splits that can be selected for each sector and site. Then, $X_{k,j} \in \mathcal{S}$ denotes the functional split configured in the j-th sector and the k-th site. Additionally, we define the vector of selected functional splits per sector and site as $\mathbf{X}=\{X_{k,j} \mid k=1,...,K_S, j=1,...,K_{sec}\}$. Besides, it is also convenient to define $\mathbf{X}_k$ as the vector with the functional splits for all the sectors of the k-th site, denoted as $\mathbf{X}_k=\{X_{k,j} \mid j=1,...,K_{sec}\}$.

The amount of computing resources at BBH and BBL is modelled as a number of *cores*, a general term that reflects a partition of compute resources in smaller processing sub-units, which can be applied to different computing architectures, such as Central Processing Unit (CPU) cores, Digital Signal Processing (DSP) cores, Graphics Processing Units (GPU) cores, etc. Then, let us assume that the BBL at one site has a total of $n_{BBL,max}$ cores, each with computational capacity Y_{BBL} in Millions of Operations Per Second (MOPS). A core in one site can host BB functions of any of the sectors in the site. Similarly, the BBH at the central location has $n_{BBH,max}$ cores to serve the needs of the K_S sites (i.e. total pooling applies). The computational capacity of a core at the BBH is Y_{BBH} MOPS.

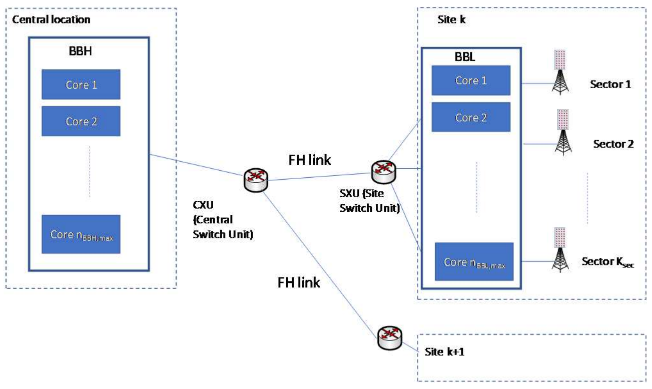

Fig. 1 High-level representation of the considered scenario.

A given split vector $\mathbf{X}$ determines the PHY layer functions executed at the BBL and BBH for each sector/site, as well as the data to be transferred over the FH in both UL and DL directions. The actual computational requirement at the BBH to support the configured splits in all the sectors/sites is denoted as $C_{BBH}(\mathbf{X})$, while the computational requirement at the BBL of the k-th site is $C_{BBL}(\mathbf{X}_k)$. The number of cores required at the BBH is $\lceil C_{BBH}(\mathbf{X})/Y_{BBH} \rceil$ where $\lceil x \rceil$ denotes the lowest integer that is greater than or equal to x. Similarly, the number of cores required at the BBL of site k is $\lceil C_{BBL}(\mathbf{X}_k)/Y_{BBL} \rceil$. To save energy, any excess cores beyond these requirements until $n_{BBL,max}$ and $n_{BBL,max}$ can be switched off. For modelling purposes, let us define as $\varepsilon=e_{BBL}/e_{BBH}$ the ratio between the energy cost of one active core at the BBL e_{BBL} and the corresponding cost of one active core at the BBH e_{BBH}. Since computational cost can be a major constraint at remote sites [18], provisioning energy resources at the BBH is assumed to be more cost-effective than at the BBL, implying $\varepsilon>1$. Moreover, the FH link bandwidth requirements between the CXU and the SXU of site k when implementing the functional splits $\mathbf{X}_k$ are denoted as $R_{FH,DL,k}(\mathbf{X}_k)$ and $R_{FH,UL,k}(\mathbf{X}_k)$, respectively, for DL and UL.

