UCLouvain

INSTITUTE OF INFORMATION AND COMMUNICATIONS TECHNOLOGIES, ELECTRONICS AND APPLIED MATHEMATICS (ICTEAM)

Collaborative On-device CNN Inference: Design and Optimization of Communication and Computation

Author:
Emre Kilcioglu

Supervisor:
Prof. Luc Vandendorpe

A thesis submitted in fulfillment of the requirements for the degree of
Ph.D. in Engineering Sciences and Technology

Dissertation committee:

Prof. David Bol (UCLouvain) (Chair)
Prof. Luc Vandendorpe (UCLouvain) (Supervisor)
Prof. Jerome Louveaux (UCLouvain)
Prof. Sofie Pollin (KU Leuven)
Prof. Eli De Poorter (Ghent University)
Prof. Jeroen Famaey (University of Antwerp)

March 2024

Declaration of Authorship

I, Emre Kilcioglu, declare that this thesis titled, "Collaborative On-device CNN Inference: Design and Optimization of Communication and Computation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

Abstract

Engineering Sciences and Technology

Ph.D.

Collaborative On-device CNN Inference: Design and Optimization of Communication and Computation

by Emre Kilcioglu

Deep learning (DL) models have experienced a notable increase in adoption across various latency-critical artificial intelligence (AI) applications, including computer vision, natural language processing, and autonomous driving. This adoption has been driven by advancements in mobile internet access and CPU capabilities, leading to the generation of vast volumes of data from mobile devices to be potentially processed in DL-driven AI applications. However, the computational demands of these DL models pose significant challenges, especially when deploying them on resource-constrained devices (RCDs), which are characterized by limited computing power, battery life, and memory. Although various model compression techniques aim to make DL models suitable for single-device execution, they often come at the cost of accuracy. Consequently, innovative approaches are needed to address these challenges and enable efficient execution of DL-driven applications on such devices.

Initially, the solution to the computational demands of DL-driven applications was a cloud-centric paradigm, known as mobile cloud computing (MCC). In MCC, computationally intensive DL operations are offloaded to centralized cloud servers, while RCDs primarily serve as data collectors without hosting DL models locally. However, this approach faces challenges such as intolerable latency, network congestion, and privacy concerns. An alternative approach is to leverage edge computing in conjunction with DL-driven AI applications, giving rise to the concept of edge intelligence (EI). EI aims to bring the computation-heavy DL execution of AI applications to edge networks. This approach offers advantages like lower latency, reduced network congestion, decentralized structures, and more reliable systems compared to the cloud-centric approach. The typical EI

solution involves deploying powerful edge servers for DL execution at the edge. In situations where deploying these servers is challenging, costly, or impossible, particularly in high-mobility and harsh scenarios, device-to-device collaborative DL execution emerges as a viable alternative for performing these applications.

Within the EI framework, this thesis introduces several schemes for collaborative on-device deep learning inference, where multiple RCDs collaboratively execute DL models without reliance on centralized servers. To achieve collaborative DL inference, this thesis explores various parallelism techniques, including data parallelism, model parallelism, and hybrid parallelism, with a specific focus on convolutional neural networks (CNNs). Data parallelism distributes input data among multiple RCDs, while model parallelism partitions DL models. Hybrid parallelism combines elements of both approaches to tackle challenges related to large input data sizes for model parallelism and the execution of multiple filters for data parallelism. In the context of hybrid parallelism, relay-assisted communication is proposed to facilitate efficient collaboration among devices. These strategies aim to distribute the computational load efficiently and reduce communication overhead to optimize the collaborative deep learning inference for specific scenarios.

In summary, this thesis presents innovative approaches and optimization techniques for collaborative on-device deep learning inference in resource-constrained environments. These contributions advance the field of collaborative edge intelligence and provide efficient solutions for DL-driven applications deployed on mobile and IoT devices.

Acknowledgements

Completing this doctoral thesis has been a long journey, and I owe a debt of gratitude to many who have supported and inspired me along the way.

First and foremost, I extend my deepest appreciation to my supervisor, Prof. Luc Vandendorpe, for his unwavering guidance, invaluable expertise, and continuous support throughout this journey. His mentorship has been instrumental in shaping my research and academic growth.

I extend my heartfelt gratitude to Dr. Ivan Stupia, who has not only been a co-author and a wellspring of inspiration for this research but has also become a trusted friend. Our collaborative efforts and shared passion for research have not only enriched this thesis but have also made this journey more enjoyable and fulfilling. I am also thankful to Dr. Hamed Mirghasemi for his invaluable contributions in the early stages of shaping this research and for being a co-author of its initial publication.

I would like to express my sincere appreciation to the members of my dissertation committee: Prof. Jerome Louveaux, Prof. Sofie Pollin, Prof. Eli De Poorter, and Prof. Jeroen Famaey. Their insightful feedback and constructive criticism have been instrumental in shaping the quality and direction of this research.

I express deep gratitude to my parents, Suat Kilcioglu and Zübeyde Kilcioglu, as well as my sister, Elif Kilcioglu, for their consistent love and support. Their enduring belief in me has served as the foundation of my determination.

My dear wife, Şeyma Kilcioglu, has been an unwavering source of strength. Her boundless patience, understanding, encouragement, and personal sacrifices have sustained me through the challenges of this academic journey. Her infinite support and motivation have been a driving force behind my completion of this research.

Last but certainly not least, my precious baby son, Ata Kilcioglu, who joined our family during this doctoral journey. He brought boundless joy and love into my life. His smiles and laughter provided a much-needed respite from the rigors of research.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial intelligence
BC	Broadcasting
CNN	Convolutional neural network
CPU	Central processing unit
C-RCD	Collaborative resource-constrained device
DL	Deep learning
DNN	Deep neural network
DO-RCD	Data owner resource-constrained device
EI	Edge intelligence
ES	Edge server
FTP	Fused tile partitioning
FRD	First round distribution
ID	Input distribution
IoT	Internet of things
MAC	Multiply-accumulate
MCC	Mobile cloud computing
OT	Output transmission
RCD	Resource-constrained device
SINR	Signal-to-interference-noise ratio

List of Symbols

Note: Some of the symbols are common in multiple chapters

Chapter 2

d_n^{ch}	Number of filters in layer-pack n
i_n^{conv}	Input data of convolutional layer n
o_n^{conv}	Output data of convolutional layer n
o_n^{pool}	Output data of pooling layer n
d^{row}	Number of rows in the input data
d^{col}	Number of columns in the input data
N^{LAY}	Number of layer-packs in the CNN model
$f_{\theta_n}^{\text{conv}}$	θ^{th} filter of convolutional layer n

Chapter 3

$o_{n,k}^{\text{conv}}$	Output data size of convolutional layer n for RCD k
$i_{n,k}^{\text{conv}}$	Input data size of convolutional layer n for RCD k
$o_{n,k}^{\text{pool}}$	Output data size of pooling layer n for RCD k
f_n^{conv}	Filter size of convolutional layer n
f_n^{pool}	Filter size of pooling layer n
ZP_n	Zero-padding setting parameter of convolutional layer n
s_n^{conv}	Stride setting parameter of convolutinal layer n
s_n^{pool}	Stride setting parameter of pooling layer n
K	Number of participating RCDs
d	2D input data of the CNN model
$d_{n,k}^{\text{row}}$	Number of rows for which RCD k is responsible during layer-pack n operation
$\widetilde{k}$	Master device index
r_k^{FRD}	Achievable data rate during the first round distribution to RCD k

p_k^{FRD}	RF transmit power of the master device during the first round distribution to RCD k
h_k^{FRD}	Channel gain between the master device and RCD k during the first round distribution
B	Communication bandwidth
N_0	Noise power density
E_k^{FRD}	Energy consumption of the master device during the first round distribution to RCD k
t_k^{FRD}	Allocated time to the first round distribution to RCD k
P_k^c	Constant power consumption of the communication circuitry of RCD k
d_k^{FRD}	Number of bits sent by the master device during the first round distribution to RCD k
$I_{n,k}^{\{\mathrm{a,b}\}}$	Indicator function; 1 if RCD k has a neighbor above or below and 0 otherwise
$p_k^{\max}$	Maximum allowed RF transmit power of RCD k
d_n^{col}	Number of columns in the input data of layer-pack n
$N_{n,k}^{\mathrm{MAC}}$	Number of MAC operations performed by RCD k in layer-pack n operation
$E_{n,k}^{\mathrm{LOC}}$	Energy consumption of RCD k during the local computation of layer-pack n operation
c_k	Number of CPU cycles per MAC operation for RCD k
κ_k	Effective hardware capacitance coefficient of RCD k
$t_{n,k}^{\mathrm{LOC}}$	Allocated time to the local computation of layer-pack n operation in RCD k
$v_k^{\max}$	Maximum CPU frequency of RCD k
$r_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}$	Achievable data rate during the exchange communication with RCD k's neighbors above and below after layer-pack n operation
$p_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}$	RF transmit power of RCD k during the exchange communication with the neighbors above and below after layer-pack n operation
$h_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}$	Channel gain between RCD k and its neighbors above and below during the exchange communication after layer-pack n operation
$E_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}$	Energy consumption of RCD k during the exchange communication with the neighbors above and below after layer-pack n operation

$t_{n,k}^{\mathrm{EXC}_{\{a,b\}}}$	Allocated time of RCD k to the exchange communication with the neighbors above and below after layer-pack n operation
$E_{n,k}^{\mathrm{EXC}}$	Total energy consumption of RCD k during the exchange communication with the neighbors above and below after layer-pack n operation
τ	Maximum allowed latency
t_n	Total allocated time to the local computation and the exchange communication of layer-pack n operation for all RCDs
$E_k^{\mathrm{FRD}_{\mathrm{RF}}}$	RF energy consumed by the master device during the first round distribution to RCD k
$E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}$	RF energy consumed by RCD k during the exchange communication with the neighbors above and below after layer-pack n operation
E^{MAC}	Minimized total energy consumption per MAC operation

Chapter 4

K_n	Number of participating RCDs for layer-pack n operation
$f_{n,k,m}^{\mathrm{conv}}$	m^{th} assigned convolutional filter of RCD k for layer-pack n operation
$d_{n,k}^{\mathrm{ch}}$	Number of filters allocated to RCD k for layer-pack n operation
$o_{n,k}$	Output data of RCD k after layer-pack n operation
r_n^{BC}	Achievable data rate during the broadcasting stage of layer-pack n operation
p_n^{BC}	RF transmit power of the master device during the broadcasting stage of layer-pack n operation
$h_{n,k}^{\mathrm{BC}}$	Channel gain between the master device and RCD k during the broadcasting stage of layer-pack n operation
$R_{n,k}^{\mathrm{BC}}$	Distance between the master device and RCD k during the broadcasting stage of layer-pack n operation
α	Path-loss exponent
c_{PL}	Path-loss constant
f_c	Carrier frequency
c_0	Speed of light
E_n^{BC}	Energy consumption of the master device during the broadcasting stage of layer-pack n operation

t_n^{BC}	Allocated time to the broadcasting stage of layer-pack n operation
d_n^{BC}	Number of bits broadcasted by the master device for layer-pack n operation
$r_{n,k}^{\mathrm{Agg}}$	Achievable data rate during the aggregation stage of layer-pack n operation for the output data of RCD k
$p_{n,k}^{\mathrm{Agg}}$	RF transmit power of RCD k during the aggregation stage of layer-pack n operation
$h_{n,k}^{\mathrm{Agg}}$	Channel gain between the master device and RCD k during the aggregation stage of layer-pack n operation
$R_{n,k}^{\mathrm{Agg}}$	Distance between the master device and RCD k during the aggregation stage of layer-pack n operation
$E_{n,k}^{\mathrm{Agg}}$	Energy consumption of RCD k during the aggregation stage of layer-pack n operation
$t_{n,k}^{\mathrm{Agg}}$	Allocated time of RCD k to the aggregation stage of layer-pack n operation
d_n^{Agg}	Number of bits sent by each RCD during the aggregation stage of layer-pack n operation
$E_n^{\mathrm{BC_{RF}}}$	RF energy consumed by the master device during the broadcasting stage of layer-pack n operation
$E_{n,k}^{\mathrm{Agg_{RF}}}$	RF energy consumed by RCD k during the aggregation stage of layer-pack n operation

Chapter 5

d_m	Number of input data rows assigned to computing segment m
d^R	Number of redundant rows additionally sent to each computing segment
N^{bl}	Number of layer-pack blocks
N_b^{LAY}	Number of layer-packs in block b
N^{rel}	Number of computing segments (or relays)
K_m	Number of C-RCDs in computing segment m
d_b^R	Number of redundant rows sent to each computing segment for block b operation
$r_{b,m}^{\mathrm{ID}}$	Achievable data rate during the input distribution to relay m for block b operation
$p_{b,m}^{\mathrm{ID}}$	RF transmit power of the DO-RCD during the input distribution to relay m for block b operation

$h^{\mathrm{ID}}_{b,m}$	Channel gain between the DO-RCD and relay m during the input distribution for block b operation
$R^{\mathrm{ID}}_{b,m}$	Distance between the DO-RCD and relay m during the input distribution for block b operation
$E^{\mathrm{ID}}_{b,m}$	Energy consumption of the DO-RCD during the input distribution to relay m for block b operation
$t^{\mathrm{ID}}_{b,m}$	Allocated time of the DO-RCD to the input distribution to relay m for block b operation
$d^{\mathrm{ID}}_{b,m}$	Number of bits sent to relay m during the input distribution for block b operation
I_m	Indicator function; 1 for top and bottom relays in the input data map (for $m = 1$ and $m = N^{\mathrm{rel}}$) and 2 otherwise
$r^{\mathrm{BC}}_{n_b,m}$	Achievable data rate during the broadcasting stage of layer-pack n operation of block b for relay m
$p^{\mathrm{BC}}_{n_b,m}$	RF transmit power of relay m during the broadcasting stage of layer-pack n operation of block b
$h^{\mathrm{BC}}_{n_b,k_m}$	Channel gain between relay m and C-RCD k during the broadcasting stage of layer-pack n operation of block b
$R^{\mathrm{BC}}_{n_b,k_m}$	Distance between relay m and C-RCD k during the broadcasting stage of layer-pack n operation of block b
$E^{\mathrm{BC}}_{n_b,m}$	Energy consumption of relay m during the broadcasting stage of layer-pack n operation of block b
$t^{\mathrm{BC}}_{n_b,m}$	Allocated time of relay m to the broadcasting stage of layer-pack n operation of block b
$d^{\mathrm{BC}}_{n_b,m}$	Number of bits broadcasted by relay m for layer-pack n operation of block b
$d_{b,m}$	Number of rows that computing segment m is responsible from for block b operation
$N^{\mathrm{MAC}}_{n_b,k_m}$	Number of MAC operations performed by C-RCD k in computing segment m for layer-pack n operation of block b
$d^{\mathrm{ch}}_{n_b,k_m}$	Number of filters allocated to C-RCD k in computing segment m for layer-pack n operation of block b
$E^{\mathrm{LOC}}_{n_b,k_m}$	Energy consumption of C-RCD k in computing segment m during the local computation of layer-pack n operation of block b
κ_{k_m}	Effective hardware capacitance coefficient of C-RCD k in computing segment m

c_{k_m}	Number of CPU cycles per MAC operation for C-RCD k in computing segment m
$t^{\mathrm{LOC}}_{n_b,k_m}$	Allocated time to the local computation of layer-pack n operation of block b in RCD k of computing segment m
$\nu^{\max}_{k_m}$	Maximum CPU frequency of C-RCD k in computing segment m
$r^{\mathrm{Agg}}_{n_b,k_m}$	Achievable data rate during the aggregation stage of layer-pack n operation of block b for the output data of C-RCD k in computing segment m
$p^{\mathrm{Agg}}_{n_b,k_m}$	RF transmit power of C-RCD k in computing segment m during the aggregation stage of layer-pack n operation of block b
$h^{\mathrm{Agg}}_{n_b,k_m}$	Channel gain between relay m and C-RCD k during the aggregation stage of layer-pack n operation of block b
$R^{\mathrm{Agg}}_{n_b,k_m}$	Distance between relay m and C-RCD k during the aggregation stage of layer-pack n operation of block b
$E^{\mathrm{Agg}}_{n_b,k_m}$	Energy consumption of C-RCD k in computing segment m during the aggregation stage of layer-pack n operation of block b
$t^{\mathrm{Agg}}_{n_b,k_m}$	Allocated time of C-RCD k in computing segment m to the aggregation stage of layer-pack n operation
$d^{\mathrm{Agg}}_{n_b,m}$	Number of bits sent by each C-RCD in computing segment m during the aggregation stage of layer-pack n operation of block b
$r^{\mathrm{OT}}_{b,m}$	Achievable data rate during the output transmission stage of block b operation for the output data of computing segment m
$p^{\mathrm{OT}}_{b,m}$	RF transmit power of relay m during the output transmission stage of block b operation
$h^{\mathrm{OT}}_{b,m}$	Channel gain between the DO-RCD and relay m during the output transmission stage of block b operation
$R^{\mathrm{OT}}_{b,m}$	Distance between the DO-RCD and relay m during the output transmission stage of block b operation
$E^{\mathrm{OT}}_{b,m}$	Energy consumption of relay m during the output transmission stage of block b operation
$t^{\mathrm{OT}}_{b,m}$	Allocated time of relay m to the output transmission stage of block b operation

$d_{b,m}^{\mathrm{OT}}$	Number of bits sent by relay m during the output transmission stage of block b operation
$d_{n_b}^{\mathrm{ch}}$	Number of filters in layer-pack n of block b
$t_{n_b,m}$	Allocated time of computing segment m to layer-pack n operation of block b
t_b	Allocated time to block b operation
$E_{b,m}^{\mathrm{ID}_{\mathrm{RF}}}$	RF energy consumed by the DO-RCD during the input distribution stage of block b operation
$E_{n_b,m}^{\mathrm{BC}_{\mathrm{RF}}}$	RF energy consumed by relay m during the broadcasting stage of layer-pack n operation of block b
$E_{n_b,k_m}^{\mathrm{Agg}_{\mathrm{RF}}}$	RF energy consumed by C-RCD k in computing segment m during the aggregation stage of layer-pack n operation of block b
$E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}}$	RF energy consumed by relay m during the output transmission stage of block b operation

Chapter 6

$\mathrm{SINR}_{\mathrm{T}}$	SINR target value
$\mathrm{SINR}_{n_b,m}^{\mathrm{BC}}$	SINR during the broadcasting stage of layer-pack n operation of block b for relay m
$h_{n_b,m,m'}^{\mathrm{BC}}$	Channel gain between relay m and relay m' during the broadcasting stage of layer-pack n operation of block b
$R_{n_b,m,m'}^{\mathrm{BC}}$	Distance between relay m and relay m' during the broadcasting stage of layer-pack n operation of block b
$\mathrm{SINR}_{n_b,k_m}^{\mathrm{Agg}}$	SINR during the aggregation stage of layer-pack n operation of block b for C-RCD k in computing segment m
$h_{n_b,k_m,k_{m'}}^{\mathrm{Agg}}$	Channel gain between C-RCD k in computing segment m and C-RCD k in computing segment m' during the aggregation stage of layer-pack n operation of block b
$R_{n_b,k_m,k_{m'}}^{\mathrm{Agg}}$	Distance between C-RCD k in computing segment m and C-RCD k in computing segment m' during the aggregation stage of layer-pack n operation of block b
$\mathrm{SINR}_{b,m}^{\mathrm{OT}}$	SINR during the output transmission stage of block b operation for relay m
$h_{b,m,m'}^{\mathrm{OT}}$	Channel gain between relay m and relay m' during the output transmission stage of block b operation

$R^{\mathrm{OT}}_{b,m,m'}$	Distance between relay m and relay m' during the output transmission stage of block b operation

Chapter 7

$E^{\mathrm{TX}}_{n,k}$	Energy consumption of the transmitter side for layer-pack n operation
$E^{\mathrm{RX}}_{n,k}$	Energy consumption of the receiver side for layer-pack n operation
$r^{\mathrm{TX}}_{n,k}$	Data rate for transmission in kbps for layer-pack n operation
$r^{\mathrm{RX}}_{n,k}$	Data rate for reception in kbps for layer-pack n operation
A^{TX}_{k}	Transceiver-specific constant for transmission
A^{RX}_{k}	Transceiver-specific constant for reception
d_{IP}	Packet overhead during each communication stage in kbits
$E^{\mathrm{commun}}_{n,k}$	Total energy consumption of each communication stage
$t^{\mathrm{commun}}_{n,k}$	Overall allocated time for each communication stage
c^{FLOPS}_{k}	Number of FLOPs per Watt for RCD k's CPU
$N^{\mathrm{FLOP}}_{n,k}$	Number of FLOPs for each computation stage in each device
$t^{\mathrm{comp}}_{n,k}$	Allocated time for each computation stage
f^{CPU}_{k}	CPU working frequency of RCD k
$P^{\mathrm{comp,max}}_{k}$	Maximum power that RCD k can consume for local computation

To my beloved wife Şeyma and my dear son Ata

Chapter 1

Introduction

Deep learning (DL) models have gained increasing prominence in recent years for their ability to deliver high-accuracy results in various latency-critical artificial intelligence (AI) applications. These applications span a wide spectrum, ranging from computer vision, video analytics, and autonomous driving to natural language processing and virtual/augmented reality [1–3]. This surge in DL adoption has been further catalyzed by the proliferation of mobile Internet and the increasing computational capability of central processing units (CPUs), which has led to an unprecedented increase in the volume of data generated by mobile devices that can potentially fuel DL-driven AI applications [4].

However, it is crucial to recognize that achieving high accuracy in DL models comes with significant challenges. Notably, the deep computational demands inherent to DL models can be a formidable hurdle [5]. Many DL-driven applications are computationally intensive, primarily due to two key challenges: huge data size problems and model complexity problems. Huge data size problems arise when the input data dimensions are substantial, necessitating significant computational resources to process. Model complexity problems, on the other hand, stem from the intricate architecture of DL models, leading to a massive number of computations required for inference. These challenges present significant barriers to deploying DL-driven applications on a single resource-constrained device (RCD). RCDs are characterized by limited computing capabilities, constrained battery life, and restricted memory resources, making them unsuitable for executing the resource-intensive operations required by DL models [6, 7]. While some model compression techniques [8, 9] have been proposed to make DL models feasible for single-device execution, they often come at the cost of sacrificing accuracy [4]. Therefore, deploying DL-driven applications on such devices necessitates innovative approaches to

address these challenges and enable efficient execution.

1.1 Deep Learning Inference Concepts

This section explores various approaches to address the demanding computational requirements of DL-driven applications. These approaches include the traditional cloud-centric approach and the emerging paradigm of edge intelligence (EI). These concepts shape the landscape of AI deployment, influencing latency, network congestion, and data privacy concerns.

1.1.1 Cloud-centric Approach

To fulfill the demanding computational requirements of DL-driven applications, the initial approach was to rely on a cloud-centric paradigm, often referred to as mobile cloud computing (MCC) [10–12]. In this paradigm, the computationally intensive DL operations are offloaded and executed on centralized powerful cloud servers, while RCDs primarily function as data collectors without hosting DL models locally [13].

This cloud-centric approach presents a series of challenges that have led to the exploration of alternative strategies:

- **Intolerable latency:** One of the prominent issues with the cloud-centric approach is the latency introduced by the distance between cloud servers and RCDs. The substantial physical separation can hinder the real-time deployment of cloud-centric architectures, especially in scenarios where low-latency responses are critical to the application's functionality [13–15]. For instance, in real-time applications like autonomous driving or augmented reality, even a slight delay in data processing can have severe consequences.
- **Network congestion:** As the number of RCDs continues to grow, along with the substantial volume of data being transmitted to cloud servers, network congestion becomes a pressing concern. The exponential increase in data traffic can lead to severe bottlenecks, making it increasingly difficult to serve all RCDs simultaneously [16,17]. This congestion not only impacts latency but also raises questions about the scalability of the cloud-centric model.
- **Privacy issues:** Another critical concern in the cloud-centric approach is related to data privacy. The raw input data collected by RCDs is typically offloaded to the cloud servers without encryption, potentially exposing sensitive information to security breaches and unauthorized access [7, 18]. This issue becomes particularly critical when

dealing with personal or confidential data, such as in healthcare or finance applications.

1.1.2 Edge Intelligence (EI)

An alternative to the cloud-centric approach is to harness the capabilities of edge computing in order to enable DL-driven AI applications to operate at the edge of the networks [19–26]. This convergence of edge computing and AI gives rise to a transformative concept known as edge intelligence (EI) [27–29]. This innovative convergence of edge computing and AI not only addresses the limitations of the cloud-centric approach but also opens up new horizons for AI deployment across various domains.

The advent of EI is driven by the increasing demand for real-time, low-latency AI applications, particularly in sectors like autonomous vehicles, industrial automation, healthcare, and the Internet of Things (IoT). EI aims to bring computationally intensive DL inference closer to the data source, commonly referred to as the edge. This proximity allows for faster decision-making, reduced latency, and enhanced performance, making it especially advantageous in applications where split-second decisions are vital. It also aims to decentralize and distribute the processing of DL inference, thereby addressing some of the key limitations of the cloud-centric approach. Here are some of the notable advantages that EI brings to the table:

- **Lower latency:** One of the most significant benefits of EI is its ability to significantly reduce latency. By executing DL operations at the edge, in close proximity to the RCDs, EI ensures that data processing occurs swiftly. This is vital for real-time applications like autonomous driving or augmented reality, where even minimal delays can pose safety concerns and affect user experience.

- **Reduced Network Congestion:** EI alleviates the network congestion issues frequently encountered in cloud-centric approaches. Through the distribution of computation across multiple devices at the edge, it minimizes the volume of data that needs to be transmitted to centralized servers. This not only enhances response times but also improves the scalability and efficiency of the network infrastructure.

- **Decentralized Structures:** EI promotes more decentralized system structures. Rather than relying on centralized cloud servers, the computational burden is distributed across multiple devices. This decentralization makes the system more scalable and reliable, especially in

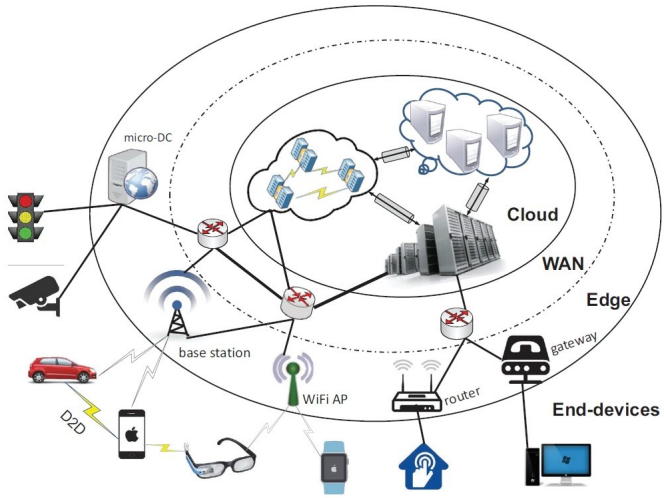


FIGURE 1.1: Edge intelligence (EI) environment [27]

scenarios where network connectivity may be intermittent or unreliable.

- **Improved Reliability:** EI offers improved reliability, as input data remains local compared to the cloud-centric approach.

- **Better Privacy Management:** Privacy is more effectively managed in contrast to the cloud-centric approach, as the data is split into numerous partitions, with each segment residing at the edge.

The typical solution within the EI framework involves deploying powerful edge servers at the network edge to facilitate DL inference. However, there are situations where deploying such edge servers can be challenging, costly, or even impossible, particularly in high-mobility or harsh environmental scenarios. In such cases, device-to-device collaborative DL inference emerges as a viable alternative approach to power DL-driven applications [30]. This collaborative paradigm allows lightweight DL computations to be distributed not only across powerful edge servers but also among multiple RCDs. This collaborative model ensures that even in challenging deployment scenarios, the benefits of EI can still be realized, ultimately expanding the reach of DL-driven AI applications to a wider range of use cases and environments.

1.2 DNN Partitioning

When it comes to collaborative DL inference, a critical issue that needs to be addressed is how to optimally distribute the workload of DL-driven applications among multiple devices. While several solutions have been proposed, deep neural network (DNN) partitioning emerges as a particularly effective strategy for workload distribution in these collaborative applications [31–52].

A prevalent approach in DNN partitioning is layer-based partitioning, where the DNN model is divided into separate layers, and specific layers of the model are allocated for inference on RCDs while others are processed on more powerful edge or cloud servers. This layer-based partitioning scheme offers several advantages, such as energy savings on RCDs, which is especially crucial for devices with limited battery life [31–35]. However, there are some considerations to keep in mind with this approach:

- **Server Dependency:** Layer-based partitioning heavily relies on the availability of powerful edge or cloud servers to offload operations effectively. This server dependence can be a limitation, particularly in scenarios where server availability is sparse or unpredictable, yet strict latency constraints must still be met.
- **Challenges with Large Input Data:** Layer-based DNN partitioning may not fully address the challenges posed by substantial input data dimensions. Partitioning the DNN model layer by layer may not be sufficient since a single RCD could face difficulties processing even a single layer operation efficiently.

To overcome these challenges and ensure efficient collaborative DL inference across multiple RCDs, it is imperative to investigate techniques that exploit computation parallelism. Parallelism can be instrumental in distributing the computational load across multiple devices in a more granular manner, allowing for the efficient utilization of the collective computational power of RCDs.

1.3 Computation Parallelism Techniques

In the context of collaborative inference, this thesis explores three distinct computation parallelism techniques: data parallelism, model parallelism, and hybrid parallelism, which combines elements of both data and model parallelism. Each technique brings its own set of advantages and presents specific challenges, especially when applied to convolutional neural networks (CNNs), which is the primary focus of our research in this thesis.

Table 1.1 presents the pros and cons of each parallelism technique, along with the research objectives specific to each technique.

1.3.1 Data Parallelism

In the data parallelism approach, input data is divided into partitions and distributed among multiple devices. Each device performs complete model computations using its allocated partial input data. To facilitate collaboration, the partial input data is sent to each device, along with additional redundant rows that account for the convolution operation's characteristics. After each layer operation, only the boundary rows are exchanged between neighboring devices on the input data map to carry out subsequent layer operations. Such a scenario is discussed in the study of Chapter 3.

Despite its effectiveness in handling large input data sizes, data parallelism faces several challenges:

- **Single Filter Limitation:** Data parallelism is applicable primarily when each convolutional layer contains a single filter. In scenarios where multiple filters are present in each layer, aggregating the outputs at the data owner after each layer operation is needed, which contradicts the efficient exchange communication between neighboring devices. On the other hand, achieving the desired accuracy from a CNN model often necessitates the inclusion of multiple filters in each convolutional layer.
- **Inefficiency:** Transmitting partial input data point-to-point to each device results in system inefficiency, negatively impacting both latency and energy consumption.

Given these limitations, the applicability of data parallelism depends on the specific requirements and constraints of collaborative CNN inference scenarios.

1.3.2 Model Parallelism

In model parallelism approach, CNN model is partitioned and distributed across multiple devices. Each device processes the entire input data using its assigned partial model. Typically, in this approach, the input data is broadcasted from the data owner device to other participating devices. After each layer operation, the output data from each device is sent back to the data owner device, which then generates the new input data for the next layer's operations by aggregating these outputs. A model parallelism approach is implemented and discussed in the study of Chapter 4.

TABLE 1.1: Different parallelism techniques for collaborative CNN inference

Parallelism Technique	**Pros**	**Cons**	**Objectives**
Data Parallelism	- Suitable for large input data sizes - Collaborative inference using partial input data	- Only feasible for CNN models with one filter per layer - Inefficient point-to-point transmission of partial data	- Divide large input data into partitions and optimally allocate them to multiple RCDs for parallel collaborative inference - Communicate with only neighboring devices, exchanging boundary rows of output data after initial distribution, reducing communication overhead
Model Parallelism	- Enables multiple-filter computation - Collaborative inference using partial models	- Struggles with large input data sizes in a limited allowed latency	- Divide each layer operation into partitions, optimally allocating each filter to an RCD for parallel collaborative inference
Hybrid Parallelism	- Combines data and model parallelism - Solves large input data size problems - Enables multiple-filter computation	- Inefficient point-to-point communication between data owner RCD and other participating RCDs, the same partial input data is sent to multiple RCDs separately	- Solve large input data size problem of model parallelism while also enabling multiple-filter computation, which is the problem of data parallelism
Hybrid Parallelism with Relay-assisted Communication	- Efficient communication with relay devices	- Complex to implement, it needs an optimization considering multiple system parameters	- Design a complete collaborative CNN inference architecture by enabling relay-assisted communication in a hybrid parallelism scenario

While model parallelism offers a promising approach for collaborative CNN inference, it also presents several challenges:

- **Single device limitations:** In scenarios where large input data sizes and low-latency constraints are in place, a single device may struggle to complete multiple filter computations efficiently.
- **Communication Constraints and Synchronization:** Communication between the data owner and other participating devices may become impractical, especially when devices are geographically distant. The latency introduced by long-distance communication can disrupt the synchronization required for collaborative model inference.
- **Communication Inefficiency:** Broadcasting complete input data to all devices can lead to communication inefficiency, primarily determined by the device with the weakest channel condition.

Given these challenges, it becomes crucial to explore alternative parallelism techniques, particularly in scenarios where the issues mentioned above are prevalent.

1.3.3 Hybrid Parallelism

The integration of both data and model parallelism emerges as a promising approach for collaborative CNN inference. This approach seeks to address the challenges posed by large input data sizes through data parallelism while accommodating computations involving multiple filters through the concept of model parallelism. However, the success of this combined approach relies heavily on efficient point-to-point communication between the data owner and other participating devices.

In the context of edge DL training techniques, such as federated learning [53], relay-assisted communication becomes a valuable consideration. Some studies [54–61] explore relay-assisted communication to reduce communication costs and enhance network coverage. This approach can be also applied to the edge inference scenarios. A hybrid parallelism scenario with the help of a relay-assisted communication scheme is proposed in the study of Chapter 5. At this point, it is crucial to clarify that the term "relays" refers to devices serving as the intermediate interface between the data owner RCD and other participating RCDs. In other words, the relay will play the role of interface to the collaborative RCDs. These relays play a pivotal role in coordinating communication, managing model parallelism, and facilitating data aggregation. Their function extends beyond the scope of classical relays in wireless communications, which typically aim to enhance the link budget or introduce diversity. In the context of this

thesis, the relays can be referred to as aggregate-forward relays due to their broader role in effectively overseeing the exchange of information between the data owner and other participating RCDs. Despite this broader functionality, the term "relay" will be used exclusively throughout the thesis for brevity.

1.4 Review of Related Research

To optimize the deployment of deep learning architectures in wireless communication and edge intelligence contexts, various studies have provided valuable insights and proposed several methodologies. In this section, we provide a comprehensive review of these studies and their contributions to the field:

- **Survey Articles:** Several comprehensive surveys, including articles such as [5, 26, 27, 62], provide valuable overviews of the landscape of deploying deep learning architectures in wireless communication and edge intelligence scenarios. These surveys provide a broad summary of existing research and identify key trends and challenges.
- **Model Compression Techniques:** In order to perform CNN execution in a single RCD without any need for powerful devices, model compression techniques have emerged as viable strategies. Han *et al.* pioneer a comprehensive approach in [8], incorporating pruning, trained quantization, and Huffman coding. This technique significantly reduces the size of the neural network models, alleviating the computational demands of complex models. However, a critical consideration here is the trade-off between model size reduction and accuracy preservation. Knowledge distillation, proposed by Hinton *et al.* [63], provides an alternative way, allowing the transfer of knowledge from a larger teacher model to a smaller student model. Despite these advancements, the compressed models suffer from accuracy loss compared to the entire CNN model and require separate training before the inference phase.
- **Early Exit Mechanisms:** Early exit mechanisms have gained considerable attention as a means to enhance the efficiency of CNN inference by skipping the unnecessary model computations by exiting the model in an earlier point. Teerapittayanon *et al.* introduce the concept of early exiting in BranchyNet [64], allowing models to terminate early when confident predictions are made. This adaptive inference approach accelerates the inference process, reducing computational overhead. Additionally, recent research by Zniber *et al.*

proposes a dynamic early exit mechanism that adapts the exit strategy based on the complexity of input instances [65]. This dynamic approach aims to find an optimal trade-off point between computational efficiency and model accuracy in different scenarios. However, the challenge persists in determining optimal exit points, especially in collaborative settings where devices may have heterogeneous computing capabilities and encounter different input data sizes.

