Interference-Aware Optimization of Energy-Efficient Relay-Assisted Collaborative CNN Execution

Emre Kilcioglu, Ivan Stupia, and Luc Vandendorpe
ICTEAM/ELEN, Université Catholique de Louvain, Louvain-la-Neuve, Belgium
{name.surname}@uclouvain.be

***Abstract*—Collaborative execution of convolutional neural networks (CNNs) holds significant promise for enhancing the performance of latency-critical artificial intelligence applications on resource-constrained devices (RCDs). Edge intelligence advocates for shifting computation-intensive tasks to the edge network to improve latency and system reliability. This paper presents an interference-aware energy-efficient architecture for relay-assisted collaborative CNN execution, integrating data and model parallelism, and addressing interference impact on communication and computation energy consumption. With the predefined signal-to-interference-noise ratio (SINR) target, the study investigates the trade-off between communication and computation and formulates a convex optimization problem for energy optimization. The proposed architecture is evaluated through simulations, demonstrating its superiority compared to several benchmark scenarios and shedding light on the role of the SINR target in shaping the trade-off between communication and computation.**

***Index Terms*—Collaborative computing, convex optimization, CNNs, data parallelism, edge intelligence, energy efficiency, interference-aware optimization, model parallelism, relay-assisted communication.**

I. Introduction

Deep learning (DL) models have found wide applications in artificial intelligence (AI) applications such as computer vision, natural language processing, and autonomous driving. These applications often demand high-accuracy inference, preventing the execution of DL models on individual resource-constrained devices (RCDs). A suitable approach is to leverage edge intelligence (EI) to deploy DL-driven AI applications at the edge [1], [2]. EI aims to shift computation-intensive DL execution to the edge networks, offering benefits like lower latency, reduced network congestion, decentralized structures, and enhanced system reliability compared to the conventional cloud-centric approach [2]. Among several approaches in EI concept, collaborative DL execution with multiple RCDs is a viable option when deploying powerful servers at the edge is challenging [3]. The effective workload distribution across multiple RCDs presents a critical challenge in collaborative DL execution scenarios. Deep neural network (DNN) partitioning emerges as a promising solution, promoting parallel collaborative computation among multiple RCDs [4]–[10]. Two primary paradigms for DNN partitioning stand out: data parallelism and model parallelism. Data parallelism involves splitting input data among RCDs, each processing the entire model computation with its partial data while model parallelism partitions the model among RCDs, with each processing complete data using its partial model.

As a pioneering instance of collaborative computing, Mao *et al.* propose MoDNN, a locally distributed mobile device computing system aimed at minimizing latency [6]. However, this approach optimizes workload distribution among devices by considering constant communication and computation parameters. DeepThings [7] employs Fused Tile Partitioning (FTP) to minimize memory usage by fusing and vertically partitioning layers in a grid pattern. However, it does not account for the heterogeneous computing capabilities of devices or varying network conditions. Our previous study [8] proposes an energy-efficient DNN partitioning for wireless collaborative fog computing systems, focusing on data parallelism. The approach assumes that each convolutional layer in the model consists of one filter. While it serves as a robust benchmark for data parallelism, a model parallelism approach becomes imperative for the CNN models with multiple filters in each convolutional layer. For this purpose, our previous study presents a distributed model parallelism scenario for on-device collaborative CNN execution, aiming to jointly optimize the number of filter assignments in each convolutional layer of the CNN model and communication-computation parameters for overall energy consumption minimization [9]. However, this model parallelism scheme suffers from the scenarios where large input data sizes exist. To mitigate the negative effects of both data and model parallelism, our recent study [10] introduces a relay-assisted, distributed, and on-device collaborative CNN execution scheme for latency-critical DL-driven applications, combining both data and model parallelism. However, the interference between relays is not considered in [10] due to the assumption of large distance between relays. Our aim in this paper is to extend the system architecture of this study by incorporating the impact of interference among relays and collaborator RCDs (C-RCDs) on the energy consumption of communication and computation.

This paper proposes an interference-aware energy efficient relay-assisted collaborative CNN execution architecture based on a predefined signal-to-interference-noise ratio (SINR) target $\mathrm{SINR_T}$ by employing hybrid parallelism, integrating data and model parallelism. The system is composed of a data owner RCD that owns the input data to be processed in the CNN model. Additionally, the system comprises multiple computing segments, each featuring a relay and multiple C-

This work was supported by F.R.S.-FNRS under the EOS research project MUSEWINET (EOS project 30452698).

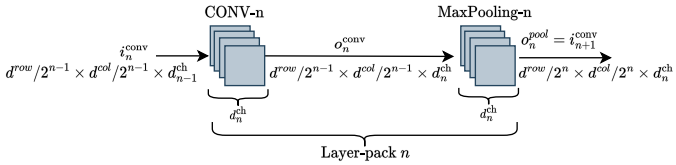

Fig. 1. The layer-pack n operation of the CNN model

RCDs. Our primary objective is to explore how interference between relays and C-RCDs across different computing segments affects overall energy consumption during collaborative CNN execution. We also demonstrate that the predefined SINR_T value significantly impacts the trade-off point between communication and computation. At the end, we formulate a convex optimization problem which can be solved using classical convex optimization techniques and Lagrange duality theory. In our simulations, we conduct a comprehensive performance analysis, comparing our approach against several benchmark scenarios with the assumptions of no interference. Furthermore, we showcase the role of the SINR_T value in shaping the trade-off between communication and computation in terms of energy consumption. The contributions of this paper include a comprehensive system architecture for collaborative CNN execution, encompassing an interference-aware relay-assisted paradigm, and embracing data and model parallelism. Moreover, we investigate how interference among different devices influences the energy consumption trade-off between communication and computation.

Section II provides a CNN model overview, while section III introduces collaborative CNN execution scenarios. Section IV defines the relay-assisted collaborative CNN execution scheme. Energy consumption modeling for each stage is detailed in section V. Sections VI and VII describe additional constraints and the problem formulation, respectively. Simulation results are presented and discussed in section VIII, followed by the conclusion in section IX.