With the above notation, the optimization problem consists in finding the vector $\mathbf{X}$ that minimizes the total energy cost while maintaining the FH bandwidth requirements in DL and UL for each site below the FH link capacity. Mathematically, this is formulated as:

$$\min_{\mathbf{X}}\left(\left\lceil\frac{C_{BBH}(\mathbf{X})}{Y_{BBH}}\right\rceil+\varepsilon\sum_{k=1}^{K_s}\left\lceil\frac{C_{BBL}(\mathbf{X}_k)}{Y_{BBL}}\right\rceil\right) \tag{1}$$

s.t.

$$\left\lceil\frac{C_{BBH}(\mathbf{X})}{Y_{BBH}}\right\rceil\leq n_{BBH,\max} \tag{2}$$

$$\left\lceil\frac{C_{BBL}(\mathbf{X}_k)}{Y_{BBL}}\right\rceil\leq n_{BBL,\max},\forall k=1,\dots,K_s \tag{3}$$

$$R_{FH,DL,k}(\mathbf{X}_k)\leq R_{FH,\max},\forall k=1,\dots,K_s \tag{4}$$

$$R_{FH,UL,k}(\mathbf{X}_k)\leq R_{FH,\max},\forall k=1,\dots,K_s \tag{5}$$

The constraints (2) and (3) impose that the number of required cores in BBH and BBL, respectively, must be lower or equal than the maximum number of available cores. Similarly, the constraints (4) and (5) impose that the FH requirements must be lower or equal than the FH link capacity.

The optimization must consider various parameters that impact the computational and FH bandwidth requirements according to the model defined in Section V of [6]. Some of these parameters, such as subcarrier spacing, channel bandwidth, time division duplex (TDD) structure, and Fast Fourier Transform (FFT) samples, can be assumed constant or semi-static. However, others like the Physical Resource Block (PRB) occupation per cell, vary dynamically based on spatial and temporal traffic conditions. Therefore, the optimal functional split $\mathbf{X}$ should be determined whenever there are changes in these dynamic parameters or traffic conditions. This paper focuses on the optimization of $\mathbf{X}$ considering the variation in the number of occupied PRBs in UL and DL, assuming average values for code rates, modulation indexes, and MIMO layers per sector.

III. Deep Neural Network solution

A. Motivation

In the considered problem there are $|\mathcal{S}|$ possible functional splits that can be selected per sector and site. Therefore, the number of potential combinations that constitute the search space is $|\mathcal{S}|^{K_S K_{sec}}$, which can escalate quickly depending on the number of splits, sectors and sites. Thus, given the potentially large search space and the complex nature of the optimization problem, an AI-based solution based on deep learning is proposed here. Due to its ability to handle highly dimensional problems, deep learning strategies have gained a lot of interest in the recent years and have been applied to multiple problems in wireless networks [19][20]. Deep learning uses DNNs (i.e. neural networks with multiple layers of neurons) to approximate a complex function through a composition of linear and non-linear transformations conducted at the different layers. In this way, the DNN can extract features from the input data and eventually make decisions. The DNN behavior depends on the weights between neurons of different layers and on the biases introduced by each neuron. Then, to learn optimal values for these weights and biases, the DNN is trained using a dataset of experiences in the so-called training stage. Once trained, the DNN can be used in the so-called inference stage to obtain adequate outputs for any given input.

For the optimization problem considered here, the choice has been to use a DNN with unsupervised training. This approach is preferred for two reasons. First, unlike supervised learning, it does not require a labelled training dataset with the optimum split vectors $\mathbf{X}$. Obtaining such a dataset would involve exploring the entire search space, which can be computationally prohibitive. Second, the DNN can directly output the selected split for each sector/site, leading to a simpler architecture compared to reinforcement learning approaches, which would normally require a DNN with as many outputs as possible actions can be selected, i.e. the search space size.

B. DNN specification

Fig. 2 depicts the considered DNN model for dynamic functional split selection. The inputs of the neural network are the number of occupied PRBs in UL and DL for each cell of a sector/site denoted as $N_{PRB,DL,k,j}$ and $N_{PRB,UL,k,j}$. Therefore, the total number of inputs to the DNN is $2\cdot K_S\cdot K_{sec}$. Each input is injected to one of the neurons of the so-called input layer.