- **Layer-Based DNN Partitioning:** To preserve accuracy and deploy a unique CNN model for all types of RCDs, DNN partitioning emerges as a promising approach. The article [31] introduces NeuroSurgeon, which employs a layer-based DNN partitioning strategy with a single cutting point. In this approach, DNN operations are executed locally on a single RCD up to a specific layer, with the output of that layer offloaded to the cloud for subsequent layer operations. However, the study [32] argues that cutting the model from a single point might not be the most efficient strategy and presents JointDNN, which explores the model layer by layer, operating some layers in a single RCD and others in the cloud, incorporating multiple cutting points for improved efficiency. The authors of [33] expand on these concepts by introducing DDNN, a distributed DNN architecture that partitions the model among multiple RCDs, edge servers, and the cloud server. While these approaches address certain challenges, they still rely heavily on the cloud server for offloading DNN operations, which can result in a centralized system. Moreover, the issue of handling large input data sizes still remains a concern.

- **Collaborative Local Computing:** As a pioneering approach in collaborative computing across multiple RCDs, Mao *et al.* propose the concept called MoDNN [36] and MeDNN [66], both locally distributed mobile device computing systems designed to minimize latency. In these scenarios, each participating device sends the output of each layer operation (intermediate data) to a master device hosting the DL-driven application. The master device then distributes the input data for the next layer operation to each device. While reducing latency, this approach introduces communication overhead. Furthermore, these approaches primarily focus on optimizing workload distribution among devices while fixing other communication and computation parameters, potentially leaving room for further optimization. To achieve truly efficient execution, it becomes crucial to jointly optimize workload distribution together with communication and computation parameters.

- **Fused Tile Partitioning (FTP):** In [37], Zhao *et al.* introduce the concept called DeepThings, which utilizes a Fused Tile Partitioning (FTP) method. Unlike partitioning DNN models layer by layer, FTP vertically partitions the model in a grid fashion. This strategy reduces the memory footprint of RCDs and enables independent distribution of computation tasks. However, DeepThings still relies on a centralized gateway device to facilitate operations between edge devices. Additionally, the workload distribution approach in DeepThings considers a homogeneous scenario where the computing capabilities of the participating devices are assumed to be equal. It does not account for the heterogeneous computing capabilities or different network conditions between the data owner and participating devices, which are the factors that are essential to consider for efficient execution.

As we go deeper into this research, it becomes clear that addressing the challenges of handling large input data sizes, centralization concerns, and the heterogeneous computing capabilities of devices and network conditions is crucial for advancing the field of collaborative computing in edge intelligence environments. In this thesis, our approaches are designed to be utilized in scenarios involving collaborative CNN inference across multiple low computing capability devices within latency-critical applications. These approaches are designed to address situations where powerful edge or cloud server resources are absent. The subsequent chapters of this thesis will build upon these insights to develop efficient and adaptable approaches for collaborative CNN inference in resource-constrained scenarios.

1.5 List of Publications

Below is a list of publications corresponding to the studies in this thesis:

- **Chapter 3: A Fine-Grained Data Parallelism Approach for Wireless Collaborative Fog Computing**
  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, doi: 10.1109/ACCESS.2021.3084689.

  In this journal paper, we present a scheme for partitioning DNNs in a manner that is energy-efficient and suitable for wireless collaborative fog computing systems. This scheme takes into account both layer-based partitioning and the approach of data parallelism.

- **Chapter 4: A Collaborative On-Device CNN Inference Scheme Considering Model Parallelism**
  E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Collaborative On-Device CNN Execution Considering Model Parallelism for Latency-Critical Applications," 2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Toronto, ON, Canada, 2023, pp. 1-7, doi: 10.1109/PIMRC56721.2023.10293767.

  This conference paper presents a method for on-device execution architecture of CNNs using collaborative model parallelism.

- **Chapter 5: A Relay-Assisted Hybrid Parallelism Architecture for Collaborative On-Device CNN Inference**
  E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Relay-Assisted Communication Scheme for Collaborative On-Device CNN Execution Considering Hybrid Parallelism," IEEE Access, vol. 11, pp. 99397-99412, 2023, doi: 10.1109/ACCESS.2023.3314381.

  This research paper investigates an on-device CNN inference method using relay assistance and collaborative strategies, emphasizing the optimization of latency-critical deep learning applications through a hybrid parallelism approach that incorporates both data and model parallelism techniques.

- **Chapter 6: Interference-Aware Relay-Assisted Collaborative CNN Inference with Hybrid Parallelism**
  E. Kilcioglu, I. Stupia and L. Vandendorpe, "Interference-Aware Optimization of Energy-Efficient Relay-Assisted Collaborative CNN Execution," Presented in 2023 IEEE Virtual Conference on Communications (VCC), pp. 1-6, 2023.

  In this conference paper, we propose a collaborative CNN inference framework that incorporates hybrid parallelism, combining data and model parallelism, to achieve interference-aware and energy-efficient relay assistance with a predefined signal-to-interference-noise ratio (SINR) target.

These publications represent the research conducted in this thesis, addressing various aspects of collaborative CNN inference in resource-constrained scenarios.

1.6 Main Contributions

In this section, we outline the main contributions of this thesis, which encompass four distinct studies. Each study explores innovative approaches to optimize collaborative CNN inference in resource-constrained scenarios, employing various forms of parallelism and communication strategies. The main contributions include:

- **Efficient Collaborative Inference:** The introduction of multiple innovative schemes for collaborative CNN inference, effectively utilizing the computing resources of RCDs without relying on powerful servers.
- **Parallelism Strategies:** Exploration and application of various parallelism strategies, including data parallelism, model parallelism, and hybrid parallelism, tailored to specific scenarios for improved energy efficiency and reduced communication overhead.
- **Joint Optimization:** The development of joint optimization frameworks that consider communication, computation, and workload distribution parameters, offering a holistic approach to energy consumption minimization.
- **Cloud-independent Architectures:** We propose cloud-independent system architectures that leverage the computing resources of participating devices. This approach promotes scalability and reliability, even in challenging deployment scenarios where the connectivity of powerful edge or cloud servers is weak.
- **Convex Optimization Solutions:** At the end of each study in each chapter, we formulate a convex optimization problem with analytical solutions included by using the primal-dual decomposition technique [67]. These analytical solutions offer practical and scalable means of achieving optimal parameter settings in resource-constrained collaborative CNN inference.
- **Interference-Aware Approach:** Specific to the fourth study in Chapter 6, addressing interference challenges and exploring solutions to improve the robustness of collaborative inference in scenarios with potential interference.

These contributions collectively advance the state of collaborative CNN inference, offering efficient and scalable solutions for resource-constrained environments. Our research provides valuable insights into optimizing the deployment of collaborative CNN architectures by leveraging the wireless communications and edge intelligence contexts. Our approaches are

validated with realistic simulation parameters derived from real measurements of a practical computation. As seen and proven from the results of Chapter 7, our collaborative CNN inference approaches are the optimal choice in terms of energy efficiency for the scenarios where there is a strict latency constraint and there exist multiple low computing capability devices to participate the collaborative operation.

1.7 Thesis Outline

This thesis is organized into several chapters, each contributing to a comprehensive exploration of different collaborative on-device CNN inference scenarios.

In Chapter 2, we provide a fundamental understanding of CNNs, a crucial component of deep learning. The chapter's content serves as the foundation for the subsequent chapters, which explore collaborative on-device CNN inference.

Chapter 3 is dedicated to the first study, titled "A Fine-Grained Data Parallelism Approach for Wireless Collaborative Fog Computing". Here, we discuss a data parallelism approach, where the input data is partitioned across multiple RCDs, each executing the same CNN model with their partial input data. This approach is effective in scenarios with large input data sizes and low CNN model complexity but may face challenges with high model complexity.

In Chapter 4, titled "A Collaborative On-Device CNN Inference Scheme Considering Model Parallelism", we propose a model parallelism approach for collaborative CNN inference in scenarios with small input data sizes and high model complexity. The study explores partitioning the model across multiple RCDs, with each RCD performing its partial models with the entire input data. However, this approach suffers in the case of large input data sizes.

Chapter 5 brings us to the third study, titled "A Relay-Assisted Hybrid Parallelism Architecture for Collaborative On-Device CNN Inference". This chapter proposes a hybrid parallelism approach, the combination of both data and model parallelism, to overcome the challenges of both approaches for some specific scenarios. A relay-assisted communication scheme is considered to reduce the communication overhead between the data owner device and other participating RCDs. The missing part of this study is the consideration of interference between RCDs and relays.

In Chapter 6, we extend the relay-assisted hybrid parallelism approach from Chapter 5 with an interference-aware paradigm. The study investigates how interference among devices impacts energy consumption, exploring the role of predefined SINR targets in shaping the trade-off between communication and computation and conducting a comprehensive performance analysis.

In Chapter 7, we design new communication and computation models for a potential practical deployment and test the feasibility of our parallelism strategies with real-world simulation parameters. At the end, we identify potential scenarios where our collaborative approaches could shine and other scenarios where individual non-collaborative CNN model execution is more appropriate.

Finally, the thesis is concluded in Chapter 8, providing a brief summary of the key findings and contributions made in the exploration of collaborative on-device CNN inference scenarios. Possible future directions after this research are also given in this chapter.

Chapter 2

Overview on Convolutional Neural Networks (CNNs)

In recent years, deep learning has become a big deal in various fields of technology, with neural networks serving as the cornerstone of intelligent algorithms. Different types of neural networks exist such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and multilayer perceptrons (MLPs). Among them, CNNs have been a game-changer in the field of image analysis and have found widespread applications in various domains, including image classification, object detection, and image generation. One of the key strengths of CNNs lies in their ability to efficiently detect and recognize patterns present in input images, including features like lines, shapes, gradients, and even complex objects like eyes and faces. This chapter serves as a foundation for understanding the principles and mechanisms behind CNNs, setting the stage for subsequent chapters.

A CNN, or ConvNet, is a specialized class of neural networks designed for processing data with a grid-like topology, such as images. Digital images, representing visual data, consist of pixels arranged in a grid, each assigned a specific brightness and color value (Fig. 2.1). Inspired by the human visual system, which processes information through receptive fields, CNN neurons similarly analyze data within their own receptive fields. These networks efficiently detect patterns, ranging from basic features like lines and shapes to more complex objects like eyes and faces.

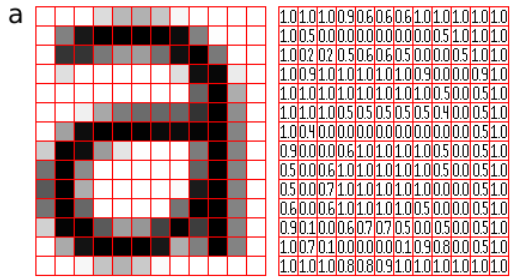


FIGURE 2.1: Representation of image as pixels [68]

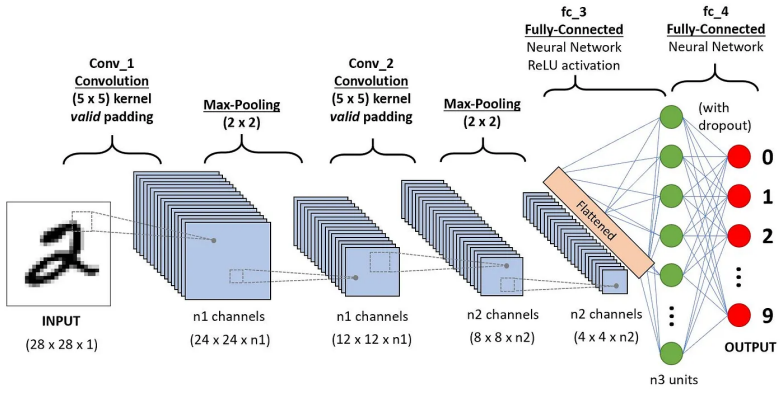


FIGURE 2.2: CNN Architecture [69]

2.1 Architecture of CNNs

CNNs are composed of multiple sequential convolutional and pooling layers and a single fully connected layer at the end (Fig. 2.2).

2.1.1 Convolutional Layers

Convolutional layers represent the fundamental building blocks of CNNs, carrying the primary computational load. These layers consist of multiple learnable filters or kernels, which perform convolution operations on a restricted portion of the input data. Convolutional layers are responsible for feature extraction, enabling the network to recognize patterns, edges, and textures within the data.

There are three hyperparameters affecting the output data size of each convolutional layer, including:

- **Number of filters** determines the number of pages in the output data. For instance, 16 filters correspond to a 16-page 3D output data.
- **Stride** defines the distance, or number of pixels, that the kernel traverses over the input matrix. A larger stride results in a smaller output.
- **Zero-padding** is applied when the filters do not align with the input data. Two types of padding exist:
  - **Valid padding:** Also known as no padding, it reduces the convolved output data dimensionality compared to the input data.
  - **Same padding:** This padding ensures equal data sizes between the input and output data of a convolutional layer by adding zeros to the outside of the input data.

Following each convolution operation, a Rectified Linear Unit (ReLU) transformation is applied to the output data, introducing nonlinearity to the model.

2.1.2 Pooling Layers

Pooling layers play a crucial role in downsampling the spatial dimensions of the output data produced by convolutional layers. Common pooling techniques include max-pooling and average-pooling, where the maximum or average value in each patch of the input data is selected, respectively. This downsampling operation helps in managing computational complexity while retaining essential information. Fig. 2.3 provides an example scenario illustrating the pooling process.

2.1.3 Fully Connected Layers

The final layer of the CNN is tasked with classification based on features extracted from preceding layers and their diverse filters. While convolutional and pooling layers typically employ ReLU functions, fully connected layers commonly leverage a softmax activation function. This function assigns probabilities ranging from 0 to 1, facilitating accurate classification of inputs.

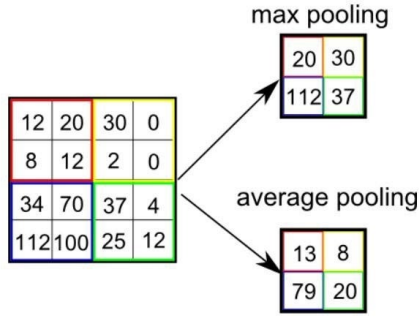


FIGURE 2.3: Pooling layer operation example [69]

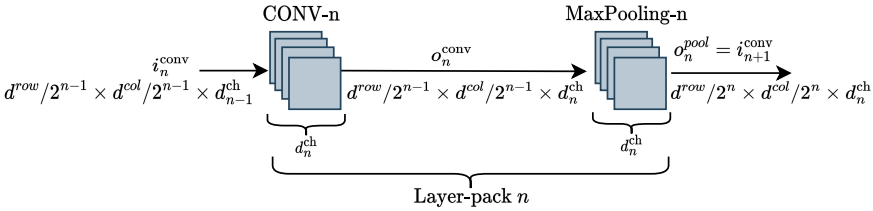


FIGURE 2.4: The layer-pack n operation of a CNN model

2.2 CNN Model Under Consideration

The CNN model under consideration in this thesis follows a typical architecture consisting of multiple pairs of convolutional and pooling layers, with a fully connected layer at the end. Throughout this thesis, we refer to each paired convolutional and pooling layers as a single "layer-pack". Fig. 2.4 illustrates a layer-pack n operation where d_n^{ch} represents the number of filters in convolutional or pooling layer n, i_n^{conv} and o_n^{conv} are the input and output data of convolutional layer n, respectively and o_n^{pool} denotes the output data of pooling layer n. The 2D input data that undergoes processing within the CNN model is referred to as i_1^{conv} and has dimensions of $d^{\text{row}} \times d^{\text{col}}$. To understand the relationship between the input and output of convolutional layer n, we turn to the fundamental properties of the convolution operation, which is mathematically defined as follows:

$$\begin{aligned} o_n^{\text{conv}}(:,:,\theta) = & f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,1) + f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,2) \\ & + \ldots + f_{\theta_n}^{\text{conv}} * i_n^{\text{conv}}(:,:,d_{n-1}^{\text{ch}}) \end{aligned} \tag{2.1}$$

where $n \in [N^{\text{LAY}}]$ is the layer-pack index, N^{LAY} is the number of layer-packs in the CNN model, $\theta \in [d_n^{\text{ch}}]$ is the page or third dimension index of the 3D output data o_n^{conv}, and $f_{\theta_n}^{\text{conv}}$ signifies the θ^{th} filter of convolutional layer n. In our model, a filter size of 3×3 is selected for each filter $f_{\theta_n}^{\text{conv}}$ due to its widespread utilization in CNN implementations. Due to the distributivity property of convolution, (2.1) is identical to

$$o_n^{\text{conv}}(:,:,\theta) = f_{\theta_n}^{\text{conv}} * \left(\sum_{\phi=1}^{d_{n-1}^{\text{ch}}} i_n^{\text{conv}}(:,:,\phi) \right) \tag{2.2}$$

This equation reveals that the summation of all pages of i_n^{conv} is convolved with each filter in convolutional layer n to obtain each page of o_n^{conv}. In the specific case of $n = 1$, where $d_0^{\text{ch}} = 1$ is assigned due to the initial input data i_1^{conv} being two-dimensional. Same padding is applied to the input data as a zero-padding technique to ensure equal data sizes between the input and output data of a convolutional layer.

The output of each convolutional layer passes through a maximum-pooling layer such that we have $o_n^{\text{pool}}(:,:,\theta) = MP\left(o_n^{\text{conv}}(:,:,\theta)\right)$ where $MP()$ is the maximum-pooling operator. In this thesis, a 2×2 filter size is used for maximum-pooling filters to halve the data size after each pooling layer.

This detailed explanation provides a clear understanding of the CNN model architecture used in this thesis. It highlights the key operations, such as convolution and pooling, as well as the dimensions of input and output data at different stages of processing. This model forms the basis for the subsequent studies presented in the following chapters.

Chapter 3

A Fine-Grained Data Parallelism Approach for Wireless Collaborative Fog Computing

In this chapter, we introduce an energy-efficient fine-grained DNN partitioning scheme designed for wireless collaborative fog computing systems, considering both layer-based partitioning and data parallelism approach. While our study is centered around CNNs, the approach can be adapted to other DNN models with minimal adjustments to mathematical modeling.

In our proposed model, the execution of each layer operation adopts a collaborative approach that efficiently partitions the workload among multiple RCDs. This collaborative process initiates with one RCD, designated as the master device, receiving a 2D input data sample. Subsequently, the other participating devices assume the role of slaves. The master device horizontally divides the 2D input data sample into slices and optimally allocates these slices among all participating RCDs, including itself. This stage is referred to as the first round distribution (FRD). Following the FRD, RCDs perform their local computation for each layer by using their assigned partial input data. After each layer operation, RCDs exchange boundary rows of the 2D output data with their neighboring RCDs, responsible for adjacent slices above and below, facilitating subsequent layer operations. This iterative process continues until all collaborative layer operations are successfully executed. Key points and contributions of this study can be listed as

- **Collaborative Fog Computing Scheme with Data Parallelism:** We propose a collaborative fog computing scheme that leverages a fine-grained data parallelism approach specifically tailored for deep learning applications.
- **Efficient Communication Management:** Our approach significantly reduces communication overhead between RCDs. Communication is optimized to minimize data transfer between devices, especially focusing on the efficient exchange of boundary rows during layer operations.
- **Joint Optimization:** Instead of focusing solely on workload distribution optimization while keeping other dynamic parameters fixed, we take a holistic approach by jointly optimizing communication parameters, computation resources, and workload distribution. This comprehensive optimization strategy enhances the overall system efficiency.
- **Server-Independent Architecture:** Our system is designed to utilize the computing resources of participating RCDs without relying on powerful central servers. This architecture promotes scalability and resource efficiency.
- **Convex Optimization Framework:** To achieve optimal parameter settings, we formulate the problem as a convex optimization problem. We employ analytical solutions through primal-dual decomposition [67] and Lagrange duality theory, with validation against the CVX software of MATLAB [70]. This procedure provides practical and scalable problem-solving.

3.1 CNN Model

In this section, we describe the CNN model used in this study, which follows the same architecture as discussed in Chapter 2. However, in this model, we make the assumption that each convolutional layer contains a single filter. Figure 3.1 visually represents this CNN model, where N^{LAY} represents the number of layers operated collaboratively within multiple RCDs.

For each dimension of the data, we can define the relationship between the input and output sizes of a single layer in each RCD as follows:

$$o_{n,k}^{\mathrm{conv}} = \left\lfloor \frac{i_{n,k}^{\mathrm{conv}} - f_n^{\mathrm{conv}} + 2\,Z P_n}{s_n^{\mathrm{conv}}} \right\rfloor + 1 \tag{3.1}$$

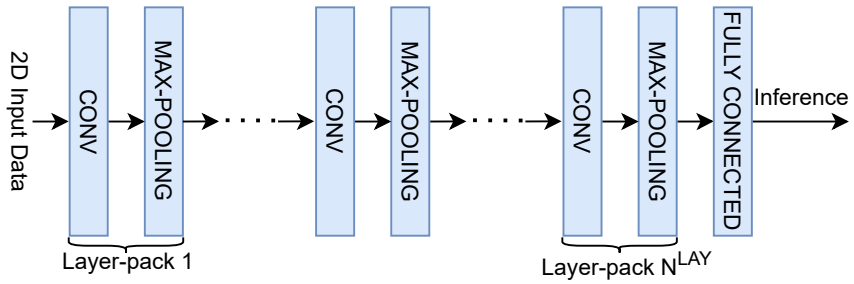


FIGURE 3.1: CNN model of the study in Chapter 3

and

$$o_{n,k}^{\text{pool}} = \left\lfloor \frac{o_{n,k}^{\text{conv}} - f_n^{\text{pool}}}{s_n^{\text{pool}}} \right\rfloor + 1 \tag{3.2}$$

Here, $\{n, k\}$ corresponds to the indices for layer-pack and RCD, respectively, $i_{n,k}^{\text{conv}}$ and $o_{n,k}^{\text{conv}}$ represent the input and output data sizes of convolutional layer n, while $o_{n,k}^{\text{pool}}$ denotes the output data size of pooling layer n. Further, f_n^{conv} and f_n^{pool} are the filter sizes of convolutional layer n and pooling layer n, respectively. ZP_n represents the zero-padding setting parameter of convolutional layer n, and s_n^{conv} and s_n^{pool} are the stride setting parameters of convolutional layer n and pooling layer n, respectively.

In this study, we employ 3×3 filters for convolutional layers ($f_n^{\text{conv}} = 3$) with a unit stride ($s_n^{\text{conv}} = 1$), and we apply zero-padding ($ZP_n = (f_n^{\text{conv}} - 1)/2$) to ensure equal input and output data sizes for each convolutional layer. For pooling layers, we use 2×2 filter sizes ($f_n^{\text{pool}} = 2$) with a stride of two ($s_n^{\text{pool}} = 2$), resulting in both the number of rows and columns being halved from their initial values, while the total dimension of the 2D output data becomes a quarter of its initial value.

Under these parameter settings, we can simplify the relationships as follows:

$$o_{n,k}^{\text{conv}} = i_{n,k}^{\text{conv}} \tag{3.3}$$

and

$$o_{n,k}^{\text{pool}} = \left\lfloor \frac{o_{n,k}^{\text{conv}}}{2} \right\rfloor = \left\lfloor \frac{i_{n,k}^{\text{conv}}}{2} \right\rfloor \tag{3.4}$$

Here, $\lfloor x \rfloor$ represents the floor function, yielding the greatest integer less

than or equal to the value of x. These definitions provide clarity on the relationships between input and output sizes within a single RCD in our CNN model.

3.2 System Model

In this section, we outline the system model for our study, which involves a set of K active single-antenna RCDs and an access point/base station connected to the cloud. The cloud resources are primarily used for the initial training of the CNN model. Once trained, the system operates among the multiple RCDs without the constant presence of the cloud server. Each RCD hosts an identical CNN model.

The key elements and operations within our proposed scenario, depicted in Fig. 3.2, are as follows:

1. **Cloud-Based CNN Training:** Initially, the CNN model is trained in the cloud using offline datasets. It is essential to note that in this study, our focus is on energy consumption during the inference phase. Typically, training is performed offline on powerful cloud servers, which is the common idea in the literature [34]. Given the latency constraints of our scenario, optimizing training within a limited time frame is not optimal since the training needs to be performed as full power as possible in order not to lose accuracy.

2. **Model Offloading:** The trained model weights w are offloaded to the RCDs.

3. **Master-Slave RCD Architecture:** When RCD k receives a 2D input data sample d, with dimensions d^{row} rows and d^{col} columns, it becomes the master device, while other participating RCDs take on the role of slaves. We ensure that each RCD is allocated to specific time frames to act as the master device, preventing collisions when multiple RCDs have 2D input data samples to be processed with collaborative CNN inference.

4. **First Round Distribution (FRD):** The master device horizontally divides the 2D input data sample into slices and optimally allocates them among all participating RCDs, taking into account their heterogeneous computing capabilities and different network conditions. Following the allocation, the master device sends partial input data to each RCD, along with an additional boundary row from its neighboring RCD, as shown in Fig. 3.3. This operation is referred to as the first round distribution (FRD). Here, it is essential to note that the

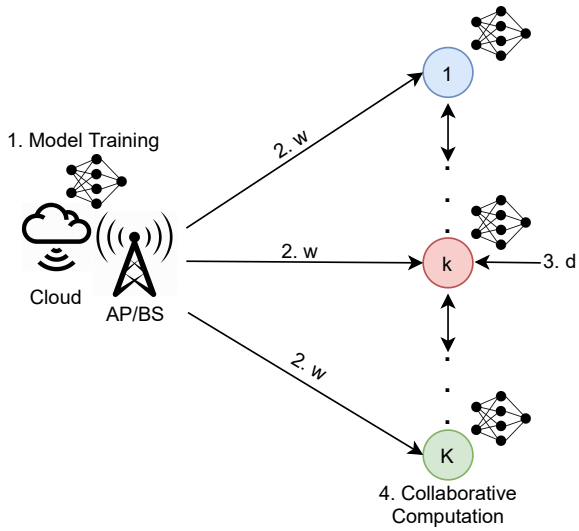


FIGURE 3.2: Illustration of the system model for the study in Chapter 3

placement of each RCD on the input data is determined by the master device before the operation begins. This ensures that every RCD is aware of its neighboring RCDs throughout the operation. Given the single-antenna nature of all RCDs, data transmission occurs in predefined time frames to avoid conflicts.

5. **Collaborative Local Computation:** Each RCD initiates its local computation after all RCDs receive their partial input data, ensuring synchronization for each layer operation.

6. **Max-Pooling and Data Exchange:** Following a single convolutional layer operation in each RCD, if max-pooling is applied to all bits, the boundary rows of the output data are transmitted to neighboring RCDs for subsequent layer operations. If a boundary row cannot undergo max-pooling due to a lack of other parts of the output data, it remains unchanged without any max-pooling. This situation is depicted in Fig. 3.3 between RCDs 2 and 3, where both lack the partial data needed for max-pooling.

7. **Iterative Process:** The process continues until the last layer operation in the CNN model. Eventually, the output data is sent back to the master device by each participating slave RCD.

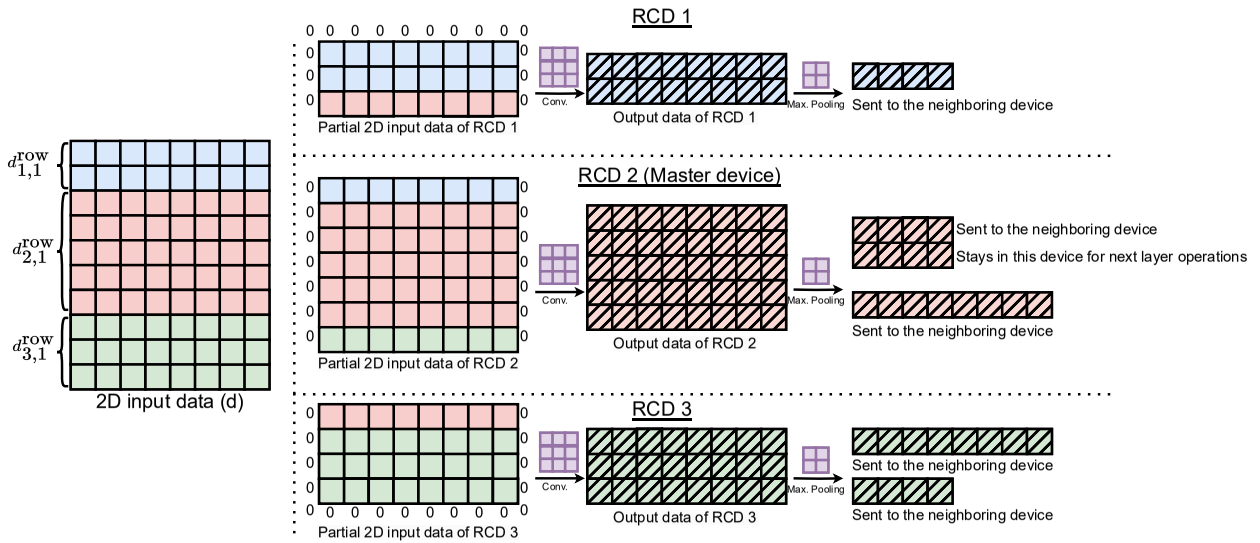


FIGURE 3.3: An example scenario of the study in Chapter 3 for the first layer operation with 3 RCDs

During the FRD stage, the overall energy consumption of collaborative CNN inference is impacted by a variety of factors, including the communication and computation parameters of each participating RCD. The master device plays a critical role in this process, as it carefully considers these factors to minimize the overall energy consumption of the system. It optimally distributes the rows of the input data among all participating RCDs, taking into account their varying computing capabilities and network conditions.

In addition to data distribution, another critical aspect addressed during FRD is the handling of intermediate boundary rows that do not undergo the max-pooling process. These boundary rows, excluded from max-pooling, require thoughtful assignment to neighboring RCDs for subsequent layer operations. A specific example of this scenario is illustrated between RCDs 2 and 3 in Fig. 3.3. In this case, the boundary rows without max-pooling in RCD 2 and RCD 3 collectively form the input data for the next layer. Thus, a critical decision arises regarding whether this resulting row should be operated on by RCD 2 or RCD 3.

This decision-making process inherently involves a trade-off between communication and computation. Specifically, if the undetermined row is operated in RCD 2, it necessitates that RCD 3 transmits its boundary row without max-pooling, along with the next set of additional max-pooled data to RCD 2. This approach, while efficient in terms of computation, incurs a cost in terms of communication overhead for RCD 3. In this study, for the sake of simplifying the optimization problem and ensuring tractability, we assume that, in such scenarios, the upper neighboring RCD always takes on the responsibility of operating the undetermined row for subsequent layer operations. This assumption, despite simplifying the decision process during FRD, has a minimal impact on the overall optimization problem, given that it revolves around only a single-row operation. As can be understood from the explanations in Fig. 3.3, the undetermined row is assigned to RCD 2, which serves as the upper neighbor of RCD 3.

3.3 Energy Consumption Modeling

In the context of collaborative CNN inference, understanding and modeling energy consumption is paramount to optimizing the system's performance. This section delves deep into the energy consumption modeling, providing insights into the critical components that influence the overall effectiveness of the collaborative CNN inference framework. The energy consumption comprises the one-time first round distribution stage from the master device to the other participating slave devices and multiple

sequential collaborative local computation and exchange communication stages for all layer-pack operations.

3.3.1 First Round Distribution (FRD)

During the initial stage of collaborative computation, all rows in the input data must be allocated to the participating RCDs. This allocation is subject to the following constraint:

$$\sum_{k=1}^{K} d_{1,k}^{\text{row}} = d^{\text{row}} \tag{3.5}$$

Here, $d_{n,k}^{\text{row}}$ represents the number of rows for which RCD k is responsible during layer-pack n operation. The master device, denoted as $\widetilde{k}$, transmits slices of the input data to $K-1$ slave RCDs at the achievable data rate

$$r_k^{\text{FRD}} = B \ln\left(1 + \frac{p_k^{\text{FRD}} h_k^{\text{FRD}}}{B N_0}\right) \tag{3.6}$$

where p_k^{FRD} is the RF transmit power of the master device when sending the input data slice to RCD k, h_k^{FRD} is the channel gain between the master device and RCD k, B is the communication bandwidth between the devices and N_0 is the noise power density. The energy consumption to transmit the data slice to RCD k is calculated as

$$E_k^{\text{FRD}} = t_k^{\text{FRD}}(p_k^{\text{FRD}} + P_{\widetilde{k}}^c) \tag{3.7}$$

where P_k^c signifies the constant power consumption of the communication circuitry of RCD k and t_k^{FRD} represents the time required to transmit the input data slice to RCD k.

Two constraints are associated with this stage. First, the transmission rate cannot exceed the achievable data rate, leading to

$$d_k^{\text{FRD}} \leq t_k^{\text{FRD}} r_k^{\text{FRD}} \tag{3.8}$$

where the number of bits d_k^{FRD} sent to RCD k by the master device $\widetilde{k}$ during FRD is defined as

$$\begin{aligned} d_k^{\text{FRD}} &= \begin{cases} d^{\text{col}}(d_{1,k}^{\text{row}} + 1) & \text{for top \& bottom RCDs} \\ d^{\text{col}}(d_{1,k}^{\text{row}} + 2) & \text{for other RCDs} \end{cases} \\ &= d^{\text{col}}(d_{1,k}^{\text{row}} + 1 + I_{1,k}^{\text{a}} I_{1,k}^{\text{b}}) \end{aligned} \tag{3.9}$$

Here, $I_{n,k}^{\{a,b\}}$ are binary indicators; equal to 1 if RCD k has a neighbor above or below, respectively, after layer-pack n operation, or 0 otherwise. The constraint in (3.8) can thus be expressed as

$$d^{\text{col}}(d_{1,k}^{\text{row}} + 1 + I_{1,k}^{\text{a}} I_{1,k}^{\text{b}}) \leq t_k^{\text{FRD}} r_k^{\text{FRD}} \tag{3.10}$$

Second, the RF transmit power should be, at most, the maximum RF power allowed for each RCD.

$$p_k^{\text{FRD}} \leq p_k^{\max} \tag{3.11}$$

where $p_k^{\max}$ is the maximum RF transmit power of RCD k.

3.3.2 Collaborative Local Computation

Following the FRD, all participating RCDs execute layer operations using their allocated partial input data. Convolutional layers dominate the energy consumption in CNNs, even though fully connected layers are more memory-intensive [36, 66, 71]. As the considered CNN model has only one fully connected layer positioned at the end of the model, which results in a considerably smaller input data dimension for this layer, the energy consumption associated with this single fully connected layer is minimal compared to the energy consumption of multiple convolutional layers in the model. Therefore, the proposed scheme primarily focuses on the energy consumption of convolutional layers.

The convolution operation in convolutional layers is decomposed into lots of multiply-accumulate (MAC) operations [71, 72]. Each MAC operation involves taking one element from the filter and one from the shaded input map, i.e., the portion of the 2D input data that the filter is applied to, multiplying them, and accumulating the result to the previous sum. For example, a single element-wise dot product of two 3×3 matrices entails 9 MAC operations.