II. Overview of the CNN Architecture

A typical CNN architecture consists of multiple pairs of convolutional and pooling layers, and a fully connected layer at the end. Fully connected layers connect all neurons within a layer to the next, while convolutional layers employ 2D filters to generate outputs from input data [5]. These outputs, generated by each filter, constitute 2D data, resulting in 3D data when multiple filters are employed. Meanwhile, maximum pooling layers perform downsampling by selecting the maximum value in the patch of the input data. We name each paired convolutional and pooling layer as a 'layer-pack' in this paper. Fig. 1 illustrates the operation of the n-th layer-pack, where d_n^{ch} represents the number of filters in convolutional or pooling layer n, i_n^{conv} and o_n^{conv} are input and output data for convolutional layer n, and o_n^{pool} represents the output data of pooling layer n. The initial 2D input data is denoted as i_1^{conv}, with dimensions $d^{\text{row}} \times d^{\text{col}}$ bits. Derived from the fundamental properties of convolution, the relation between the input and output of the n-th convolutional layer is

$$\begin{aligned} o_n^{\text{conv}}(:,:,\theta) &= f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,1) + f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,2) \\ &\quad + ... + f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,d_{n-1}^{\text{ch}}) \\ &= f_{\theta_n}^{\text{conv}} * \left(\sum_{\phi=1}^{d_{n-1}^{\text{ch}}} i_n^{\text{conv}}(:,:,\phi) \right) \end{aligned} \tag{1}$$

Here, $n \in [N^{\text{LAY}}]$ denotes the index of the layer-pack, N^{LAY} represents the number of layer-packs in the CNN model, $\theta \in [d_n^{\text{ch}}]$ is the third-dimensional index (or page) of the 3D output data o_n^{conv}, and $f_{\theta_n}^{\text{conv}}$ signifies the θ^{th} filter of convolutional layer n. To maintain consistency, we adopt a filter size of 3×3 for each filter $f_{\theta_n}^{\text{conv}}$, a widely employed choice in CNN implementations. In the case of $n = 1$, $d_0^{\text{ch}} = 1$ is assigned to account for the initial two-dimensional input data i_1^{conv}. The technique of zero-padding is employed to ensure consistent data sizes between input and output data of a convolutional layer by adding zeros to the external regions of the 2D input data. Subsequently, the output of each convolutional layer undergoes maximum-pooling, denoted as $o_n^{\text{pool}}(:,:,\theta) = MP\left(o_n^{\text{conv}}(:,:,\theta)\right)$, with $MP()$ signifying the maximum-pooling operator. This paper utilizes a filter size of 2×2 for maximum-pooling filters, resulting in a halving of data size after each pooling layer.

III. Collaborative CNN Execution Scenarios

To execute the CNN models collaboratively, there are several concepts, mentioned in the following subsections.

A. Model Parallelism

In the model parallelism concept, the model is partitioned among multiple devices. The partial models are executed collaboratively by processing the entire input data. The input data is broadcasted from the data owner device to the other devices and the output in each participating device after each layer-pack operation is sent to the data owner to produce the new input data for the next layer-pack operations. The adoption of a pure model parallelism approach for collaborative CNN execution presents several challenges, including:

- A single device may struggle to complete multiple filter computations with large input data sizes under a low latency constraint.
- Communication between the data owner and other participating devices may become impractical, especially if they are geographically distant.
- Broadcasting complete input data can result in communication inefficiency, which is primarily determined by the device with the weakest channel condition.

As a consequence, it becomes essential to explore alternative parallelism techniques for collaborative CNN execution.

B. Data Parallelism

In the data parallelism concept, the input data is partitioned and allocated to multiple devices which then conduct full model computations using partial input data. The partial input

data is dispatched to each device, along with additional redundant rows that accommodate the convolution operation's inherent characteristics. Following each layer-pack operation, only boundary rows are exchanged between neighboring devices on the input data map to perform next layer-pack operations. Such a scenario was already proposed in our previous study [8]. Data parallelism proves effective in managing large input data sizes, but it has several critical issues such as

- It is applicable only when each convolutional layer contains a single filter. With multiple filters in each layer-pack, aggregation at the data owner after each layer-pack operation is needed which contradicts the idea of exchange communication between neighboring devices. Also, achieving the desired accuracy from a CNN model necessitates multiple filters in each convolutional layer, rendering data parallelism impractical.
- Transmitting partial input data point-to-point to each device results in system inefficiency, adversely impacting both latency and energy consumption.

C. Hybrid Parallelism

The integration of data and model parallelism emerges as a promising approach for collaborative CNN execution. It addresses challenges posed by large input data sizes through data parallelism, while accommodating computations involving multiple filters through model parallelism. This combined approach necessitates efficient point-to-point communication between the data owner and other participating devices. To tackle this challenge, our previous work [10] introduces relay devices within each computing segment, comprising multiple C-RCDs and a single relay. The relay coordinates communication, model parallelism, and data aggregation, managing the exchange between the data owner and C-RCDs. In this paper, we extend this relay-assisted approach to an interference-aware scenario, integrating SINR targets into the communication model. The subsequent section explains the detailed structure of this extended architecture.

IV. Relay-assisted Collaborative CNN Execution Considering Hybrid Parallelism

The system model for relay-assisted collaborative CNN execution, depicted in Fig. 2, includes a data owner RCD (DO-RCD) with input data, N^{rel} number of computing segments, each of which contains a relay and K_m C-RCDs assigned to computing segment m. The rectangle shape in Fig. 2 represents the 2D input data in the DO-RCD.