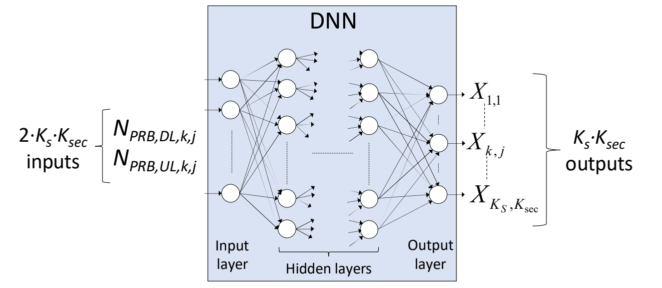

Fig. 2 DNN for dynamic functional split selection.

The outputs of the DNN are the various components $X_{k,j}$ of the vector $\mathbf{X}$. Each one is provided by a neuron of the so-called output layer. Thus, the number of neurons in this layer is $K_{sec}\cdot K_S$. A sigmoid activation function is applied at the output layer so that the output of a neuron is comprised between 0 and 1. Then, the output of each neuron is discretized in steps of $1/|\mathcal{S}|$ in order to map it to a specific split. In this way, if the output is between 0 and $1/|\mathcal{S}|$ the selected split is the first one in the set $\mathcal{S}$, if it is between $1/|\mathcal{S}|$ and $2/|\mathcal{S}|$ the selected split is the second one, and so on.

The DNN can be configured with multiple hidden layers, each one composed by a certain number of neurons. Fully connected layers are considered, meaning that all the neurons of one layer are connected with all the neurons of the subsequent layer. The resulting vector of a fully connected layer with N inputs and M outputs can be expressed as:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{6}$$

where $\mathbf{x} = [x_1,...,x_N]^T \in \mathbb{R}^{N\times 1}$ is the input vector to the layer containing the outputs of the N neurons of the preceding layer, $\mathbf{W} \in \mathbb{R}^{M\times N}$ is a matrix containing the weights applied to $\mathbf{x}$, $\mathbf{b} = [b_1,...,b_M]^T \in \mathbb{R}^{M\times 1}$ contains M biases for the output neurons, $\mathbf{y} = [y_1,...,y_M]^T \in \mathbb{R}^{M\times 1}$ is the output vector representing the values of the M output neurons of the layer, and $f(\cdot)$ is the so-called activation function, which can take several forms, such as a sigmoid function, a hyperbolic tangent (tanh) function, a rectified linear unit (ReLu), etc.

The activation function, the number of hidden layers, and the number of neurons per layer are design-time configuration hyperparameters of the neural network. In turn, the weights and biases to be applied by the different neurons have to be learnt during the training stage.

C. Training process

The unsupervised training process uses a training dataset of T input samples denoted as z_i, where i=1, ..., T. Each sample z_i consists of $2\cdot K_S\cdot K_{sec}$ elements, corresponding to the DNN's input as described in the previous section. The DNN, with a given set of weights and biases, processes each training sample to generate an output $\mathbf{X}$ that represents a combination of functional splits for each sector and site. This output $\mathbf{X}$ is evaluated using a loss function $L(\mathbf{X})$, which measures the quality of the output in relation to the optimization problem. Finally, the training objective is to find the optimal set of weights and biases that minimizes the loss function across all training samples.

For the optimization problem considered here, since the target is to minimize the total energy cost in (1) subject to the constraints (2)-(5), the loss function $L(\mathbf{X})$ is defined as the energy cost when the constraints are fulfilled and an infinite value when they are not, that is:

$$L(\mathbf{X}) = \begin{cases} \left\lceil \dfrac{C_{BBH}(\mathbf{X})}{Y_{BBH}} \right\rceil + \varepsilon \displaystyle\sum_{k=1}^{K_S} \left\lceil \dfrac{C_{BBL}(\mathbf{X}_k)}{Y_{BBL}} \right\rceil & \text{if } \mathbf{X} \text{ fulfils constraints} \\ \infty & \text{if } \mathbf{X} \text{ does not fulfil constraints} \end{cases} \tag{7}$$

For conciseness, the computational requirements $C_{BBH}(\mathbf{X})$ and $C_{BBL}(\mathbf{X}_k)$, as defined in (7), are calculated using the equations in Table 5-1 of [6], for each functional split $\mathbf{X}$ and its subset $\mathbf{X}_k$ at the BBH and BBL respectively. Similarly, the FH bandwidth requirements are derived from the formulas in Table 5-2 of [6].