To determine the energy consumption of local computation per layer for a single RCD, we need to calculate the number of CPU cycles required. For this purpose, we first calculate the number of MAC operations needed for this computation and then multiply it by c_k, which represents the number of CPU cycles needed for a single MAC operation in RCD k. Considering the properties of the convolution operation, the number of MACs $N_{n,k}^{\text{MAC}}$ performed by RCD k in layer-pack n operation can be expressed as

$$N_{n,k}^{\text{MAC}} = (f_n^{\text{conv}})^2 d_n^{\text{col}} d_{n,k}^{\text{row}} = 9 d_n^{\text{col}} d_{n,k}^{\text{row}} \tag{3.12}$$

where d_n^{col} is the number of columns in the 2D input data at the beginning of layer-pack n operation and $d_n^{\text{col}} = \lfloor d_1^{\text{col}} / 2^{n-1} \rfloor$. The energy consumption of layer-pack n operation in RCD k under the low CPU voltage assumption [73–76] is given by

$$E_{n,k}^{\text{LOC}} = \frac{\kappa_k c_k^3 (N_{n,k}^{\text{MAC}})^3}{(t_{n,k}^{\text{LOC}})^2} = \frac{3^6 \kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{n,k}^{\text{row}})^3}{(t_{n,k}^{\text{LOC}})^2} \tag{3.13}$$

where κ_k is the effective capacitance coefficient depending on the hardware architecture and $t_{n,k}^{\text{LOC}}$ is the local computation time allocated to layer-pack n operation for RCD k. Within this stage, the following constraint is imposed:

$$c_k N_{n,k}^{\text{MAC}} = 9 c_k d_n^{\text{col}} d_{n,k}^{\text{row}} \leq t_{n,k}^{\text{LOC}} \nu_k^{\max} \tag{3.14}$$

which signifies the existence of a maximum CPU frequency, denoted as $\nu_k^{\max}$, for each RCD, ensuring that their CPUs do not exceed this value.

3.3.3 Exchange Communication

After convolutional layer operations, the distribution of the input data leads to various scenarios. In the example scenario depicted in Fig. 3.3, the following scenarios arise:

- If an RCD has a neighbor below:
  - If the row number index of the boundary row ends with an even number, the maximum-pooled boundary row is sent to the neighbor below ($d_n^{\text{col}}/2$ bits). This is the case for RCD 1 in Fig. 3.3.
  - If the row number index of the boundary row ends with an odd number, the row that cannot be max-pooled due to the lack of an adjacent row is sent without max-pooling to the neighbor below (d_n^{col} bits). This is the case for RCD 2 in Fig. 3.3.
- If an RCD has a neighbor above:
  - If the row number index of the boundary row starts with an odd number, the maximum-pooled boundary row is sent to the neighbor above ($d_n^{\text{col}}/2$ bits). This is the case for RCD 2 in Fig. 3.3.
  - If the row number index of the boundary row starts with an even number, the row that cannot be max-pooled due to the lack

of an adjacent row and the previous max-pooled row are sent together to the neighbor above for the next layer-pack operations ($3d_n^{\text{col}}/2$ bits). This is the case for RCD 3 in Fig. 3.3.

As a result, there exists a relationship between $d_{n,k}^{\text{row}}$ and $d_{1,k}^{\text{row}}$ depending on the previous starting row number index in the overall input data of a layer and the previously distributed number of rows $d_{n,k-1}^{\text{row}}$ assigned to RCD k. While this relationship cannot be analytically expressed as a general expression, it is approximated that $d_{n,k}^{\text{row}} = d_{1,k}^{\text{row}}/2^{n-1}$. Thus, we can rewrite (3.12), (3.13), and (3.14) as follows

$$N_{n,k}^{\text{MAC}} = \frac{9d_n^{\text{col}}}{2^{n-1}} d_{1,k}^{\text{row}} \tag{3.15}$$

$$E_{n,k}^{\text{LOC}} = \frac{3^6 \kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{1,k}^{\text{row}})^3}{(2^{n-1})^3 (t_{n,k}^{\text{LOC}})^2} \tag{3.16}$$

and

$$\frac{9c_k d_n^{\text{col}}}{2^{n-1}} d_{1,k}^{\text{row}} \leq t_{n,k}^{\text{LOC}} \nu_k^{\max} \tag{3.17}$$

Additionally, it can be seen from the scenario itemized previously that each RCD sends either $d_n^{\text{col}}/2$, d_n^{col} or $3d_n^{\text{col}}/2$ number of bits to its neighbors after each layer-pack operation. Therefore, the number of bits sent to each neighbor after each layer-pack operation is assumed to be as d_n^{col} bits on average.

After local computation, each RCD exchanges the boundary rows with its neighbors at the achievable data rate

$$r_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} = B \ln(1 + \frac{p_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} h_{n,k}^{\text{EXC}_{\{\text{a,b}\}}}}{BN_0}) \tag{3.18}$$

where $\{\text{a},\text{b}\}$ index can take values either a (for the exchange communication with the neighbor above) or b (for the exchange communication with the neighbor below). Additionally, $p_{n,k}^{\text{EXC}_{\{\text{a,b}\}}}$ and $h_{n,k}^{\text{EXC}_{\{\text{a,b}\}}}$ are the RF transmit power of RCD k and the channel gain when transmitting the boundary row of layer-pack n output data to its neighbors above and below, respectively.

The energy consumption of exchanging data after layer-pack n operation in RCD k is

$$E_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} = t_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} (p_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} + P_k^c) \tag{3.19}$$

where $t_{n,k}^{\mathrm{EXC}_{\{a,b\}}}$ are the allocated exchange time for RCD k to transmit the boundary row to its neighbor above and below. Additionally, since the number of bits sent during exchange communication cannot exceed the number of bits that can be sent with the achievable data rate, we have

$$I_{n,k}^{\{a,b\}} d_n^{\mathrm{col}} \leq t_{n,k}^{\mathrm{EXC}_{\{a,b\}}} r_{n,k}^{\mathrm{EXC}_{\{a,b\}}} \tag{3.20}$$

Furthermore, each RCD is subject to a maximum transmitted power constraint such that

$$p_{n,k}^{\mathrm{EXC}_{\{a,b\}}} \leq p_k^{\max} \tag{3.21}$$

Finally, the total energy consumption of exchange communication after layer-pack n operation for RCD k is calculated as

$$E_{n,k}^{\mathrm{EXC}} = I_{n,k}^{a} E_{n,k}^{\mathrm{EXC}_a} + I_{n,k}^{b} E_{n,k}^{\mathrm{EXC}_b} \tag{3.22}$$

3.4 Problem Formulation

In this section, we outline the problem formulation for optimizing collaborative CNN inference. Due to the necessity for participating RCDs to receive previous layer boundary rows from their neighbors before proceeding with the next layer-pack operation, we assume that for each layer, the total time t_n spent on both local computation and exchange communication is consistent across all RCDs. This value is optimized by the master device and serves as an upper bound for the following inequality:

$$t_{n,k}^{\mathrm{LOC}} + I_{n,k}^{a} t_{n,k}^{\mathrm{EXC}_a} + I_{n,k}^{b} t_{n,k}^{\mathrm{EXC}_b} \leq t_n \tag{3.23}$$

Additionally, we impose a latency requirement to ensure that the entire operation is completed within a specified time frame of τ seconds, which can be expressed mathematically as

$$\sum_{\substack{k=1 \\ k \neq \tilde{k}}}^{K} t_k^{\mathrm{FRD}} + \sum_{n=1}^{N^{\mathrm{LAY}}} t_n \leq \tau \tag{3.24}$$

The final optimization problem is defined as

$$\begin{aligned}
\min_{\mathbf{x}_1} \quad & \sum_{\substack{k=1 \\ k\neq\widetilde{k}}}^{K} E_k^{\mathrm{FRD}} + \sum_{k=1}^{K}\sum_{n=1}^{N^{\mathrm{LAY}}} (E_{n,k}^{\mathrm{LOC}} + E_{n,k}^{\mathrm{EXC}}) \\
\text{s.t.} \quad & (3.5), (3.10), (3.11), (3.17), (3.20), (3.21), (3.23), (3.24), \\
& 0 \leq p_k^{\mathrm{FRD}}, p_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}, \\
& 0 \leq d_{1,k}^{\mathrm{row}} \leq d^{\mathrm{row}}, \\
& 0 \leq t_{n,k}^{\mathrm{LOC}}, t_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}} \leq t_n, \\
& 0 \leq t_k^{\mathrm{FRD}} \leq \tau
\end{aligned} \tag{3.25}$$

where $\mathbf{x}_1 = \{\{t_k^{\mathrm{FRD}}, p_k^{\mathrm{FRD}}\}_{\substack{k=1 \\ k\neq\widetilde{k}}}^{K}, \{\{p_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}, t_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}, t_{n,k}^{\mathrm{LOC}}\}_{k=1}^{K}\}_{n=1}^{N^{\mathrm{LAY}}} \{t_n\}_{n=1}^{N^{\mathrm{LAY}}}$, $\{d_{1,k}^{\mathrm{row}}\}_{k=1}^{K}\}$ represents a set of variables involved in the optimization process, including time, data distribution and RF transmit power values.

3.5 Problem Solution

In this section, we address the solution to the optimization problem outlined in (3.25). Notably, this problem is non-convex due to the presence of communication energy expressions. For this purpose, we employ a convexification approach by introducing two new parameters: $E_k^{\mathrm{FRD}_{\mathrm{RF}}} = p_k^{\mathrm{FRD}} t_k^{\mathrm{FRD}}$ and $E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{\mathrm{a,b}\}}}} = t_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}} p_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}$. These parameters represent the RF energy consumed during the first round distribution and exchange communication, respectively. Accordingly, we modify the expressions in (3.6), (3.7), (3.11), (3.18), (3.19), and (3.21) as follows:

$$r_k^{\mathrm{FRD}} = B \ln(1 + \frac{(E_k^{\mathrm{FRD}_{\mathrm{RF}}} / t_k^{\mathrm{FRD}}) h_k^{\mathrm{FRD}}}{BN_0}) \tag{3.26}$$

$$E_k^{\mathrm{FRD}} = E_k^{\mathrm{FRD}_{\mathrm{RF}}} + t_k^{\mathrm{FRD}} P_{\widetilde{k}}^c \tag{3.27}$$

$$E_k^{\mathrm{FRD}_{\mathrm{RF}}} \leq t_k^{\mathrm{FRD}} p_{\widetilde{k}}^{\max} \tag{3.28}$$

$$r_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}} = B \ln(1 + \frac{(E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{\mathrm{a,b}\}}}} / t_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}) h_{n,k}^{\mathrm{EXC}_{\{\mathrm{a,b}\}}}}{BN_0}) \tag{3.29}$$

$$E_{n,k}^{\mathrm{EXC}_{\{a,b\}}} = E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}} + t_{n,k}^{\mathrm{EXC}_{\{a,b\}}} P_k^c \tag{3.30}$$

and

$$E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}} \leq t_{n,k}^{\mathrm{EXC}_{\{a,b\}}} p_k^{\max} \tag{3.31}$$

By introducing these new parameters, we convert the non-convex problem into a convex optimization problem, which can be more readily solved using optimization techniques. This modification allows us to optimize the energy consumption of collaborative CNN inference efficiently. The final convex optimization problem can be expressed as follows:

$$\begin{aligned}
\min_{\mathbf{x}_2} \quad & \sum_{\substack{k=1 \\ k\neq\tilde{k}}}^{K} E_k^{\mathrm{FRD}} + \sum_{k=1}^{K}\sum_{n=1}^{N^{\mathrm{LAY}}} (E_{n,k}^{\mathrm{LOC}} + E_{n,k}^{\mathrm{EXC}}) \\
\text{s.t.} \quad & (3.5), (3.10), (3.17), (3.20), (3.23), (3.24), (3.28), (3.31), \\
& 0 \leq E_k^{\mathrm{FRD}_{\mathrm{RF}}}, E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}, \\
& 0 \leq d_{1,k}^{\mathrm{row}} \leq d^{\mathrm{row}}, \\
& 0 \leq t_{n,k}^{\mathrm{LOC}}, t_{n,k}^{\mathrm{EXC}_{\{a,b\}}} \leq t_n, \\
& 0 \leq t_n, t_k^{\mathrm{FRD}} \leq \tau
\end{aligned} \tag{3.32}$$

where $\mathbf{x}_2 = \{\{t_k^{\mathrm{FRD}}, E_k^{\mathrm{FRD}_{\mathrm{RF}}}\}_{\substack{k=1 \\ k\neq\tilde{k}}}^{K}, \{\{E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}, t_{n,k}^{\mathrm{EXC}_{\{a,b\}}}, t_{n,k}^{\mathrm{LOC}}\}_{k=1}^{K}\}_{n=1}^{N^{\mathrm{LAY}}}$, $\{t_n\}_{n=1}^{N^{\mathrm{LAY}}}, \{d_{1,k}^{\mathrm{row}}\}_{k=1}^{K}\}$.

The problem can be easily solved via convex optimization techniques. However, to facilitate the solution of this problem and gain analytical insights into each optimizing parameter, a decomposition approach is employed. This approach, inspired by the primal-dual decomposition method [67], divides the problem into two distinct sub-problems: an external master primal problem and an internal sub-problem executed at each slave device for each layer-pack operation.

The external problem focuses on optimizing parameters like the time duration t_n for each layer-pack operation, the allocation of rows $d_{1,k}^{\mathrm{row}}$ to RCD k, and the communication parameters for the first round distribution, namely t_k^{FRD} and $E_k^{\mathrm{FRD}_{\mathrm{RF}}}$. On the other hand, the internal sub-problem is designed to optimize local computation and exchange communication parameters individually for each RCD. These parameters include $t_{n,k}^{\mathrm{LOC}}$, $t_{n,k}^{\mathrm{EXC}_{\{a,b\}}}$, and

$E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}$, given the values of t_n and $d_{1,k}^{\mathrm{row}}$.

The internal optimization problem is represented as

$$\begin{aligned}(E_{n,k}^{\mathrm{PL}}(t_n, d_{1,k}^{\mathrm{row}}))^* = \min_{\mathbf{x}_3} \quad & E_{n,k}^{\mathrm{LOC}} + E_{n,k}^{\mathrm{EXC}} \\ \text{s.t.} \quad & (3.17), (3.20), (3.23), (3.31), \\ & 0 \leq E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}, \\ & 0 \leq t_{n,k}^{\mathrm{LOC}}, t_{n,k}^{\mathrm{EXC}_{\{a,b\}}} \leq t_n\end{aligned} \tag{3.33}$$

where $\mathbf{x}_3 = \{t_{n,k}^{\mathrm{LOC}}, t_{n,k}^{\mathrm{EXC}_{\{a,b\}}}, E_{n,k}^{\mathrm{EXC}_{\mathrm{RF}_{\{a,b\}}}}\}$. The objective is to minimize the total energy consumption, which is the sum of local computation and exchange communication energy during layer-pack n operation for RCD k, subject to specific constraints related to energy, time, and communication parameters, given the time duration t_n and the input data distribution $d_{1,k}^{\mathrm{row}}$.

The external optimization problem is expressed as

$$\begin{aligned}\min_{\mathbf{x}_4} \quad & \sum_{\substack{k=1 \\ k \neq \tilde{k}}}^{K} E_k^{\mathrm{FRD}} + \sum_{k=1}^{K} \sum_{n=1}^{N^{\mathrm{LAY}}} (E_{n,k}^{\mathrm{PL}}(t_n, d_{1,k}^{\mathrm{row}}))^* \\ \text{s.t.} \quad & (3.5), (3.10), (3.24), (3.28), \\ & 0 \leq E_k^{\mathrm{FRD}_{\mathrm{RF}}}, \\ & 0 \leq t_n, t_k^{\mathrm{FRD}} \leq \tau, \\ & 0 \leq d_{1,k}^{\mathrm{row}} \leq d^{\mathrm{row}}\end{aligned} \tag{3.34}$$

where $\mathbf{x}_4 = \{\{t_n\}_{n=1}^{N^{\mathrm{LAY}}}, \{d_{1,k}^{\mathrm{row}}\}_{k=1}^{K}, \{t_k^{\mathrm{FRD}}, E_k^{\mathrm{FRD}_{\mathrm{RF}}}\}_{\substack{k=1 \\ k \neq \tilde{k}}}^{K}\}$. The goal is to minimize the overall energy consumption of all RCDs and layer-packs, considering the solutions obtained from the internal sub-problems.

By iteratively solving these sub-problems using the primal-dual decomposition approach, the overall problem can be effectively solved, as demonstrated in the corresponding section of the article [67]. The iterative convergence algorithm is outlined in detail in Algorithm 1.

3.5.1 Solution of Internal Optimization Problem

In this section, we present a comprehensive solution to the internal optimization problem introduced earlier. The detailed steps for solving this

Algorithm 1 Iterative convergence algorithm

1: Initialize the parameters $d_{1,k}^{\text{row}}$ and t_n.
2: Given the parameters $d_{1,k}^{\text{row}}$ and t_n, solve the convex internal optimization problem (3.33) for the parameters $t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{EXC}_{\{a,b\}}}, E_{n,k}^{\text{EXC}_{\text{RF}_{\{a,b\}}}}$.
3: By using $(E_{n,k}^{\text{PL}}(t_n, d_{1,k}^{\text{row}}))^*$, solve the convex external optimization problem (3.34) for the parameters $t_n, d_{1,k}^{\text{row}}, t_k^{\text{FRD}}, E_k^{\text{FRD}_{\text{RF}}}$.
4: By using the parameters $d_{1,k}^{\text{row}}$ and t_n found in item 3, repeat the process from item 2 until convergence is reached and the results do not change anymore.

problem can be found in Appendix A.

For the parameter $t_{n,k}^{\text{LOC}}$, the optimal solution is as follows:

$$(t_{n,k}^{\text{LOC}})^* = \begin{cases} \frac{m_{n,k}}{\nu_k^{\max}} & \beta_{n,k} \geq 2\kappa_k(\nu_k^{\max})^3 \\ m_{n,k}\left(\frac{2\kappa_k}{\beta_{n,k}}\right)^{1/3} & \frac{2\kappa_k(m_{n,k})^3}{(t_n)^3} < \beta_{n,k} < 2\kappa_k(\nu_k^{\max})^3 \\ t_n & \beta_{n,k} \leq \frac{2\kappa_k(m_{n,k})^3}{(t_n)^3} \end{cases} \tag{3.35}$$

This equation provides the optimal values of $t_{n,k}^{\text{LOC}}$ based on different conditions involving the Lagrange multiplier $\beta_{n,k}$ associated with (3.23) and other problem parameters.

For the parameters $p_{n,k}^{\text{EXC}_b}$ and $t_{n,k}^{\text{EXC}_b}$, the optimal solutions are given as follows:

$$(p_{n,k}^{\text{EXC}_b})^* = \begin{cases} 0 & \mu_{n,k}^{①} \leq \frac{N_0}{h_{n,k}^{\text{EXC}_b}} \\ B\left(\mu_{n,k}^{①} - \frac{N_0}{h_{n,k}^{\text{EXC}_b}}\right) & \frac{N_0}{h_{n,k}^{\text{EXC}_b}} < \mu_{n,k}^{①} < \frac{N_0}{h_{n,k}^{\text{EXC}_b}} + \frac{p_k^{\max}}{B} \\ p_k^{\max} & \frac{N_0}{h_{n,k}^{\text{EXC}_b}} + \frac{p_k^{\max}}{B} \leq \mu_{n,k}^{①} \end{cases} \tag{3.36}$$

and

$$(t_{n,k}^{\mathrm{EXC_b}})^* = \begin{cases} t_n & \mu_{n,k}^{①} \leq \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} e^{\frac{d_n^{\mathrm{col}}}{Bt_n}} \\ \frac{d_n^{\mathrm{col}}}{B \ln\left(\frac{h_{n,k}^{\mathrm{EXC_b}}}{N_0} \mu_{n,k}^{①}\right)} & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} e^{\frac{d_n^{\mathrm{col}}}{Bt_n}} < \mu_{n,k}^{①} < \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \\ \frac{d_n^{\mathrm{col}}}{B \ln\left(1+\frac{h_{n,k}^{\mathrm{EXC_b}}}{BN_0} p_k^{\max}\right)} & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \leq \mu_{n,k}^{①} \end{cases} \tag{3.37}$$

These equations provide the optimal values of $p_{n,k}^{\mathrm{EXC_b}}$ and $t_{n,k}^{\mathrm{EXC_b}}$ based on different conditions involving the Lagrange multiplier $\mu_{n,k}^{①}$ associated with (3.20). The variable $m_{n,k} = 9c_k d_n^{\mathrm{col}} d_{1,k}^{\mathrm{row}} / 2^{n-1}$ aggregates multiple parameters into a single term for simplified notation.

3.5.2 Solution of External Optimization Problem

For the parameter t_n, the optimal solution is

$$(t_n)^* = \begin{cases} 0 & \phi > \sum_{k=1}^{K} \beta_{n,k} \\ (0, \tau) & \phi = \sum_{k=1}^{K} \beta_{n,k} \\ \tau & \phi < \sum_{k=1}^{K} \beta_{n,k} \end{cases} \tag{3.38}$$

This equation provides the optimal values of t_n based on different conditions involving the Lagrange multiplier ϕ associated with (3.24) and other problem parameters.

For the parameter $d_{1,k}^{\mathrm{row}}$, the optimal solution is

$$(d_{1,k}^{\mathrm{row}})^* = \begin{cases} 0 & \psi_k \leq 0 \\ \sqrt{\frac{\psi_k}{3\left(\sum_{n=1}^{N^{\mathrm{LAY}}} cons_{n,k}^{①}\right)}} & 0 < \psi_k < 3\left(\sum_{n=1}^{N^{\mathrm{LAY}}} cons_{n,k}^{①}\right)(d^{\mathrm{row}})^2 \\ d^{\mathrm{row}} & \psi_k \geq 3\left(\sum_{n=1}^{N^{\mathrm{LAY}}} cons_{n,k}^{①}\right)(d^{\mathrm{row}})^2 \end{cases} \tag{3.39}$$

where

$$\psi_k = \begin{cases} \lambda - \sum_{n=1}^{N^{\mathrm{LAY}}} \gamma_{n,k}^{①} cons_{n,k}^{②} & k = \widetilde{k} \\ \lambda - \sum_{n=1}^{N^{\mathrm{LAY}}} \gamma_{n,k}^{①} cons_{n,k}^{②} - \theta_k d^{\mathrm{col}} & k \neq \widetilde{k} \end{cases} \tag{3.40}$$

$$cons_{n,k}^{①} = \frac{3^6 \kappa_k c_k^3 (d_n^{\mathrm{col}})^3}{(2^{n-1})^3 ((t_{n,k}^{\mathrm{LOC}})^*)^2} \tag{3.41}$$

$$cons_{n,k}^{②} = \frac{9 c_k d_n^{\mathrm{col}}}{2^{n-1} \nu_k^{\max}} \tag{3.42}$$

Here, θ_k and $\gamma_{n,k}^{①}$ are the Lagrange multipliers associated with (3.10) and (3.17), respectively.

For the parameters p_k^{FRD} and t_k^{FRD}, the optimal solutions are given as follows:

$$(p_k^{\mathrm{FRD}})^* = \begin{cases} 0 & \theta_k \leq \frac{N_0}{h_k^{\mathrm{FRD}}} \\ B\left(\theta_k - \frac{N_0}{h_k^{\mathrm{FRD}}}\right) & \frac{N_0}{h_k^{\mathrm{FRD}}} < \theta_k < \frac{N_0}{h_k^{\mathrm{FRD}}} + \frac{p_{\widetilde{k}}^{\max}}{B} \\ p_{\widetilde{k}}^{\max} & \frac{N_0}{h_k^{\mathrm{FRD}}} + \frac{p_{\widetilde{k}}^{\max}}{B} \leq \theta_k \end{cases} \tag{3.43}$$

and

$$(t_k^{\mathrm{FRD}})^* = \begin{cases} 0 & \theta_k < cons_k^{③} \\ (0, \tau) & \theta_k = cons_k^{③} \\ \tau & \theta_k > cons_k^{③} \end{cases} \tag{3.44}$$

where

$$cons_k^{③} = \frac{P_{\widetilde{k}}^c + \phi - \xi_k^{②} p_{\widetilde{k}}^{\max}}{\left(r_k^{\mathrm{FRD}} \left((p_k^{\mathrm{FRD}})^* \right) - \frac{\frac{h_k^{\mathrm{FRD}}}{N_0} (p_k^{\mathrm{FRD}})^*}{1 + \frac{h_k^{\mathrm{FRD}}}{BN_0} (p_k^{\mathrm{FRD}})^*} \right)} \tag{3.45}$$

$$\xi_k^{②} = \begin{cases} 0 & (p_k^{\mathrm{FRD}})^* < p_{\widetilde{k}}^{\max} \\ \theta_k \frac{h_k^{\mathrm{FRD}}}{N_0 + \frac{h_k^{\mathrm{FRD}}}{B} p_{\widetilde{k}}^{\max}} - 1 & (p_k^{\mathrm{FRD}})^* = p_{\widetilde{k}}^{\max} \end{cases} \tag{3.46}$$

The complete proofs and derivations of these solutions can be found in Appendix B.

3.6 Simulation Results

In this section, we present the simulation results of this study, which aims to evaluate the performance of the optimal analytical solutions derived for the iterative two-stage primal-dual decomposed problem. One of our goals is to demonstrate that the results obtained using the decomposed approach converge to those achieved through MATLAB CVX optimization of the overall problem described in (3.32). Additionally, we compare the proposed algorithm with a scenario denoted as *NoOptDist*, where the distribution of $d_{1,k}^{\text{row}}$ among participating RCDs is not optimized. In the *NoOptDist* scenario, $d_{1,k}^{\text{row}}$ is distributed equally among the participating RCDs ($d_{1,k}^{\text{row}} = d^{\text{row}}/K$). While this scenario does not optimize the distribution of $d_{1,k}^{\text{row}}$, it still optimizes communication and computation parameters, as well as the time t_n allocated for each layer-pack operation, making it still a strong benchmark scenario to compare.

We have selected the simulation parameters based on prior research articles [13, 20, 24, 74]. The specific parameter values employed are summarized in Table 3.1. The simulation results are derived from averaging outcomes across 1000 distinct random channel realizations. The obtained simulation results are dependent on various random performance parameters, including c_k (uniformly distributed between 250 and 750 CPU cycles/MAC) and $\nu_k^{\max}$ (uniformly distributed between 2 and 4 GHz). It is worth mentioning that, despite testing with different random scenarios, no significant differences have been observed in the general trend of all simulation results. Thus, the proposed approach can be used in different scenarios, involving different types of devices with different specifications. This robustness indicates that the proposed approach is versatile and can be effectively applied in various scenarios, accommodating different types of devices with diverse specifications.

Throughout the simulation results of this study, we focus on the minimized collaborative computing energy consumption E^{MAC} per MAC operation of the CNN model, which encompasses the first round distribution, local computation, and exchange communication of all layer-pack operations. Mathematically, it is expressed as $E_{\text{total}}/\sum_{k=1}^{K}\sum_{n=1}^{N^{\text{LAY}}} N_{n,k}^{\text{MAC}}$ where E_{total} is the overall energy consumption of the system, i.e., the solution of the external optimization problem (3.34). We investigate three different scenarios:

- *Opt − Decomp*: This scenario leverages the analytical results obtained by decomposing the overall optimization problem and represents our proposed collaborative computing approach.

TABLE 3.1: Selected parameters for the simulation results of Chapter 3

Parameters	Values	Units
κ_k	Unif($[10^{-28}; 10^{-27}]$)	/
c_k	Unif($[250; 750]$)	[CPU cycles/MAC]
$\nu_k^{\max}$	Unif($[2; 4]$)	[GHz]
h_k	$\mathcal{CN}(0, 10^{-3})$(Rayleigh)	/
$p_k^{\max}$	Unif($[10; 25]$)	[mW]
P_k^c	Unif($[10; 25]$)	[mW]
B	1	[MHz]
N_0	10^{-16}	[W/Hz]

- $Opt - Overall$: In this case, we solve the overall optimization problem without decomposition using MATLAB CVX toolbox.
- $NoOptDist$: This scenario assumes an equal distribution of $d_{1,k}^{\text{row}}$ among participating RCDs ($d_{1,k}^{\text{row}} = d^{\text{row}}/K$) but optimizes communication and computation parameters as well as t_n for each layer-pack operation.

Across all simulation scenarios, a consistent finding is that optimizing the distribution of input data among heterogeneous RCDs leads to a significant improvement in energy efficiency, resulting in a substantial reduction in energy consumption per MAC operation. Given that CNN models often involve millions of MAC operations, even small differences in E^{MAC} are highly significant when scaled to the entire model's energy consumption.

Furthermore, the results obtained from the decomposed approach ($Opt - Decomp$) converge with those achieved using the overall optimization problem ($Opt - Overall$). This convergence demonstrates that we can analytically determine the minimized total energy consumption without relying on external optimization tools. This analytical approach offers greater flexibility in tailoring energy and latency requirements to specific applications, making it a valuable contribution to collaborative computing optimization.

In Fig. 3.4, we illustrate the optimal time allocations (t_n) for layer-pack operations while varying the total number of rows (d^{row}) in a 2D input data sample, with fixed parameters ($N^{\text{LAY}} = 4$, $d^{\text{col}} = 512$ bits, $K = 10$, and $\tau = 0.5$ s). Notably, as data progresses from the front-end to the back-end layers, the allocated time exponentially decreases due to the decreasing data dimensions in the back-end compared to the front-end layer-packs. Additionally, when increasing the total number of rows (d^{row}), the system

allocates less time for collaborative computing, primarily because larger input data sizes demand more time for the first round distribution stage, consequently reducing the time spent for collaborative computation. These insights shed light on optimizing time allocation in scenarios with varying input data sizes and diverse computational requirements across layers, providing valuable guidance for improving collaborative computing efficiency.

In Fig. 3.5, we explore the optimal distribution $d_{1,k}^{\text{row}}$ among RCDs for various values of d^{row}, by considering the same setup with Fig. 3.4 ($N^{\text{LAY}} = 4$, $d^{\text{col}} = 512$ bits, $K = 10$, and $\tau = 0.5$ s). Each distinct color represents the allocation to a specific RCD. Notably, the optimal distribution does not converge to an equal-sharing scenario due to the heterogeneity in computing capabilities among RCDs. Instead, it assigns a greater portion of the input data with a higher number of rows to the RCDs with higher computing capabilities. This optimized distribution strategy significantly enhances system efficiency compared to the equal distribution approach (*NoOptDist*), as evident in subsequent figures. This analysis underscores the importance of tailored data distribution in heterogeneous computing environments to improve overall system performance.

In Fig. 3.6, we investigate the impact of varying the maximum latency τ allowed for the entire operation, considering scenarios with different data sizes ($d^{\text{col}} = 256, 512$ bits) while keeping $d^{\text{row}} = 1024$ bits, $K = 20$, and $N^{\text{LAY}} = 3$. For the case where $d^{\text{col}} = 512$ bits, we observe that the energy consumption E^{MAC} in the *NoOptDist* scenario is approximately 50% higher than that in the proposed optimal distribution, particularly for small τ values. However, as τ increases, these energy consumption curves approach each other, converging towards similar values. This convergence occurs because both scenarios have sufficient latency to achieve very low energy consumption for large values of τ. On the other hand, for the scenario with $d^{\text{col}} = 256$ bits, the impact of the optimal distribution on energy consumption is less pronounced. This is primarily due to the reduced data size, which reduces the overall computation and communication requirements. As a result, the energy consumption difference between the proposed approach and the equal data distribution scenario is smaller. This figure underscores the significance of joint optimization of input data distribution, communication, and computation parameters, particularly in latency-critical applications where minimizing latency leads to a substantial energy consumption difference between the proposed approach and equal data distribution scenario.

In Fig. 3.7, we evaluate the system's performance with varying numbers of columns d^{col} in the input data while considering two scenarios, one with

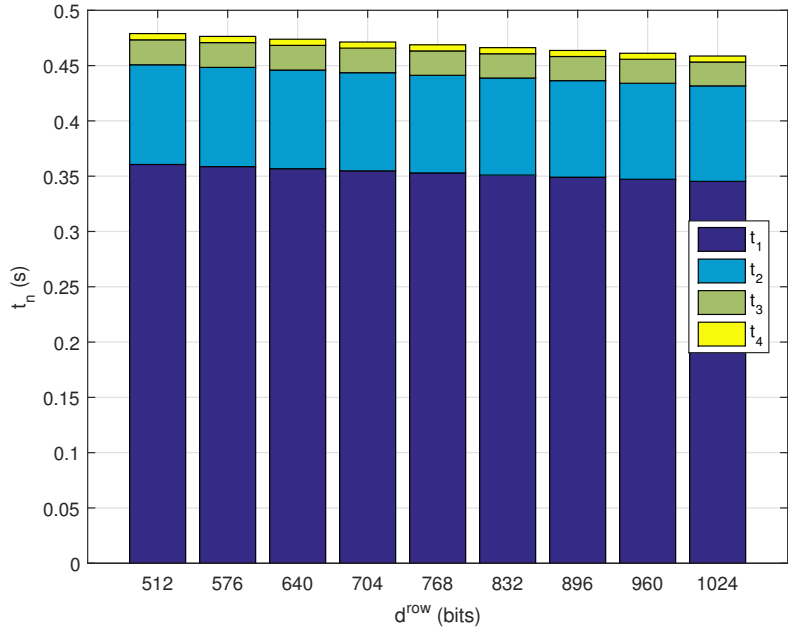


FIGURE 3.4: t_n vs. d^{row} for $N^{\text{LAY}} = 4$, $d^{\text{col}} = 512$ bits, $K = 10$ and $\tau = 0.5$ s

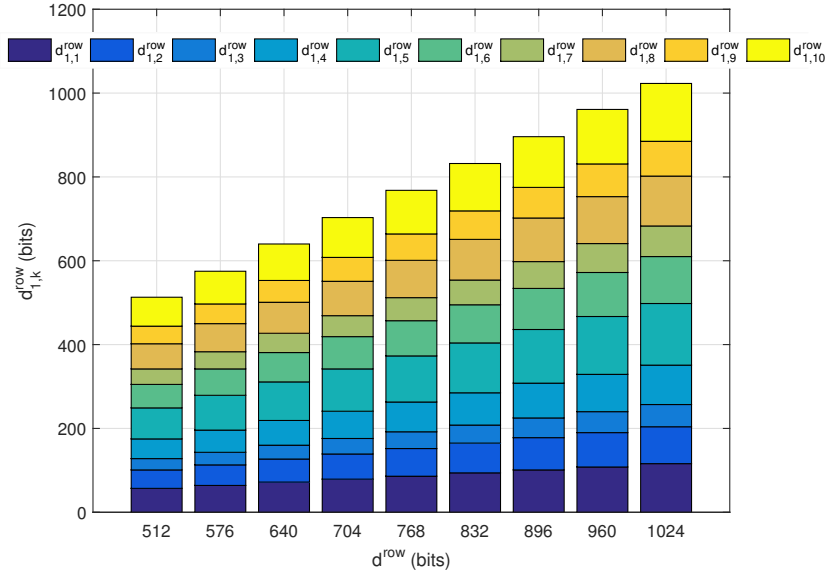


FIGURE 3.5: $d^{\text{row}}_{1,k}$ vs. d^{row} for $N^{\text{LAY}} = 4$, $d^{\text{col}} = 512$ bits, $K = 10$ and $\tau = 0.5$ s

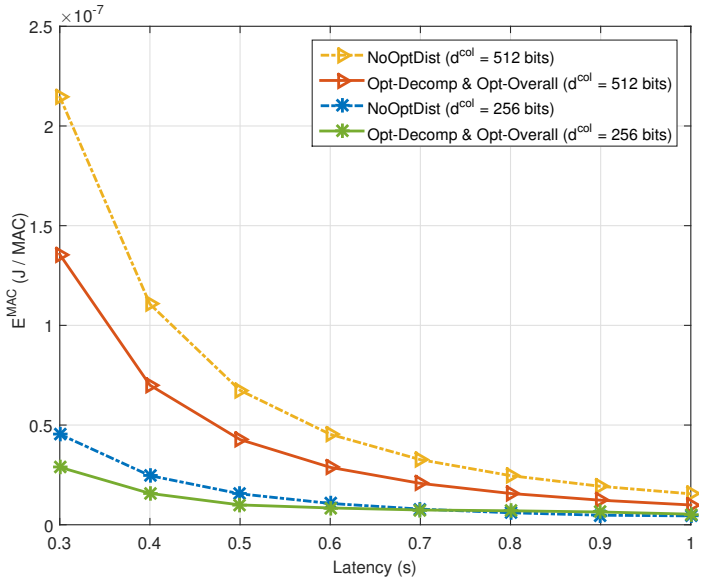


FIGURE 3.6: E^{MAC} vs. latency (τ) for $d^{\text{row}} = 1024$ bits, $K = 20$ and $N^{\text{LAY}} = 3$

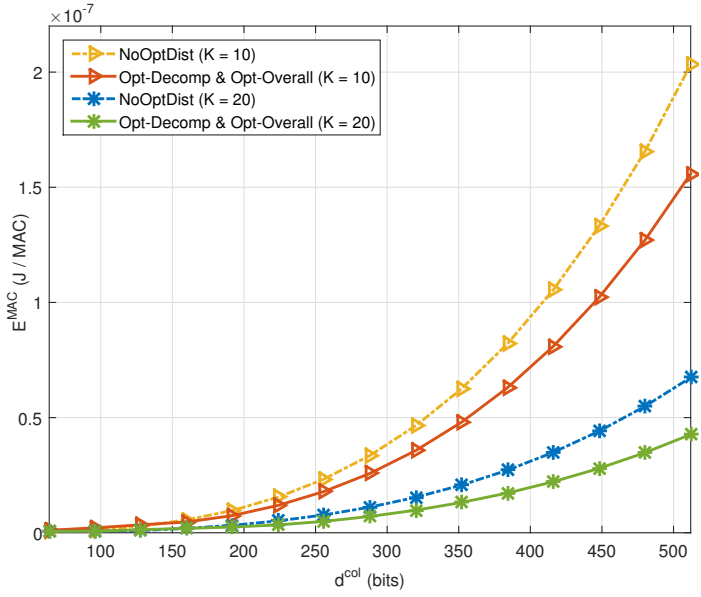


FIGURE 3.7: E^{MAC} vs. d^{col} for $d^{\text{row}} = 1024$ bits, $N^{\text{LAY}} = 3$ and $\tau = 0.5$ s

$K = 10$ and the other with $K = 20$ participating RCDs. As the number of columns d^{col} in the input data increases, the energy consumption difference between the jointly optimized scenario and the equal data distribution scenario (*NoOptDist*) becomes more pronounced. This difference is especially significant when the data size is large, emphasizing the critical role of optimal input data distribution in heterogeneous computing environments. Furthermore, increasing the number of participating RCDs (K) leads to reduced energy consumption. Notably, the curve for $K = 20$ exhibits nearly three times lower energy consumption per MAC operation compared to the curve for $K = 10$. This demonstrates the advantages of involving more RCDs in the collaborative process, as it enables better utilization of heterogeneous computing resources and contributes to energy efficiency.

