Using Early-Exit Deep Neural Networks to Accelerate Spectrum Classification in O-RAN

Roberto Goncalves Pacheco, Rodrigo de Souza Couto, Sahar Hoteit

Using Early-Exit Deep Neural Networks to Accelerate Spectrum Classification in O-RAN

Roberto G. Pacheco*, Rodrigo S. Couto †, and Sahar Hoteit‡§
* Universidade Federal Fluminense - Rio das Ostras, RJ, Brazil
† Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - Rio de Janeiro, RJ, Brazil
‡ Universite Paris-Saclay, CNRS, CentraleSupélec, L2S, Gif-sur-Yvette, France
§ Institut Universitaire de France (IUF), France
Emails: robertopacheco@id.uff.br, rodrigo@gta.ufrj.br, sahar.hoteit@centralesupelec.fr

***Abstract*—O-RAN architecture introduces a new level of flexibility in managing Radio Access Networks (RANs), facilitating the development of different applications. One of these applications is spectrum sharing, in which cellular traffic can share the unlicensed band with WLAN technologies, such as Wi-Fi. A key component of this application is a spectrum classification unit that identifies the communication technology used in the medium to support decision making in the RAN. This classification can be performed using Deep Neural Networks (DNNs) that receive I/Q samples and infer which communication technology is generating the traffic. Despite the high accuracy of DNNs in this task, the inference must be performed quickly to allow timely action to avoid interference. One promising approach to enhancing the performance of DNNs is to use early-exit DNNs (EE-DNNs), which are designed to reduce computations by allowing the inference process to terminate at intermediate layers when a certain confidence level is achieved. In this paper, we explore the application of EE-DNNs for spectrum classification by applying early exits to the Convolutional Neural Network (CNN) used by the ChARM (Channel-Aware Reacting Mechanism) framework. Using the ChARM dataset, we show that an EE-DNN can accelerate inference by 10% and even achieve higher accuracy than a conventional CNN by approximately 2%.**

I. Introduction

O-RAN (Open Radio Access Network) architecture aims to enhance interoperability, flexibility, and innovation within Radio Access Networks (RANs). By opening and disaggregating proprietary and closed interfaces in mobile network equipment, O-RAN provides a more open and collaborative ecosystem than traditional RANs [1]. In O-RAN, the Base Station (BS) comprises three key components: O-RU (O-RAN Radio Unit), O-DU (O-RAN Distributed Unit), and O-CU (O-RAN Central Unit), aligned with the Split Option 7.2x as defined by the 3GPP New Radio (3GPP NR) specifications. The O-RU handles radio frequency operations and lower physical layer tasks, while the O-DU manages the upper physical layer and link layer, and the O-CU is responsible for the networking layer.

The key O-RAN components are managed by Non-Real-Time (Non-RT) and Near-Real-Time (Near-RT) RAN Intelligent Controllers (RICs), which are critical for effective O-RAN management and orchestration, leveraging machine learning to enhance network performance and adaptability. The Non-RT RIC provides a global infrastructure view and executes rApps, running on timescales exceeding one second. Conversely, the Near-RT RAN Intelligent Controller (Near-RT RIC) manages specific RAN components through xApps, operating within timescales from ten milliseconds to one second [1]. Near-RT RICs handle user sessions and medium access, employing xApps for tasks such as load balancing, scheduling, and RAN slicing [2]. For instance, xApps may utilize MAC-level Key Performance Indicators (KPIs) for PRB (Physical Resource Block) allocation or determine the optimal BS for user equipment (UE) assignment. Non-RT RICs, through rApps, configure xApps by setting policies, models, and parameters, and can dictate the machine learning models used by xApps and their deployment locations.