The most usual strategies of finding the weights and biases that minimize the loss function are the gradient-based algorithms, such as the Stochastic Gradient Descent (SGD) [21]. SGD involves computing the loss function for each training sample, calculating the average gradient of this function for all the samples, and then adjusting the weights accordingly. This process is repeated iteratively until the loss function converges. However, SGD requires a differentiable loss function, which is not the case for the optimization problem in (7). For this reason, a genetic algorithm (GA)-based approach, inspired by recent work in [22], is adopted for this non-differentiable problem.

In the terminology of GAs, an *individual* or *chromosome* refers to a combination of weights/biases of the DNN. The individual is represented by a unidimensional array, so a preliminary step is to encode the weights and biases (i.e. matrices $\mathbf{W}$ and vectors $\mathbf{b}$) of all the layers in a single vector. In this way, different individuals represent different configurations of the DNN. Moreover, a *generation* refers to a group of N_c chromosomes, where N_c is the *population* size.

The GA begins by initializing a first generation of chromosomes with randomly assigned values. Then, each chromosome of the generation is evaluated individually. To evaluate a chromosome c, the DNN processes each training sample z_i using the weights and biases associated with c and generates an output $\mathbf{X}(z_i,c)$. The quality of the chromosome c is given by its fitness function $F(c)$, which is defined as the average loss function of this chromosome for all the training samples T as:

$$F(c) = \frac{1}{T}\sum_{i=1}^{T} L(\mathbf{X}(z_i,c)) \tag{8}$$

Next, the GA creates a new generation of chromosomes based on individuals of the previous generation and genetic operators such as selection, crossover and mutation. First, a fraction of best-performing individuals given by the *elite selection rate* are directly copied. Then, pairs of chromosomes (parents) are selected based on their fitness and are recombined through crossover to create new offspring. The ratio of individuals obtained by crossover in the new generation is given by the *crossover rate*. The remaining ones are obtained by mutation, which is applied to the previous generation individuals with certain *mutation probability* to introduce genetic diversity. This consists in modifying some of the entries in a chromosome using probabilistic distributions (e.g. Gaussian, uniform, etc.). As a result of these operators, a new generation of N_c chromosomes is obtained. This iterative process of selection, crossover, and mutation continues until a termination condition, such as a maximum number of generations or runtime, is met. Then, the algorithm outputs the best chromosome found across all generations, which corresponds to the solution with the lowest fitness.

IV. RESULTS

This section presents the considered scenario, the training process assessment, and the performance results of the DNN-based functional split selection approach, targeted to study its capability to provide optimum solutions.

A. Scenario

To evaluate the DNN-based functional split selection model, we consider a scenario with K_s=1 site with K_{sec}=3 sectors. The set $\mathcal{S}$ of candidate functional splits for a sector is shown in Fig. 3 and considers different splitting points denoted as (8, 7a, 7b, 7c, 7d, 6) where split 8 places all PHY layer functions at the BBH, split 6 places them at the BBL, and splits 7a to 7d are ordered from more to less functions at the BBH. The 5G NR cells operate with TDD and the parameters and algorithms listed in Table I. Furthermore, to characterize the computational requirements and the UL/DL FH bandwidth requirements for a given offered load of a sector, we make use of the model presented in [6] and the algorithms in Table II of [12], which shows the required operations per algorithm execution (see [6] for details and references on each algorithm).

For the computational capacity, we assume Intel Xeon 6140 processors [23] at both BBH and BBL, each with 18 cores that can be dynamically switched on and off depending on the needs. The computational capacity per core is $Y_{BBL}=Y_{BBH}=58240$ MOPS. For this single site scenario, the number of cores available at the BBL is $n_{BBL,max}=3\cdot 18=54$ (i.e. one processor of 18 cores per sector), and the number of cores at the BBH is set to the same value $n_{BBH,max}=54$. Results are presented for $\varepsilon=2$ and an FH capacity of $R_{FH,max}=40$ Gb/s.