In Fig. 3.8, we analyze the impact of varying the number of participating RCDs (K) on the energy consumption (E^{MAC}). As the number of participating RCDs increases, we observe a significant reduction in energy consumption, approaching near-zero levels. This phenomenon occurs because, with a greater number of RCDs in the system, the influence of less energy-efficient RCDs becomes less significant in the overall system performance. These less efficient RCDs handle a smaller portion of the data due to the increasing number of RCDs, resulting in a reduced impact on energy consumption. It is important to note that the effectiveness of adding more RCDs to the system decreases after a certain point in the plots. For instance, there is a substantial drop in energy consumption when increasing from $K = 4$ to $K = 6$, but the reduction in energy consumption becomes less pronounced when increasing from $K = 18$ to $K = 20$. This observation highlights the diminishing returns of adding more RCDs beyond a specific threshold. This effect occurs because adding more RCDs after a certain point leads to a substantial increase in communication costs compared to the computational cost savings, creating a trade-off between communication and computation.

In Fig. 3.9, we examine the impact of the number of rows (d^{row}) in the input data on the energy consumption for two different scenarios, one with $N^{\text{LAY}} = 2$ and the other with $N^{\text{LAY}} = 4$. The energy consumption difference between the optimized and non-optimized *NoOptDist* scenarios is evident. For example, with $d^{\text{row}} = 1024$ bits and $N^{\text{LAY}} = 4$, the optimized scenario is almost 33% more energy-efficient than the *NoOptDist* scenario. This result highlights the substantial energy savings that can be achieved through the optimized input data distribution, especially when dealing with larger input data sizes and a larger number of layers.

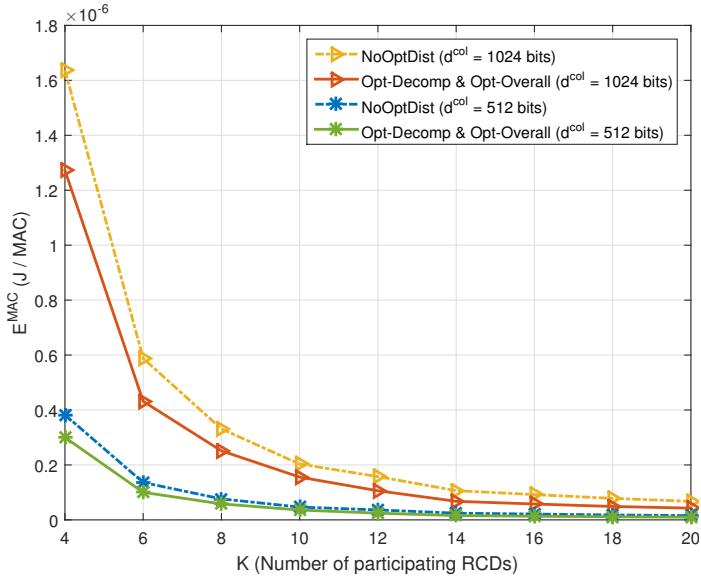


FIGURE 3.8: E^{MAC} vs. K for $d^{\text{row}} = 1024$ bits, $N^{\text{LAY}} = 3$ and $\tau = 0.5$ s

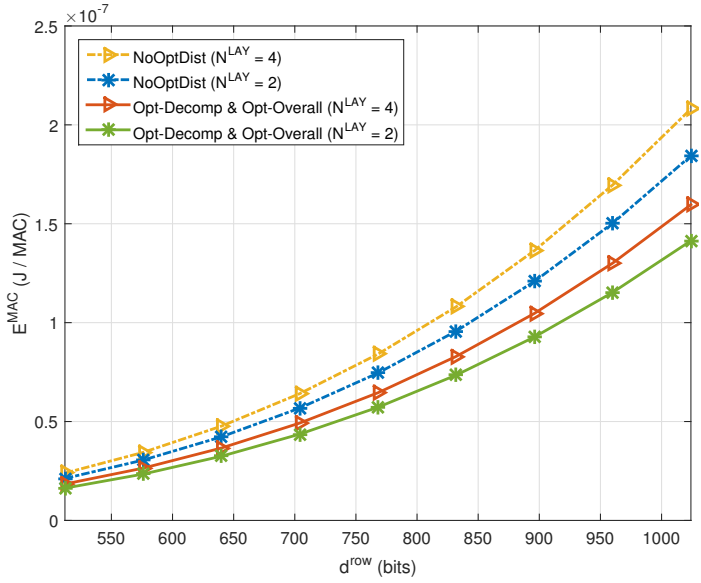


FIGURE 3.9: E^{MAC} vs. d^{row} for $d^{\text{col}} = 512$ bits, $K = 10$ and $\tau = 0.5$ s

3.7 Summary

In summary, Chapter 3 presents an energy-efficient fine-grained data parallelism approach designed for wireless collaborative fog computing systems, with a particular focus on deep learning-driven applications, using CNNs as a primary model. This approach optimally distributes the computational workload among multiple RCDs by jointly optimizing communication and computation parameters together with workload distribution. The key contributions of this chapter include a significant reduction in communication overhead, efficient resource utilization within a server-independent architecture, and the utilization of a robust convex optimization framework. These contributions are prominent in addressing the challenges of deep learning in the context of collaborative fog computing.

The findings of this study reveal crucial insights into enhancing energy efficiency within the system. Joint optimization of workload distribution, along with communication and computation parameters, proves to be essential for improved energy efficiency compared to scenarios where optimization is focused solely on communication and computation parameters, while workload is evenly distributed among all participating RCDs. Additionally, the benefits of the joint optimization show up more for the scenarios with large input data sizes, low allowed latency, and a smaller number of participating devices.

In Table 1.1, one of the cons highlighted for the data parallelism approach was the inefficient point-to-point transmission of partial input data. This chapter addresses this issue by introducing a communication scheme. The proposed scheme streamlines the process, requiring the exchange of only the boundary rows among neighboring devices after each layer operation.

Despite this improvement, the data parallelism approach still faces challenges, particularly in the execution of CNN models with multiple filters in each convolutional layer. While the data parallelism approach shines in scenarios with large input data sizes and low CNN model complexity, it faces difficulties as model complexity increases. In the forthcoming chapter, we explore a model parallelism approach designed to facilitate collaborative CNN inference in scenarios characterized by high model complexity but lower input data sizes. This approach seeks to address the limitations associated with managing multiple filters in convolutional layers.

Chapter 4

A Collaborative On-Device CNN Inference Scheme Considering Model Parallelism

The previous chapter addressed data parallelism by optimizing the distribution of input data among multiple RCDs. While it provides an excellent foundation, CNNs typically involve convolutional layers with multiple filters for extracting various features from input data. Consequently, in addition to data parallelism, a model parallelism scenario becomes necessary.

In this chapter, we shift our focus from data parallelism, as discussed in Chapter 3, to model parallelism in the context of collaborative on-device CNN inference. Specifically, we aim to efficiently distribute the computational workload across RCDs by partitioning the model's filters among them. This approach allows us to harness the capabilities of RCDs collaboratively, resulting in enhanced efficiency and performance.

In contrast to the previous chapter, where data partitioning was the primary focus, in this chapter, we divide the model due to specific scenarios with relatively small input data dimensions, such as the MNIST dataset [77], where the input size is 28×28 bits. In such cases, dividing the data might not be optimal, and instead, dividing the model itself provides advantages. This approach leverages broadcasting to transmit the entire input data from the data owner master device to all participating RCDs, saving energy by avoiding additional communication stages for each RCD to send the input data.

For each layer operation, the data owner master device broadcasts the entire input data and assigns filters to the slave devices based on their computing capabilities and network conditions. After local computation, slave devices transmit their filter outputs back to the master device, which aggregates these outputs to form the new input data for subsequent layers.

It is worth noting that the CNN model considered throughout this study is the one discussed in Chapter 2. Each convolutional layer is composed of multiple filters, and these filters are allocated to multiple RCDs for parallel computation, considering their heterogeneous computing capabilities and different network conditions.

To tackle the optimization problem associated with this approach, we formulate a convex optimization problem aimed at minimizing overall energy consumption. This optimization includes communication and computation times, RF energy consumption, and the allocation of filters to multiple RCDs. Similar to Chapter 3, we employ a primal-dual decomposition technique [67] and an iterative algorithm to find analytical solutions, effectively distributing filter computations among the RCDs to improve overall efficiency.

Our key contributions in this chapter can be summarized as follows:

- **Collaborative Model Parallelism:** We introduce a collaborative model parallelism approach for on-device CNN inference, jointly optimizing filter assignments and communication and computation parameters to minimize energy consumption.
- **Server-Free Inference:** Similar to Chapter 3, our scheme eliminates the need for powerful edge or cloud servers during the inference phase, making it suitable for resource-constrained environments.
- **Output Aggregation Method:** We investigate an output aggregation method after each layer operation to reduce the communication overhead by making use of the fundamental mathematical properties of the convolution operation.
- **Convex Optimization and Iterative Algorithm:** We introduce a convex optimization problem and an iterative algorithm that decomposes the problem, distributing filter computations among multiple RCDs, thus improving overall efficiency.

In subsequent sections, we go into the details of our proposed collaborative inference scheme, its optimization strategies, and the analytical insights gained through this study.

4.1 System Model

In this section, we present the system model, which consists of K RCDs and an access point connected to a powerful cloud server, as illustrated in Fig. 4.1. The system operates through the following sequential steps:

1. **Cloud-Based CNN Training:** The CNN model is trained entirely on powerful cloud servers using offline datasets, similar to the approach outlined in Chapter 3. This approach ensures that the training process can utilize maximum computational power and is not subject to latency constraints, thereby preventing any loss in accuracy. As a result, this study primarily focuses on the inference phase rather than the training phase.
2. **Model Offloading:** After the training phase, the optimized model weights, denoted as w, are distributed to all RCDs. This ensures that every RCD possesses an identical pre-trained CNN model.
3. **Master-Slave RCD Architecture:** A random RCD k receives local input data to process in the CNN model. It behaves as a master RCD while the others become slaves. It is assumed that each RCD is allocated to a predefined time frame during which it acts as the master RCD in order to prevent any collision between multiple RCDs that request to be the master device of the system at the same time. The master RCD is responsible for coordinating the execution of the CNN model, specifically:
   (a) Determining the optimal number of participating RCDs, represented as K_n (including itself), for layer-pack n operation.
   (b) Optimizing the allocation of filters to each participating RCD for each layer-pack operation.
   (c) Allocating a total time, t_n, to each RCD to complete each layer-pack n operation.
4. **Data Broadcasting:** The master RCD broadcasts the entire input data to all slave RCDs before initiating each layer-pack operation.
5. **Collaborative Local Computation:** Within the allocated time, t_n, for layer-pack n operation, all slave RCDs receive the input data, perform their local computation and transmit the resulting output data back to the master RCD. The master RCD, having already the input data, can dedicate the entire time, t_n, to local computation.
6. **Data Aggregation:** After receiving the output data from all RCDs participating in the layer-pack n operation, the master RCD combines

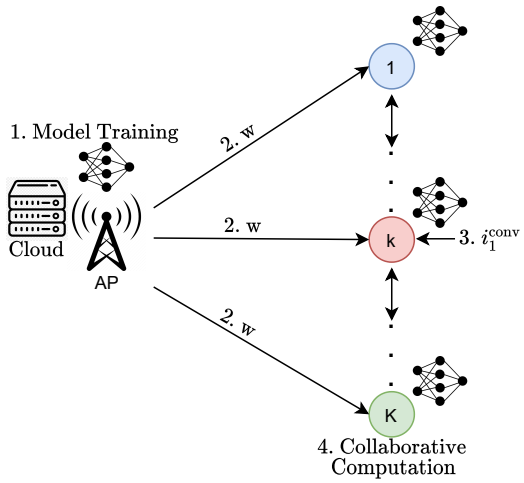


FIGURE 4.1: Illustration of the system model for the study in Chapter 4

it to generate the new input data and broadcasts it to the slave RCDs for the next layer-pack operation.

7. **Iterative Process:** This iterative process continues until the final layer-pack operation is completed.

Fig. 4.2 depicts a typical scenario for the distribution of layer-pack n operation within the collaborative model parallelism approach. Each blue square symbolizes a filter within each layer. Specifically, $f_{n,k,m}^{\text{conv}}$ represents the m^{th} assigned convolutional filter of RCD k for layer-pack n operation, and $d_{n,k}^{\text{ch}}$ indicates the number of filters allocated to RCD k for this operation.

At the beginning of layer-pack n operation, all slave RCDs receive the input data o_{n-1} from the master RCD. Subsequently, all RCDs independently execute their local computations using the filters assigned to them. The max-pooling outputs from all filters allocated to RCD k are then internally summed to generate the output data of RCD k for layer-pack n operation, represented as

$$o_{n,k} = \sum_{m=1}^{d_{n,k}^{\text{ch}}} MP(o_{n-1} * f_{n,k,m}^{\text{conv}}) \tag{4.1}$$

where $o_n = \sum_{k=1}^{K_n} o_{n,k}$, $\forall k \in [K_n]$, $\forall n \in [N^{\text{LAY}}]$. The initial value, o_0, corresponds to i_1^{conv}, which represents the input data at the beginning of the CNN model.

Importantly, RCDs are not required to transmit the entire 3D output data, which encompasses the outputs of all filters. Instead, they only transmit the 2D data $o_{n,k}$ to the master RCD. This 2D data is the summation of all pages in the 3D output data, leading to a substantial reduction in communication overhead within the system. We call this approach as output aggregation. Following the reception of each output data $o_{n,k}$, the master RCD aggregates them to generate the final output data o_n for layer-pack n operation. This final output data is then forwarded to the K_{n+1} participating RCDs for layer-pack $n+1$ operation.

4.2 Energy Consumption Modeling

The energy consumption of the system is segmented into three distinct stages: broadcasting, collaborative local computation, and aggregation.

4.2.1 Broadcasting (BC)

During this stage, the master RCD broadcasts the entire input data for layer-pack n operation to the slave RCDs at the achievable data rate

$$r_n^{\text{BC}} = B \ \ln(1 + \frac{p_n^{\text{BC}} \min_k \left(\frac{h_{n,k}^{\text{BC}}}{l(R_{n,k}^{\text{BC}})} \right)}{BN_0}) \tag{4.2}$$

where p_n^{BC} represents the RF transmit power of the master RCD $\widetilde{k}$ during broadcasting, B denotes the communication bandwidth, N_0 signifies the noise power density, $h_{n,k}^{\text{BC}}$ characterizes the channel gain between the master RCD $\widetilde{k}$ and RCD k, and $R_{n,k}^{\text{BC}}$ specifies the distance between them during broadcasting. The path-loss model, described by $l(R) = c_{\text{PL}} R^{\alpha}$, incorporates parameters such as α as the path-loss exponent, c_{PL} as the path-loss constant, f_c as the carrier frequency, and c_0 as the speed of light.

The energy consumption of broadcasting for layer-pack n operation is calculated as

$$E_n^{\text{BC}} = t_n^{\text{BC}} (p_n^{\text{BC}} + P_{\widetilde{k}}^c) \tag{4.3}$$

In this equation, P_k^c represents the constant power consumption of the communication circuitry in RCD k during communication, and t_n^{BC} signifies the

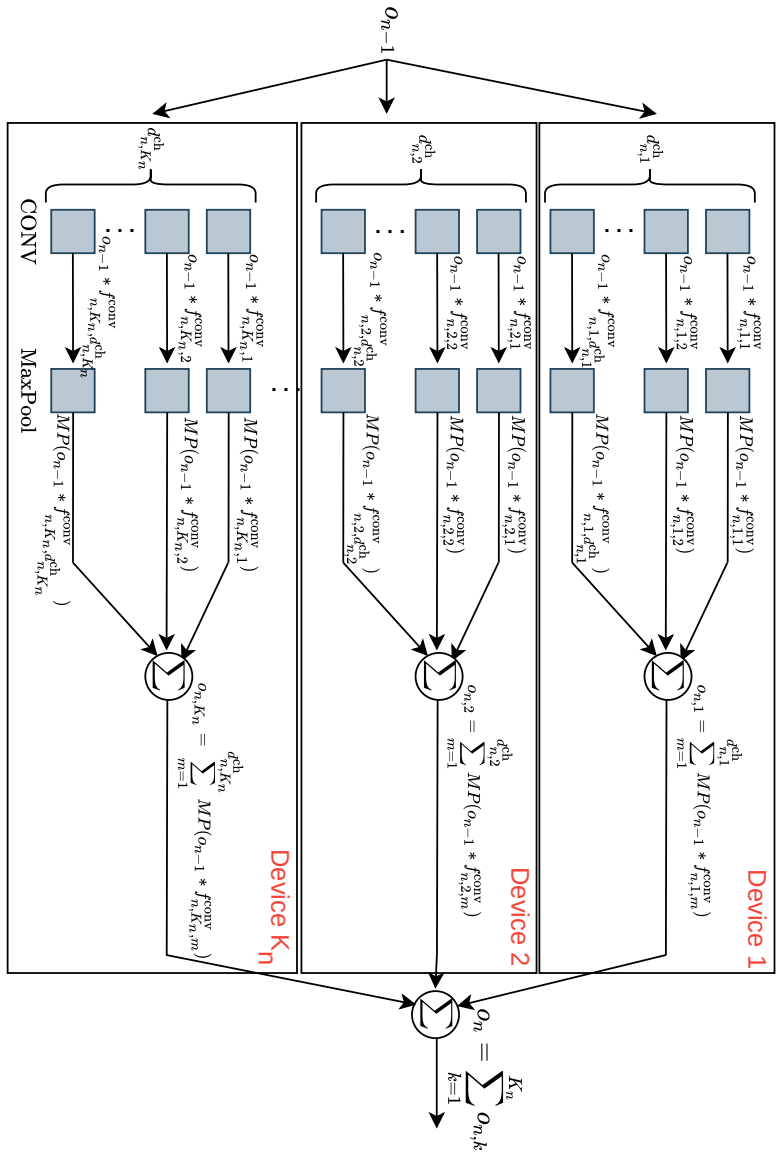


FIGURE 4.2: Collaborative layer-pack n operation diagram of the study in Chapter 4

time required to broadcast the input data for layer-pack n operation. The following two constraints are imposed regarding this stage:

- **Constraint 1 (Transmission Rate Limitation):** The data rate of the communication during broadcasting cannot exceed the achievable data rate such that

$$d_n^{\mathrm{BC}} = \frac{d^{\mathrm{row}}}{2^{n-1}} \frac{d^{\mathrm{col}}}{2^{n-1}} = 4^{1-n} d^{\mathrm{row}} d^{\mathrm{col}} \leq t_n^{\mathrm{BC}} r_n^{\mathrm{BC}} \tag{4.4}$$

  where d_n^{BC} is the number of bits of the input data with dimensions $\frac{d^{\mathrm{row}}}{2^{n-1}} \times \frac{d^{\mathrm{col}}}{2^{n-1}}$ for layer-pack n operation.

- **Constraint 2 (RF Transmit Power Limitation):** The RF transmit power of the master RCD during broadcasting cannot be greater than the maximum RF transmit power such that

$$p_n^{\mathrm{BC}} \leq p_k^{\max} \tag{4.5}$$

where $p_k^{\max}$ represents the maximum RF transmit power that RCD k can attain.

4.2.2 Collaborative Local Computation

Upon receiving the input data, each RCD initiates its local computation process. Given that the energy consumption of convolutional layers dominates over max-pooling layers [36,71], the primary focus in this study centers around the energy consumption of convolutional layers. Specifically, as in the study in Chapter 3, we examine MAC operations within convolutional layers, where a single MAC operation involves multiplying an element from the filter with an element from the shaded input data and accumulating the result to the previous sum.

The number of MAC operations, denoted as $N_{n,k}^{\mathrm{MAC}}$, that RCD k must perform for layer-pack n operation is determined by the following equation:

$$N_{n,k}^{\mathrm{MAC}} = \sum_{m=1}^{d_{n,k}^{\mathrm{ch}}} \frac{d^{\mathrm{row}}}{2^{n-1}} \frac{d^{\mathrm{col}}}{2^{n-1}} (d_{f_{n,k,m}^{\mathrm{conv}}})^2 = 36 \frac{d^{\mathrm{row}} d^{\mathrm{col}}}{4^n} d_{n,k}^{\mathrm{ch}} \tag{4.6}$$

where $d_{f_{n,k,m}^{\mathrm{conv}}} = 3$ is the row and column numbers of convolutional layer filters. The energy consumption of local computation in RCD k for layer-pack n operation, under the assumption of low CPU voltage [73–76], is

given by

$$E_{n,k}^{\mathrm{LOC}} = \frac{\kappa_k c_k^3 (N_{n,k}^{\mathrm{MAC}})^3}{(t_{n,k}^{\mathrm{LOC}})^2} = \frac{36^3 (d^{\mathrm{row}})^3 (d^{\mathrm{col}})^3 \kappa_k c_k^3}{64^n} \frac{(d_{n,k}^{\mathrm{ch}})^3}{(t_{n,k}^{\mathrm{LOC}})^2} \tag{4.7}$$

where κ_k denotes the effective capacitance coefficient of the hardware, c_k represents the number of CPU cycles required for a single MAC operation, and $t_{n,k}^{\mathrm{LOC}}$ is the time needed to complete the local computation for layer-pack n operation in RCD k. The following constraint is established for the local computation stage:

$$c_k N_{n,k}^{\mathrm{MAC}} = 36 c_k \frac{d^{\mathrm{row}} d^{\mathrm{col}}}{4^n} d_{n,k}^{\mathrm{ch}} \leq t_{n,k}^{\mathrm{LOC}} \nu_k^{\max} \tag{4.8}$$

which ensures that the CPU frequencies of RCDs do not exceed their maximum CPU frequency $\nu_k^{\max}$.

4.2.3 Aggregation

After the completion of local computation, each RCD transmits the output data $o_{n,k}$ to the master RCD. This transmission occurs at the achievable data rate defined as

$$r_{n,k}^{\mathrm{Agg}} = B \ \ln(1 + \frac{p_{n,k}^{\mathrm{Agg}} \frac{h_{n,k}^{\mathrm{Agg}}}{l(R_{n,k}^{\mathrm{Agg}})}}{BN_0}) \tag{4.9}$$

where $p_{n,k}^{\mathrm{Agg}}$ is the RF transmit power of RCD k, $h_{n,k}^{\mathrm{Agg}}$ and $R_{n,k}^{\mathrm{Agg}}$ are the channel gain and the distance between the master RCD and RCD k during aggregation, respectively.

The energy consumption associated with the aggregation in RCD k after layer-pack n operation is computed as

$$E_{n,k}^{\mathrm{Agg}} = t_{n,k}^{\mathrm{Agg}} \left(p_{n,k}^{\mathrm{Agg}} + P_k^c \right) \tag{4.10}$$

where $t_{n,k}^{\mathrm{Agg}}$ represents the time required to transmit the output data to the master RCD by RCD k for layer-pack n operation. We apply constraints to this stage similar to those in Section 4.2.1, ensuring that the data rate of the communication during aggregation does not exceed the achievable data rate and that the RF transmit power of RCDs during aggregation does not exceed their maximum RF transmit power.

- **Constraint 1 (Transmission Rate Limitation):**

$$d_n^{\text{Agg}} = \frac{d^{\text{row}}}{2^n}\frac{d^{\text{col}}}{2^n} = 4^{-n} d^{\text{row}} d^{\text{col}} \leq t_{n,k}^{\text{Agg}} r_{n,k}^{\text{Agg}} \tag{4.11}$$

- **Constraint 2 (RF Transmit Power Limitation):**

$$p_{n,k}^{\text{Agg}} \leq p_k^{\max} \tag{4.12}$$

where d_n^{Agg} is the number of bits sent to the master RCD from each slave RCD after layer-pack n operation.

4.3 Additional Constraints

We have several additional constraints to ensure the proper functioning of our model:

- **Constraint on the Number of Filters Assigned:**

$$\sum_{k=1}^{K_n} d_{n,k}^{\text{ch}} = d_n^{\text{ch}} \tag{4.13}$$

  This constraint ensures that the total number of filters assigned to all RCDs in layer-pack n operation is equal to the number of filters d_n^{ch} in convolutional layer n.

- **Constraint on the Time Allocation:**

$$t_n^{\text{BC}} + t_{n,k}^{\text{LOC}} + t_{n,k}^{\text{Agg}} \leq t_n \tag{4.14}$$

  This constraint specifies that the time required for broadcasting, local computation, and aggregation in layer-pack n operation should be less than or equal to the allocated time t_n. The master RCD does not have the communication stages since the data is already in the master RCD.

- **Constraint on the Master RCD's Local Computation Time:**

$$t_{n,\widetilde{k}}^{\text{LOC}} \leq t_n \tag{4.15}$$

  This constraint ensures that the master RCD's local computation time in layer-pack n operation does not exceed the allocated time t_n.

- **Constraint on the Maximum Allowed Latency:**

$$\sum_{n=1}^{N^{\text{LAY}}} t_n \leq \tau \tag{4.16}$$

This constraint states that the total time taken to complete the entire CNN inference, considering all layer-pack operations, must be less than or equal to the maximum allowed latency τ.

4.4 Problem Formulation

The final problem formulation is given by the following optimization problem:

$$\begin{aligned}
\min_{\mathbf{x}_1} \quad & \sum_{n=1}^{N^{\text{LAY}}} (E_n^{\text{BC}} + \sum_{k=1}^{K_n} E_{n,k}^{\text{LOC}} + \sum_{\substack{k=1 \\ k\neq\tilde{k}}}^{K_n} E_{n,k}^{\text{Agg}}) \\
\text{s.t.} \quad & (4.4), (4.5), (4.8), (4.11), (4.12), (4.13), (4.14), (4.15), (4.16), \\
& 0 \leq d_{n,k}^{\text{ch}}, t_n^{\text{BC}}, p_n^{\text{BC}}, p_{n,k}^{\text{Agg}}, \\
& 0 \leq t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{Agg}} \leq t_n
\end{aligned} \tag{4.17}$$

where $\mathbf{x}_1 = \{\{t_n, t_n^{\text{BC}}, p_n^{\text{BC}}\}_{n=1}^{N^{\text{LAY}}}, \{\{d_{n,k}^{ch}, t_{n,k}^{\text{LOC}}\}_{k=1}^{K_n}\}_{n=1}^{N^{\text{LAY}}}, \{\{t_{n,k}^{\text{Agg}}, p_{n,k}^{\text{Agg}}\}_{\substack{k=1 \\ k\neq\tilde{k}}}^{K_n}\}_{n=1}^{N^{\text{LAY}}}\}$.

To address the non-convexity introduced by communication constraints, as in Chapter 3, we introduce two new parameters, $E_n^{\text{BC}_{\text{RF}}} = p_n^{\text{BC}} t_n^{\text{BC}}$ and $E_{n,k}^{\text{Agg}_{\text{RF}}} = p_{n,k}^{\text{Agg}} t_{n,k}^{\text{Agg}}$, which represent the RF energy consumed during broadcasting and aggregation, respectively. Then, we modify the related constraints (4.5), (4.12) and equations (4.2), (4.3), (4.9), (4.10) as

$$r_n^{\text{BC}} = B \ \ln(1 + \frac{(E_n^{\text{BC}_{\text{RF}}}/t_n^{\text{BC}}) \min_k \left(\frac{h_{n,k}^{\text{BC}}}{I(R_{n,k}^{\text{BC}})}\right)}{BN_0}) \tag{4.18}$$

$$E_n^{\text{BC}} = E_n^{\text{BC}_{\text{RF}}} + t_n^{\text{BC}} P_{\tilde{k}}^{c} \tag{4.19}$$

$$E_n^{\text{BC}_{\text{RF}}} \leq t_n^{\text{BC}} \ p_{\tilde{k}}^{\max} \tag{4.20}$$

$$r_{n,k}^{\text{Agg}} = B\ \ln(1 + \frac{(E_{n,k}^{\text{Agg}_{\text{RF}}} / t_{n,k}^{\text{Agg}}) \frac{h_{n,k}^{\text{Agg}}}{l(R_{n,k}^{\text{Agg}})}}{BN_0}) \tag{4.21}$$

$$E_{n,k}^{\text{Agg}} = E_{n,k}^{\text{Agg}_{\text{RF}}} + t_{n,k}^{\text{Agg}} P_k^c \tag{4.22}$$

$$E_{n,k}^{\text{Agg}_{\text{RF}}} \leq t_{n,k}^{\text{Agg}}\ p_k^{\max} \tag{4.23}$$

Thus, the final convex optimization problem becomes

$$\begin{aligned} \min_{\mathbf{x}_2} \quad & \sum_{n=1}^{N^{\text{LAY}}} (E_n^{\text{BC}} + \sum_{k=1}^{K_n} E_{n,k}^{\text{LOC}} + \sum_{\substack{k=1 \\ k \neq \tilde{k}}}^{K_n} E_{n,k}^{\text{Agg}}) \\ \text{s.t.} \quad & (4.4), (4.8), (4.11), (4.13), (4.14), (4.15), (4.16), (4.20), (4.23), \\ & 0 \leq d_{n,k}^{\text{ch}}, t_n^{\text{BC}}, E_n^{\text{BC}_{\text{RF}}}, E_{n,k}^{\text{Agg}_{\text{RF}}}, \\ & 0 \leq t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{Agg}} \leq t_n \end{aligned} \tag{4.24}$$

where $\mathbf{x}_2 = \{\{t_n, t_n^{\text{BC}}, E_n^{\text{BC}_{\text{RF}}}\}_{n=1}^{N^{\text{LAY}}}, \{\{d_{n,k}^{ch}, t_{n,k}^{\text{LOC}}\}_{k=1}^{K_n}\}_{n=1}^{N^{\text{LAY}}}, \{\{t_{n,k}^{\text{Agg}}, E_{n,k}^{\text{Agg}_{\text{RF}}}\}_{\substack{k=1 \\ k \neq \tilde{k}}}^{K_n}\}_{n=1}^{N^{\text{LAY}}}\}$.

4.5 Problem Decomposition

The final optimization problem can be effectively solved using classical convex optimization techniques. However, to gain an analytical interpretation of each parameter and facilitate distributed optimization, we decompose the problem into two sub-problems without sacrificing generality. This decomposition leverages the primal-dual decomposition technique, with a detailed proof available in [67]. The decomposed problem consists of two sub-problems: internal and external. The external problem's solution relies on the optimal solution of the internal problem, while the internal problem's solution is determined based on the optimal parameters of the external problem. The decomposition algorithm, outlined in Algorithm 2, provides a distributed optimization framework for the system. Although it introduces an additional communication stage for exchanging optimal parameters, this overhead is negligible compared to the benefits of distributed optimization and the reduced computational burden on the master RCD. Nevertheless, the optimization can also be performed without decomposition if minimizing this additional communication stage is desirable.

Algorithm 2 Decomposition algorithm

1: The parameters $d_{n,k}^{\text{ch}}$, K_n, t_n^{BC} and t_n are initialized.
2: Given the parameters $d_{n,k}^{\text{ch}}$, t_n^{BC} and t_n, the internal optimization problem (4.25), (4.26) is solved for the parameters $t_{n,k}^{\text{LOC}}$, $t_{n,k}^{\text{Agg}}$ and $E_{n,k}^{\text{Agg}_{\text{RF}}}$.
3: By using the optimal solution $(E_{n,k}^{\text{INT}})^*$ of the internal optimization problem, the external optimization problem (4.27) is solved for the parameters $t_n, t_n^{\text{BC}}, E_n^{\text{BC}_{\text{RF}}}$ and $d_{n,k}^{\text{ch}}$ for each $0 \leq K_n \leq K$ value and the optimal K_n value is selected.
4: The procedure is repeated from item 2 with the new parameters found in item 3 until the system reaches a consensus.

4.5.1 Internal Optimization Problem

We analyze the internal optimization problem in two distinct scenarios: one for the master RCD and one for the slave RCDs.