In the literature, different rApps and xApps have been proposed. Baldesi *et al.* [3] propose a framework to allow spectrum sharing in O-RAN infrastructures. The idea is to allow coexistence of different wireless technologies (e.g., Wi-Fi and LTE) by sensing the medium and configuring O-RUs and O-DUs to avoid interference and implement spectrum sharing policies. The main component of the proposed framework is the `ChARM` (Channel-Aware Reacting Mechanism) xApp. This xApp has a Spectrum Classification Unit (SDU) that communicates with the O-DU to receive, via the O-DU, I/Q samples collected from the O-RU. The SDU runs a Deep Neural Network (DNN) to classify the spectrum and infer which wireless technology is being employed at specific frequencies in the O-RU. This information is used by another module in this xApp, called Policy Decision Unit (PDU), to reconfigure the parameters from the O-RU and the O-DU. For example, if an LTE communication is using an unlicensed band and the xApp detects the presence of Wi-Fi traffic, the PDU can change the center frequency used by LTE.

The `ChARM` xApp needs to perform a fast computation to react accordingly to changes in the state of the spectrum. As such, the DNN used for spectrum classification must perform a fast inference. There are different proposals to accelerate DNN inference, such as simplifying the DNN model by compression and using specialized hardware [4]. In this work, we explore the technique of Early-exit Deep Neural Networks (EE-DNN) to accelerate spectrum classification. EE-DNNs try to reduce the time and energy consumption by stopping the inference in the early layers [5]. This concept is widely explored in the areas of image classification. However, there is no work in

the literature using EE-DNNs for spectrum classification.

The purpose of this work is to show the effectiveness of EE-DNNs for spectrum classification. Hence, we insert early exits into the Convolutional Neural Network (CNNs) used by the `ChARM` proposal in [3] and compare it to the original CNN. Our results, evaluated using the `ChARM` dataset [6], show that an EE-DNN decreases the processing cost by approximately 10% when compared to the original DNN. In addition, EE-DNN even shows an improvement in accuracy, due to its reduction in the problem of overthinking [7].

This paper is structured as follows. Section II reviews related work, while Section III describes early-exit DNNs. Section IV overviews the ChARM dataset. Section V evaluates the performance of EE-DNNs in the context of spectrum classification, while Section VI concludes the paper and outlines future research directions.

II. Related Work

Many studies in O-RAN leverage machine learning solutions, such as Deep Reinforcement Learning (DRL) [2], [8], Graph Neural Networks (GNNs) [9], Residual Networks (RNs) [3], and Recurrent Neural Networks (RNNs) [10]. Given the scale and stringent timing requirements of O-RAN application, specially those in Near-RT RICs, ensuring fast inference performance of neural networks in this context is crucial [11]. To address this challenge, Early-exit Deep Neural Networks (EE-DNNs) is a promising approach, since EE-DNNs have shown significant performance improvements in CNNs for image classification, by reducing inference time and computational load while maintaining accuracy [12], [13]. Despite EE-DNNs being widely employed in image classification [14]–[16], this approach has not yet been explored for spectrum classification in the context of RANs. Therefore, this study aims to bridge this gap by demonstrating the applicability of EE-DNNs to spectrum classification tasks within RAN environments, highlighting their potential to enhance high performance of O-RAN applications.

Beyond its applicability in spectrum classification in O-RAN, DNNs can also be employed on other wireless systems. For example, CNN and LSTMs (Long Short-Term Memory) models are applied to classify the type of signal modulation, which is a task called Automatic Modulation Classification (AMC) [17]–[19]. Recently, EE-DNNs were used in AMC, showing [20], [21] that this type of neural network can induce a significant reduction in inference time without compromising accuracy. The knowledge of modulation provided by AMC can help distinguish between different wireless technologies. However, the AMC task is distinct from the spectrum classification considered in this work, as both LTE and Wi-Fi, among other technologies, can utilize the same modulation techniques, such as OFDM and QAM [22], [23].

III. Early-Exit DNNs for Spectrum Classification

Early-exit DNNs (EE-DNNs) are designed with multiple early-exit branches integrated into their intermediate layers, enabling inference to be concluded at these intermediate stages, as illustrated in Figure 1. The early classification in the intermediate stages can reduce the required computation to classify an input, accelerating the inference. The idea behind EE-DNNs is that some inputs can be easy to classify and thus do not require processing all the DNN layers. For example, in the domain of image classification, distorted images may need to process more layers than a pristine ones [15].