For seamless operation, aggregated output data from each relay must return to the DO-RCD after every layer-pack operation to form new input data, which introduces communication overhead to the system in the scenarios where the DO-RCD and relays are distant. To alleviate this, a potential solution involves the creation of layer-pack blocks, each being composed of multiple layer-packs [10]. These layer-packs within a block can be executed inside each computing segment without DO-RCD-relay communication. At the beginning of each block, additional redundant rows are dispatched to each

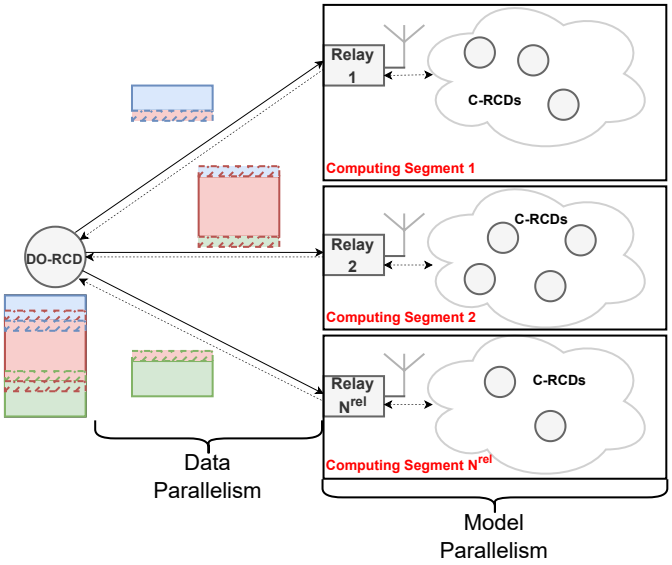

Fig. 2. The proposed relay-assisted collaborative CNN execution architecture considering hybrid parallelism

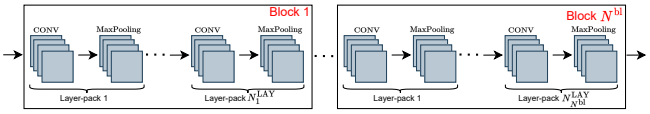

Fig. 3. Layer-pack block formation

relay to compute its boundary rows, which is shown as shaded regions in the 2D input data in Fig. 2. This approach mitigates the communication while introducing additional computation due to the calculation of the same redundant rows in multiple computing segments. The layer-pack block formation is depicted in Fig. 3 where N^{bl} represents the number of blocks and N_b^{LAY} denotes the number of layer-packs within block b.

The system workflow for each block execution entails:

- Pre-training the CNN model on powerful cloud servers using offline datasets to avoid latency constraints during training.
- Dividing input into N^{rel} pieces by the DO-RCD before the first block, sent to relays along with d_1^R redundant rows where d_b^R is the number of redundant rows that must be additionally sent to each relay for block b computation. Due to the properties of the convolution operation, d_b^R for block b computation is expressed as $d_b^R = \sum_{n=1}^{N_b^{\text{LAY}}} 2^{n-1}$. For subsequent blocks, only d_b^R redundant rows are sent, as relays already possess partial input data. This step is referred to as the input distribution (ID).
- Executing model parallelism within each computing segment.
- At the end of all layer-pack operations within the block, relays send boundary rows to the DO-RCD for the next layer-pack operation. In the final block, entire output data is sent to complete collaborative CNN execution.
- DO-RCD collects boundary rows, and the process continues for the next block.

Model parallelism calculation inside each computing seg-

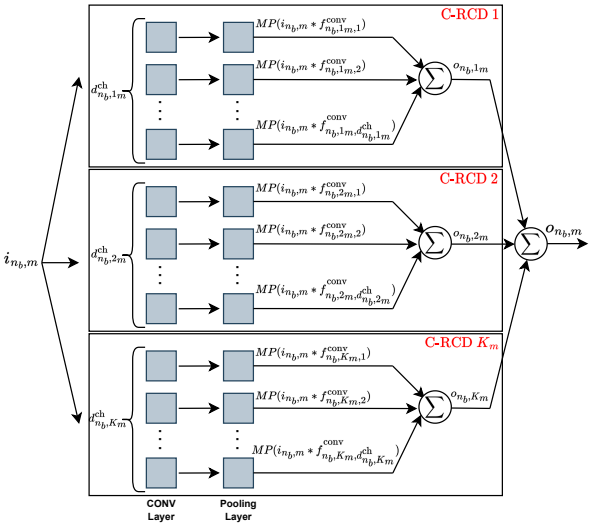

Fig. 4. Layer-pack n execution of collaborative CNN model parallelism

ment is depicted in Fig. 4, where filters for the n-th layer-pack are distributed among K_m C-RCDs. Each blue square corresponds to a filter, $f^{\text{conv}}_{n_b,k_m,j}$ represents the j^{th} convolutional filter and $d^{\text{ch}}_{n_b,k_m}$ indicates the number of filters allocated to C-RCD k of computing segment m for layer-pack n operation of block b. Model parallelism comprises three stages for each layer-pack operation in a block:

- Relay broadcasts input data with redundant rows to assigned C-RCDs.
- C-RCDs execute local computation, and the max-pooling outputs of these filters are summed within each C-RCD, yielding the output data o_{n_b,k_m} for layer-pack n operation $o_{n_b,k_m} = \sum_{j=1}^{d^{\text{ch}}_{n_b,k_m}} MP(i_{n_b,m} * f^{\text{conv}}_{n_b,k_m,j})$.
- After each layer-pack operation, each C-RCD returns the summation of its assigned filter's output data to its relay, which aggregates the output data o_{n_b,k_m} from all C-RCDs to derive the final output data $o_{n_b,m} = \sum_{k=1}^{K_m} o_{n_b,k_m}$, used as input for next layer-pack operation.

In the next section, the energy consumption of each stage is modeled.

V. Energy Consumption Modeling

In this section, the energy consumption stages are categorized into three parts for each block, which are defined in the following subsections.

A. Input Distribution (ID)

The DO-RCD communicates with each relay m to send the input data for block b computation with the achievable rate

$$r^{\text{ID}}_{b,m} = B\ \ln(1 + \frac{p^{\text{ID}}_{b,m}\frac{h^{\text{ID}}_{b,m}}{l(R^{\text{ID}}_{b,m})}}{BN_0}) \tag{2}$$

Here, $p^{\text{ID}}_{b,m}$ represents the RF transmit power of the DO-RCD (denoted as $\widetilde{k}$) during ID, B is the communication bandwidth, N_0 is the noise power density, $h^{\text{ID}}_{b,m}$ and $R^{\text{ID}}_{b,m}$ are the channel gain and distance between DO-RCD and relay m for block b computation during ID, respectively and $l(R) = c_{\text{PL}}R^{\alpha}$ is the path-loss model where α is the path-loss exponent, $c_{\text{PL}} = (4\pi f_c/c_0)^2$ is the path-loss constant, f_c is the carrier frequency and c_0 is the speed of light.