The master RCD (for $k = \widetilde{k}$) internal optimization problem can be formulated as

$$\begin{aligned}(E_{n,\widetilde{k}}^{\text{INT}})^* = \min_{t_{n,\widetilde{k}}^{\text{LOC}}} \quad & E_{n,\widetilde{k}}^{\text{LOC}} \\ \text{s.t.} \quad & 36\frac{c_{\widetilde{k}}}{\nu_{\widetilde{k}}^{\max}}\frac{d^{\text{row}}d^{\text{col}}}{4^n}d_{n,\widetilde{k}}^{\text{ch}} \leq t_{n,\widetilde{k}}^{\text{LOC}} \leq t_n\end{aligned} \tag{4.25}$$

Meanwhile, the internal optimization problem for the other participating RCDs (with $k \in [K_n] - \widetilde{k}$) is

$$\begin{aligned}(E_{n,k}^{\text{INT}})^* = \min_{\mathbf{x}_3} \quad & E_{n,k}^{\text{LOC}} + E_{n,k}^{\text{Agg}} \\ \text{s.t.} \quad & (4.8), (4.11), (4.14), (4.23), \\ & 0 \leq E_{n,k}^{\text{Agg}_{\text{RF}}}, \\ & 0 \leq t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{Agg}} \leq t_n\end{aligned} \tag{4.26}$$

where $\mathbf{x}_3 = \{t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{Agg}}, E_{n,k}^{\text{Agg}_{\text{RF}}}\}$ and $(E_{n,k}^{\text{INT}})^*$ is the optimal solution of the internal optimization problem given the parameters t_n, t_n^{BC} and $d_{n,k}^{\text{ch}}$ from the external optimization problem.

4.5.2 External Optimization Problem

The external optimization problem can be stated as

$$\begin{aligned} \min_{\mathbf{x}_4} \quad & \sum_{n=1}^{N^{\mathrm{LAY}}} \left(E_n^{\mathrm{BC}} + \sum_{k=1}^{K_n} (E_{n,k}^{\mathrm{INT}})^* \right) \\ \text{s.t.} \quad & (4.4), (4.13), (4.16), (4.20), \\ & 0 \leq d_{n,k}^{\mathrm{ch}}, t_n, t_n^{\mathrm{BC}}, E_n^{\mathrm{BC_{RF}}} \end{aligned} \tag{4.27}$$

where $\mathbf{x_4} = \{t_n, t_n^{\mathrm{BC}}, E_n^{\mathrm{BC_{RF}}}, d_{n,k}^{\mathrm{ch}}\}$, $\forall n \in [N^{\mathrm{LAY}}]$ and $\forall k \in [K_n]$.

This decomposition approach allows for the analytical interpretation of each parameter and facilitates distributed optimization while introducing minimal communication overhead, ultimately reducing the optimization workload on the master RCD. The analytical solutions of internal and external optimization problems for each optimizing parameter are given in Appendices C and D, respectively.

4.6 Simulation Results

In this section, we showcase the performance of the proposed scheme with a series of simulations. Table 4.1 provides an overview of the selected parameters used in our simulations. Similar to the previous study, the simulation outcomes presented here are based on averaging results from 1000 distinct random channel realizations. These simulation results rely on various random performance parameters, such as c_k (ranging uniformly between 250 and 550 CPU cycles/MAC) and $\nu_k^{\max}$ (uniformly distributed between 1 and 3 GHz). It is important to note that, despite experimenting with different random scenarios, there were no significant variations observed in the overall pattern of the simulation results. This suggests that the proposed approach is adaptable and can be applied effectively in diverse scenarios involving devices with varying specifications.

To conduct these simulations, we harnessed the capabilities of the MATLAB CVX toolbox [70] and the MOSEK optimizer. MOSEK is particularly well-suited for solving optimization problems involving a combination of integer and real parameters, ensuring accurate and efficient results.

In Fig. 4.3, the overall energy consumption of different benchmark scenarios in the literature is compared by varying the maximum allowed latency.

TABLE 4.1: Selected parameters for the simulation results of Chapter 4

Parameters	Values	Units
κ_k	Unif$([10^{-28}; 10^{-27}])$	/
c_k	Unif$([250; 550])$	CPU cycles/MAC
$\nu_k^{\max}$	Unif$([1; 3])$	GHz
h_k	$\mathcal{CN}(0, 10^{-3})$(Rayleigh)	/
$p_k^{\max}$	Unif$([10; 25])$	mW
P_k^c	Unif$([10; 25])$	mW
B	2	MHz
N_0	0.5×10^{-11}	W/MHz
d^{col} & d^{row}	28	bits
f_c	2.1	GHz
α	2.5	/
$R_{n,k}^{\text{BC}}$ & $R_{n,k}^{\text{Agg}}$	Unif$([1; 20])$	m

The blue curve represents the energy consumption achieved by our proposed approach in this chapter, involving 6 RCDs. The green curve depicts a scenario where the workload distribution is equal among RCDs (i.e., an equal number of filters allocated to each RCD), but the communication and computation parameters are still optimized. The red curve represents a non-distributed scenario where the master RCD attempts to perform the entire operation single-handedly. Another benchmark scenario involves including a powerful MEC server within a range of R meters to offload the entire operation. It's worth mentioning that for $\tau < 0.5$ s, the red curve is not present because a single RCD cannot complete the operation within such a short timeframe. Furthermore, for $\tau > 0.5$ s, the red curve exhibits poorer performance compared to the collaborative scenarios (green and blue curves). Our proposed approach demonstrates a substantial reduction in energy consumption, particularly in low-latency scenarios. This is attributed to the significance of jointly optimizing filter distribution and communication/computation parameters. However, as the maximum allowed latency increases, the performance gap between these scenarios diminishes significantly. This trend suggests that optimization becomes less critical in scenarios with higher latencies since each RCD has sufficient time to perform its computations efficiently. Additionally, both MEC server scenarios, represented by the curves for $R = 40$ m and $R = 80$ m, consume more energy compared to our proposed approach. This is due to the absence of parallel computing and the long distance between the master RCD and the MEC server, which leads to higher communication costs.

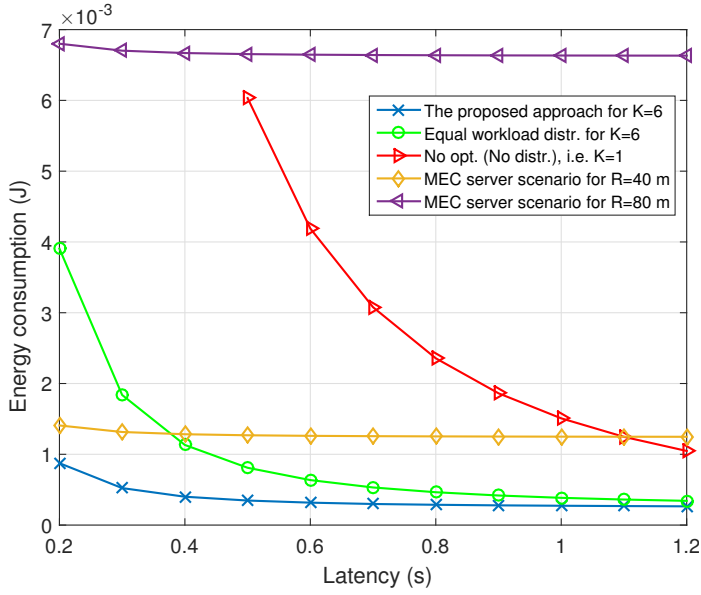


FIGURE 4.3: Overall energy consumption vs. latency for different benchmark scenarios for $N^{\mathrm{LAY}} = 3$ and $d_n^{\mathrm{ch}} = 32, \forall n \in [N^{\mathrm{LAY}}]$

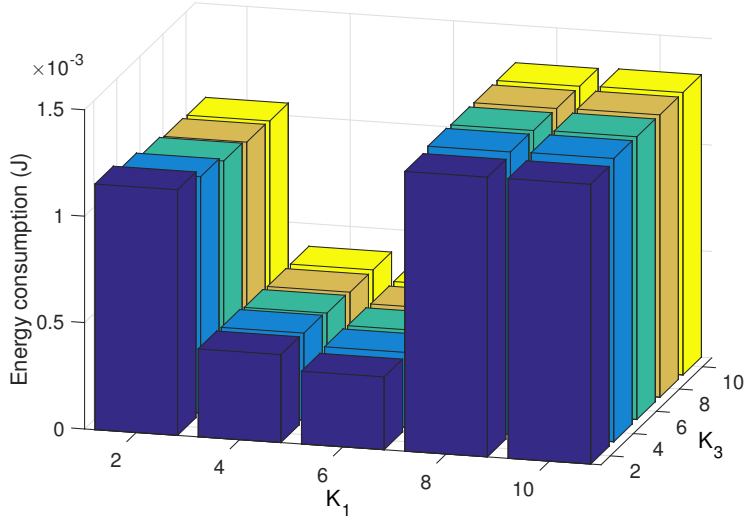


FIGURE 4.4: Overall energy consumption vs. K_1, K_3 for $\tau = 0.5$ s, $K = 10$, $K_2 = 6$, $N^{\mathrm{LAY}} = 3$ and $d_n^{\mathrm{ch}} = 32$, $\forall n \in [N^{\mathrm{LAY}}]$

In Fig. 4.4, a 3D visualization illustrates the relationship between overall energy consumption and various combinations of K_1 and K_3 by fixing $K_2 = 6$. The results show that increasing the number of RCDs in each layer initially improves energy efficiency, but after a certain point, adding more RCDs increases energy consumption. This emphasizes the importance of the critical trade-off between communication and computation. The addition of more RCDs enhances the computational capabilities of the system up to a specific point, making the overall system more energy efficient. Beyond this point, requesting further collaboration becomes inefficient as communication costs start to outweigh computational costs. Furthermore, it's important to note that varying the number of RCDs (K_3) in the third layer-pack operation has a different impact compared to changing the number of RCDs (K_1) in the first layer-pack operation. This distinction arises due to differences in input data dimensions and the number of MAC operations in the back-end and front-end layer-packs. For instance, when K_1 varies from 2 to 6, a substantial decrease occurs in the overall energy consumption of the system. In contrast, when K_3 varies within the same range, the energy consumption variation is minimal. This underscores the necessity of optimizing the number of participating RCDs differently for each layer-pack operation. Front-end layer-pack operations, characterized by large input data sizes and a higher number of MAC operations, appear to benefit more from increased collaboration. On the other hand, back-end layer-pack operations involve smaller input data sizes and fewer MAC operations. Consequently, using fewer RCDs for back-end layer-packs makes sense in these scenarios, as increased collaboration does not yield significant energy efficiency gains in terms of computation but consumes additional energy in communication.

Fig. 4.5 is the 4D visualization version of Fig. 4.4 by relaxing K_2. Each data point in this plot corresponds to a specific combination of K_n, and its color represents the corresponding energy consumption value. This figure presents a more general view to the problem, but more complex to observe the trade-off compared to Fig. 4.4. Consistent with our earlier observations, this figure reaffirms that increasing the number of RCDs within a layer can indeed be advantageous up to a certain threshold. Beyond this point, however, further increments in the number of RCDs become inefficient, primarily due to the increasing dominance of communication costs over computation costs.

Fig. 4.6 presents the energy consumption of each separate stage for three different number of filters d_n^{ch} in each layer-pack. For configurations with a relatively small number of filters (specifically, when $d_n^{\text{ch}} = 16$), we observe

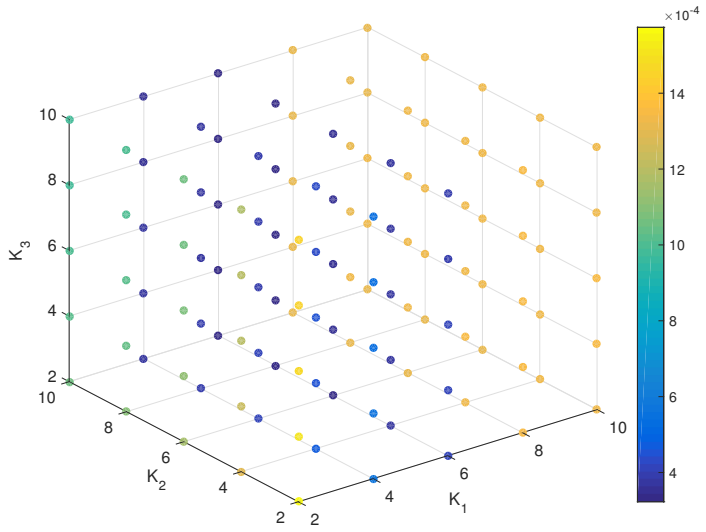


FIGURE 4.5: Overall energy consumption vs. K_1, K_2, K_3 for $\tau = 0.5$ s, $K = 10$, $N^{\text{LAY}} = 3$ and $d_n^{\text{ch}} = 32$, $\forall n \in [N^{\text{LAY}}]$

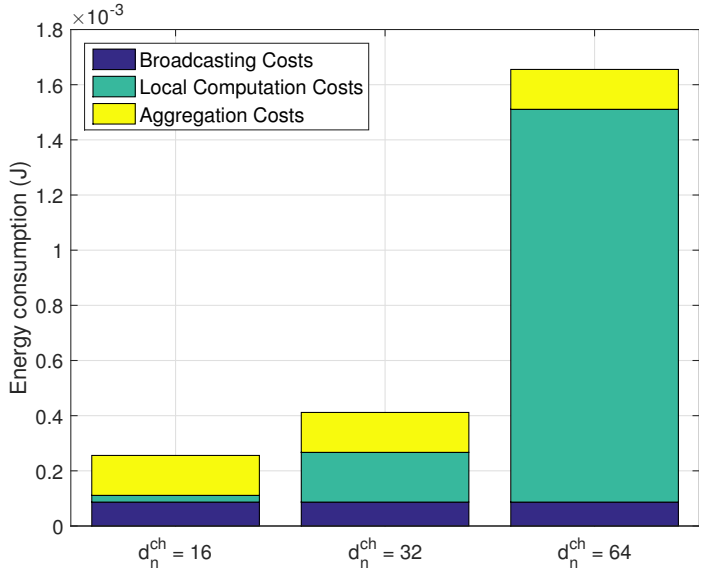


FIGURE 4.6: Energy consumption of each stage for different values of d_n^{ch} for $\tau = 0.5$ s, $N^{\text{LAY}} = 3$ and $K = K_n = 5$, $\forall n \in [N^{\text{LAY}}]$

that the dominant factor contributing to energy consumption is communication costs. In these scenarios, the energy spent on data transfer and inter-device communication outweighs the computational costs associated with executing the CNN model. However, as we increase the value of d_n^{ch}, the energy consumption becomes increasingly driven by computation costs. This finding supports the idea that the trade-off between communication and computation needs to be optimized given a specific amount of filters in a CNN model.

Fig. 4.7 presents an analysis of the energy consumption of each layer-pack for CNN models with a different number of layers N^{LAY}. One notable trend that emerges from this figure is the exponential decrease in energy consumption as we move towards the back-end layer-packs. This decline can be attributed to the reduction in input data dimensions as we progress through the layer-packs. As the data dimensions become smaller, the associated communication and computation costs decrease accordingly, resulting in a more energy-efficient operation in the back-end layers. Additionally, we observe a slight decrease in energy consumption as N^{LAY} decreases. With fewer layer-pack operations to process, the system allocates more time to the initial layer-pack. This extended processing time translates to improved energy efficiency, as both communication and local computation can be executed more relaxed. As a result, the energy consumption in the first layer-pack decreases, demonstrating the impact of model architecture on energy efficiency.

4.7 Summary

In this chapter, we introduce a collaborative model parallelism approach for on-device CNN inference architectures. We leverage communication capabilities around to divide the model across multiple RCDs, optimizing filter assignments and parameters for communication and computation. This approach eliminates the need for powerful servers during inference, making it suitable for resource-constrained environments. We also investigate an output aggregation method after each layer operation, ensuring adaptability for collaborative CNN inference. By formulating a convex optimization problem and employing an iterative algorithm, we efficiently distribute filter computations among RCDs, with the aim of energy consumption minimization for latency-constrained scenarios.

The results of this study provide important insights into improving energy efficiency within the system. In contrast to a non-collaborative approach where a single RCD attempts to execute the CNN independently, collaborative execution across multiple RCDs significantly improves performance

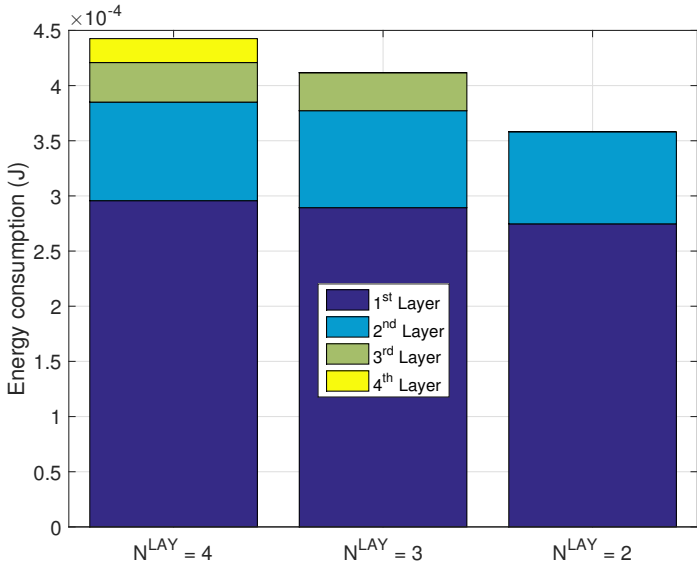


FIGURE 4.7: Energy consumption of each layer-pack for CNN models with different number of layers N^{LAY} for $\tau = 0.5$ s, $d_n^{\text{ch}} = 32$ and $K = K_n = 5, \forall n \in [N^{\text{LAY}}]$

in terms of energy efficiency, particularly in low-latency scenarios. Furthermore, the findings suggest a trade-off between communication and computation concerning the number of participating devices parameter. Initially, increasing system collaborativeness proves beneficial for energy efficiency until a threshold point, as the dominant factor in scenarios with a low number of devices is computation consumption. However, beyond this threshold, further increasing the number of participating devices does not yield additional benefits, as communication consumption begins to dominate overall energy consumption. Therefore, the proposed approach is most effective for scenarios with low input data sizes, low latency, high model complexity, and a moderate number of participating devices, depending on other system parameters.

The model parallelism approach studied in this chapter makes multiple filter computation feasible for collaborative CNN inference and solves the problems of data parallelism approach. However, a challenge arises when dealing with large input data sizes. The model parallelism approach involves several consecutive communication stages, leading to overhead as the entire data is sent between the data owner RCD and C-RCDs unlike the data parallelism approach where only the boundary rows are exchanged

between neighboring devices.

To address this concern, the next chapter proposes a hybrid parallelism scenario by merging the data parallelism approach discussed in Chapter 3 with the model parallelism approach outlined in Chapter 4. This hybrid approach not only makes multiple-filter computation feasible but also resolves the challenges associated with handling large input data sizes. An additional enhancement is introduced by incorporating relay devices into the system, further boosting the efficiency of our collaborative CNN inference framework.

Chapter 5

A Relay-Assisted Hybrid Parallelism Architecture for Collaborative On-Device CNN Inference

This chapter explores a relay-assisted collaborative on-device CNN inference approach, with a focus on latency-critical deep learning-driven applications. We introduce a hybrid parallelism strategy that combines both data and model parallelism techniques, complemented by relay-assisted communication to optimize the execution of CNNs on RCDs. The key contributions of this chapter's study include:

- **Relay-assisted collaborative on-device CNN inference scheme:** The proposed approach introduces a system architecture that leverages relay-assisted communication to enable collaborative CNN inference on RCDs. This distributed approach reduces the computational burden on individual RCDs and enables efficient and scalable inference of CNNs on RCDs.
- **Hybrid parallelism combining data and model parallelism:** Our approach considers a hybrid parallelism strategy that combines both data and model parallelism techniques to optimize the collaborative inference of CNNs on RCDs. This hybrid approach allows for the efficient utilization of resources on each RCD while minimizing communication overhead.
- **Communication-aware energy consumption minimization:** We focus on achieving the optimal trade-off between communication and

computation to minimize energy consumption. By effectively managing communication overhead through relay-assisted communication, our approach significantly reduces the overall energy requirements for collaborative CNN inference on RCDs.

It's worth noting that the CNN model employed in this chapter is consistent with the architecture discussed in Chapter 2, comprising multiple pairs of convolutional and max-pooling layers, each equipped with multiple filters.

5.1 Parallelism Techniques

In this section, the parallelism techniques in the proposed hybrid parallelism in this study are discussed and analyzed.

5.1.1 Model Parallelism

Model parallelism involves partitioning both the layer-packs and the filters within each layer-pack across multiple devices, and the partial models are executed collaboratively by using the entire input data. Chapter 4 of this thesis explores a model parallelism scenario.

To recap the model parallelism approach, Fig. 5.1 illustrates an example model parallelism scenario for distributing filters of layer-pack n operation among K number of collaborative RCDs (C-RCDs). In the figure, $f^{\text{conv}}_{n,k,j}$ is the j^{th} assigned convolutional filter of C-RCD k, and $d^{\text{ch}}_{n,k}$ corresponds to the number of filters assigned to C-RCD k for layer-pack n operation. First, the input data i_n is broadcasted by the data owner RCD (DO-RCD) to all C-RCDs, which perform local computation with their assigned filters. Then, the max-pooling outputs of all filters assigned to C-RCD k are internally summed within each C-RCD to produce the output data $o_{n,k}$ for layer-pack n operation, which is represented as

$$o_{n,k} = \sum_{j=1}^{d^{\text{ch}}_{n,k}} MP(i_n * f^{\text{conv}}_{n,k,j}) \tag{5.1}$$

The resulting output data $o_{n,k}$ from all C-RCDs is then transmitted to the DO-RCD. Finally, the DO-RCD sums up all received output data $o_{n,k}$ from all C-RCDs to obtain the final output data $o_n = \sum_{k=1}^{K} o_{n,k}$. This output data serves as the input for the next layer-pack operation ($o_n = i_{n+1}$).

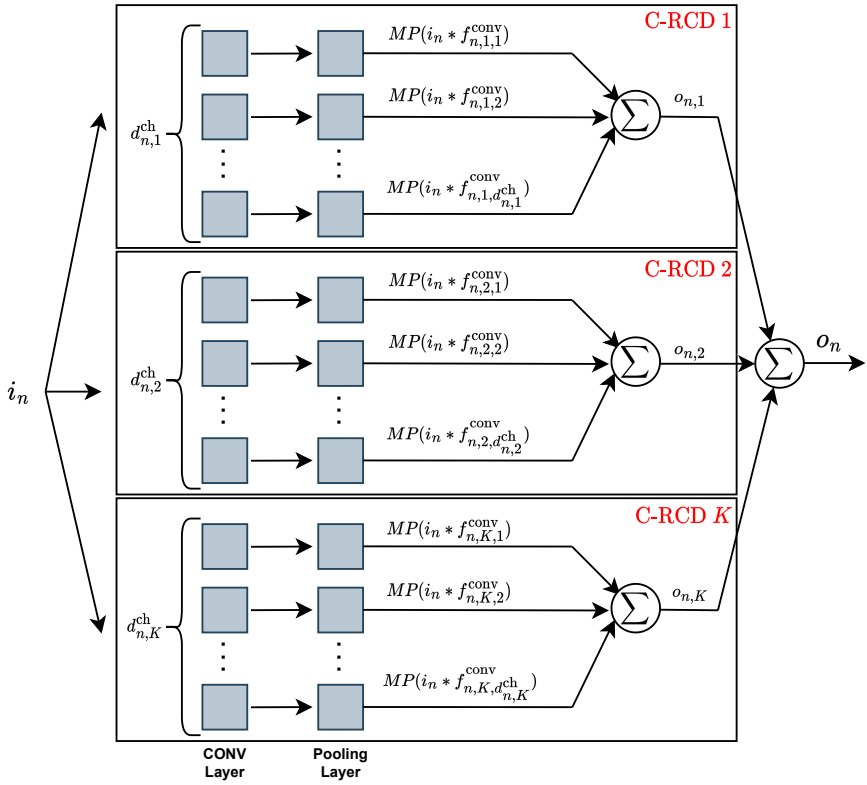


FIGURE 5.1: Layer-pack n execution of a collaborative CNN model parallelism inference

It's noteworthy that each C-RCD does not send the output data of each filter individually because the summation of the output data from each filter is sufficient for the next layer-pack operation, given the nature of the convolution operation. In this way, the communication overhead in the system is considerably reduced, as also stated in Chapter 4.

5.1.2 Data Parallelism

In the data parallelism approach, the input data is cut in the DO-RCD and partitioned among multiple computing segments, each of which executes the complete model computations with its partial input data. Depending on the scenario, the computing segment could be composed of a single C-RCD or multiple C-RCDs. A single layer-pack operation in a typical data

parallelism scenario can be seen in Fig. 5.2 where d_m is the number of input data rows assigned to computing segment m. Such a scenario has been previously investigated and optimized in our study in Chapter 3. The partial input data is sent to each computing segment along with some amount of redundant rows d^R that must be additionally sent to each computing segment due to the nature of the convolution operation, which is shown as a shaded area in Fig. 5.2. After each layer-pack operation, the boundary rows are exchanged between neighboring computing segments to be able to perform the next layer-pack operation, which introduces communication overhead to the system.

What is the difference between the data parallelism approach in the study in Chapter 3 and the data parallelism part of this chapter's study is the following: To mitigate the communication overhead, a solution is proposed in this study, which is to form layer-pack blocks. Each layer-pack block contains multiple layer-packs, which are performed within the block without communication with the neighboring computing segments. At the start of each block, additional redundant rows are also dispatched to each computing segment to compute its own boundary rows, which are intended to be sent from its neighboring computing segments after each layer-pack operation. Thus, the exchange communication costs can be mitigated while incurring additional computation costs due to the computation of additional redundant rows in multiple computing segments.

If each layer-pack corresponds to a block, implying that no block is formed, exchanging boundary rows after each layer-pack operation with the neighboring computing segments leads to the previously mentioned communication overhead. On the other hand, if a single block consisting of all layer-packs in the CNN model is formed, there will be a significant amount of additional computation costs that exceed the aforementioned communication costs. Thus, there exists a trade-off between communication and computation.

The layer-pack block formation is shown in Fig. 5.3 where N^{bl} is the number of blocks and N_b^{LAY} is the number of layer-packs in block b.

5.1.3 Hybrid Parallelism

The utilization of both data and model parallelism for collaborative CNN inference can be regarded as a promising solution. Data parallelism effectively addresses challenges associated with large input data sizes, while model parallelism brings a multiple-filter computation opportunity to the system. However, merging data and model parallelism in scenarios where multiple filters are present in each convolutional layer is quite challenging.

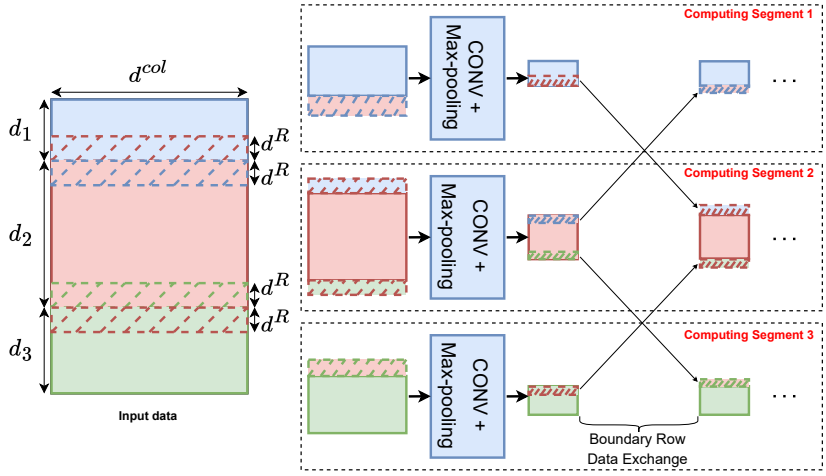


FIGURE 5.2: Single layer-pack operation and boundary row data exchange in a data parallelism scenario

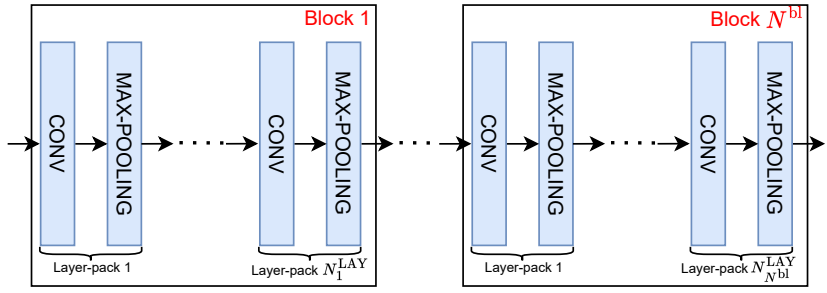


FIGURE 5.3: Layer-pack block formation

This is because, after the first layer-pack operation, each C-RCD requires the complete output data from all other C-RCDs to proceed to the next layer-pack operation. However, in hybrid parallelism, C-RCDs lack this entire data because of the data parallelism that divides the input data at the start of the operation.

One potential solution for merging these parallelism techniques involves segmenting the C-RCDs and assigning different partial input data to C-RCDs within different segments. With this approach, the DO-RCD assigns and distributes each partial input data to each segment having multiple C-RCDs, which is the data parallelism part. With the assigned partial input data, the model parallelism inference is then performed in the C-RCDs inside each segment. However, implementing this approach requires an excessive amount of point-to-point communication between the DO-RCD and C-RCDs to send each partial input data and receive the output data after each block operation.

To address this challenge, this study introduces an intermediate device, such as a relay device, within each segment to coordinate the communication between the DO-RCD and C-RCDs. This device is responsible for receiving the partial input data from the DO-RCD, broadcasting it to the C-RCDs in its segment, orchestrating the model parallelism part of the system, and sending the final output data after each block operation to the DO-RCD to form the new input data for the next block operation. This relay device plays a crucial role in facilitating efficient communication and orchestration in hybrid parallelism scenarios, reducing communication complexity and enhancing the overall system's performance.

5.2 Relay-assisted Collaborative CNN Inference Considering Hybrid Parallelism

The relay-assisted collaborative CNN inference framework, incorporating hybrid parallelism, is illustrated in Fig. 5.4. This model comprises a single DO-RCD that possesses the input data, N^{rel} number of computing segments, each of which contains a single relay and K_m number of C-RCDs assigned to segment m to participate in the collaborative computation.

To address the convolution operation's characteristics, which require additional redundant rows, the number of redundant rows d_b^R for block b

operation is determined as

$$d_b^R = \sum_{n=1}^{N_b^{\mathrm{LAY}}} 2^{n-1} \tag{5.2}$$

Before the operation starts, the CNN model is entirely pre-trained in powerful cloud servers by offline datasets. Therefore, this study, as well as other studies in the previous chapters, mainly focuses on the inference phase rather than the training phase.

The workflow for each block operation within the relay-assisted collaborative CNN inference framework is as follows:

- The input data is initially divided into N^{rel} number of pieces by the DO-RCD before the first block operation. It is then sent to each relay along with d_1^R number of redundant rows. On the other hand, only d_b^R number of rows are sent before the blocks other than the first block since the partial input data already exists in each relay from the previous block operation. This step is referred to as the input distribution (ID) in the data parallelism part of the system. It is important to note that retaining the majority of the input and output data in relay devices significantly reduces communication overhead.
- Once the partial input data is received by each relay, it orchestrates the model parallelism part of the system with its assigned C-RCDs. The model parallelism consists of three stages, which are repeated for each layer-pack operation within the same block.
  - First, each relay broadcasts the input data with the redundant rows to the assigned C-RCDs.
  - Next, each C-RCD executes its local computation using the partial input data and assigned filters.
  - After each layer-pack operation, each C-RCD returns the summation of its assigned filter's output data to its relay. The relay aggregates the output data from all C-RCDs and uses it as the input data for the next layer-pack operation.
- At the end of all layer-pack operations within the same block, each relay sends the boundary rows to the DO-RCD to be used for the next layer-pack operation. For the final block operation, each relay sends the entire output data to the DO-RCD, rather than just the boundary rows, to complete the collaborative CNN inference.

FIGURE 5.4: A single block operation of the relay-assisted collaborative CNN inference considering hybrid parallelism

- DO-RCD collects all boundary rows, and the process continues from the input distribution stage of the next block operation.

In the next section, the energy consumption of each stage in this relay-assisted collaborative CNN inference framework with hybrid parallelism is modeled and analyzed.

5.3 Energy Consumption Modeling

In this study, we assume that each computing segment is spatially sparse, meaning that interference between segments is negligible. However, it's important to note that in scenarios where segments are closer together, the impact of interference on energy consumption optimization must be further investigated. This will be a focus of our study in Chapter 6.

The energy consumption stages can be divided into three parts for each block, namely,

- partial input distribution by the DO-RCD to each relay (First stage of data parallelism),
- the collaborative computation in each computing segment, composed of a single relay and multiple C-RCDs (Model parallelism), including the broadcasting stage, local computation stage, and aggregation stage,
- output transmission (OT) to the DO-RCD by each relay after all layer-pack operations in the block are performed (Last stage of data parallelism).

5.3.1 Input Distribution (ID)

This stage marks the beginning of each block operation, during which the DO-RCD transmits partial input data to each relay. These relays coordinate the intra-block collaborative computation in the model parallelism segment with their assigned C-RCDs. The achievable data rate for communication between the DO-RCD and relay m for block b operation is given by

$$r_{b,m}^{\mathrm{ID}} = B \ \ln(1 + \frac{p_{b,m}^{\mathrm{ID}} \frac{h_{b,m}^{\mathrm{ID}}}{l(R_{b,m}^{\mathrm{ID}})}}{BN_0}) \tag{5.3}$$

where $p_{b,m}^{\mathrm{ID}}$ is the RF transmit power of the DO-RCD (shown as $\widetilde{k}$) during ID, B is the communication bandwidth, N_0 is the noise power density, $h_{b,m}^{\mathrm{ID}}$

and $R^{\text{ID}}_{b,m}$ are the channel gain and distance between DO-RCD and relay m for block b operation 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 operation is

$$E^{\text{ID}}_{b,m} = t^{\text{ID}}_{b,m}(p^{\text{ID}}_{b,m} + P^{c}_{\tilde{k}}) \tag{5.4}$$

where $P^{c}_{\tilde{k}}$ is the constant power consumption of the communication circuitry in the DO-RCD and $t^{\text{ID}}_{b,m}$ is the time required to send the input data to relay m during block b operation. The number of bits $d^{\text{ID}}_{b,m}$ that must be sent during ID is upper-bounded by the following constraint:

$$d^{\text{ID}}_{b,m} \leq t^{\text{ID}}_{b,m} r^{\text{ID}}_{b,m} \tag{5.5}$$

where $d^{\text{ID}}_{b,m}$ is defined as

$$d^{\text{ID}}_{b,m} = \begin{cases} (d_m + I_m d^{R}_{1}) d^{\text{col}} & \text{if } b = 1 \\ I_m d^{R}_{b} d^{\text{col}}_{b} & \text{otherwise} \end{cases} \tag{5.6}$$

Here, $d^{\text{col}}_{b} = \frac{d^{\text{col}}}{2^{\sum_{b'=1}^{b-1} N^{LAY}_{b'}}}$ 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. For the first block operation, the DO-RCD sends both the partial input data and the redundant rows during the ID stage. For subsequent block operations, only redundant rows are sent by the DO-RCD, as the majority of the input data is already present in the relays.

5.3.2 Model Parallelism

Once the relay devices receive the partial input data from the DO-RCD before each block operation, the relay devices play a crucial role in orchestrating the model parallelism part of the system. During this part, the relay devices optimally allocate the convolutional layer filters among K_m number of participating C-RCDs assigned to relay m, taking into account their heterogeneous computing capabilities, as well as different network conditions, including channel gain, environmental noise, and path-loss components. These network conditions collectively influence the optimal distribution of computational resources, affecting signal strength and signal attenuation.

The process for each layer-pack operation within a given block consists of three distinct stages:
(1) Broadcasting: In this stage, the partial input data received from the DO-RCD is broadcasted to all participating C-RCDs through the relay devices.
(2) Collaborative Local Computation: Multiple C-RCDs execute local computation.
(3) Aggregation: Following the local computation, the output data generated by each C-RCD's assigned filters is aggregated at the relay to create the output of that layer-pack.