Once trained, EE-DNNs can receive an input to classify. In this paper, the EE-DNN receive a stream of I/Q samples as input. The input is processed, layer-by-layer, until it reaches an early-exit branch. At this exit branch, a fully-connected layer generates a logit vector $\boldsymbol{z}_i$, from which a probability vector $\boldsymbol{p}_i$ is derived using a softmax layer:

$$\boldsymbol{p}_i = \text{softmax}(\boldsymbol{z}_i) \propto \exp(\boldsymbol{z}_i), \tag{1}$$

where the exponential function is applied element-wise. The probability vector $\boldsymbol{p}_i$ consolidates the probabilities that an input belongs to predefined classes. The class with the highest probability in $\boldsymbol{p}_i$ corresponds to the predicted class, and its confidence level is indicated by the maximum probability $\max \boldsymbol{p}_i$. During inference, the input is processed until it reaches an exit branch. If the confidence level provided by the exit branch exceeds a predefined threshold, the exit branch classifies the input, and the inference process terminates. Consequently, the input sample bypasses subsequent layers, reducing processing delay and computational load. Alternatively, if the confidence value is below the threshold, the input continues through subsequent layers until it encounters the next exit branch, following the same procedure as described previously. If no exit branch confidence exceeds the threshold, the input is classified by the conventional DNN's output layer.

TABLE I
Convolutional Neural Network (CNN) employed in [3].

Layer	Output dim
Input	2 x 20,000
Conv/ReLU	7 x 20,000
MaxPool	7 x 10,000
Conv/ReLU	7 x 10,000
MaxPool	7 x 2,000
Conv/ReLU	7 x 2,000
MaxPool	7 x 1,000
Conv/ReLU	7 x 1,000
MaxPool	7 x 200
Conv/ReLU	7 x 200
MaxPool	7 x 100
Conv/ReLU	7 x 100
MaxPool	7 x 20
Conv/ReLU	7 x 20
MaxPool	7 x 10
FC/Tanh	18
FC/Tanh	16
FC/Softmax	3

In this work, we add two exit branches in the DNN presented in Table I and employed in [3]. Figure 1 illustrates the employed EE-DNN. The first one is placed after the second MaxPool layer (i.e., the one with output dimension 7 x 2,000). The second one is placed after the fourth MaxPool layer (i.e., the one with output dimension 7 x 200). We chose this

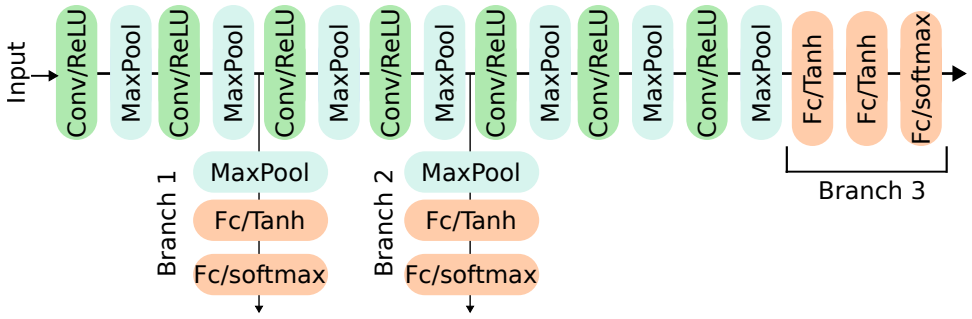

Fig. 1. The EE-DNN proposed in this work.

exit branch placement to balance the number of convolutional layers between each exit branch. Therefore, the number of convolutional layers between the input and Branch 1 is equal to the number of layers between Branch 1 and Branch 2. Since the total number of convolutional layers is not divisible by three, Branch 3 has an additional convolutional layer. This decision aligns with the concept of EE-DNNs, which aim to classify as many inputs as possible in the earlier exit branches, leaving the more complex inputs for the final exit branch. Another way of balancing the number of convolutional layers between exit branches would be to add six exit branches to the original DNN. However, this would result in a high overhead when applying the concept of EE-DNNs.