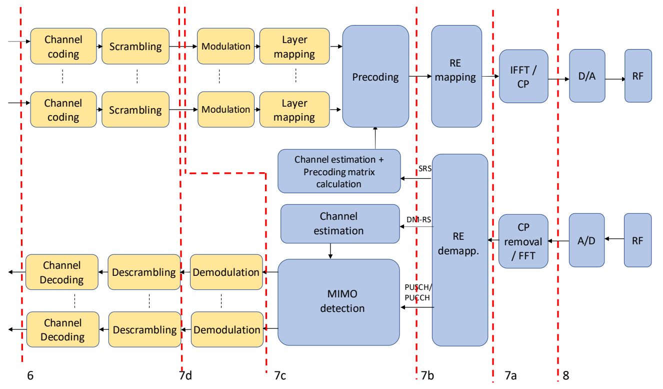

Fig. 3 PHY layer functions in DL transmission and UL reception and possible functional splits [6].

The DNN and GA were implemented using the MATLAB Deep Learning Toolbox and the MATLAB Global Optimization Toolbox, respectively. Detailed information about the different configuration options of the GA can be found in [24].

B. *DNN-based functional split selection training process*

In the following results, we explore the behavior of the training process using a randomly generated dataset composed of T=1000 traffic mixes across the three sectors. A traffic mix specifies the offered DL and UL load in each sector normalized to the sector capacity. Table II lists the GA hyperparameters and DNN structure that achieved the best results after extensive tuning to balance quality and training time.

Fig. 4 illustrates the evolution of the *best fitness* and *mean fitness* values throughout the DNN training process. The best fitness refers to the minimum fitness value (8) attained by the best individual in each generation[1], while the mean fitness is the average fitness of all individuals in a generation. The GA preserves the top 5% of individuals (i.e. elite individuals) from each generation, therefore the best fitness can only decrease or remain the same. As shown in Fig. 4, the best fitness steadily decreases in the initial generations and stabilizes around 38.52. Regarding the mean fitness, it is progressively decreased from the 1st generation to approximately the 130th generation. However, beyond this point, the best individual already provides a near-optimal solution, making small mutations and crossovers more likely to produce infeasible solutions (i.e., those not fulfilling the problem constraints (2)-(5)). As a result, the mean fitness begins to increase significantly across generations.

C. *DNN-based functional split selection performance results*

Once the model is trained, the performance of the DNN is assessed using an evaluation dataset consisting of 1000 new samples not included in the training dataset. To establish a reference for comparison, the optimum solution for each traffic mix is used as a performance upper bound. In the considered scenario, the optimum solution per traffic mix can be found using an exhaustive search. The following metrics were used to assess the DNN's performance:

- *Individual degradation*: the percentage increase in energy cost between the DNN solution and the optimal solution for a given traffic mix.
- *Average degradation*: the average individual degradation across all samples.
- *Optimum ratio*: the percentage of optimum solutions found by the DNN across all samples.
- *Unfeasible ratio*: the percentage of solutions found by the DNN that violate constraints (2)-(5) across all samples.

TABLE I. CELL PARAMETERS [6].

Parameter	Value
Frequency	3.5 GHz
Subcarrier spacing	Δf=30 kHz
Channel bandwidth and number of PRBs	100 MHz, N_{PRB}=273 PRBs
Slot duration	T_{slot}=0.5 ms
Symbol duration	$T_s=T_{slot}/14$=35.7 µs
IFFT/FFT size	N_{FFT}=4096
Number of time samples of cyclic prefix	$N_{CP}=N_{FFT}/14$=292
Number of antennas at the BS	B=64
Number of antennas at the UEs (equivalently number of layers)	U=16 (aggregate number for all UEs)
Length of training sequences for channel estimation	L_{SRS}=12, $L_{DM\text{-}RS}$=8
Modulation and coding scheme	64QAM (m=6 bits/symbol) and code rate r=666/1024.
TDD configuration	3 DL slots, 1 special slot (with 10 DL symbols, 2 guard symbols and 2 UL symbols), 1 UL slot
Number of bits to encode an IQ sample	n_{IQ}=32
Number of bits to encode a softbit at the output of the demodulator	n_{soft}=8
Channel decoding parameters (see section IV.C.2 of [6])	$BLER$=0.1, I_{max}=10, d_c=2, n=8424/r=12952, d_v=0.699, E=4528

TABLE II. HYPERPARAMETER CONFIGURATION

Parameter	Value
GA	
Mutation function	Uniform with prob. 0.01.
Crossover function	Scattered
Crossover rate	0.2
Elite selection rate	0.05
Selection function	Roulette
Population of a generation	200
Number of generations	10000
DNN structure	
Number of hidden layers and neurons	2 (10,6)
Activation function for hidden layers	Sigmoid
Activation function for output layer	Sigmoid

[1] For practical computation, the ∞ value in (7) is set to 200.