The energy consumption of ID to relay m for block b computation is represented as

$$E^{\text{ID}}_{b,m} = t^{\text{ID}}_{b,m}(p^{\text{ID}}_{b,m} + P^{c}_{\widetilde{k}}) \tag{3}$$

where $P^{c}_{\widetilde{k}}$ denotes the constant power consumption of the communication circuitry in the DO-RCD and $t^{\text{ID}}_{b,m}$ is the time to send the input data to relay m for block b computation. The number of bits $d^{\text{ID}}_{b,m}$ required to be sent during ID is constrained by

$$d^{\text{ID}}_{b,m} \leq t^{\text{ID}}_{b,m} r^{\text{ID}}_{b,m} \tag{4}$$

$$d^{\text{ID}}_{b,m} = \begin{cases} (d_m + I_m d^R_1)d^{\text{col}} & \text{if } b = 1 \\ I_m d^R_b d^{\text{col}}_b & \text{otherwise} \end{cases} \tag{5}$$

where d_m is the number of rows assigned to computing segment m, $d^{\text{col}}_b = \frac{d^{\text{col}}}{2^{\sum_{b'=1}^{b-1} N^{LAY}_{b'}}}$ and I_m is the indicator function; it is 1 for top and bottom relays in the input data map (for $m = 1$ and $m = N^{\text{rel}}$) and 2 otherwise.

B. Model Parallelism Stages

After receiving input data from the DO-RCD before each block computation, the relay devices take charge of orchestrating the model parallelism aspect of the system. In this phase, the relay devices optimally distribute convolutional layer filters among the K_m participating C-RCDs associated with each relay m, considering their heterogeneous computing capabilities and different network conditions. Within the same block, the process consists of three distinct stages:
(1) Broadcasting (BC) the partial input data by each relay to assigned C-RCDs,
(2) Local computation in each C-RCD,
(3) Aggregation of the output data of each C-RCD at the assigned relay.
These stages are repeated for each layer-pack operation until the end of the block.

1) Broadcasting (BC): Relay m broadcasts its partial input data for layer-pack n computations of block b to the participating C-RCDs with the following SINR:

$$\text{SINR}^{\text{BC}}_{n_b,m} = \frac{p^{\text{BC}}_{n_b,m}\min_{k_m}\left(\frac{h^{\text{BC}}_{n_b,m,k_m}}{l(R^{\text{BC}}_{n_b,m,k_m})}\right)}{BN_0 + \sum_{\substack{m'=1 \\ m'\neq m}}^{N^{\text{rel}}} p^{\text{BC}}_{n_b,m'}\frac{h^{\text{BC}}_{n_b,m,m'}}{l(R^{\text{BC}}_{n_b,m,m'})}} \geq \text{SINR}_{\text{T}} \tag{6}$$

where an SINR target SINR_{T} is defined. Here, $p^{\text{BC}}_{n_b,m}$ is the RF transmit power of broadcasting in relay m, $h^{\text{BC}}_{n_b,k_m}$ and $R^{\text{BC}}_{n_b,k_m}$ are the channel gain and distance between relay m and C-RCD k during layer-pack n computation of block b,

respectively. The energy consumption of broadcasting by relay m for layer-pack n computations of block b is

$$E^{\text{BC}}_{n_b,m} = t^{\text{BC}}_{n_b,m}(p^{\text{BC}}_{n_b,m} + P^c_m) \tag{7}$$

where P^c_m and $t^{\text{BC}}_{n_b,m}$ are the constant power consumption of the communication circuitry and the time taken for broadcasting by relay m during the computation of layer-pack n in block b, respectively. The broadcasting time $t^{\text{BC}}_{n_b,m}$ is subject to the following lower-bound constraint

$$\frac{d^{\text{BC}}_{n_b,m}}{B\ln(1+\text{SINR}_\text{T})} \leq t^{\text{BC}}_{n_b,m} \tag{8}$$

and

$$d^{\text{BC}}_{n_b,m} = \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^{n-1})}{2^{n-1}}\right)\frac{d^{\text{col}}_b}{2^{n-1}} \tag{9}$$

where $d^{\text{BC}}_{n_b,m}$ is the broadcasted number of bits, $d^{\text{col}}_b = \frac{d^{\text{col}}}{2^{\sum_{b'=1}^{b-1} N^{LAY}_{b'}}}$ and $d_{b,m} = \frac{d_m}{2^{\sum_{b'=1}^{b-1} N^{LAY}_{b'}}}$ is the number of rows associated with relay m except for the redundant rows.

2) Local Computation: Following the reception of input data, each C-RCD executes its local computation. This paper emphasizes the energy consumption of convolutional layers, as they predominantly contribute to energy usage over other layers [6], [11].

A convolution operation encompasses multiple multiply-accumulate (MAC) operations [11]. Each MAC operation involves the multiplication of an element from the filter with a corresponding element from a patch of input data, followed by the accumulation of the result to the existing sum. The number of MAC operations $N^{\text{MAC}}_{n_b,k_m}$ carried out by C-RCD k of relay m for layer-pack n operation of block b is given by

$$\begin{aligned} N^{\text{MAC}}_{n_b,k_m} &= (d^{\text{filter}})^2\left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^n)}{2^{n-1}}\right)\frac{d^{\text{col}}_b}{2^{n-1}}d^{\text{ch}}_{n_b,k_m} \\ &= \alpha_{n_b,m}d^{\text{ch}}_{n_b,k_m} \end{aligned} \tag{10}$$

where $\alpha_{n_b,m}$ is introduced for clarity in notation and $d^{\text{filter}} = 3$ represents the preferred dimension of convolutional filters. The energy consumption associated with local computation in C-RCD k of relay m for layer-pack n operation in block b under low CPU voltage assumption [12] is

$$E^{\text{LOC}}_{n_b,k_m} = \frac{\kappa_{k_m}c^3_{k_m}(N^{\text{MAC}}_{n_b,k_m})^3}{(t^{\text{LOC}}_{n_b,k_m})^2} = \kappa_{k_m}c^3_{k_m}(\alpha_{n_b,m})^3\frac{(d^{\text{ch}}_{n_b,k_m})^3}{(t^{\text{LOC}}_{n_b,k_m})^2} \tag{11}$$

where κ_{k_m} is the effective hardware capacitance coefficient, c_{k_m} is the number of CPU cycles per MAC operation, and $t^{\text{LOC}}_{n_b,k_m}$ is the time needed to complete the local computation for layer-pack n computations in block b for C-RCD k of relay m. The following constraint holds

$$c_{k_m}N^{\text{MAC}}_{n_b,k_m} = c_k\alpha_{n_b,m}d^{\text{ch}}_{n_b,k_m} \leq \nu^{\max}_{k_m}t^{\text{LOC}}_{n_b,k_m} \tag{12}$$

This constraint places an upper limit on the frequency of each CPU in the C-RCDs, where $\nu^{\max}_{k_m}$ is the maximum allowed CPU frequency for C-RCD k of relay m.