In Figure 1, each exit branch includes an additional MaxPool layer to further reduce the dimensionality of the input data, followed by two fully connected (FC) layers with tanh activation functions, and concluding with a softmax layer for classification. This structure not only enables faster inference by providing early exits, but also ensures efficient processing of the input data by reducing its dimensionality at each exit branch. In this work, we call the exit of the main DNN backbone as Branch 3. We train the early-exit DNN following the methodology employed in [5], [12], using the dataset described in Section IV. In this work, the EE-DNN receives a stream of I/Q samples and classifies its signal as Wi-Fi, LTE, or Clear (i.e., is related to the background noise, when there is no LTE or Wi-Fi signals).

IV. ChARM Dataset

The ChARM dataset [6] consists of spectrum data, in which each input is an I/Q sample derived from LTE and Wi-Fi traffic, along with background noise. An I/Q sample captures an instantaneous snapshot of the modulated signal, containing both the In-phase (I) and Quadrature (Q) components. The I component represents the amplitude of the baseband signal that is in-phase with the I carrier (0-degree phase), while the Q component represents the amplitude of the baseband signal that is in-phase with the Q carrier (90-degree phase).

The data was collected at the Colosseum testbed, using a central frequency of 5.2GHz and a bandwidth of 20MHz. The dataset is available at the Northeastern University Digital Repository Service (DRS) [6] and contains the following types of traffic:

- LTE and Wi-Fi traces with idle traffic. In this case, there is only control traffic from the BS (LTE) or the access point (Wi-Fi);
- LTE and Wi-Fi traces with a 1Mbps flow between the BS and the UE, generated using iperf3;
- LTE and Wi-Fi traces with ping flooding with 1KB packets between the BS and the UE, representing bursty traffic with high throughput;
- LTE and Wi-Fi traces with 300-byte packets of ping between the BS (or the access point) and the UE, representing bursty traffic with low throughput;
- Background noise, collected when there is no LTE or Wi-Fi communication. This data corresponds to the Clear category previously mentioned.

In this paper, we employ this dataset to train the EE-DNN to classify traffic types based on received streams of I/Q samples. Our EE-DNN model can classify the traffic in Clear, Wi-Fi, and LTE. For training and test purposes, we split this dataset into 50%/25%/25% for train/validation/test, according to the methodology presented in [3] and available in an open repository[1].

V. Results

This section evaluates the performance of the EE-DNN for the spectrum classification task. To this end, we compare the performance of the EE-DNN with the conventional DNN (i.e., with no exit branches) presented in Table I. This evaluation is performed using the test set for different values of confidence thresholds in the exit branches. In this experiment, the same threshold is employed in both exit branches. We provide the code developed for this paper in an open repository[2].

A. Accuracy, Precision, and Recall

This section compares the performance of EE-DNN with the conventional DNN in terms of the accuracy, precision, and recall. For a given threshold, the accuracy is defined as the fraction of test inputs correctly classified, regardless of the exit branch that classifies the input. Precision measures the proportion of correctly predicted positive examples out of the total predicted positives, indicating the model's ability

[1]https://github.com/lucabaldesi/charm_trainer/tree/c06ce57de5842951d2ff24b0b7fcdfc314044313

[2]https://github.com/GTA-UFRJ/ORAN_DNN

to avoid false positives. Recall measures the proportion of correctly predicted positive examples out of the total actual positives, reflecting the model's ability to identify all positive examples. Let $\mathcal{C}$ represent the set of classes, with $|\mathcal{C}|$ indicating the number of classes. In our case, we have $|\mathcal{C}| = 3$ classes: *Clear*, *LTE*, and *Wi-Fi*. We compute the above metrics in the following way:

$$\text{Accuracy} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}, \quad (2)$$

$$\text{Precision} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\frac{TP_i}{TP_i + FP_i}, \quad (3)$$

$$\text{Recall} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\frac{TP_i}{TP_i + FN_i}, \quad (4)$$

where TP_i is the number of true positive of i-th class, while TN_i is the number of true negative of i-th class, FP_i is the number of false positive of i-th class and FN_i is the number of false negative of i-th class.