These stages are performed iteratively for each layer-pack operation within the block. After each layer-pack operation, each C-RCD transmits the summation of the output data generated by its assigned filters back to the relay. The relay device then combines these outputs to generate the new input data for the subsequent layer-pack operation.

Broadcasting (BC)

Relay m broadcasts the input data for layer-pack n operation of block b to the participating C-RCDs at the achievable data rate

$$r^{\mathrm{BC}}_{n_b,m} = B\ \ln(1+\frac{p^{\mathrm{BC}}_{n_b,m}\ \min\limits_{k_m}\left\{\frac{h^{\mathrm{BC}}_{n_b,k_m}}{l(R^{\mathrm{BC}}_{n_b,k_m})}\right\}}{BN_0}) \tag{5.7}$$

where $p^{\mathrm{BC}}_{n_b,m}$ is the RF transmit power of relay m, $h^{\mathrm{BC}}_{n_b,k_m}$ and $R^{\mathrm{BC}}_{n_b,k_m}$ are the channel gain and distance between relay m and C-RCD k during layer-pack n operation of block b, respectively. The achievable data rate is dominated by the C-RCD, with the weakest channel condition among the others.

The energy consumption associated with broadcasting by relay m to transmit the partial input data for layer-pack n operation of block b is

$$E^{\mathrm{BC}}_{n_b,m} = t^{\mathrm{BC}}_{n_b,m}(p^{\mathrm{BC}}_{n_b,m} + P^c_m) \tag{5.8}$$

where P^c_m and $t^{\mathrm{BC}}_{n_b,m}$ are the constant power consumption of the communication circuitry and the broadcasting time for relay m during layer-pack n operation of block b, respectively. The number of bits broadcasted, denoted as $d^{\mathrm{BC}}_{n_b,m}$, is constrained by

$$d^{\mathrm{BC}}_{n_b,m} = \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^{n-1})}{2^{n-1}}\right)\frac{d^{\mathrm{col}}_b}{2^{n-1}} \leq t^{\mathrm{BC}}_{n_b,m} r^{\mathrm{BC}}_{n_b,m} \tag{5.9}$$

where $d_{b,m} = \frac{d_m}{2^{\sum_{b'=1}^{b-1} N_{b'}^{LAY}}}$ represents the number of rows held by relay m, excluding the redundant rows before block b operation.

Collaborative Local Computation

Once the input data is received, each C-RCD starts its local computation. Given that convolutional layers are the primary energy consumers in such operations, we focus on their energy consumption, similar to previous chapters.

The number of MAC operations N_{n_b,k_m}^{MAC} to be performed by C-RCD k of relay m for layer-pack n operation of block b is

$$\begin{aligned} N_{n_b,k_m}^{\text{MAC}} &= (d^{\text{filter}})^2 \left(\frac{d_{b,m} + I_m(d_b^R + 1 - 2^n)}{2^{n-1}} \right) \frac{d_b^{\text{col}}}{2^{n-1}} d_{n_b,k_m}^{\text{ch}} \\ &= \alpha_{n_b,m} d_{n_b,k_m}^{\text{ch}} \end{aligned} \tag{5.10}$$

where d_{n_b,k_m}^{ch} corresponds to the number of filters assigned to C-RCD k of relay m for layer-pack n operation in block b, $\alpha_{n_b,m}$ is defined for easier notation. We also consider $d^{\text{filter}} = 3$ as the preferred filter dimension.

The energy consumption of local computation in C-RCD k of relay m for layer-pack n operation of block b under the low CPU voltage assumption [73–76] is

$$E_{n_b,k_m}^{\text{LOC}} = \frac{\kappa_{k_m} c_{k_m}^3 (N_{n_b,k_m}^{\text{MAC}})^3}{(t_{n_b,k_m}^{\text{LOC}})^2} = \kappa_{k_m} c_{k_m}^3 (\alpha_{n_b,m})^3 \frac{(d_{n_b,k_m}^{\text{ch}})^3}{(t_{n_b,k_m}^{\text{LOC}})^2} \tag{5.11}$$

where κ_{k_m} is the effective hardware capacitance coefficient, c_{k_m} is the number of CPU cycles per MAC operation, and t_{n_b,k_m}^{LOC} is the time required to complete the local computation of layer-pack n operation of block b for C-RCD k of relay m. Regarding this stage, the following constraint holds:

$$c_{k_m} N_{n_b,k_m}^{\text{MAC}} = c_{k_m} \alpha_{n_b,m} d_{n_b,k_m}^{\text{ch}} \leq \nu_{k_m}^{\max} t_{n_b,k_m}^{\text{LOC}} \tag{5.12}$$

which puts an upper-bound for the frequency of each CPU in the C-RCDs where $\nu_{k_m}^{\max}$ is the maximum allowed CPU frequency for C-RCD k of relay m.

Aggregation

Each C-RCD k transmits the summation of the output data of their assigned filters for layer-pack n operation to its relay m. The achievable data rate for this transmission is given by

$$r^{\text{Agg}}_{n_b,k_m} = B \ \ln(1 + \frac{p^{\text{Agg}}_{n_b,k_m} \frac{h^{\text{Agg}}_{n_b,k_m}}{l(R^{\text{Agg}}_{n_b,k_m})}}{BN_0}) \tag{5.13}$$

where $p^{\text{Agg}}_{n_b,k_m}$ is the RF transmit power of 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 during aggregation of layer-pack n operation of block b, respectively.

The energy consumption of aggregation by C-RCD k of relay m to send the summation of the output data for layer-pack n operation of block b is

$$E^{\text{Agg}}_{n_b,k_m} = t^{\text{Agg}}_{n_b,k_m}(p^{\text{Agg}}_{n_b,k_m} + P^c_{k_m}) \tag{5.14}$$

where $P^c_{k_m}$ is the constant power consumption of the communication circuitry and $t^{\text{Agg}}_{n_b,k_m}$ is the transmission time for C-RCD k of relay m. Regarding the aggregation stage, we have the following constraint:

$$d^{\text{Agg}}_{n_b,m} = \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^n)}{2^n}\right) \frac{d^{\text{col}}_b}{2^n} \leq t^{\text{Agg}}_{n_b,k_m} r^{\text{Agg}}_{n_b,k_m} \tag{5.15}$$

where $d^{\text{Agg}}_{n_b,m}$ is the number of transmitted bits during each aggregation stage.

5.3.3 Output Transmission (OT)

This stage is the final part of the data parallelism phase for each block operation. After completing the model parallelism phase for each layer-pack operation within the block, each relay m is responsible for sending either only the redundant boundary rows required for the next block operation or the entire output data if the current block is the last block in the model.

The achievable data rate for the communication between relay m and the DO-RCD during OT is given by

$$r_{b,m}^{\mathrm{OT}} = B\ \ln(1 + \frac{p_{b,m}^{\mathrm{OT}} \frac{h_{b,m}^{\mathrm{OT}}}{l(R_{b,m}^{\mathrm{OT}})}}{BN_0}) \tag{5.16}$$

where $p_{b,m}^{\mathrm{OT}}$ is the RF transmit power of relay m, $h_{b,m}^{\mathrm{OT}}$ and $R_{b,m}^{\mathrm{OT}}$ are the channel gain and distance between relay m and the DO-RCD for block b operation during OT, respectively.

The energy consumption of OT for relay m to send the output data for block b operation is

$$E_{b,m}^{\mathrm{OT}} = t_{b,m}^{\mathrm{OT}}(p_{b,m}^{\mathrm{OT}} + P_m^c) \tag{5.17}$$

where $t_{b,m}^{\mathrm{OT}}$ is the transmission time for relay m during block b operation. The constraint of this stage for the number of bits $d_{b,m}^{\mathrm{OT}}$ that needs to be sent during OT is

$$d_{b,m}^{\mathrm{OT}} \leq t_{b,m}^{\mathrm{OT}} r_{b,m}^{\mathrm{OT}} \tag{5.18}$$

where $d_{b,m}^{\mathrm{OT}}$ is defined as

$$d_{b,m}^{\mathrm{OT}} = \begin{cases} \frac{d_{b,m}}{2^{N_b^{\mathrm{LAY}}}} \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{if } b = N^{\mathrm{bl}} \\ I_m d_{b+1}^R \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{otherwise} \end{cases} \tag{5.19}$$

where the redundant boundary rows for the next block operation are sent by each relay to the DO-RCD, which collects all the redundant boundary rows and sends them to each relay for the next block operation. Thus, each relay keeps most of the output data in itself since it will be used for the next block operation in the same relay. The only exception to this procedure is that if the current block is the last block, each relay needs to send the entire output data to the DO-RCD to complete the collaborative CNN inference.

5.4 Problem Formulation

Before formulating the problem, we have some additional constraints that need to be considered, such as

$$\sum_{k=1}^{K_m} d_{n_b,k_m}^{\mathrm{ch}} = d_{n_b}^{\mathrm{ch}} \tag{5.20}$$

$$t^{\mathrm{BC}}_{n_b,m} + t^{\mathrm{LOC}}_{n_b,k_m} + t^{\mathrm{Agg}}_{n_b,k_m} \leq t_{n_b,m} \tag{5.21}$$

$$t^{\mathrm{ID}}_{b,m} + \sum_{n=1}^{N^{\mathrm{LAY}}_b} t_{n_b,m} + t^{\mathrm{OT}}_{b,m} \leq t_b \tag{5.22}$$

$$\sum_{b=1}^{N^{\mathrm{bl}}} t_b \leq \tau \tag{5.23}$$

where $t_{n_b,m}$, t_b, and τ are the allocated times for each layer-pack operation in each block, each block operation, and the whole operation, respectively. Constraint (5.20) states that the summation of the number of filters assigned to each C-RCD should be equal to the total number of filters $d^{\mathrm{ch}}_{n_b}$ in convolutional layer n of block b. Constraint (5.21) means that each layer-pack operation of the blocks in the model parallelism part needs to be completed in the allocated time $t_{n_b,m}$ for each relay. Constraint (5.22) is a similar constraint of data parallelism for the DO-RCD to receive the output data of each relay after each block operation at the same time. Last, constraint (5.23) simply indicates that the whole procedure needs to be completed under τ seconds, which is the maximum allowed latency of the system.

To formulate the convex optimization problem, some non-convex constraints related to communication stages are replaced with equivalent RF energy parameters, following the approach used in previous studies in Chapters 3 and 4. These parameters are $E^{\mathrm{ID_{RF}}}_{b,m} = p^{\mathrm{ID}}_{b,m} t^{\mathrm{ID}}_{b,m}$, $E^{\mathrm{BC_{RF}}}_{n_b,m} = p^{\mathrm{BC}}_{n_b,m} t^{\mathrm{BC}}_{n_b,m}$, $E^{\mathrm{Agg_{RF}}}_{n_b,k_m} = p^{\mathrm{Agg}}_{n_b,k_m} t^{\mathrm{Agg}}_{n_b,k_m}$ and $E^{\mathrm{OT_{RF}}}_{b,m} = p^{\mathrm{OT}}_{b,m} t^{\mathrm{OT}}_{b,m}$, which are the consumed RF energy in each communication stage. Then, we modify the related constraints (5.3), (5.4), (5.7), (5.8), (5.13), (5.14), (5.16), (5.17) in each communication-related stage by substituting the RF energy parameters wherever the RF transmit power parameters are seen.

After convexifying the problem, the final convex optimization problem is formulated as

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \sum_{b=1}^{N^{\mathrm{bl}}} \sum_{m=1}^{N^{\mathrm{rel}}} \left(E^{\mathrm{ID}}_{b,m} + E^{\mathrm{OT}}_{b,m} + \sum_{n=1}^{N^{\mathrm{LAY}}_b} \left(E^{\mathrm{BC}}_{n_b,m} + \sum_{k=1}^{K_m} \left(E^{\mathrm{LOC}}_{n_b,k_m} + E^{\mathrm{Agg}}_{n_b,k_m} \right) \right) \right) \\
\text{s.t.} \quad & (5.5), (5.9), (5.12), (5.15), (5.18), (5.20), (5.21), (5.22), (5.23), \\
& 0 \leq t^{\mathrm{ID}}_{b,m}, t^{\mathrm{BC}}_{n_b,m}, t^{\mathrm{Agg}}_{n_b,k_m}, t^{\mathrm{OT}}_{b,m}, t^{\mathrm{LOC}}_{n_b,k_m}, d^{\mathrm{ch}}_{n_b,k_m}, E^{\mathrm{ID_{RF}}}_{b,m}, E^{\mathrm{BC_{RF}}}_{n_b,m}, E^{\mathrm{Agg_{RF}}}_{n_b,k_m}, E^{\mathrm{OT_{RF}}}_{b,m}
\end{aligned} \tag{5.24}$$

where $\mathbf{x} = \{t^{\mathrm{ID}}_{b,m}, E^{\mathrm{ID_{RF}}}_{b,m}, t^{\mathrm{BC}}_{n_b,m}, E^{\mathrm{BC_{RF}}}_{n_b,m}, t^{\mathrm{Agg}}_{n_b,k_m}, E^{\mathrm{Agg_{RF}}}_{n_b,k_m}, t^{\mathrm{OT}}_{b,m}, E^{\mathrm{OT_{RF}}}_{b,m}, t^{\mathrm{LOC}}_{n_b,k_m}, d^{\mathrm{ch}}_{n_b,k_m},$

$t_{n_b,m}, t_b\}$, $\forall n \in [N_b^{\text{LAY}}]$, $\forall k \in [K_m]$, $\forall b \in [N^{\text{bl}}]$ and $\forall m \in [N^{\text{rel}}]$. The optimization problem can be easily solved via classical convex optimization techniques and Lagrange duality theory. However, a decomposition strategy can be employed to solve the problem independently at different locations without any loss of generality, utilizing the primal-dual decomposition technique as proposed in Section IV-A1 of [67]. The resulting problem formulation aligns with the problem (31) stated in [67]. This decomposition technique entails solving an internal model parallelism optimization problem at each relay in the network, as well as an external data parallelism optimization problem at the DO-RCD. This distributed approach enables the sharing and distribution of the optimization workload among multiple devices, as opposed to being executed solely on a single device.

Given the allocated time $t_{b,m} = \sum_{n=1}^{N_b^{LAY}} t_{n_b,m}$ for each block operation, the internal model parallelism optimization problem is solved for the model parallelism parameters as follows:

$$\begin{aligned}(E_{n_b,m}^{\text{INT}})^* = \min_{\mathbf{x}_{\text{int}}} \quad & E_{n_b,m}^{\text{BC}} + \sum_{k=1}^{K_m} \left(E_{n_b,k_m}^{\text{LOC}} + E_{n_b,k_m}^{\text{Agg}} \right) \\ \text{s.t.} \quad & (5.9), (5.12), (5.15), (5.20), (5.21) \\ & 0 \leq t_{n_b,m}^{\text{BC}}, E_{n_b,m}^{\text{BC}_{\text{RF}}}, t_{n_b,k_m}^{\text{Agg}}, E_{n_b,k_m}^{\text{Agg}_{\text{RF}}}, t_{n_b,k_m}^{\text{LOC}}, d_{n_b,k_m}^{\text{ch}}\end{aligned} \tag{5.25}$$

Here, $(E_{n_b,m}^{\text{INT}})^*$ is the minimized energy consumption for layer-pack n operation in block b for relay m and

$$\mathbf{x}_{\text{int}} = \{t_{n_b,m}^{\text{BC}}, E_{n_b,m}^{\text{BC}_{\text{RF}}}, t_{n_b,k_m}^{\text{Agg}}, E_{n_b,k_m}^{\text{Agg}_{\text{RF}}}, t_{n_b,k_m}^{\text{LOC}}, d_{n_b,k_m}^{\text{ch}}, t_{n_b,m}\} \tag{5.26}$$

where $\forall n \in [N_b^{\text{LAY}}]$, $\forall k \in [K_m]$, $\forall b \in [N^{\text{bl}}]$ and $\forall m \in [N^{\text{rel}}]$. By using the minimized energy consumption $(E_{n_b,m}^{\text{INT}})^*$ values, the external data parallelism optimization problem can be solved as

$$\begin{aligned}\min_{\mathbf{x}_{\text{ext}}} \quad & E_{b,m}^{\text{ID}} + E_{b,m}^{\text{OT}} + (E_{b,m}^{\text{INT}})^* \\ \text{s.t.} \quad & (5.5), (5.18), (5.22), (5.23) \\ & 0 \leq t_{b,m}^{\text{ID}}, E_{b,m}^{\text{ID}_{\text{RF}}}, t_{b,m}^{\text{OT}}, E_{b,m}^{\text{OT}_{\text{RF}}}, t_{b,m}\end{aligned} \tag{5.27}$$

where $(E_{b,m}^{\mathrm{INT}})^* = \sum_{n=1}^{N_b^{\mathrm{LAY}}} (E_{n_b,m}^{\mathrm{INT}})^*$ and

$$\mathbf{x}_{\mathrm{ext}} = \{t_{b,m}^{\mathrm{ID}}, E_{b,m}^{\mathrm{ID}_{\mathrm{RF}}}, t_{b,m}^{\mathrm{OT}}, E_{b,m}^{\mathrm{OT}_{\mathrm{RF}}}, t_{b,m}, t_b\} \tag{5.28}$$

where $\forall b \in [N^{\mathrm{bl}}]$ and $\forall m \in [N^{\mathrm{rel}}]$. The DO-RCD and relays can iteratively exchange the allocated time $t_{b,m}$ and the result of the internal optimization problem $(E_{b,m}^{\mathrm{INT}})^*$ for each block operation until the convergence is reached. In this study, we assume that the number of rows assigned to each relay is equal, i.e., $d_m = d^{\mathrm{row}} / N^{\mathrm{rel}}$.

5.5 Simulation Results

This section presents the simulation results that benchmark the proposed hybrid parallelism approach with relay-assisted communication against other possible scenarios and show the efficacy of the proposed approach in improving the achievable trade-off between communication and computation. Table 5.1 provides an overview of the parameters employed, inspired by the articles [20, 24]. Similar to the previous chapters, the simulation results provided here derive from the average of outcomes obtained from 1000 different random channel realizations.

Throughout the simulation results, the layer-pack block formation scenario is denoted through the vector $[N_1^{\mathrm{LAY}}\ N_2^{\mathrm{LAY}}\ ...\ N_{N^{\mathrm{bl}}}^{\mathrm{LAY}}]$ whose number of entries is equal to the number of blocks N^{bl}. Notably, communication costs in the figures include the overall energy consumption of all communication-related stages, including input distribution, broadcasting, aggregation, and output transmission, while computation costs are the overall energy consumption of collaborative local computation among all C-RCDs.

5.5.1 Comparison of the Proposed Approach With Benchmark Scenarios

In this subsection, the proposed collaborative approach with hybrid parallelism and relay-assisted communication is compared to other benchmark non-collaborative and collaborative scenarios. The integration of relay-assisted communication, hybrid parallelism, and layer-pack block formation proves to be essential in achieving superior performance when compared to other benchmark scenarios.

In Fig. 5.5, the proposed collaborative, hybrid parallelism, and relay-assisted CNN inference approach is compared to a scenario where the input data

TABLE 5.1: Selected parameters for the simulation results of Chapter 5

Parameters	Values	Units
κ_{k_m}	Unif$([10^{-28}; 10^{-27}])$	/
c_{k_m}	Unif$([150; 350])$	CPU cycles/MAC
$\nu_{k_m}^{\max}$	Unif$([1; 3])$	GHz
h(Each channel gain)	$\mathcal{CN}(0, 10^{-3})$(Rayleigh)	/
$P_{\tilde{k}}^c$ & P_m^c & $P_{k_m}^c$	Unif$([10; 25])$	mW
B	2	MHz
N_0	0.5×10^{-11}	W/MHz
f_c	2.1	GHz
α	2.5	/
$R_{b,m}^{\text{ID}}$ & $R_{b,m}^{\text{OT}}$	Unif$([100; 150])$	m
R_{n_b,k_m}^{BC} & R_{n_b,k_m}^{Agg}	Unif$([10; 20])$	m
d^{row}	384	bits
d^{col}	128	bits
$d_{n_b}^{\text{ch}}$	16	/

is entirely offloaded to a powerful edge server (ES) by the DO-RCD. This scenario comprises three stages: the transmission of the entire input data to the ES by the DO-RCD, model computation, and output transmission to the DO-RCD by the ES. The parameters of ES are selected as $\kappa_{\text{ES}} = \text{Unif}([10^{-28}; 10^{-27}])$, $P_{\text{ES}}^c = \text{Unif}([10; 25])$ mW, $c_{\text{ES}} = \text{Unif}([55; 75])$ CPU cycles/MAC and $\nu_{\text{ES}}^{\max} = \text{Unif}([3; 5])$ GHz to guarantee a higher computing capability compared to the RCDs. The scenarios are evaluated for different network coverage distances where the coverage of the proposed approach extends from the DO-RCD to the relays, while the ES scenario coverage extends from the DO-RCD to the ES. As illustrated in Fig. 5.5, collaborative model execution across multiple RCDs significantly reduces energy consumption in comparison to the ES scenario across all network coverage distances, highlighting the efficiency of collaborative and parallel CNN model inference among multiple RCDs.

Fig. 5.6 illustrates the comparison in terms of energy consumption between the proposed collaborative approach and three other collaborative scenarios. The first benchmark scenario adopts data parallelism with multiple computing segments. In this scenario, each relay within each computing segment chooses a single C-RCD that consumes the minimum energy among other C-RCDs in the same segment. As a result of a lack of model partitioning, this selected C-RCD must carry out all filter computations for

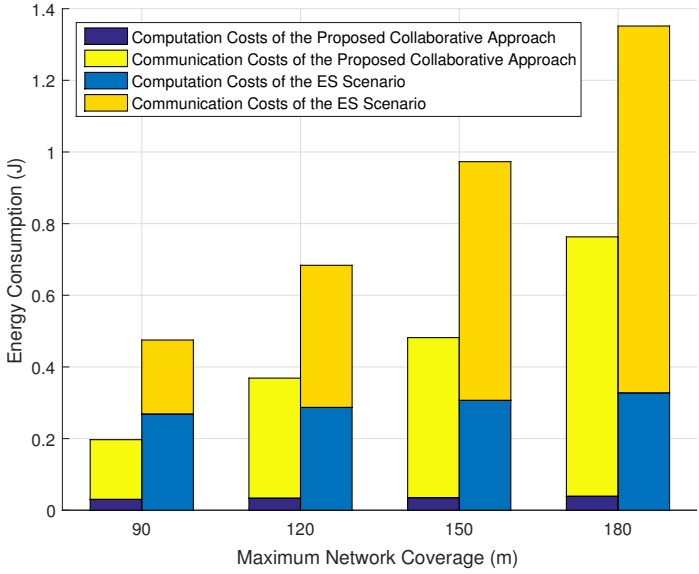


FIGURE 5.5: Comparison of the proposed approach with the ES scenario for $K_m = 8$, $N^{\text{rel}} = 4$, $N_b^{\text{LAY}} = [1\ 2]$, and $\tau = 0.5$ s

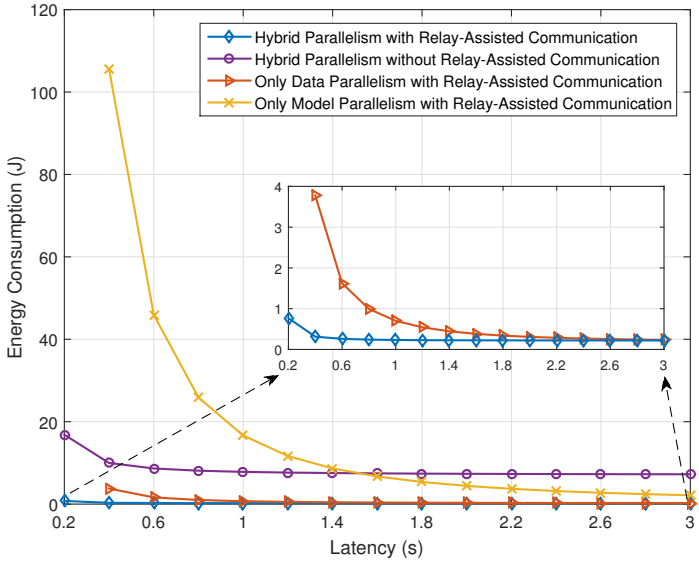


FIGURE 5.6: Comparison of the proposed approach with different collaborative scenarios for the total number of C-RCDs $K = 20$, and $N^{\text{LAY}} = 3$

each layer-pack operation. The parameters selected for this scenario are the number of relays $N^{\text{rel}} = 3$, the layer-pack distribution $N_b^{\text{LAY}} = [1\ 2]$ and the number of blocks $N^{\text{bl}} = 2$. The second scenario employs only model parallelism, where the complete input data is transmitted to the computing segment that consumes the minimum energy among other computing segments. Due to the absence of data partitioning, only one computing segment is involved in the collaborative computation. This computing segment comprises a single relay and multiple C-RCDs, the same as the proposed approach. Since there is no data parallelism, no layer-pack block formation exists in this scenario. In the third scenario, hybrid parallelism with both model and data parallelism is considered without relay-assisted communication to compare with the proposed approach and investigate the performance boost achieved by having relay-assisted communication. In this scenario, partial input data is transmitted point-to-point to each C-RCD in each computing segment, and the output data of each layer-pack operation is fully transmitted back to the DO-RCD due to the absence of relays and layer-block formation. All these scenarios are visually represented in Fig. 5.7. Also, the system workflows of all benchmark scenarios are explained step by step in Tables 5.2 and 5.3 where the communication and computation stages are highlighted with blue and red font colors, respectively.

Based on the results presented in Fig. 5.6, it can be inferred that hybrid parallelism plays an important role in meeting the low latency requirements, particularly for $\tau < 0.4$ s. In instances where low latency is essential, the data parallelism scenario suffers due to the lack of model partitioning since all filter computations are handled by a single C-RCD, whereas the model parallelism scenario fails to satisfy the computational demands of large input data, which are sent and allocated to a single computing segment. For high latency values, the data parallelism scenario exhibits a performance that is nearly equivalent to the proposed approach since each C-RCD has sufficient time to overcome its computational workload. The third scenario to compare with the proposed approach considers hybrid parallelism without relay-assisted communication. For low latency values below 0.4 s, this scenario incurs an energy consumption that is approximately 25 times higher than that of the proposed approach, while a noticeable difference in energy consumption persists even for higher latencies. This observation underscores the significance of incorporating relay-assisted communication into a hybrid parallelism system. In the absence of relays, a large number of point-to-point communication stages would be required to transmit partial input data from the DO-RCD to multiple C-RCDs within each segment. Moreover, the output data of each layer-pack operation is transmitted directly to the DO-RCD, which may be located far

TABLE 5.2: System workflow of each parallelism technique for each block operation

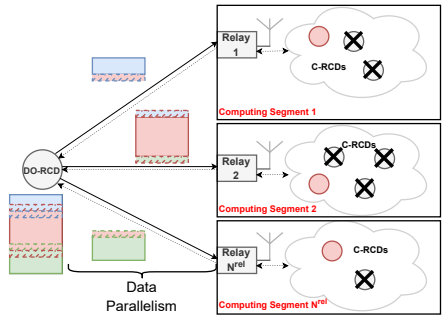
	Only Data Parallelism	**Only Model Parallelism**	**Hybrid Paralellism**
Input Distribution	Partial input data is sent from the DO-RCD to its assigned computing segment	Entire input data is sent from the DO-RCD to the computing segment that consumes the minimum energy among other computing segments.	Partial input data is sent from the DO-RCD to its assigned computing segment.
Broadcasting	Each relay transmits partial input data to the selected C-RCD.	Each relay broadcasts partial input data to its C-RCDs.	Each relay broadcasts partial input data to its C-RCDs.
Local Computation	The selected C-RCD carries out all computation by itself.	All C-RCDs perform collaborative local computation with their assigned filters.	All C-RCDs perform collaborative local computation with their assigned filters.
Aggregation	The selected C-RCD transmits the output data back to its relay.	All C-RCDs transmit the output data back to their relay.	All C-RCDs transmit the output data back to their relay.
Repetition	Process is repeated from broadcasting stage for each layer-pack operation in the block.	Process is repeated from broadcasting stage for each layer-pack operation in the block.	Process is repeated from broadcasting stage for each layer-pack operation in the block.
Output Transmission	The final output data of each block operation is sent back to the DO-RCD by each relay.	The final output data of each block operation is sent back to the DO-RCD by the relay.	The final output data of each block operation is sent back to the DO-RCD by each relay.

• Communication stages • Computation stages

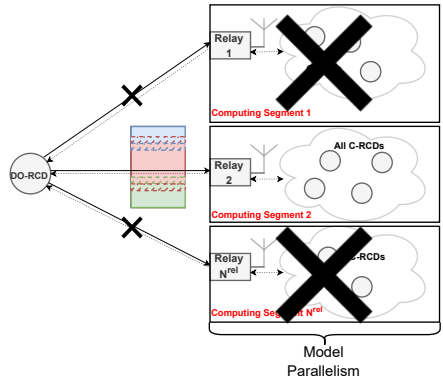
TABLE 5.3: System workflow of the scenarios with and without relay deployment

	Without Relay-assisted Communication	**With Relay-assisted Communication**
Input Distribution	Partial input data is sent point-to-point from the DO-RCD to multiple C-RCDs separately.	Partial input data is sent from the DO-RCD to its assigned computing segment.
Broadcasting	Not applicable.	Each relay broadcasts partial input data to its C-RCDs.
Local Computation	All C-RCDs perform collaborative local computation with their assigned filters.	All C-RCDs perform collaborative local computation with their assigned filters.
Aggregation	All C-RCDs transmit the output data back to the DO-RCD.	All C-RCDs transmit the output data back to their relay.
Repetition	Process is repeated from input distribution stage for each layer-pack operation.	Process is repeated from broadcasting stage for each layer-pack operation in the block.
Output Transmission	Not applicable.	The final output data of each block operation is sent back to the DO-RCD by each relay.

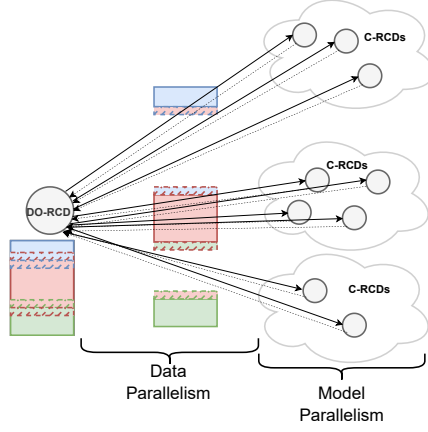
• Communication stages • Computation stages



(A) Only Data Parallelism Scenario with Relay-Assisted Communication



(B) Only Model Parallelism Scenario with Relay-Assisted Communication



(C) Hybrid Parallelism Scenario without Relay-Assisted Communication

FIGURE 5.7: Different benchmark collaborative scenarios

from the C-RCDs, thereby destroying the advantages of layer-pack block formation.

5.5.2 Impact of Various Block Formation Configurations on Energy Consumption

In this subsection, we present the impact of various block formation configurations on both communication and computation energy consumption in two distinct scenarios. The simulation results for this impact are depicted in Fig. 5.8 for the scenario where no optimization has been applied to the workload, communication, and computation parameters. In this setup, the partial input data size is assumed to be distributed equally among all computing segments, along with an equal allocation of filters to each C-RCD within the same computing segment. Furthermore, the maximum allowed latency τ is uniformly distributed to each energy stage, including input distribution, broadcasting, local computation, aggregation, and output transmission. Fig. 5.9 showcases the simulation results regarding the impact of block formation configuration for our approach, where the proposed joint optimization effect is included.

When the model consists of three blocks, with each layer-pack represented as an individual block, it leads to higher energy consumption. This is attributed to the increased communication that occurs after each layer-pack between the DO-RCD and the relays. Conversely, if a single block consisting of all layer-packs exists in the model, i.e., the 1-block scenario, the computation costs escalate due to the exponential increase in the amount of redundant row computation. The best scenario is to have two blocks, with the layer-pack distribution [1 2]. As such, the block formation configuration significantly affects the communication-computation trade-off, necessitating its consideration for energy consumption optimization.

Upon comparing the simulation results between our proposed approach and the non-optimized scenario, a huge reduction in energy consumption is observed compared to the non-optimized scenario when the optimization is applied to the system. Also, as seen in Fig. 5.9, the performance gap between different block formation configurations is minimized for the proposed approach so that the system can adapt to different block formation configurations easily.

Our findings in this subsection underscore the critical role of block formation configuration in shaping energy consumption dynamics. Choosing the best block formation configuration can lead to considerable energy savings.

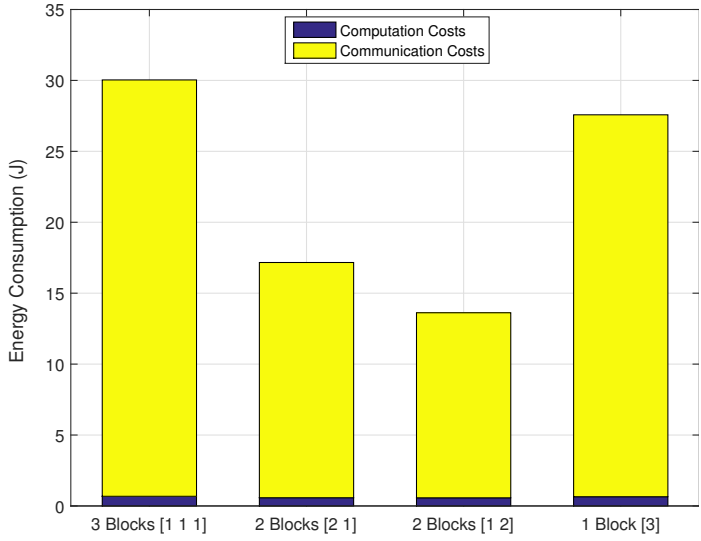


FIGURE 5.8: Energy consumption of various block formation configurations for the non-optimized scenario for $N^{\mathrm{LAY}} = 3$, $N^{\mathrm{rel}} = 3$, $K_m = 8$, and $\tau = 0.3$ s

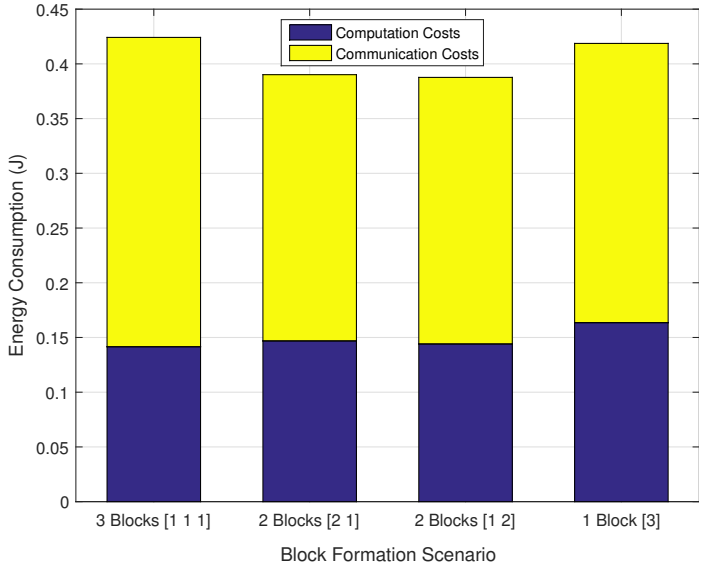


FIGURE 5.9: Energy consumption of various block formation configurations for the proposed approach for $N^{\mathrm{LAY}} = 3$, $N^{\mathrm{rel}} = 3$, $K_m = 8$, and $\tau = 0.3$ s

5.5.3 Impact of the Number of C-RCDs per Relay on Energy Consumption

Fig. 5.10 investigates the impact of the number of C-RCDs per relay K_m on the energy consumption, where the number of relays is fixed at $N^{\text{rel}} = 3$. Increasing the number of C-RCDs per relay leads to achieving more energy efficiency until a certain point, as the computation costs are the primary factor in the energy consumption for a low number of C-RCDs due to the lack of enough computing capability of the system. However, energy consumption begins to rise for more than 8 C-RCDs per relay. The reason for this observation is that after this point, the addition of more C-RCDs to each computing segment brings additional communication costs, which start to dominate over what the system gains from the computation costs. This finding emphasizes the significance of identifying the best trade-off point between communication and computation. Leveraging communication opportunities enables obtaining more computing capabilities until a specific trade-off point. However, requesting more collaboration beyond this point is inefficient since communication costs start to outweigh computation costs.