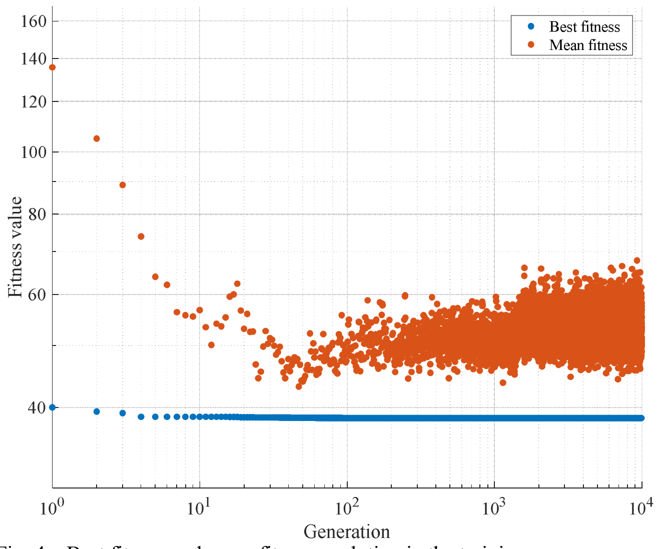

Fig. 4 Best fitness and mean fitness evolution in the training process.

Table III presents the performance metrics of the trained DNN (configured with the weights/biases of the best individual found after the training process) when applied to both training and evaluation datasets. For the training dataset, the GA successfully adjusted the DNN's weights and biases to avoid unfeasible solutions. In addition, the DNN achieved the optimal solution in 62.1% of the samples, with an average degradation of 2.63%. Similar performance was observed for the evaluation dataset in terms of average degradation and optimum ratio. However, a small percentage of unfeasible solutions, i.e. 0.9%, were observed, in which the FH bandwidth exceeds the maximum limit.

Fig. 5 compares the cumulative distribution function (CDF) of the energy cost for the DNN solutions and optimal solutions. The DNN closely approximates the optimal solutions, with minor differences observed primarily for traffic mixes with low PRB occupation. In these cases, the BB functions require few computations, and a change in the split configuration have minimal impact on core utilization. This poses a challenge for the GA to find significant energy improvements through small modifications in the individuals, so it gets stuck in suboptimal splits. Conversely, for traffic mixes with high PRB occupation, which require more computations in both BBL and BBH, the energy is more sensitive to the selected split. Therefore, the GA is more effective in finding optimal splits.

TABLE III. BEST INDIVIDUAL METRICS FOR THE TRAINING AND EVALUATION DATASETS

Metric	Value
Training dataset	
Average degradation	2.63%
Optimum ratio	62.1%
Unfeasible ratio	0%
Evaluation dataset	
Average degradation	2.85%
Optima ratio	61.8%
Unfeasible ratio	0.9%

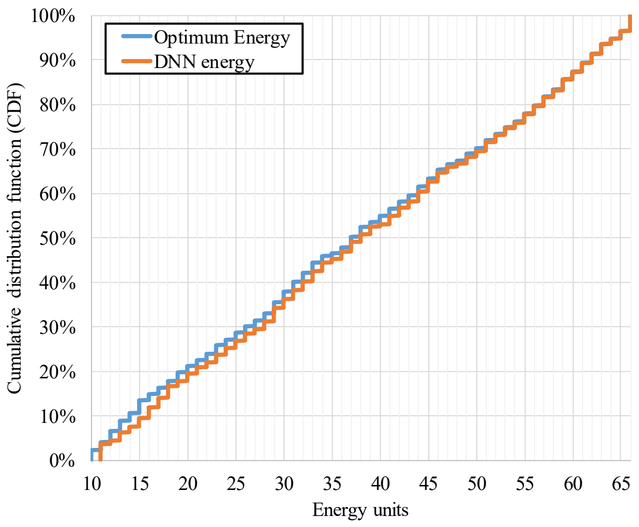

Fig. 5 Cumulative distribution function (CDF) of the energy cost over the evaluation dataset.