Figure 2 shows the comparison in terms of accuracy between the EE-DNN and the conventional DNN. Notably, the DNN exhibits a constant accuracy curve, indicative of its single output layer architecture where all inputs are classified independent of the threshold. Thus, it underscores the independence of DNN performance with respect to thresholds.

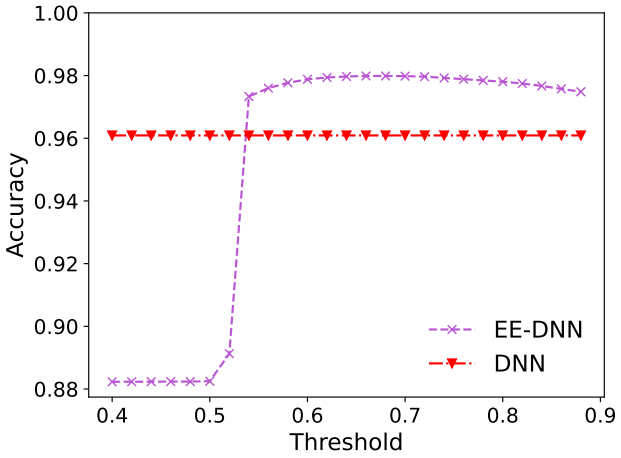

Fig. 2. Accuracy on EE-DNN and conventional DNN.

Figure 2 shows that, for lower threshold values, the EE-DNN initially exhibits lower accuracy compared to the conventional DNN. This outcome arises because the first exit branch can classify the majority of inputs, including unreliable inputs, as it operates with a less stringent threshold. Consequently, the performance of the EE-DNN is constrained by the classification capability of the first exit branch. However, after a more stringent threshold, the performance is even better than the original DNN. This is explained since stopping the inference earlier can reduce the overthinking problem that can happen when a sample is processed by multiple layers [7], [24]. The overthinking problem in EE-DNNs occurs when an input is misclassified by the final exit of an early-exit DNN but could have been correctly classified by an earlier exit, resulting in unnecessary computational overhead and a performance degradation. Additionally, this result demonstrates the improvement in the decision-making capability of each exit branch as the confidence threshold is appropriately adjusted. The same behavior occurs when evaluating the average precision and recall of all classes, as shown in Figure 3 and Figure 4. To analyze in more detail EE-DNN's behavior, we separately assess the performance of each exit branch in the next subsection.

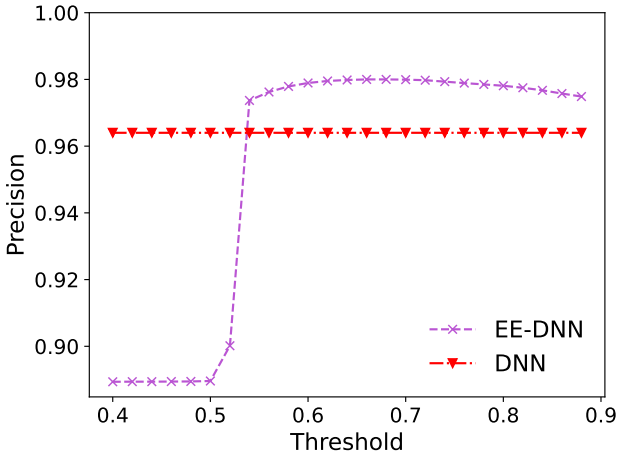

Fig. 3. Precision on EE-DNN and conventional DNN.