5.5.4 Impact of the Number of Computing Segments on Energy Consumption

Fig. 5.11 presents an alternative scenario where the number of computing segments is varied to investigate the impact on the energy consumption of the system. Each computing segment is composed of a single relay and 8 C-RCDs. For a small number of computing segments, the computation costs are relatively high compared to the communication costs due to the high input data size per computing segment and the low total computing capability of the system. Adding a computing segment reduces the input data size per computing segment and introduces additional computing capability to the system, resulting in an exponential decrease in computation costs at the price of increased communication costs due to the increased amount of communication between the DO-RCD and the growing number of computing segments. Hence, there is a specific point beyond which adding more segments does not achieve a more energy-efficient system despite decreasing computation costs.

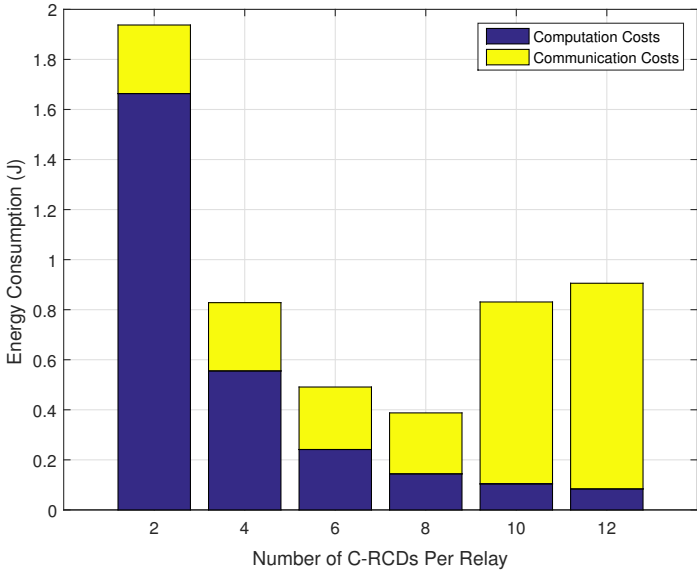


FIGURE 5.10: Energy consumption of communication and computation vs. number of C-RCDs per relay (K_m) for $N_b^{\mathrm{LAY}} = [1\ 2]$, $N^{\mathrm{rel}} = 3$, and $\tau = 0.3$ s

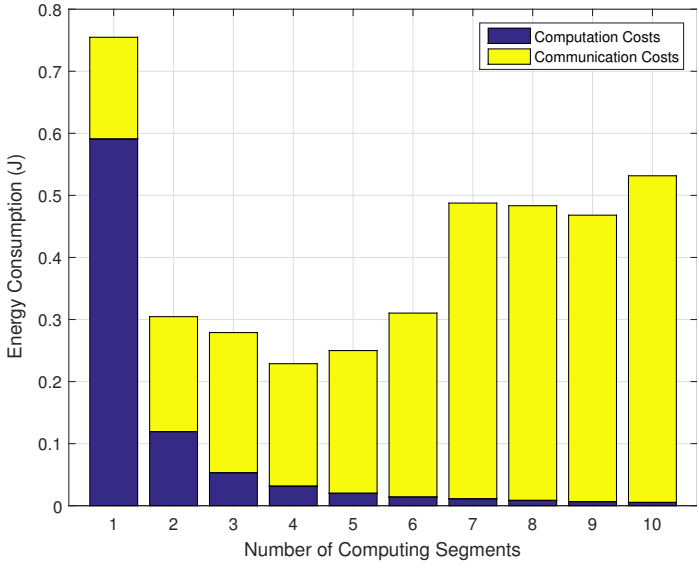


FIGURE 5.11: Energy consumption of communication and computation vs. number of computing segments for $K_m = 8$, $N_b^{\mathrm{LAY}} = [1\ 2]$, and $\tau = 0.5$ s

5.5.5 Effect of the Number of Relays and Latency on Energy Consumption

In this subsection, we show the impact of varying the number of relays and latency on energy consumption and investigate how different number of relays and latency values affect both communication and computation costs in the system. The results reveal valuable insights into the trade-off between communication and computation, leading to the identification of energy-efficient configurations for diverse scenarios.

Fig. 5.12 shows the effect of the number of relays on the energy consumption when the total number of C-RCDs in the system is kept constant, meaning that the total computing capability of the system remains unchanged regardless of the number of relays. In this case, the total number of C-RCDs is assumed to be evenly shared among multiple computing segments, each with a single relay. For a low number of relays, the communication costs are dominated by the high data size that must be exchanged between the DO-RCD and relays during data parallelism and between relays and C-RCDs during model parallelism. For a high number of relays, the communication costs are dominated mainly by the increased amount of communication between the DO-RCD and multiple relays during the ID and OT stages. Changing the number of relays does not significantly affect the computation costs, as the total computing capability of the system does not change.

In Fig. 5.13, we investigate the effect of latency on the energy consumption with several number of relay N^{rel} values while keeping the number of C-RCDs per relay fixed at $K_m = 8$, $\forall m \in [N^{\text{rel}}]$. For low latency scenarios, where computation costs dominate, having more relays is beneficial as it reduces the workload on each relay and its associated C-RCDs, as shown for $\tau = 0.2$ s, where the best-performing case is the one with $N^{\text{rel}} = 6$. As the latency increases, having fewer relays is more energy-efficient since computation costs begin to decrease, and communication costs become the dominant factor in high allowed latency scenarios. As seen in the figure, for high latencies, the best performing cases are the cases where $N^{\text{rel}} = 2$ or $N^{\text{rel}} = 4$.

Fig. 5.14 illustrates the communication and computation costs of different number of relay values for three distinct latency scenarios for the fixed number of C-RCDs per relay $K_m = 8$. The results show that an increase in the number of relays N^{rel} leads to a consistent reduction in the computation costs, as when the number of relays is increased, more C-RCDs are introduced to the system, enhancing the overall computing capability. In

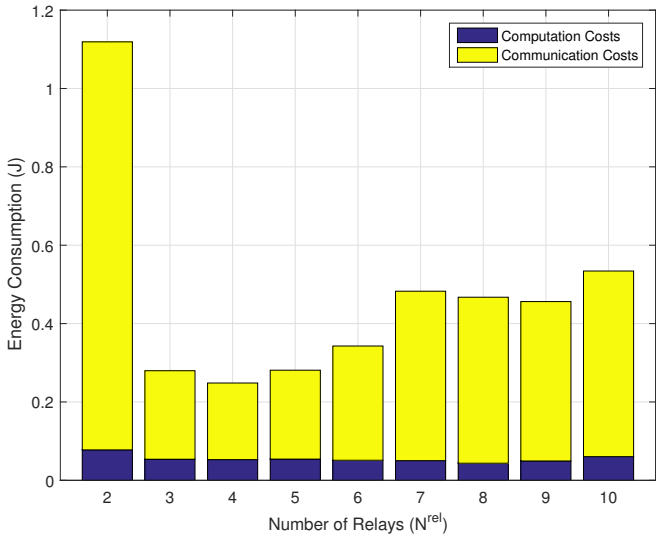


FIGURE 5.12: Energy consumption of communication and computation vs. number of relays for the fixed total number of C-RCDs $K = 24$, $N_b^{\mathrm{LAY}} = [1\ 2]$, and $\tau = 0.5$ s

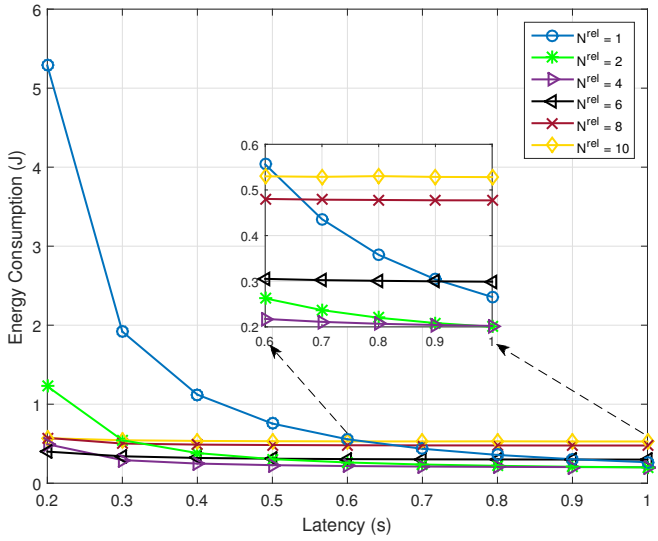


FIGURE 5.13: Energy consumption of communication and computation vs. latency for the fixed number of C-RCDs per relay $K_m = 8$, and $N_b^{\mathrm{LAY}} = [1\ 2]$

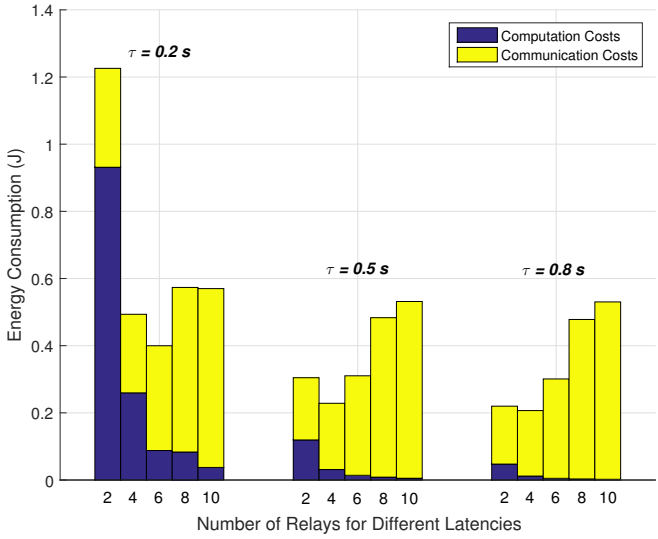


FIGURE 5.14: Energy consumption of communication and computation vs. number of relays for different latency scenarios for the fixed number of C-RCDs per relay $K_m = 8$, and $N_b^{\text{LAY}} = [1\ 2]$

return, the communication costs increase due to the communication overhead between the DO-RCD and relays during the ID and OT stages. These issues together establish the best trade-off point between communication and computation for the number of relay values, which is seen as $N^{\text{rel}} = 6$ for low latency values and $N^{\text{rel}} = 4$ for mid and high latency values.

In Fig. 5.15, a scenario similar to Fig. 5.13 is presented, where the total number of C-RCDs K is fixed at 20 instead of the number of C-RCDs per relay K_m, meaning that the total computing capability of the system is not affected with the number of relays and latency. The figure reveals that in contrast to the scenario depicted in Fig. 5.13, the best number of relays N^{rel} does not change over all latency scenarios. The communication and computation costs for this scenario, with three different latency values, are presented separately in Fig. 5.16. Unlike Fig. 5.14, the computation costs after $N^{\text{rel}} = 4$ start to increase. This is because adding more relays to the system after a certain point results in a smaller number of C-RCDs per relay, causing the system to have insufficient computing capability for model parallelism in each computing segment. As a result, the system consumes more energy during computation. Meanwhile, communication costs continue to rise with additional relays due to the increased ID and OT communication between the DO-RCD and multiple relays.

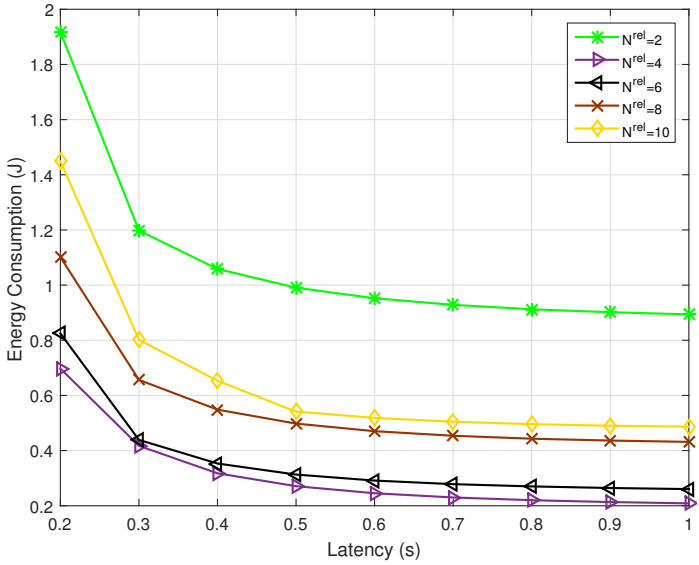


FIGURE 5.15: Overall energy consumption vs. latency for the fixed total number of C-RCDs $K = 20$, and $N_b^{\mathrm{LAY}} = [1\ 2]$

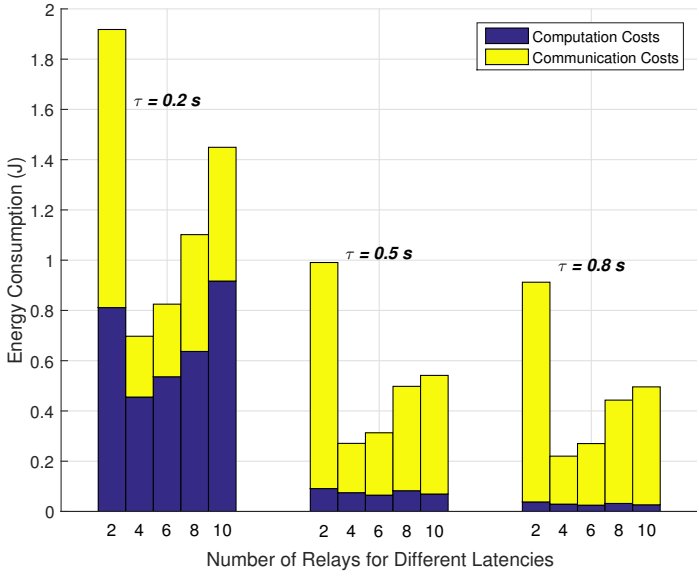


FIGURE 5.16: Energy consumption of communication and computation vs. number of relays for the fixed total number of C-RCDs $K = 20$, $N_b^{\mathrm{LAY}} = [1\ 2]$

5.6 Summary

This study proposes a relay-assisted and collaborative on-device execution scheme for CNNs in deep learning-driven applications. Our approach leverages hybrid parallelism, combining data and model parallelism, to optimize collaborative CNN inference on RCDs. To address the communication overhead associated with hybrid parallelism, we employ relay-assisted communication and layer block formation techniques. These strategies reduce the input data size per RCD and minimize point-to-point communication between the data owner RCD and collaborative RCDs. Our approach aims to find the best trade-off point between communication and computation for energy consumption minimization by formulating a convex optimization problem, which optimizes the assignment of each filter in each layer, as well as communication and computation parameters. We also explore a decomposed approach to solve the problem in a distributed manner. Our proposed approach holds significant potential for enhancing the efficiency and performance of collaborative CNN inference.

The results of this study provide important insights for improving energy efficiency in the system. The suggested relay-assisted hybrid parallelism method performs better than other benchmark scenarios, such as solely relying on data parallelism, only using model parallelism, and employing hybrid parallelism without relay-assisted communication. This superior performance is observed particularly in scenarios with large input data sizes, low latency, and high model complexity. Notably, data parallelism struggles with high model complexity, while model parallelism faces challenges with large input data sizes. The inefficiency of the system is evident when using hybrid parallelism without relay assistance due to excessive communication between the data owner device and other participating devices. Additionally, our findings align with those of a previous study (Chapter 4), emphasizing the trade-off between communication and computation with respect to the number of computing segments. Initially, increased system collaborativeness is advantageous for energy efficiency until a certain threshold, where computation consumption becomes the dominant factor in scenarios with a low number of computing segments. However, beyond this threshold, further increasing the number of computing segments does not result in additional benefits, as communication consumption begins to dominate overall energy consumption.

In the upcoming chapter, we extend the proposed relay-assisted hybrid parallelism approach for collaborative CNN inference, considering the impact of interference between devices.

Chapter 6

Interference-Aware Relay-Assisted Collaborative CNN Inference with Hybrid Parallelism

In this chapter, we propose an interference-aware energy-efficient relay-assisted collaborative CNN inference architecture based on a predefined signal-to-interference-noise ratio (SINR) target SINR_T by employing hybrid parallelism, integrating data and model parallelism. The proposed system model is an extension of the one presented in Chapter 5, comprising a data owner RCD responsible for the input data and multiple computing segments, each equipped with a relay and multiple C-RCDs. The main objective of this chapter is to analyze the impact of interference between relays and C-RCDs across various computing segments on the overall energy consumption during collaborative CNN inference. We also demonstrate that the predefined SINR_T value significantly impacts the trade-off point between communication and computation. In our simulations, we conduct a comprehensive performance analysis, comparing our approach against several benchmark scenarios with the assumption 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 study include a comprehensive system architecture for collaborative CNN inference, incorporating an

interference-aware relay-assisted paradigm and hybrid parallelism. Moreover, we investigate how interference among different devices influences the energy consumption trade-off between communication and computation.

6.1 Recap for Relay-Assisted Collaborative CNN Inference with Hybrid Parallelism

In this section, we recap the relay-assisted collaborative CNN inference scheme with hybrid parallelism, introduced in the study in Chapter 5. In this chapter, we extend this relay-assisted approach to encompass an interference-aware scenario, where we integrate SINR targets into the communication model.

The system model for relay-assisted collaborative CNN inference, depicted in Fig. 5.4, includes the DO-RCD with the input data, N^{rel} number of computing segments, each of which contains a relay and K_m number of C-RCDs assigned to computing segment m.

The system workflow for each block execution entails:

- Pre-training the CNN model on powerful cloud servers using offline datasets.
- Dividing input data into N^{rel} pieces by the DO-RCD before the first block, sent to relays along with d_1^R number of redundant rows where d_b^R is the number of redundant rows that must be additionally sent to each relay for block b operation. 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 operations. In the final block, the entire output data is sent to complete collaborative CNN inference.
- DO-RCD collects boundary rows, and the process continues for the next block until the last block operation.

The subsequent sections go into the details of the extended interference-aware relay-assisted collaborative CNN inference architecture.

6.2 Energy Consumption Modeling

In this section, we break down the energy consumption into three distinct parts for each block, which are defined and explained in the following subsections.

6.2.1 Input Distribution (ID)

The energy consumption modeling of the input distribution of this study is identical to the modeling of the study in Chapter 5. To recap, during the input distribution stage, the DO-RCD communicates with each relay m to send the input data required for block b operation. The achievable data rate $r_{b,m}^{\mathrm{ID}}$ for this communication can be expressed as

$$r_{b,m}^{\mathrm{ID}} = B\ \ln(1 + \frac{p_{b,m}^{\mathrm{ID}} \frac{h_{b,m}^{\mathrm{ID}}}{l(R_{b,m}^{\mathrm{ID}})}}{BN_0}) \tag{6.1}$$

Here, $p_{b,m}^{\mathrm{ID}}$ 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, $h_{b,m}^{\mathrm{ID}}$ and $R_{b,m}^{\mathrm{ID}}$ are the channel gain and distance between DO-RCD and relay m for block b operation during ID, respectively.

The energy consumption of ID for relay m during block b operation is given by

$$E_{b,m}^{\mathrm{ID}} = t_{b,m}^{\mathrm{ID}}(p_{b,m}^{\mathrm{ID}} + P_{\widetilde{k}}^{c}) \tag{6.2}$$

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

$$d_{b,m}^{\mathrm{ID}} \leq t_{b,m}^{\mathrm{ID}} r_{b,m}^{\mathrm{ID}} \tag{6.3}$$

where

$$d_{b,m}^{\mathrm{ID}} = \begin{cases} (d_m + I_m d_1^R) d^{\mathrm{col}} & \text{if } b = 1 \\ I_m d_b^R d_b^{\mathrm{col}} & \text{otherwise} \end{cases} \tag{6.4}$$

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

6.2.2 Model Parallelism

After receiving the input data from the DO-RCD, the relay devices take on the responsibility of managing the model parallelism part 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.

Broadcasting (BC)

During the broadcasting stage, each relay broadcasts its partial input data for layer-pack n operation of block b to the participating C-RCDs. Interference may occur between relays in different computing segments. The SINR for this broadcasting is given by

$$\mathrm{SINR}^{\mathrm{BC}}_{n_b,m} = \frac{p^{\mathrm{BC}}_{n_b,m} \min_k \left(\frac{h^{\mathrm{BC}}_{n_b,k_m}}{l(R^{\mathrm{BC}}_{n_b,k_m})} \right)}{BN_0 + \sum\limits_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} p^{\mathrm{BC}}_{n_b,m'} \frac{h^{\mathrm{BC}}_{n_b,m,m'}}{l(R^{\mathrm{BC}}_{n_b,m,m'})}} \geq \mathrm{SINR}_{\mathrm{T}} \tag{6.5}$$

where a desired SINR target $\mathrm{SINR}_{\mathrm{T}}$ is given to the system. Here, $p^{\mathrm{BC}}_{n_b,m}$ is the RF transmit power of broadcasting in relay m, $h^{\mathrm{BC}}_{n_b,k_m}$ and $R^{\mathrm{BC}}_{n_b,k_m}$ are the channel gain and distance between relay m and C-RCD k during layer-pack n operation of block b, respectively.

The energy consumption of broadcasting by relay m for layer-pack n operation of block b is calculated as

$$E^{\mathrm{BC}}_{n_b,m} = t^{\mathrm{BC}}_{n_b,m}(p^{\mathrm{BC}}_{n_b,m} + P^{c}_{m}) \tag{6.6}$$

where P^{c}_{m} and $t^{\mathrm{BC}}_{n_b,m}$ are the constant power consumption of the communication circuitry and the time taken for broadcasting by relay m during the operation of layer-pack n in block b, respectively.

The broadcasting time $t^{\mathrm{BC}}_{n_b,m}$ is subject to the following constraint:

$$\frac{d^{\mathrm{BC}}_{n_b,m}}{B \ln(1 + \mathrm{SINR}_{\mathrm{T}})} \leq t^{\mathrm{BC}}_{n_b,m} \tag{6.7}$$

and

$$d^{\mathrm{BC}}_{n_b,m} = \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^{n-1})}{2^{n-1}} \right) \frac{d^{\mathrm{col}}_b}{2^{n-1}} \tag{6.8}$$

where $d^{\mathrm{BC}}_{n_b,m}$ is the broadcasted number of bits, $d^{\mathrm{col}}_b = \frac{d^{\mathrm{col}}}{2^{\sum_{b'=1}^{b-1} N^{LAY}_{b'}}}$ 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.

Collaborative Local Computation

After receiving the input data, each C-RCD performs its local computation. This study emphasizes the energy consumption of convolutional layers, as they predominantly contribute to energy usage over other layers [36, 71], as stated in the other studies in Chapters 3, 4, 5.

The number of MAC operations $N^{\mathrm{MAC}}_{n_b,k_m}$ carried out by C-RCD k of relay m for layer-pack n operation of block b is given by

$$\begin{aligned} N^{\mathrm{MAC}}_{n_b,k_m} &= (d^{\mathrm{filter}})^2 \left(\frac{d_{b,m} + I_m(d^R_b + 1 - 2^n)}{2^{n-1}} \right) \frac{d^{\mathrm{col}}_b}{2^{n-1}} d^{\mathrm{ch}}_{n_b,k_m} \\ &= \alpha_{n_b,m} d^{\mathrm{ch}}_{n_b,k_m} \end{aligned} \tag{6.9}$$

where $d^{\mathrm{filter}} = 3$ 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 the low CPU voltage assumption [73–76] is

$$E^{\mathrm{LOC}}_{n_b,k_m} = \frac{\kappa_{k_m} c^3_{k_m} (N^{\mathrm{MAC}}_{n_b,k_m})^3}{(t^{\mathrm{LOC}}_{n_b,k_m})^2} = \kappa_{k_m} c^3_{k_m} (\alpha_{n_b,m})^3 \frac{(d^{\mathrm{ch}}_{n_b,k_m})^3}{(t^{\mathrm{LOC}}_{n_b,k_m})^2} \tag{6.10}$$

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

$$c_{k_m} N^{\mathrm{MAC}}_{n_b,k_m} = c_k \alpha_{n_b,m} d^{\mathrm{ch}}_{n_b,k_m} \leq \nu^{\max}_{k_m} t^{\mathrm{LOC}}_{n_b,k_m} \tag{6.11}$$

This constraint sets 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.

Aggregation

In this stage, each C-RCD k transmits the summation of the output data of its assigned filters for layer-pack n operation to its corresponding relay m, adhering to the following criterion for SINR:

$$\mathrm{SINR}^{\mathrm{Agg}}_{n_b,k_m} = \frac{p^{\mathrm{Agg}}_{n_b,k_m}\left(\frac{h^{\mathrm{Agg}}_{n_b,k_m}}{l(R^{\mathrm{Agg}}_{n_b,k_m})}\right)}{\frac{B}{K_m}N_0 + \sum\limits_{\substack{m'=1\\ m'\neq m}}^{N^{\mathrm{rel}}} p^{\mathrm{Agg}}_{n_b,k_{m'}}\frac{h^{\mathrm{Agg}}_{n_b,k_m,k_{m'}}}{l(R^{\mathrm{Agg}}_{n_b,k_m,k_{m'}})}} \geq \mathrm{SINR}_{\mathrm{T}} \tag{6.12}$$

where $p^{\mathrm{Agg}}_{n_b,k_m}$ is the RF transmit power of aggregation in C-RCD k of relay m, $h^{\mathrm{Agg}}_{n_b,k_m}$ and $R^{\mathrm{Agg}}_{n_b,k_m}$ 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 operation of block b is

$$E^{\mathrm{Agg}}_{n_b,k_m} = t^{\mathrm{Agg}}_{n_b,k_m}(p^{\mathrm{Agg}}_{n_b,k_m} + P^{c}_{k_m}) \tag{6.13}$$

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

$$\frac{d^{\mathrm{Agg}}_{n_b,k_m}}{\frac{B}{K_m}\ln(1+\mathrm{SINR}_{\mathrm{T}})} \leq t^{\mathrm{Agg}}_{n_b,k_m} \tag{6.14}$$

where

$$d^{\mathrm{Agg}}_{n_b,k_m} = \left(\frac{d_{b,m} + I_m(d^{R}_{b} + 1 - 2^{n})}{2^{n}}\right)\frac{d^{\mathrm{col}}_{b}}{2^{n}} \tag{6.15}$$

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

6.2.3 Output Transmission (OT)

Following the model parallelism stage 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 the SINR value $\mathrm{SINR}^{\mathrm{OT}}_{b,m}$ for the communication between relay m and DO-RCD

throughout OT is

$$\mathrm{SINR}_{b,m}^{\mathrm{OT}} = \frac{p_{b,m}^{\mathrm{OT}} \left(\frac{h_{b,m}^{\mathrm{OT}}}{l(R_{b,m}^{\mathrm{OT}})} \right)}{BN_0 + \sum\limits_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} p_{b,m'}^{\mathrm{OT}} \frac{h_{b,m,m'}^{\mathrm{OT}}}{l(R_{b,m,m'}^{\mathrm{OT}})}} \geq \mathrm{SINR}_{\mathrm{T}} \tag{6.16}$$

where $p_{b,m}^{\mathrm{OT}}$ is the RF transmit power of relay m, $h_{b,m}^{\mathrm{OT}}$ and $R_{b,m}^{\mathrm{OT}}$ are the channel gain and distance between relay m and the DO-RCD for block b operation during OT, respectively. Energy consumed for OT by relay m during block b operation is

$$E_{b,m}^{\mathrm{OT}} = t_{b,m}^{\mathrm{OT}} (p_{b,m}^{\mathrm{OT}} + P_m^c) \tag{6.17}$$

where $t_{b,m}^{\mathrm{OT}}$ is the transmission time for relay m and block b operation. The constraint of this stage for the transmission time $t_{b,m}^{\mathrm{OT}}$ is

$$\frac{d_{b,m}^{\mathrm{OT}}}{B \ln(1 + \mathrm{SINR}_{\mathrm{T}})} \leq t_{b,m}^{\mathrm{OT}} \tag{6.18}$$

and

$$d_{b,m}^{\mathrm{OT}} = \begin{cases} \frac{d_{b,m}}{2^{N_b^{\mathrm{LAY}}}} \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{if } b = N^{\mathrm{bl}} \\ I_m d_{b+1}^R \frac{d_b^{\mathrm{col}}}{2^{N_b^{\mathrm{LAY}}}} & \text{otherwise} \end{cases} \tag{6.19}$$

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

6.3 Additional Constraints

The system imposes several crucial additional constraints:

1. Filter Channel Distribution Constraint:

$$\sum_{k=1}^{K_m} d_{n_b,k_m}^{\mathrm{ch}} = d_{n_b}^{\mathrm{ch}} \tag{6.20}$$

This constraint ensures that the filters are properly distributed among the C-RCDs for each layer-pack operation within a block.

2. Timing Constraint for Layer-Pack Operations:

$$t^{\mathrm{BC}}_{n_b,m} + t^{\mathrm{LOC}}_{n_b,k_m} + t^{\mathrm{Agg}}_{n_b,k_m} \leq t_{n_b,m} \tag{6.21}$$

where $t_{n_b,m}$ denotes the allocated time for layer-pack n operation of block b for relay m. This constraint ensures that the time allocated for broadcasting, local computation, and aggregation within a layer-pack operation does not exceed the total time available for that operation.

3. Timing Constraint for Block Operations:

$$t^{\mathrm{ID}}_{b,m} + \sum_{n=1}^{N^{\mathrm{LAY}}_b} t_{n_b,m} + t^{\mathrm{OT}}_{b,m} \leq t_b \tag{6.22}$$

where t_b is the allocated time for block b. This constraint ensures that the time allocated for input distribution, layer-pack operations, and output transmission within a block does not exceed the total time available for that block.

4. Overall Latency Constraint:

$$\sum_{b=1}^{N^{\mathrm{bl}}} t_b \leq \tau \tag{6.23}$$

This constraint enforces the overall system latency requirement, ensuring that the total execution time for all blocks does not exceed the specified system latency, denoted as τ.

6.4 Problem Formulation

To transform the problem into a convex optimization problem, inspired by our previous studies in Chapters 3, 4 and 5, new parameters are introduced: $E^{\mathrm{ID}_{\mathrm{RF}}}_{b,m} = p^{\mathrm{ID}}_{b,m} t^{\mathrm{ID}}_{b,m}$, $E^{\mathrm{BC}_{\mathrm{RF}}}_{n_b,m} = p^{\mathrm{BC}}_{n_b,m} t^{\mathrm{BC}}_{n_b,m}$, $E^{\mathrm{Agg}_{\mathrm{RF}}}_{n_b,k_m} = p^{\mathrm{Agg}}_{n_b,k_m} t^{\mathrm{Agg}}_{n_b,k_m}$ and $E^{\mathrm{OT}_{\mathrm{RF}}}_{b,m} = p^{\mathrm{OT}}_{b,m} t^{\mathrm{OT}}_{b,m}$, representing RF energy consumption in various communication stages.

The final convex optimization problem can be formally described as follows:

$$\min_{\mathbf{x}} \quad \sum_{b=1}^{N^{\mathrm{bl}}} \sum_{m=1}^{N^{\mathrm{rel}}} \left(E^{\mathrm{ID}}_{b,m} + E^{\mathrm{OT}}_{b,m} + \sum_{n=1}^{N_b^{\mathrm{LAY}}} \left(E^{\mathrm{BC}}_{n_b,m} + \sum_{k=1}^{K_m} \left(E^{\mathrm{LOC}}_{n_b,k_m} + E^{\mathrm{Agg}}_{n_b,k_m} \right) \right) \right)$$

$$\text{s.t.} \quad (6.3), (6.7), (6.11), (6.14), (6.18), (6.20), (6.21), (6.22), (6.23),$$

$$E^{\mathrm{BC}_{\mathrm{RF}}}_{n_b,m} \min_k \left(\frac{h^{\mathrm{BC}}_{n_b,k_m}}{l(R^{\mathrm{BC}}_{n_b,k_m})} \right) \geq \mathrm{SINR}_{\mathrm{T}} \left(B N_0 t^{\mathrm{BC}}_{n_b,m} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E^{\mathrm{BC}_{\mathrm{RF}}}_{n_b,m'} \frac{h^{\mathrm{BC}}_{n_b,m,m'}}{R^{\mathrm{BC}}_{n_b,m,m'}} \right),$$

$$E^{\mathrm{Agg}_{\mathrm{RF}}}_{n_b,k_m} \frac{h^{\mathrm{Agg}}_{n_b,k_m}}{l(R^{\mathrm{Agg}}_{n_b,k_m})} \geq \mathrm{SINR}_{\mathrm{T}} \left(\frac{B}{K_m} N_0 t^{\mathrm{Agg}}_{n_b,k_m} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E^{\mathrm{Agg}_{\mathrm{RF}}}_{n_b,k_{m'}} \frac{h^{\mathrm{Agg}}_{n_b,k_m,k_{m'}}}{R^{\mathrm{Agg}}_{n_b,k_m,k_{m'}}} \right),$$

$$E^{\mathrm{OT}_{\mathrm{RF}}}_{b,m} \frac{h^{\mathrm{OT}}_{b,m}}{l(R^{\mathrm{OT}}_{b,m})} \geq \mathrm{SINR}_{\mathrm{T}} \left(B N_0 t^{\mathrm{OT}}_{b,m} + \sum_{\substack{m'=1 \\ m' \neq m}}^{N^{\mathrm{rel}}} E^{\mathrm{OT}_{\mathrm{RF}}}_{b,m'} \frac{h^{\mathrm{OT}}_{b,m,m'}}{R^{\mathrm{OT}}_{b,m,m'}} \right),$$

$$0 \leq t^{\mathrm{ID}}_{b,m}, t^{\mathrm{BC}}_{n_b,m}, t^{\mathrm{Agg}}_{n_b,k_m}, t^{\mathrm{OT}}_{b,m}, t^{\mathrm{LOC}}_{n_b,k_m}, d^{\mathrm{ch}}_{n_b,k_m}, E^{\mathrm{ID}_{\mathrm{RF}}}_{b,m}, E^{\mathrm{BC}_{\mathrm{RF}}}_{n_b,m}, E^{\mathrm{Agg}_{\mathrm{RF}}}_{n_b,k_m}, E^{\mathrm{OT}_{\mathrm{RF}}}_{b,m} \tag{6.24}$$

where $\mathbf{x} = \{t^{\mathrm{ID}}_{b,m}, E^{\mathrm{ID}_{\mathrm{RF}}}_{b,m}, t^{\mathrm{BC}}_{n_b,m}, E^{\mathrm{BC}_{\mathrm{RF}}}_{n_b,m}, t^{\mathrm{Agg}}_{n_b,k_m}, E^{\mathrm{Agg}_{\mathrm{RF}}}_{n_b,k_m}, t^{\mathrm{OT}}_{b,m}, E^{\mathrm{OT}_{\mathrm{RF}}}_{b,m}, t^{\mathrm{LOC}}_{n_b,k_m}, d^{\mathrm{ch}}_{n_b,k_m}, t_{n_b,m}, t_b\}$, $\forall n \in [N_b^{\mathrm{LAY}}]$, $\forall k \in [K_m]$, $\forall b \in [N^{\mathrm{bl}}]$ 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.

6.5 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.