3) Aggregation: In this phase, each C-RCD k transmits the cumulative output data of its assigned filters for layer-pack n to its corresponding relay m, adhering to the following criterion

$$\text{SINR}^{\text{Agg}}_{n_b,k_m} = \frac{p^{\text{Agg}}_{n_b,k_m}\left(\frac{h^{\text{Agg}}_{n_b,m,k_m}}{l(R^{\text{Agg}}_{n_b,m,k_m})}\right)}{\frac{B}{K_m}N_0 + \sum\limits_{\substack{m'=1\\m'\neq m}}^{N^{\text{rel}}} p^{\text{Agg}}_{n_b,k_{m'}}\frac{h^{\text{Agg}}_{n_b,k_m,k_{m'}}}{l(R^{\text{Agg}}_{n_b,k_m,k_{m'}})}} \geq \text{SINR}_\text{T} \tag{13}$$

where $p^{\text{Agg}}_{n_b,k_m}$ is the RF transmit power of aggregation in C-RCD k of relay m, $h^{\text{Agg}}_{n_b,k_m}$ and $R^{\text{Agg}}_{n_b,k_m}$ are the channel gain and distance between relay m and C-RCD k, respectively. The energy consumption related to aggregation by C-RCD k of relay m for layer-pack n computations of block b is

$$E^{\text{Agg}}_{n_b,k_m} = t^{\text{Agg}}_{n_b,k_m}(p^{\text{Agg}}_{n_b,k_m} + P^c_{k_m}) \tag{14}$$

where $P^c_{k_m}$ is the constant power consumption of the communication circuitry and $t^{\text{Agg}}_{n_b,k_m}$ denotes the time required for aggregation by C-RCD k of relay m. The following constraint applies to the aggregation time $t^{\text{Agg}}_{n_b,k_m}$

$$\frac{d^{\text{Agg}}_{n_b,k_m}}{\frac{B}{K_m}\ln(1+\text{SINR}_\text{T})} \leq t^{\text{Agg}}_{n_b,k_m} \tag{15}$$

$$d^{\text{Agg}}_{n_b,k_m} = \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^n)}{2^n}\right)\frac{d^{\text{col}}_b}{2^n} \tag{16}$$

where $d^{\text{Agg}}_{n_b,k_m}$ is the number of bits sent by C-RCD k of relay m for layer-pack n operation of block b.

C. Output Transmission (OT)

Following the model parallelism phase in each block, each relay m transmits either only redundant boundary rows or the complete output data if the current block is the last block in the model. The lower-bound of $\text{SINR}^{\text{OT}}_{b,m}$ for the communication between relay m and DO-RCD throughout OT is

$$\text{SINR}^{\text{OT}}_{b,m} = \frac{p^{\text{OT}}_{b,m}\left(\frac{h^{\text{OT}}_{b,m}}{l(R^{\text{OT}}_{b,m})}\right)}{BN_0 + \sum\limits_{\substack{m'=1\\m'\neq m}}^{N^{\text{rel}}} p^{\text{OT}}_{b,m'}\frac{h^{\text{OT}}_{b,m,m'}}{l(R^{\text{OT}}_{b,m,m'})}} \geq \text{SINR}_\text{T} \tag{17}$$

where $p^{\text{OT}}_{b,m}$ is the RF transmit power of relay m, $h^{\text{OT}}_{b,m}$ and $R^{\text{OT}}_{b,m}$ are the channel gain and distance between relay m and the DO-RCD for block b computation during OT, respectively. Energy consumed for OT by relay m during block b computation is

$$E^{\text{OT}}_{b,m} = t^{\text{OT}}_{b,m}(p^{\text{OT}}_{b,m} + P^c_m) \tag{18}$$

where $t^{\text{OT}}_{b,m}$ is the transmission time for relay m and block b computation. The constraint of this stage for the transmission time $t^{\text{OT}}_{b,m}$ is

$$\frac{d^{\text{OT}}_{b,m}}{B\ln(1+\text{SINR}_\text{T})} \leq t^{\text{OT}}_{b,m} \tag{19}$$

and

$$d_{b,m}^{\mathrm{OT}} = \begin{cases} \frac{d_{b,m}}{2^{N_b^{\mathrm{LAY}}}} \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{if } b = N^{\mathrm{bl}} \\ I_m d_{b+1}^R \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{otherwise} \end{cases} \tag{20}$$

where $d_{b,m}^{\mathrm{OT}}$ is the number of transmitted bits from relay m to the DO-RCD after block b computation.

VI. Additional Constraints

The system imposes several crucial additional constraints:

$$\sum_{k=1}^{K_m} d_{n_b,k_m}^{\mathrm{ch}} = d_{n_b}^{\mathrm{ch}} \tag{21}$$

$$t_{n_b,m}^{\mathrm{BC}} + t_{n_b,k_m}^{\mathrm{LOC}} + t_{n_b,k_m}^{\mathrm{Agg}} \leq t_{n_b,m} \tag{22}$$

$$t_{b,m}^{\mathrm{ID}} + \sum_{n=1}^{N_b^{\mathrm{LAY}}} t_{n_b,m} + t_{b,m}^{\mathrm{OT}} \leq t_b \tag{23}$$

$$\sum_{b=1}^{N^{\mathrm{bl}}} t_b \leq \tau \tag{24}$$

where $t_{n_b,m}$, t_b and τ denote the allocated times for individual layer-pack operations, block operations, and the entire procedure, respectively. Constraint (21) ensures the proper distribution of filter channels, while (22) and (23) guarantee timing synchronization within layer-pack and block operations. Lastly, (24) enforces the overall system latency requirement.