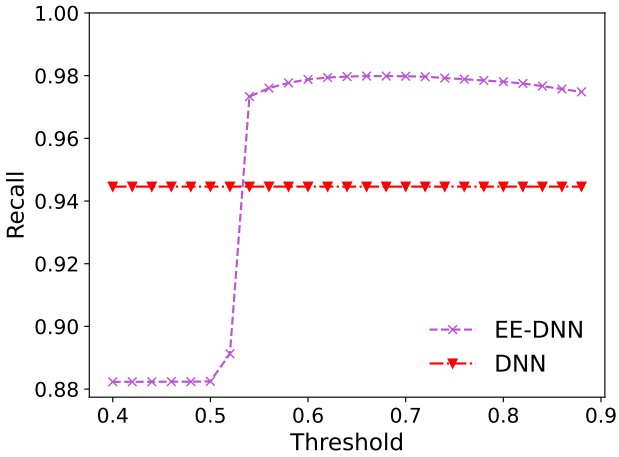

Fig. 4. Recall on EE-DNN and conventional DNN.

B. Per-branch Evaluation

This subsection conducts an analysis of each exit branch in the EE-DNN to elucidate their individual contributions to the

overall EE-DNN's performance. Therefore, we evaluate the accuracy and early-exit probability for each exit branch. The accuracy on each exit branch is computed as the ratio of correctly classified inputs to the total number of inputs processed by that exit branch. Meanwhile, the early-exit probability for the i-th exit branch is computed as the fraction of inputs that reach this specific exit branch and can be classified by it. For instance, the early-exit probability for the first exit branch is calculated as the number of test set inputs that achieve classification confidence greater than the threshold, divided by the total number of test set inputs. Similarly, this probability for the second exit branch is the fraction of inputs classified by this branch that were not classified by the first exit branch.

Figure 5 shows the early-exit probability, denoted as P[Inference] in y-axis, as a function of the thresholds. At a first glance, this figure shows that as we increase the threshold, it becomes more stringent for early classification, reducing the percentage of inputs classified earlier. Additionally, this figure demonstrates that the first exit branch is capable of classifying a high percentage of inputs, thereby reducing the computation required for classification. For instance, with a threshold of 0.54, approximately 80% of the inputs are classified by the first exit branch, thus avoiding processing by the second and third exit branches. In Figure 5, we observe that Branch 1 shows a decreasing trend, while Branches 2 and 3 are increasing as the threshold varies. This occurs because a higher threshold imposes stricter conditions, leading to a decrease in the proportion of inputs meeting the confidence criterion. Consequently, more inputs are left to be processed and classified by Branches 2 and 3.

Figure 6 highlights the benefits of an EE-DNN, showing the accuracy of each exit branch as a function of the thresholds. Note that, for a threshold of 0.54, EE-DNN classifies approximately 80% of the inputs (see Figure 5) in Branch 1 with an accuracy of 98%. This means that we can speed up the inference process by maintaining high accuracy. Also, Figure 5 shows that, for all threshold values considered, a very low percentage of inputs needs to be classified on the main DNN backbone (i.e., Branch 3). This means that a low-complexity DNN can be employed in most cases.

The results presented before show that the majority of inputs are classified on the first exit branch which can reduce the processing power required to perform an inference. To quantify this reduction, we measure the average number of Floating-point Operations (FLOPs) for the original DNN and the EE-DNN. The idea of using FLOPs is to provide a metric agnostic to the hardware architecture that runs the classification. This metric is obtained using the `pthflops`[3] tool. Figure 7 shows the average number of million FLOPs needed to process the images of the test set, comparing the original DNN and the EE-DNN. The results show that EE-DNN reduces the processing cost in 10% or more, showing its ability to accelerate spectrum classification. In addition, the reduction in FLOPs can improve the energy efficiency of the classification task.

[3]https://pypi.org/project/pthflops/

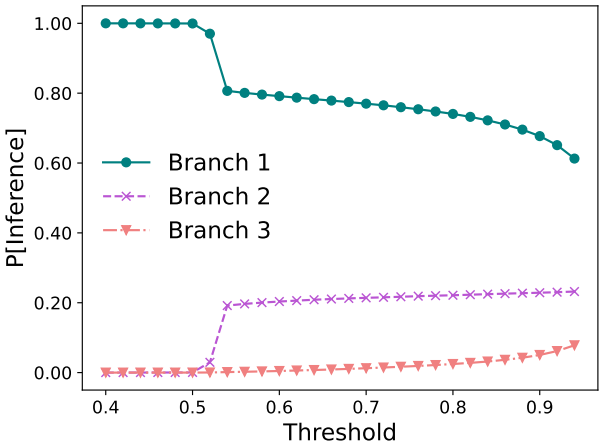

Fig. 5. Early-exit probability on each EE-DNN's branch.