Fig. 6 presents a histogram plot of the individual degradation for the DNN solutions in the evaluation dataset, excluding the 0.9% of unfeasible solutions. The plot reveals that 95.5% of the DNN solutions have a degradation of less than 15%, and only 3.6% of the cases show greater degradation. It is worth highlighting that, in traffic mixes where the DNN solution is suboptimal, the absolute energy excess typically ranges from 1 to 3 units. This energy excess has different impact on the individual degradation depending on the traffic mix, since a 3-unit energy excess has a greater impact on traffic mixes with lower optimum energy consumption (e.g. 12 units) than on those with higher optimum consumption (e.g. 63 units).

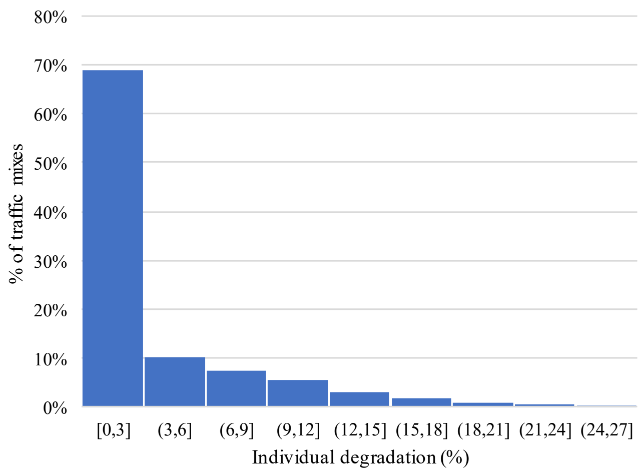

Fig. 6. Individual degradation histogram for the evaluation dataset.

V. CONCLUSIONS

This paper has presented an AI-based solution for dynamic functional split selection in disaggregated RANs with varying traffic load conditions. The proposed solution aims to select the optimal functional split per sector and site to minimize the total energy cost associated with the BBH and BBL computational resources subject to FH capacity constraints. After formulating the optimization problem, a DNN-based approach with unsupervised training that relies on a GA has been proposed to handle the potentially large search space, the prohibitive computational cost of obtaining labeled data, and the non-differentiable nature of the loss function.

The performance evaluation of the AI-based approach has been conducted by comparing the results obtained with the DNN against those of the optimum configurations. Overall, the DNN results lead to energy costs that are, on average, only 2.85% higher than the optimum. Moreover, in 61.8% of the cases the DNN is able to find optimum solutions. Overall, 95.5% of the solutions achieved an individual degradation of less than 15% with respect to the optimum, and only 0.9% were found to be unfeasible. Additionally, it has been found that the DNN performs better in scenarios with high PRB occupation.

Based on the results presented here, future work will assess the benefits, performance, and generalization capability of the genetic-based DNN under different FH bandwidth constraints and cell parameters (e.g., changing the number of antennas at the base station, or TDD configuration).

Acknowledgement

The work by J. Pérez-Romero, O. Sallent and D. Campoy has been partially funded by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096034 (VERGE project), grant ARTIST (PID2020-115104RB-I00) and TRAINER-6G (PID2023-146748OB-I00) funded by MCIN/AEI/10.13039/501100011033 and by ERDF/EU and by Huawei Technologies. The work by X. Gelabert and B. Klaiqi has been partially funded by the European Union's Horizon Europe research and innovation programme (HORIZON-MSCA-2021-DN-0) under the Marie Skłodowska-Curie grant agreement No. 101073265 [25]. The expressed views and opinions are, however, those of the authors only and may not reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] A. Akman *et al.*, "O-RAN Minimum Viable Plan and Acceleration towards Commercialization," O-RAN Alliance White Paper, 2021.

[2] D. Sabella *et al.*, "RAN as a Service: Challenges of Designing a Flexible RAN Architecture in a Cloud-based Heterogeneous Mobile Network," in *Proc. Future Netw. & Mobile Summit*, 2013.