VII. Problem Formulation

To transform the problem into a convex optimization one, inspired by our previous paper [8], new parameters are introduced: $E_{b,m}^{\mathrm{ID}_{\mathrm{RF}}} = p_{b,m}^{\mathrm{ID}} t_{b,m}^{\mathrm{ID}}$, $E_{n_b,m}^{\mathrm{BC}_{\mathrm{RF}}} = p_{n_b,m}^{\mathrm{BC}} t_{n_b,m}^{\mathrm{BC}}$, $E_{n_b,k_m}^{\mathrm{Agg}_{\mathrm{RF}}} = p_{n_b,k_m}^{\mathrm{Agg}} t_{n_b,k_m}^{\mathrm{Agg}}$ and $E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}} = p_{b,m}^{\mathrm{OT}} t_{b,m}^{\mathrm{OT}}$, representing RF energy consumption in various communication stages.

The final convex optimization problem is defined as follows

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \sum_{b=1}^{N^{\mathrm{bl}}} \sum_{m=1}^{N^{\mathrm{rel}}} \left(E_{b,m}^{\mathrm{ID}} + E_{b,m}^{\mathrm{OT}} + \sum_{n=1}^{N_b^{\mathrm{LAY}}} \left(E_{n_b,m}^{\mathrm{BC}} + \sum_{k=1}^{K_m} \left(E_{n_b,k_m}^{\mathrm{LOC}} + E_{n_b,k_m}^{\mathrm{Agg}} \right) \right) \right) \\
\text{s.t.} \quad & (4), (8), (12), (15), (19), (21), (22), (23), (24), \\
& E_{n_b,m}^{\mathrm{BC}_{\mathrm{RF}}} \min_{k_m} \left(\frac{h_{n_b,m,k_m}^{\mathrm{BC}}}{l(R_{n_b,m,k_m}^{\mathrm{BC}})} \right) \geq \mathrm{SINR_T} \left(B N_0 t_{n_b,m}^{\mathrm{BC}} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E_{n_b,m'}^{\mathrm{BC}_{\mathrm{RF}}} \frac{h_{n_b,m,m'}^{\mathrm{BC}}}{R_{n_b,m,m'}^{\mathrm{BC}}} \right), \\
& E_{n_b,k_m}^{\mathrm{Agg}_{\mathrm{RF}}} \frac{h_{n_b,m,k_m}^{\mathrm{Agg}}}{l(R_{n_b,m,k_m}^{\mathrm{Agg}})} \geq \mathrm{SINR_T} \left(\frac{B}{K_m} N_0 t_{n_b,k_m}^{\mathrm{Agg}} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E_{n_b,k_{m'}}^{\mathrm{Agg}_{\mathrm{RF}}} \frac{h_{n_b,k_m,k_{m'}}^{\mathrm{Agg}}}{R_{n_b,k_m,k_{m'}}^{\mathrm{Agg}}} \right), \\
& E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}} \frac{h_{b,m}^{\mathrm{OT}}}{l(R_{b,m}^{\mathrm{OT}})} \geq \mathrm{SINR_T} \left(B N_0 t_{b,m}^{\mathrm{OT}} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E_{b,m'}^{\mathrm{OT}_{\mathrm{RF}}} \frac{h_{b,m,m'}^{\mathrm{OT}}}{R_{b,m,m'}^{\mathrm{OT}}} \right), \\
& 0 \leq t_{b,m}^{\mathrm{ID}}, t_{n_b,m}^{\mathrm{BC}}, t_{n_b,k_m}^{\mathrm{Agg}}, t_{b,m}^{\mathrm{OT}}, t_{n_b,k_m}^{\mathrm{LOC}}, d_{n_b,k_m}^{\mathrm{ch}}, E_{b,m}^{\mathrm{ID}_{\mathrm{RF}}}, E_{n_b,m}^{\mathrm{BC}_{\mathrm{RF}}}, E_{n_b,k_m}^{\mathrm{Agg}_{\mathrm{RF}}}, E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}}
\end{aligned} \tag{25}$$

where $\mathbf{x} = \{t_{b,m}^{\mathrm{ID}}, E_{b,m}^{\mathrm{ID}_{\mathrm{RF}}}, t_{n_b,m}^{\mathrm{BC}}, E_{n_b,m}^{\mathrm{BC}_{\mathrm{RF}}}, t_{n_b,k_m}^{\mathrm{Agg}}, E_{n_b,k_m}^{\mathrm{Agg}_{\mathrm{RF}}}, t_{b,m}^{\mathrm{OT}}, E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}}, t_{n_b,k_m}^{\mathrm{LOC}}, d_{n_b,k_m}^{\mathrm{ch}}, t_{n_b,m}, t_b\}$, $\forall n \in [N_b^{\mathrm{LAY}}]$, $\forall k \in [K_m]$, $\forall b \in [N^{\mathrm{bl}}]$ and $\forall m \in [N^{\mathrm{rel}}]$. The optimization problem can be solved using classical convex optimization methods and Lagrange duality theory. The problem seeks to minimize the overall energy consumption while satisfying communication quality and timing constraints.

TABLE I
Selected parameters

κ_{k_m}: Unif($[10^{-28}; 10^{-27}]$)	τ: 0.5 sec
$\nu_{k_m}^{\max}$: Unif($[1; 3]$) GHz	B: 2 MHz
$P_{\tilde{k}}^c$ & P_m^c & $P_{k_m}^c$: Unif($[10; 25]$) mW	α: 2.5
N_0: 0.5×10^{-11} W/MHz	f_c: 2.1 GHz
h(Each channel gain): $\mathcal{CN}(0, 10^{-3})$(Rayleigh)	d_n^{ch}: 32
$R_{n_b,k_m}^{\mathrm{BC}}$ & $R_{n_b,k_m}^{\mathrm{Agg}}$: Unif($[10; 20]$) m	N^{LAY}: 3
$R_{b,m}^{\mathrm{ID}}$ & $R_{b,m}^{\mathrm{OT}}$: Unif($[100; 150]$) m	N_1^{LAY}: 1
c_{k_m}: Unif($[150; 350]$) CPU cycles/MAC	N_2^{LAY}: 2

VIII. Simulation Results

This section presents simulation results evaluating the performance of the proposed approach, referred to as *DoubleRel*. The aim is to compare it against alternative scenarios and demonstrate its effectiveness in enhancing the achievable trade-offs between communication and computation.