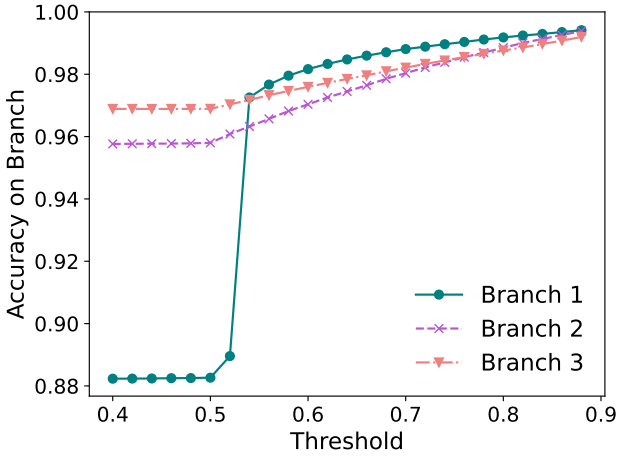

Fig. 6. Accuracy on each EE-DNN's branch.

VI. Conclusions and Future Work

Spectrum classification is a fundamental task for spectrum sharing xApps, being used by the ChARM (Channel-Aware Reacting Mechanism) framework. The ChARM xApp uses a DNN to classify I/Q samples into LTE, Wi-Fi, and Clear categories, supporting the implementation of spectrum sharing policies. Given that an xApp operates within a Near-RT RIC (Near-Real-Time RAN Intelligent Controller), it is crucial that the DNN inference time remains low. Hence, in this work we have investigated the use of Earl-Exit DNNs (EE-DNNs) to reduce the amount of computation of a DNN employed by the ChARM framework. Our results show that EE-DNNs can reduce the number of floating-point operations by 10%, based on the test set, while also improving the accuracy of the original DNN approximately to 2%. These findings underscore the practical benefits of employing EE-DNNs

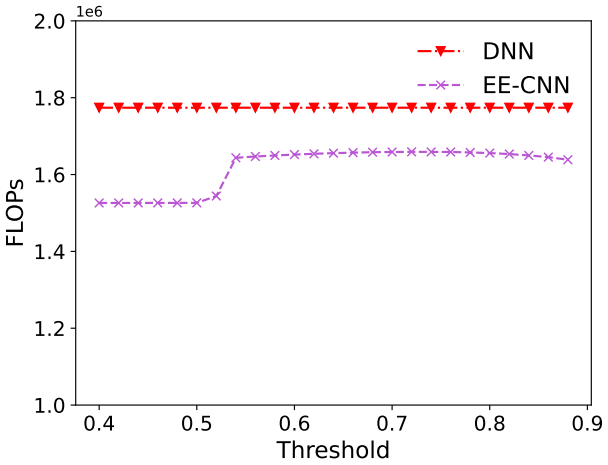

Fig. 7. FLOPs on EE-DNN vs DNN.

for spectrum classification, providing enhanced performance without additional computational cost.

As a future work, we plan to complement the results of floating-point operations by measuring the inference time on typical hardware used by Near-RT RICs to ensure it is compliant with the time scale of the Near-RT RICs between 10 ms and 1 s. Another direction for future research is to analyze the influence of the signal-to-noise ratio (SNR) of I/Q samples on the effectiveness of EE-DNNs. For instance, inputs with low SNR may require passing through all DNN layers to complete the inference, whereas inputs with high SNR could be classified at an earlier stage.