[3] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 1, pp. 146–72, 2019.

[4] M. Peng *et al.*, "Recent Advances in Cloud Radio Access Networks: System Architectures, Key Techniques and Open Issues," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 3, pp. 2282-2304, 2016.

[5] B. Khan *et al.*, "Survey on 5G Second Phase RAN Architectures and Functional splits," *TechRxiv*, 2022.

[6] J. Pérez-Romero *et al.*, "A Tutorial on the Characterisation and Modelling of Low Layer Functional Splits for Flexible Radio Access Networks in 5G and Beyond," *IEEE Commun. Surveys & Tutorials*, 2023.

[7] Haoran Mei, Limei Peng, "Flexible functional split for cost-efficient C-RAN," *Comp. Commun.*, Volume 161, pp 368-374, 2020.

[8] I. Koutsopoulos, "The Impact of Baseband Functional Splits on Resource Allocation in 5G Radio Access Networks," *IEEE Conf. on Comp. Commun.*, pp. 1-10, 2021.

[9] A. M. Alba, J. H. G. Velásquez and W. Kellerer, "An adaptive functional split in 5G networks," *IEEE Conf. Comp. Commun.*, pp. 410-416, 2019.

[10] A. M. Alba, S. Janardhanan and W. Kellerer, "Enabling Dynamically Centralized RAN Architectures in 5G and Beyond," *IEEE Trans. Netw. and Service Manage.*, vol. 18, no. 3, pp. 3509-3526, 2021.

[11] A. M. Alba and W. Kellerer, "Dynamic Functional Split Adaptation in Next-Generation Radio Access Networks," *IEEE Trans. on Netw. and Service Manage.*, vol. 19, no. 3, pp. 3239-3263, 2022.

[12] J. Pérez-Romero *et al.*, "Low Layer Functional Split Management in 5G and Beyond: Architecture and Self-adaptation," *Int. Symp. on Wireless Commun. Systems*, pp. 1-6, 2024.

[13] D. A. Temesgene, M. Miozzo and P. Dini, "Dynamic Functional Split Selection in Energy Harvesting Virtual Small Cells Using Temporal Difference Learning," *IEEE Annu. Int. Symp. on Personal, Indoor and Mobile Radio Commun.*, pp. 1813-1819, 2018.

[14] D. A. Temesgene *et al.*, "Distributed Deep Reinforcement Learning for Functional Split Control in Energy Harvesting Virtualized Small Cells," *IEEE Trans. on Sustain. Comput.*, vol. 6, no. 4, pp. 626-640, 2021.

[15] F. W. Murti *et al.*, "Learning-Based Orchestration for Dynamic Functional Split and Resource Allocation in vRANs," *Joint European Conf. Netw. and Commun. & 6G Summit*, pp. 243-248, 2022.

[16] S. Mollahasani *et al.*, "M. Energy-Aware Dynamic DU Selection and NF Relocation in O-RAN Using Actor–Critic Learning," *Sensors*, 2022.

[17] G. Dulac-Arnold *et al.*, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learn.,* vol. 110, pp. 2419-2468, 2021.

[18] J. Liu *et al.*, "Graph-based framework for flexible baseband function splitting and placement in C-RAN," *IEEE Int. Conf. Commun.*, 2015.

[19] C. Zhang, P. Patras, H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Comms. Surveys & Tutorials*, vol. 21, no. 3, 2019.

[20] A. Grönland *et al.*, "Constrained Deep Reinforcement Learning for Fronthaul Compression Optimization," *IEEE ICMLCN'24*, May 2024.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.

[22] E. Galván and P. Mooney, "Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges," *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 476-492, 2021.

[23] J. K. Chaudhary *et al.*, "C-RAN Employing xRAN Functional Split: Complexity Analysis for 5G NR Remote Radio Unit," *European Conf. on Netw. and Commun.*, pp. 580-585, 2019.

[24] Mathworks, "Genetic Algorithm Options," [Online]. Available: https://es.mathworks.com/help/gads/genetic-algorithm-options.html

[25] Z. Vujicic *et al.*, "Towards Virtualized Optical-Wireless Heterogeneous Networks," IEEE Access, pp. 1–1, 2024.