The first benchmark scenario, denoted as *NoIntfc-SingleRel*, involves a single computing segment containing a relay and multiple C-RCDs. The second scenario, *NoIntfc-AllocBW*, includes multiple computing segments, each allocated a portion of the bandwidth to mitigate interference.

In our simulations, we utilized parameter settings outlined in Table I, inspired by our prior research [8]–[10].

A. Impact of the Number of C-RCDs per Relay K_m on Energy Consumption of Different Scenarios

In figures 5 and 6, we examine the energy consumption of different scenarios as the number of C-RCDs per relay (K_m) varies, while considering different $\mathrm{SINR_T}$ values. The objective is to analyze the trade-off between communication and computation and identify the optimal K_m value for each scenario.

Fig. 5 illustrates a configuration with input data sizes of $d^{\mathrm{row}} = 256$ bits and $d^{\mathrm{col}} = 128$ bits. For scenarios with a low number of C-RCDs per relay, the *NoIntfc-SingleRel* scenario exhibits the poorest performance. This is due to the limitations of a single computing segment in handling the computational demands of the CNN model within the specified latency, despite the absence of interference. As the number of C-RCDs per relay increases, performance differences between scenarios diminish. This trend occurs because the *NoIntfc-SingleRel* scenario reaches an optimal C-RCDs per relay configuration, leveraging the advantages of a single computing segment. However, excessive C-RCDs per relay in other scenarios lead to diminishing returns beyond a certain threshold.

Across various K_m values, the proposed *DoubleRel* scenario consistently outperforms alternative scenarios. This is primarily due to its larger computing capacity compared to the *NoIntfc-SingleRel* scenario, and its ability to handle interference better than the *NoIntfc-AllocBW* scenario. Regardless of the scenario, higher $\mathrm{SINR_T}$ values contribute to improved energy efficiency despite increased communication costs. This outcome arises from the fact that higher $\mathrm{SINR_T}$ values reduce the lower bounds in equations (8), (15), and (19), thus requiring less time for communication-related phases. This allows

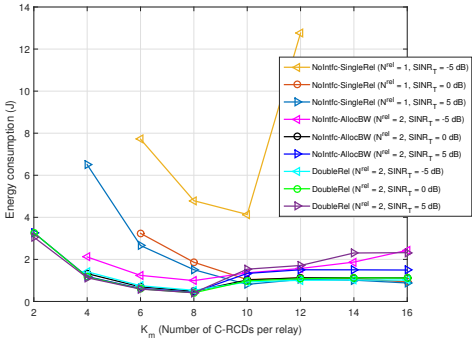

Fig. 5. The energy consumption of different scenarios vs. K_m for several SINR_T values for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits,

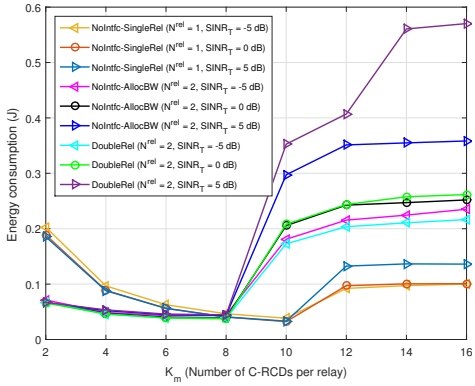

Fig. 6. The energy consumption of different scenarios vs. K_m for several SINR_T values for $d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits

the system to allocate more time for local computation, resulting in significantly reduced computation costs. In scenarios with high computation loads, a slightly higher SINR_T value could potentially enhance overall energy efficiency.

In Figure 6, the same configuration is used with smaller input data sizes, specifically $d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits. In this case, a clear relationship between energy consumption and SINR_T values is observed. Increasing SINR_T leads to higher energy consumption. Unlike in Figure 5, where energy consumption decreases with higher SINR_T values, here, the dominant factor in energy consumption is communication costs due to the reduced computational load resulting from smaller input data sizes. Notably, the *NoIntfc-SingleRel* scenario exhibits significantly lower energy consumption than other scenarios when there is a high number of C-RCDs per relay. This is due to the decreased input data sizes, which alleviate the computational demands for this scenario. However, it still performs poorly for a low number of C-RCDs per relay due to the lack of sufficient computing capability compared to the other scenarios.

These two figures underscore the trade-off between communication and computation and emphasize the importance of tailoring the system implementation to the computational load of the CNN model. Notably, the K_m value is predefined and not optimized in this system. Thus, the key takeaway is to compare the energy consumption of different scenarios at the K_m value where they perform optimally.

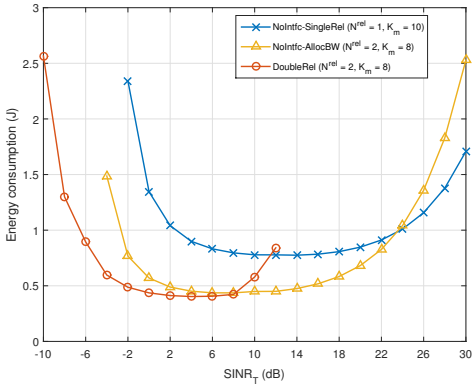

Fig. 7. The energy consumption of different scenarios with the optimal K_m values vs. SINR_T for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits

B. Impact of SINR_T Value on Energy Consumption of Different Scenarios

Figure 7 depicts the energy consumption of different scenarios at their respective optimal K_m values. The investigation varies SINR_T values to assess their impact on the trade-off between communication and computation. At low SINR_T values, all scenarios exhibit significantly higher energy consumption due to the inverse relationship between SINR_T value and the lower bounds specified by equations (8), (15), and (19). A decrease in SINR_T value leads to higher lower bounds, prompting the system to allocate more time for communication and less for computation. As a result, computation costs substantially increase.