Acknowledgement

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, FAPERJ grant SEI-260003/004771/2021, FAPESP grants 23/00673-7 and 23/00811-0, and CNPq grant 408255/2023-4. This work was partially funded by the MERR 2024 program of Paris-Saclay University, it was also supported by the ANR HEIDIS (https://heidis.roc.cnam.fr; ANR-21-CE25-0019) project.

References

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[2] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[3] L. Baldesi, F. Restuccia, and T. Melodia, "ChARM: NextG Spectrum Sharing through Data-Driven Real-Time O-RAN Dynamic Control," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022, pp. 240–249.

[4] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2020.

[5] R. G. Pacheco, R. S. Couto, and O. Simeone, "On the impact of deep neural network calibration on adaptive edge offloading for image classification," *Journal of Network and Computer Applications*, p. 103679, 2023.

[6] L. Baldesi, F. Restuccia, and T. Melodia, "ChARM (Channel-Aware Reactive Mechanism) dataset," Available: http://hdl.handle.net/2047/D20423481, 2021.

[7] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3301–3310.

[8] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Pérez, and C. Verikoukis, "On the Specialization of FDRL Agents for Scalable and Distributed 6G RAN Slicing Orchestration," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3473–3487, 2023.

[9] O. Orhan, V. N. Swamy, T. Tetzlaff, M. Nassar, H. Nikopour, and S. Talwar, "Connection Management xAPP for O-RAN RIC: A Graph Neural Network and Reinforcement Learning Approach," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 936–941.

[10] H. Hojeij, M. Sharara, S. Hoteit, and V. Vèque, "Dynamic placement of O-CU and O-DU functionalities in open-ran architecture," in *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2023.

[11] E. Municio, G. Garcia-Aviles, A. Garcia-Saavedra, and X. Costa-Pérez, "O-ran: Analysis of latency-critical interfaces and overview of time sensitive networking solutions," *IEEE Communications Standards Magazine*, vol. 7, no. 3, pp. 82–89, 2023.

[12] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *IEEE international conference on pattern recognition (ICPR)*, 2016, pp. 2464–2469.

[13] R. G. Pacheco and R. S. Couto, "Inference time optimization using branchynet partitioning," in *IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6.

[14] R. G. Pacheco, R. S. Couto, and O. Simeone, "Calibration-aided edge inference offloading via adaptive model partitioning of deep neural networks," in *IEEE International Conference on Communications (ICC)*, 2021, pp. 1–6.

[15] R. G. Pacheco, F. D. Oliveira, and R. S. Couto, "Early-exit deep neural networks for distorted images: Providing an efficient edge offloading," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[16] R. G. Pacheco, K. Bochie, M. S. Gilbert, R. S. Couto, and M. E. M. Campista, "Towards edge computing using early-exit convolutional neural networks," *Information*, vol. 12, no. 10, p. 431, 2021.

[17] N. E. West and T. O'shea, "Deep architectures for modulation recognition," in *IEEE International Symposium on Dynamic Spectrum Access networks (DySPAN)*, 2017, pp. 1–6.

[18] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[19] S. Peng, S. Sun, and Y.-D. Yao, "A survey of modulation classification using deep learning: Signal representation and data preprocessing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7020–7038, 2021.

[20] E. Mohammed, O. Mashaal, and H. Abou-Zeid, "Using early exits for fast inference in automatic modulation classification," in *IEEE Global Communications Conference (GLOBECOM)*, 2023, pp. 291–296.

[21] D. Verbruggen, S. Pollin, and H. Sallouha, "Computational efficient width-wise early exits in modulation classification," *arXiv preprint arXiv:2405.03222*, 2024.

[22] F. Schaich and T. Wild, "Waveform contenders for 5g—ofdm vs. fbmc vs. ufmc," in *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, 2014, pp. 457–460.

[23] A. Elnashar and M. A. El-Saidny, "Looking at lte in practice: A performance analysis of the lte system based on field test results," *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 81–92, 2013.

[24] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive inference through early-exit networks: Design, challenges and directions," in *International Workshop on Embedded and Mobile Deep Learning (EMDL)*, 2021, pp. 1–6.