Conversely, with increasing SINR_T values beyond a certain threshold, communication costs start to dominate the overall energy consumption. Therefore, raising SINR_T values beyond this threshold does not yield energy efficiency benefits. The optimal SINR_T value strikes a balance between communication and computation costs, ensuring the system allocates an appropriate amount of time for both phases. This trade-off is clearly visible in the figures and underlines the importance of selecting an optimal SINR_T value that suits the specific system's characteristics and computational demands.

C. Energy Consumption of Communication and Computation with Different K_m and SINR_T Values

In this subsection, the energy consumption of communication and computation in *DoubleRel* scenario is individually investigated for different K_m and SINR_T values where the aim is to attain a clearer comprehension of the trade-off between communication and computation.

In Fig. 8, with input data sizes set to $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits, the enhancement in energy efficiency is evident as the number of C-RCDs per relay increases, up to a certain threshold. This increase in efficiency occurs due to computation costs dominating the overall energy consumption when the C-RCDs are few, resulting in insufficient computation capacity. Beyond 8 C-RCDs per relay, energy consumption starts to rise. This is due to the point where additional C-RCDs introduce higher communication costs, overshadowing the benefits from reduced computation costs.

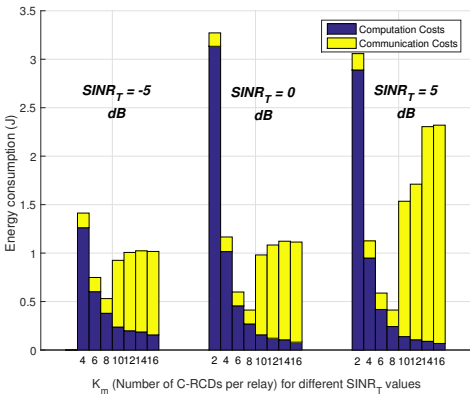

Fig. 8. The energy consumption of communication and computation with different K_m and SINR_T for *DoubleRel* scenario for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits

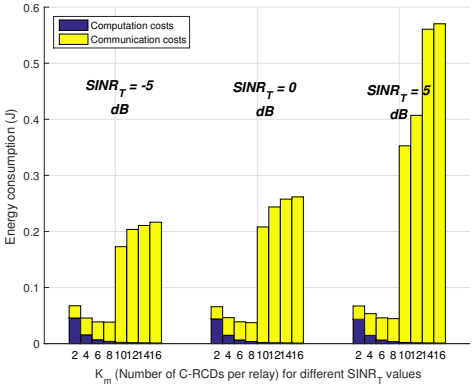

Fig. 9. The energy consumption of communication and computation with different K_m and SINR_T for *DoubleRel* scenario for $d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits

While collaboration improves computational capabilities until a particular threshold, further collaboration becomes inefficient as communication costs surpass computation costs.

Fig. 9 employs the same experimental setup but with smaller input data sizes ($d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits). The conclusions are aligned with those from Fig. 8, yet notable differences emerge due to the reduced computational load. Communication costs gain prominence relative to computation costs, leading to a diminished energy consumption difference between the optimal K_m value (8) and lower K_m values.

IX. Conclusion

This study presents an interference-aware relay-assisted collaborative convolutional neural network (CNN) execution framework that integrates hybrid parallelism, encompassing data and model parallelism. By addressing challenges like communication overhead and interference effects, the framework aims to minimize energy consumption during collaborative CNN execution. The investigation into the impact of the signal-to-interference-noise ratio (SINR) target value on communication-related stages underscores its role in shaping the trade-off between communication and computation energy consumption. The proposed approach is formalized as a convex optimization problem, which can be effectively solved using classical convex optimization techniques and Lagrange duality theory. Extensive simulations corroborate the effectiveness of the proposed approach when compared to benchmark scenarios, underscoring the importance of the chosen SINR target value in optimizing collaborative CNN execution within interference-prone environments.

References

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[2] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," *IEEE Internet of Things J.*, vol. 7, no. 8, pp. 7457-7469, Aug. 2020.

[3] W. Ren *et al.*, "A Survey on Collaborative DNN Inference for Edge Intelligence," *arXiv:2207.07812, [Online]*, 2022, Available: https://arxiv.org/abs/2207.07812.

[4] L. Zeng, X. Chen, Z. Zhou, L. Yang and J. Zhang, "CoEdge: Cooperative DNN Inference With Adaptive Workload Partitioning Over Heterogeneous Edge Devices," *IEEE/ACM Trans. on Netw.*, vol. 29, no. 2, pp. 595-608, April 2021.

[5] E. Li, L. Zeng, Z. Zhou and X. Chen, "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 1, pp. 447-457, Jan. 2020.

[6] J. Mao, X. Chen, K. W. Nixon, C. Krieger and Y. Chen, "MoDNN: Local distributed mobile computing system for Deep Neural Network," *Design, Automat. Test in Europe Conf. Exhibition (DATE)*, Lausanne, pp. 1396-1401, 2017.

[7] Z. Zhao, K. M. Barijough and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," *IEEE Trans. on Comput.-Aided Design Integr. Circuits and Syst.*, vol. 37, no. 11, pp. 2348-2359, Nov. 2018.

[8] E. Kilcioglu, H. Mirghasemi, I. Stupia and L. Vandendorpe, "An Energy-Efficient Fine-Grained Deep Neural Network Partitioning Scheme for Wireless Collaborative Fog Computing," *IEEE Access*, vol. 9, pp. 79611-79627, 2021.

[9] E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Collaborative On-Device CNN Execution Considering Model Parallelism for Latency-Critical Applications," *will be presented at 2023 IEEE Int. Symp. on Personal, Indoor and Mobile Radio Commun. (PIMRC)*, Toronto, Canada, pp. 1-7, 2023.

[10] E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Relay-Assisted Communication Scheme for Collaborative On-Device CNN Execution Considering Hybrid Parallelism," *Submitted to IEEE Access*, 2023.

[11] T. Yang, Y. Chen, J. Emer and V. Sze, "A method to estimate the energy consumption of deep neural networks," *51st Asilomar Conf. on Signals, Syst., and Comput.*, Pacific Grove, CA, pp. 1916-1920, 2017.

[12] F. Wang, J. Xu, X. Wang and S. Cui, "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 3, pp. 1784-1797, March 2018.