TABLE 6.1: Selected parameters for the simulation results of Chapter 6

Parameters	Values	Units
κ_{k_m}	Unif$([10^{-28}; 10^{-27}])$	/
c_{k_m}	Unif$([150; 350])$	CPU cycles/MAC
$\nu_{k_m}^{\max}$	Unif$([1; 3])$	GHz
h(Each channel gain)	$\mathcal{CN}(0, 10^{-3})$(Rayleigh)	/
$P_{\widetilde{k}}^c$ & P_m^c & $P_{k_m}^c$	Unif$([10; 25])$	mW
B	2	MHz
N_0	0.5×10^{-11}	W/MHz
f_c	2.1	GHz
α	2.5	/
$R_{b,m}^{\mathrm{ID}}$ & $R_{b,m}^{\mathrm{OT}}$	Unif$([100; 150])$	m
$R_{n_b,k_m}^{\mathrm{BC}}$ & $R_{n_b,k_m}^{\mathrm{Agg}}$	Unif$([10; 20])$	m
τ	0.5	s
d_n^{ch}	16	/
N^{LAY}	3	/
N_1^{LAY}	1	/
N_2^{LAY}	2	/

Throughout our simulations, we employed parameter settings as detailed in Table 6.1. These settings were inspired by the findings in the articles [20, 24]. Just like in the previous chapters, the simulation results presented here are based on averaging outcomes obtained from 1000 distinct random channel realizations.

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

In Figures 6.1 and 6.2, 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. To ensure a fair comparison, we equalize the data rates of the proposed *DoubleRel* and the *NoIntfc-AllocBW* scenarios in each communication stage, expressed as $r^{DoubleRel} = r^{AllocBW}$. The data rates are defined as

$$r^{DoubleRel} = B \ln(1 + \mathrm{SINR_T}), \qquad r^{AllocBW} = \frac{B}{N^{\mathrm{rel}}} \ln(1 + \mathrm{SINR_T^O}) \quad (6.25)$$

where $\mathrm{SINR}_{\mathrm{T}}^{O}$ is the SINR target value of the scenario labeled *NoIntfc-AllocBW*. We express $\mathrm{SINR}_{\mathrm{T}}^{O}$ as a function of $\mathrm{SINR}_{\mathrm{T}}$ as follows:

$$\mathrm{SINR}_{\mathrm{T}}^{O} = (1 + \mathrm{SINR}_{\mathrm{T}})^{N^{\mathrm{rel}}} - 1 \tag{6.26}$$

Fig. 6.1 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 larger allocated bandwidth per computing segment than the *NoIntfc-AllocBW* scenario. Regardless of the scenario, higher $\mathrm{SINR}_{\mathrm{T}}$ values contribute to improved energy efficiency despite increased communication costs. This outcome arises from the fact that higher $\mathrm{SINR}_{\mathrm{T}}$ values reduce the lower bounds in (6.7), (6.14), and (6.18), thus requiring less time for communication-related stages. This allows the system to allocate more time for local computation, resulting in significantly reduced computation costs. In scenarios with high computation loads, surprisingly, a slightly higher $\mathrm{SINR}_{\mathrm{T}}$ value could potentially enhance overall energy efficiency.

In Fig. 6.2, the same configuration is used with smaller input data sizes, specifically $d^{\mathrm{row}} = 128$ bits and $d^{\mathrm{col}} = 64$ bits. In this case, a clear relationship between energy consumption and $\mathrm{SINR}_{\mathrm{T}}$ values is observed. Increasing $\mathrm{SINR}_{\mathrm{T}}$ leads to higher energy consumption. Unlike in Fig. 6.1, where energy consumption decreases with higher $\mathrm{SINR}_{\mathrm{T}}$ 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.

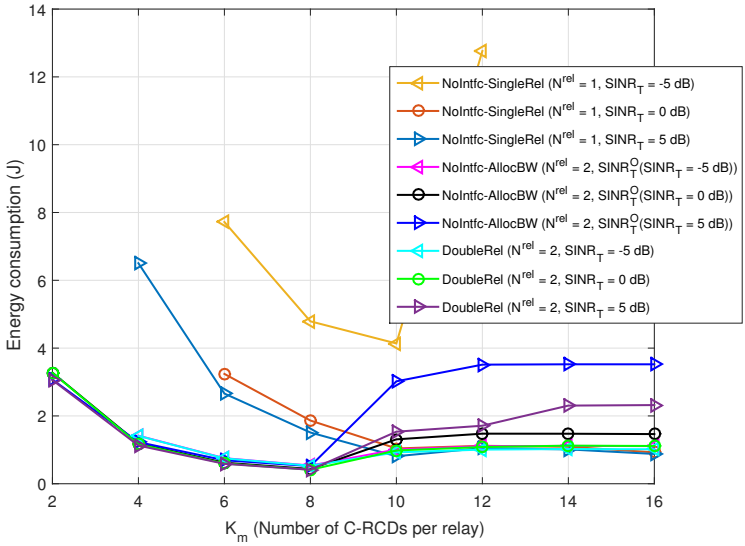


FIGURE 6.1: Energy consumption of different scenarios vs. K_m for several SINR_T values for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits

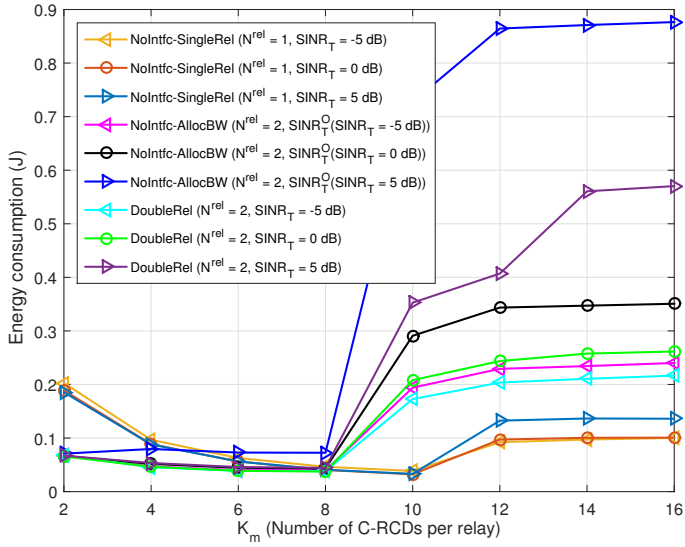


FIGURE 6.2: Energy consumption of different scenarios vs. K_m for several SINR_T values for $d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits

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 their best K_m value where they perform optimally.

6.5.2 Impact of SINR_T Value on Energy Consumption of Different Scenarios

Fig. 6.3 depicts the energy consumption of different scenarios at their 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 (6.7), (6.14), and (6.18). 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, increasing SINR_T values beyond this threshold does not yield energy efficiency benefits. The optimal SINR_T value strikes the trade-off between communication and computation costs, ensuring the system allocates an appropriate amount of time for both stages. 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.

6.5.3 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. 6.4, 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

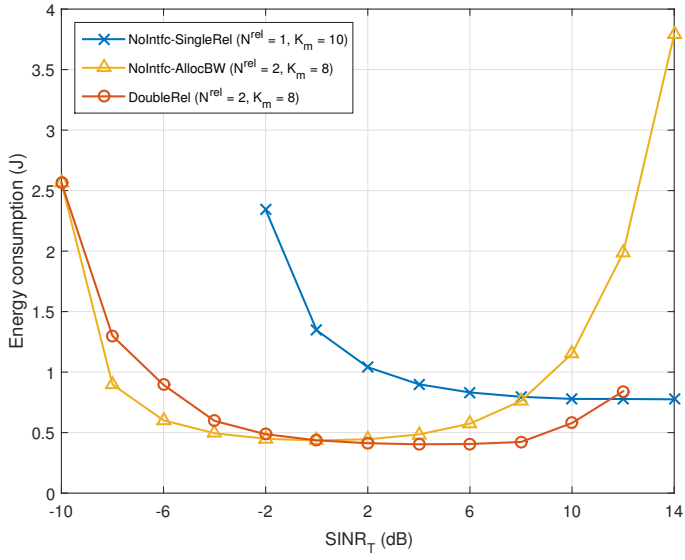


FIGURE 6.3: Energy consumption of different scenarios with the optimal K_m values vs. SINR_T for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits

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

Fig. 6.5 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. 6.4, 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.

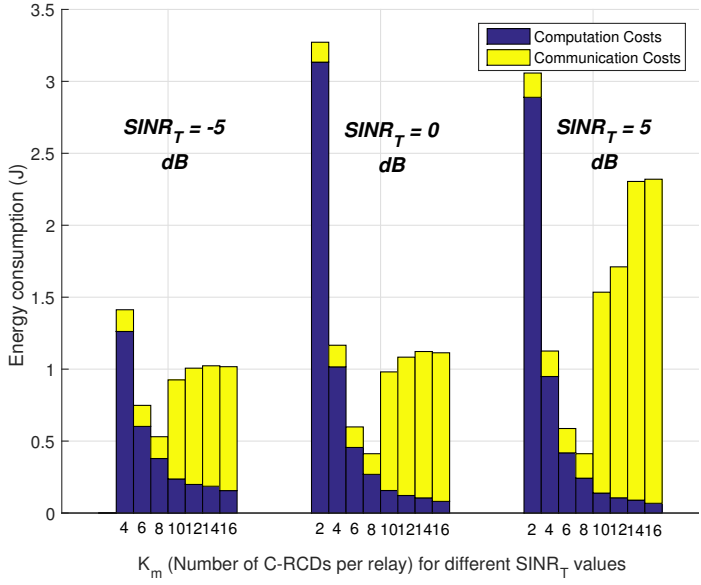


FIGURE 6.4: Energy consumption of communication and computation with different K_m and SINR_T for *DoubleRel* scenario for $d^{\text{row}} = 256$ bits and $d^{\text{col}} = 128$ bits

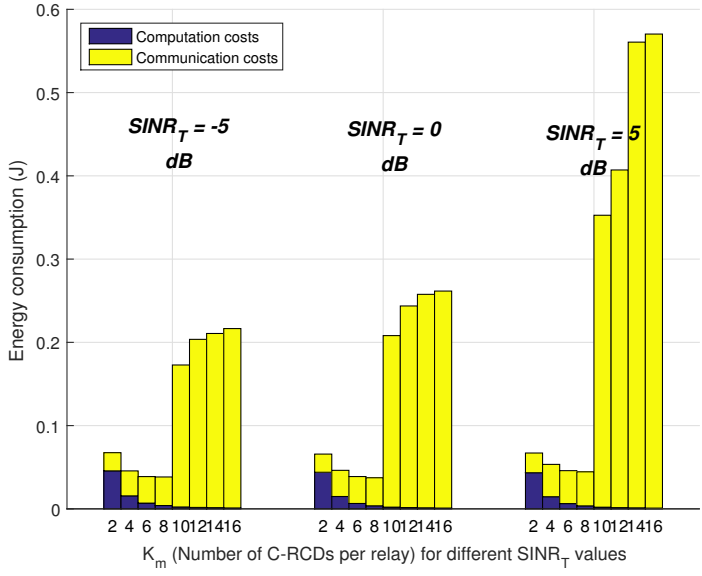


FIGURE 6.5: Energy consumption of communication and computation with different K_m and SINR_T for *DoubleRel* scenario for $d^{\text{row}} = 128$ bits and $d^{\text{col}} = 64$ bits

6.6 Summary

This study presents an interference-aware relay-assisted collaborative CNN inference framework that integrates hybrid parallelism, composed of both data and model parallelism. By addressing challenges like communication overhead and interference effects, the framework aims to minimize energy consumption during collaborative CNN inference.

The investigation into the impact of SINR target value on communication-related stages underscores its role in shaping the trade-off between communication and computation energy consumption. At lower SINR_T values, the system experiences notably higher energy consumption because a reduction in SINR_T results in the system allocating more time to communication and less to computation, consequently raising computation costs. Conversely, as SINR_T values surpass a certain threshold, communication costs begin to dominate overall energy consumption. Hence, increasing SINR_T beyond this threshold does not lead to energy efficiency gains. This emphasizes the importance of selecting an optimal SINR_T value tailored to the specific characteristics and computational demands of the system.

Chapter 7

New Communication and Computation Models for Practical Deployment

In this chapter, we define new communication and computation models and employ simulation parameters that are more practical-oriented. The primary objective is to demonstrate the feasibility of the parallelism techniques mentioned in this thesis for potential real-world deployment.

7.1 Practical Communication Model

To assess the energy consumption of each communication stage, we adopt the power model detailed in [78], which accounts for the IEEE 802.11g standard for WiFi-based communication. There are certain assumptions that we consider in our calculations. Firstly, we consider the energy consumption of both the transmitter and receiver sides at each communication stage. Additionally, we assume ideal packet scheduling so that no scheduling time is taken into account. Another assumption is that radios can turn on and off immediately, eliminating the need to account for switching time in our calculations.

By inspiring from [78], we define energy consumption models separately for the transmitter (TX) and receiver (RX) sides of layer-pack n operation in device k as

$$E_{n,k}^{\mathrm{TX}} = (A_k^{\mathrm{TX}}\, r_{n,k}^{\mathrm{TX}} + F_k) t_{n,k}^{\mathrm{TX}} = A_k^{\mathrm{TX}}(d_{n,k} + d_{\mathrm{IP}}) + F_k\, t_{n,k}^{\mathrm{TX}} \tag{7.1}$$

$$E_{n,k}^{\mathrm{RX}} = (A_k^{\mathrm{RX}}\, r_{n,k}^{\mathrm{RX}} + F_k) t_{n,k}^{\mathrm{RX}} = A_k^{\mathrm{RX}}(d_{n,k} + d_{\mathrm{IP}}) + F_k\, t_{n,k}^{\mathrm{RX}} \tag{7.2}$$

where $d_{n,k} + d_{\text{IP}} = r_{n,k}^{\text{TX}} \, t_{n,k}^{\text{TX}} = r_{n,k}^{\text{RX}} \, t_{n,k}^{\text{RX}}$. Here, $r_{n,k}^{\text{TX}}$ and $r_{n,k}^{\text{RX}}$ represent the data rates for transmission and reception in kilobits per second (kbps), respectively. A_k^{TX} and A_k^{RX} denote positive values in Joules/kbit specific to the transceiver for the data rates $r_{n,k}^{\text{TX}}$ and $r_{n,k}^{\text{RX}}$, respectively. F_k signifies the constant power consumption of the communication circuitry in Watts for each transceiver. $d_{n,k}$ is the number of kbits to be sent in each communication stage and d_{IP} is the packet overhead in kbits.

The total energy consumption in each communication stage is given by

$$E_{n,k}^{\text{commun}} = E_{n,k}^{\text{TX}} + E_{n,k}^{\text{RX}} = (A_k^{\text{TX}} + A_k^{\text{RX}})(d_{n,k} + d_{\text{IP}}) + F_k \, t_{n,k}^{\text{commun}} \tag{7.3}$$

where $t_{n,k}^{\text{commun}} = t_{n,k}^{\text{TX}} + t_{n,k}^{\text{RX}}$ is the total allocated time for each communication stage.

We consider a single constraint for each communication stage, similar to the communication model in previous chapters. This constraint is related to the maximum number of bits that can be sent via each communication channel and it can be expressed as

$$(d_{n,k} + d_{\text{IP}})kbits(1000bits/kbits) \leq t_{n,k}^{\text{commun}} \, B \, \ln(1 + \frac{\frac{E_{n,k}^{\text{RF}}}{t_{n,k}^{\text{commun}}} \frac{h_{n,k}}{l(R_{n,k})}}{BN_0}) \tag{7.4}$$

where $E_{n,k}^{\text{RF}}$ is the RF energy consumption of each communication stage in Joules.

7.2 Practical Computation Model

The computation model that we follow in this chapter relies on the constant parameter c_k^{FLOPS}, representing the number of FLOPs per Watt. This parameter is device specific and the values of this parameter for several devices can be found in [79]. The energy consumption of each computation stage is modeled as

$$E_{n,k}^{\text{comp}} = \frac{N_{n,k}^{\text{FLOP}}}{c_k^{\text{FLOPS}}} t_{n,k}^{\text{comp}} = \frac{2 \, N_{n,k}^{\text{MAC}}}{c_k^{\text{FLOPS}}} t_{n,k}^{\text{comp}} \tag{7.5}$$

where $N_{n,k}^{\text{FLOP}}$ is the number of FLOPs for each computation stage in each device, and $t_{n,k}^{\text{comp}}$ is the allocated time for each computation stage. We have the relationship $N_{n,k}^{\text{FLOP}} = 2 \, N_{n,k}^{\text{MAC}}$ since one MAC operation corresponds to two FLOPs (one for multiplication and one for addition).

Unlike our computation model used in earlier chapters where the computation time is inversely proportional to the energy consumption of computation, here, the computation time is directly proportional to the energy consumption. This is because CPU frequency of each device is considered to be constant in this computation model throughout all operation while it is allowed to change in each CPU cycle for the computation model of the previous chapters. Thus, the allocated time for each computation stage becomes one of the optimizing parameters for the optimization problem in the computation model of the previous chapters, where increasing the time allocated to computation decreases the working CPU frequency of devices, leading to more energy-efficient execution. In the present computation model, with a constant CPU frequency, for a given number of MAC operations $N_{n,k}^{\mathrm{MAC}}$, $t_{n,k}^{\mathrm{comp}}$ becomes a constant parameter, which can be calculated as

$$t_{n,k}^{\mathrm{comp}} = \frac{N_{n,k}^{\mathrm{FLOP}}}{f_k^{\mathrm{CPU}}} = \frac{2\, N_{n,k}^{\mathrm{MAC}}}{f_k^{\mathrm{CPU}}} \tag{7.6}$$

where f_k^{CPU} is the CPU working frequency of device k. Consequently, the computation energy consumption expression can be modified as

$$E_{n,k}^{\mathrm{comp}} = \frac{4\, (N_{n,k}^{\mathrm{MAC}})^2}{c_k^{\mathrm{FLOPS}} f_k^{\mathrm{CPU}}} \tag{7.7}$$

We impose a single constraint for each computation stage, stating that the power consumed in each computation stage for each device cannot exceed its maximum allowed power $P_k^{\mathrm{comp,max}}$:

$$\frac{2\, N_{n,k}^{\mathrm{MAC}}}{c_k^{\mathrm{FLOPS}}} \leq P_k^{\mathrm{comp,max}} \tag{7.8}$$

This constraint ensures that the power consumption during each computation stage adheres to the device's specified maximum power limit.

7.3 Applying the Models for Model Parallelism Scenario

In this section, we apply the introduced practical communication and computation models to the model parallelism approach, as mentioned in Chapter 4. The model parallelism scenario is selected to deploy these models for its straightforward stages compared to other approaches, such as the relay-assisted hybrid parallelism scenario.

For the model parallelism approach, the energy consumption of the system is divided into three distinct stages: broadcasting, collaborative local computation, and aggregation.

7.3.1 Broadcasting

The energy consumption of broadcasting for layer-pack n operation is calculated as

$$E_n^{\mathrm{BC}} = (A_k^{\mathrm{TX}} + K_n A_k^{\mathrm{RX}})(d_n^{\mathrm{BC}} + d_{\mathrm{IP}}) + F_k\, t_n^{\mathrm{BC}} \tag{7.9}$$

where the data parameters d_n^{BC} and d_{IP} are in kbits. The following constraint is imposed regarding this stage:

$$1000(d_n^{\mathrm{BC}} + d_{\mathrm{IP}}) \leq t_n^{\mathrm{BC}}\, B\, \ln(1 + \frac{\frac{E_n^{\mathrm{BC_{RF}}}}{t_n^{\mathrm{BC}}}\, \min_k \left(\frac{h_{n,k}^{\mathrm{BC}}}{l(R_{n,k}^{\mathrm{BC}})} \right)}{BN_0}) \tag{7.10}$$

The meaning of this constraint is explained in Section 7.1.

7.3.2 Collaborative Local Computation

The energy consumption of local computation in device k for layer-pack n operation is given by

$$E_{n,k}^{\mathrm{LOC}} = \frac{4\ (N_{n,k}^{\mathrm{MAC}})^2}{c_k^{\mathrm{FLOPS}} f_k^{\mathrm{CPU}}} = \frac{4\ 36^2 (d^{\mathrm{row}} d^{\mathrm{col}})^2}{16^n c_k^{\mathrm{FLOPS}} f_k^{\mathrm{CPU}}} (d_{n,k}^{\mathrm{ch}})^2 \tag{7.11}$$

where

$$N_{n,k}^{\mathrm{MAC}} = 36 \frac{d^{\mathrm{row}} d^{\mathrm{col}}}{4^n} d_{n,k}^{\mathrm{ch}} \tag{7.12}$$

We consider a single constraint for each computation stage, which is expressed as

$$\frac{2\ N_{n,k}^{\mathrm{MAC}}}{c_k^{\mathrm{FLOPS}}} = \frac{72\ d^{\mathrm{row}} d^{\mathrm{col}}}{4^n c_k^{\mathrm{FLOPS}}} d_{n,k}^{\mathrm{ch}} \leq P_k^{\mathrm{comp,max}} \tag{7.13}$$

stating that the power consumed in each computation stage in each device cannot exceed its maximum allowed power $P_k^{\mathrm{comp,max}}$.

7.3.3 Aggregation

The energy consumption for the aggregation in device k after layer-pack n operation is computed as

$$E_{n,k}^{\text{Agg}} = (A_k^{\text{TX}} + A_k^{\text{RX}})(d_n^{\text{Agg}} + d_{\text{IP}}) + F_k \, t_{n,k}^{\text{Agg}} \tag{7.14}$$

where we have a similar constraint with the broadcasting stage:

$$1000(d_n^{\text{Agg}} + d_{\text{IP}}) \leq t_{n,k}^{\text{Agg}} \; B \; \ln(1 + \frac{\frac{E_{n,k}^{\text{Agg}_{\text{RF}}}}{t_{n,k}^{\text{Agg}}} \frac{h_{n,k}^{\text{Agg}}}{l(R_{n,k}^{\text{Agg}})}}{BN_0}) \tag{7.15}$$

where the data parameter d_n^{Agg} is in kbits.

7.3.4 Problem Formulation

By considering the same additional constraints (4.13), (4.14), (4.15), (4.16) in Chapter 4, the final problem formulation looks like the following:

$$\begin{aligned} \min_{\mathbf{x}_1} \quad & \sum_{n=1}^{N^{\text{LAY}}} (E_n^{\text{BC}} + \sum_{k=1}^{K_n} E_{n,k}^{\text{LOC}} + \sum_{\substack{k=1 \\ k \neq \tilde{k}}}^{K_n} E_{n,k}^{\text{Agg}}) \\ \text{s.t.} \quad & (4.13), (4.14), (4.15), (4.16), (7.10), (7.13), (7.15), \\ & 0 \leq d_{n,k}^{\text{ch}}, E_n^{\text{BC}_{\text{RF}}}, E_{n,k}^{\text{Agg}_{\text{RF}}}, \\ & 0 \leq t_n^{\text{BC}}, t_{n,k}^{\text{Agg}} \leq t_n \end{aligned} \tag{7.16}$$

where $\mathbf{x}_1 = \{\{t_n, t_n^{\text{BC}}, E_n^{\text{BC}_{\text{RF}}}\}_{n=1}^{N^{\text{LAY}}}, \{\{d_{n,k}^{ch}\}_{k=1}^{K_n}\}_{n=1}^{N^{\text{LAY}}}, \{\{t_{n,k}^{\text{Agg}}, E_{n,k}^{\text{Agg}_{\text{RF}}}\}_{\substack{k=1 \\ k \neq \tilde{k}}}^{K_n}\}_{n=1}^{N^{\text{LAY}}}\}$.

7.4 Simulation Results

This section presents the simulation results obtained by incorporating the new communication and computation models to the model parallelism scenario proposed in Chapter 4. The selected parameters for these results are shown in Table 7.1. The coefficients A_k^{TX} and A_k^{RX} were measured in [78] as 1.4×10^{-5} J/kbits and 1.8×10^{-6} J/kbits, respectively. The constant power consumption F_k of the communication circuitry is set to 4.8×10^{-4}

W [78]. It is assumed that there is packet overhead, and the minimum length of an IP header is 20 bytes (160 bits), i.e., $d_{\text{IP}} = 0.16$ kbits.

The simulation results are obtained based on averaging over 100 channel realizations, with bandwidth B and carrier frequency f_c values assigned according to the IEEE 802.11g standard, specifically 20 MHz and 2.4 GHz, respectively. To explore the impact of varying channel realizations on the system performance, the results of all channel realizations are compared and they give almost identical results. This implies that the system exhibits robustness to variations in channel realizations.

For a potential real-time deployment, selecting an appropriate maximum allowed latency is crucial. For this purpose, a latency-critical value is chosen as $\tau = 10$ ms.

As the CNN model, unlike the previous chapters, the network design of a popular CNN architecture, AlexNet [80], is employed. AlexNet comprises 5 convolutional layers with varying numbers of filters in each layer ($[96, 256, 384, 384, 256]$). The motivation behind choosing AlexNet is that it has a suitable amount of convolutional layers for our model parallelism approach and different number of filters in each convolutional layer, which provides us the opportunity to explore different scenarios for filter distribution for each convolutional layer.

For participating devices, specifications from three different Raspberry Pi models are utilized: Raspberry Pi-4B, Raspberry Pi-3B+, and Raspberry Pi-2. The specifications of these devices can be found in Table 7.2, taken from [79].

Three different scenarios are explored for the employment of participating devices. The first scenario, called the mixed scenario, involves the equal inclusion of all three Raspberry Pi models. In the second scenario, termed the low computing capability scenario, only Raspberry Pi-2 devices are used as participating devices. The third scenario, referred to as the high computing capability scenario, employs only Raspberry Pi-4B devices.

Fig. 7.1 illustrates the energy consumption values of the mixed scenario corresponding to varying number of participating devices for $\tau = 10$ ms. The plot also includes the energy consumption of communication and computation separately to show the trade-off using these models as well. Consistent with the findings by using the models in previous chapters, the energy consumption is initially high for a low number of participating devices, primarily driven by the dominant factor of the energy consumption of computation. However, as the number of participating devices increases, a critical point is reached where further additions provide no

TABLE 7.1: Selected parameters for the simulation results of Chapter 7

Parameters	Values	Units
A_k^{TX}	1.4×10^{-5}	J/kbits
A_k^{RX}	1.8×10^{-6}	J/kbits
F_k	4.8×10^{-4}	W
d_{IP}	0.16	kbits
$h_{n,k}$	$\mathcal{CN}(0, 10^{-3})$(Rayleigh)	/
$R_{n,k}$	Unif$([1;20])$	m
B	20	MHz
N_0	10^{-12}	W/MHz
f_c	2.4	GHz
α	2.5	/
N^{LAY}	5	/
d_n^{ch}	$[96, 256, 384, 384, 256]$	/
d^{col} & d^{row}	128	/
τ	10	ms

TABLE 7.2: Specifications of participating devices for the simulation results of Chapter 7

Device	$\mathbf{c_k^{FLOPS}}$ (GFLOPs/W)	$\mathbf{P_k^{comp,max}}$ (W)	$\mathbf{f_k^{CPU}}$ (GHz)
Raspberry Pi-2	0.432	3.6	0.9
Raspberry Pi-3B+	0.73	9.4	1.4
Raspberry Pi-4B	1.35	8.2	1.5

more energy-efficiency. This is due to the energy consumption of communication starting to become the dominant factor.

Fig. 7.2 is created to compare different deployment scenarios for participating devices. In the case of low computing capability with Raspberry Pi-2's, there is a clear trade-off point between communication and computation for the number of participating devices. However, when we increase the capability of participating devices by replacing Raspberry Pi-2's with Raspberry Pi-4B's, the optimal number of participating devices becomes much lower, even as low as 2, which is the minimum for a collaborative scenario. This comparison helps us define the practical boundaries where our approaches could potentially shine. Additionally, we include the data-point 1 on the x-axis of the figure to emphasize that, given the latency $\tau = 10$ ms, all types of devices fail to complete the entire CNN model execution when performed individually. For the given parameters, the time required for the highest computing capability device (Raspberry Pi-4B) to complete the CNN model execution is 37.6 ms, whereas it is 62.6 ms if a Raspberry Pi-2 is used.

To assess the energy consumption level of non-collaborative individual CNN model execution, we initially relax the latency constraint to 37.6 ms, presenting the results in Fig. 7.3. Examining the figure, it becomes evident that the energy consumption associated with non-collaborative execution using Raspberry Pi-4B is significantly lower than other collaborative approaches across all scenarios. The key insight from this observation is that, under conditions where participating devices are comparably powerful, and sufficient time is allocated to complete the CNN execution, the individual execution is more favorable. However, in alternative scenarios, particularly those with less powerful devices or constrained latency, a collaborative approach should be the preferred option.

Furthermore, to explore the energy consumption of non-collaborative execution with Raspberry Pi-2, the latency constraint is further relaxed to 62.6 ms, and the corresponding results are illustrated in Fig. 7.4. In this particular scenario, it becomes apparent that even with the relaxation of latency constraints, the non-collaborative approach does not become more favorable. Consequently, our collaborative approaches remain the optimal choices in this configuration.

Lastly, to explore the threshold where collaborative execution becomes less energy-efficient, it's crucial to consider the energy efficiency parameter c_k^{FLOPS} and the CPU working frequency f_k^{CPU} of devices. The logical expectation is as follows: Devices with high f_k^{CPU}, sufficient allowed latency for the executed CNN-driven application, and a high c_k^{FLOPS} value may

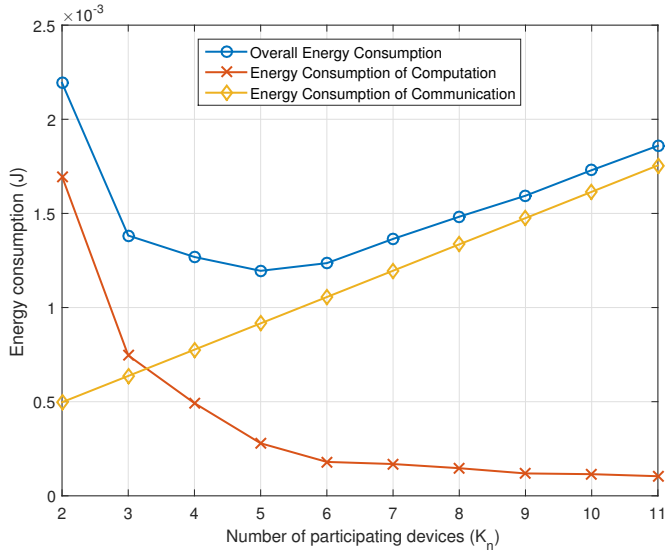


FIGURE 7.1: Energy consumption of communication and computation for the mixed scenario vs. number of participating devices for $\tau = 10$ ms

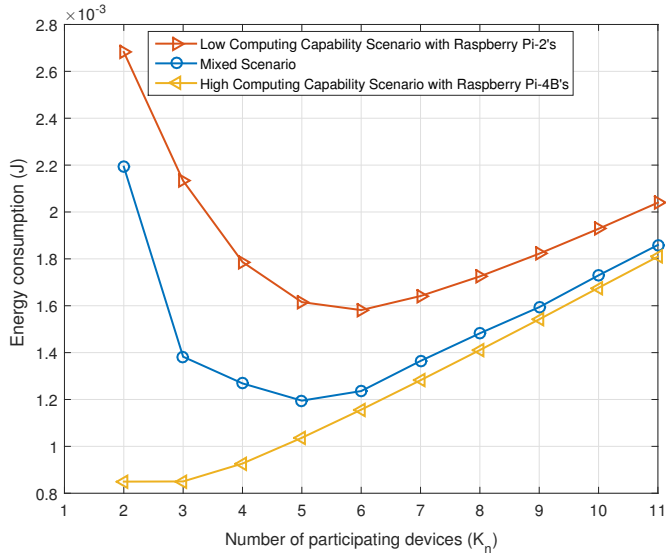


FIGURE 7.2: Overall energy consumption of different scenarios vs. number of participating devices for $\tau = 10$ ms

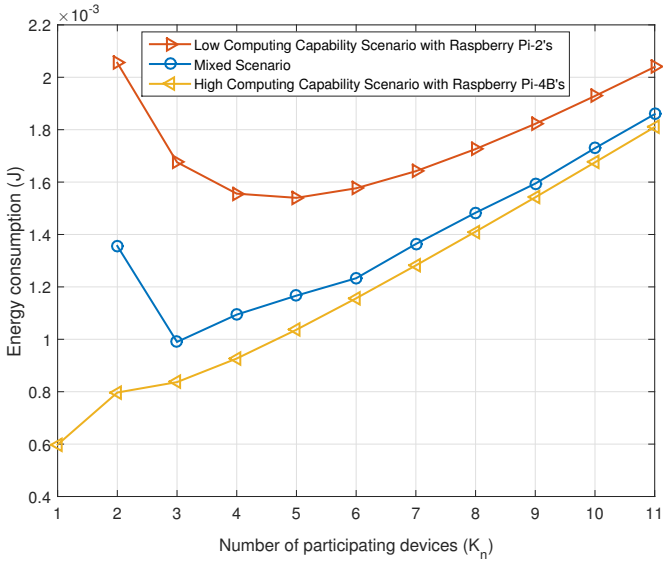


FIGURE 7.3: Overall energy consumption of different scenarios vs. number of participating devices for $\tau = 37.6$ ms

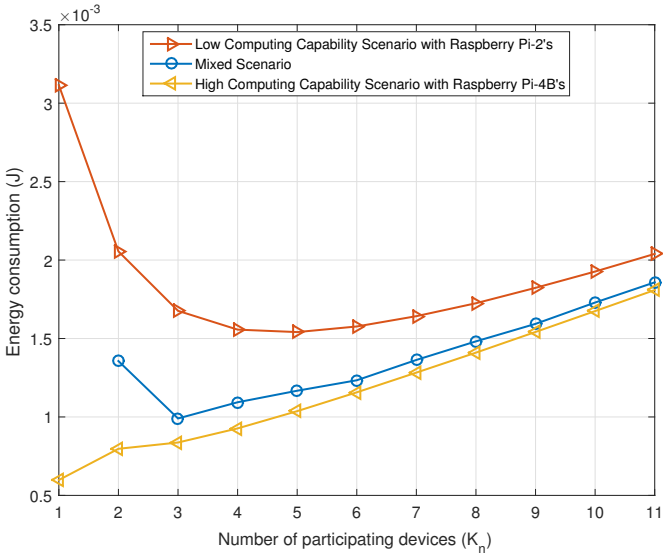


FIGURE 7.4: Overall energy consumption of different scenarios vs. number of participating devices for $\tau = 62.6$ ms

find single-device execution more energy efficient. To investigate this, two homogeneous scenarios are established, where all devices operate at the same CPU frequency ($f_k^{\text{CPU}} = 0.9$ GHz in Fig. 7.5 or $f_k^{\text{CPU}} = 1.5$ GHz in Fig. 7.6), and the energy efficiency parameter c_k^{FLOPS} is varied. The optimal number of participating devices for each c_k^{FLOPS} configuration is determined. This methodology allows us to identify the threshold c_k^{FLOPS} value where collaborative execution becomes less energy-efficient. With a lower CPU frequency ($f_k^{\text{CPU}} = 0.9$ GHz) in Fig. 7.5, collaborative execution appears to be the preferred choice, even for high c_k^{FLOPS} values, indicating that the CPU frequency is insufficient to handle CNN execution efficiently on a single device. However, if f_k^{CPU} of each device is increased to 1.5 GHz, the threshold condition for favoring non-collaborative execution is reached at $c_k^{\text{FLOPS}} = 0.86$ GFLOPs/W. The observations from these figures align with our initial expectations: if a device has a sufficiently high f_k^{CPU} value, adequate allowed latency, and a sufficiently high c_k^{FLOPS}, performing CNN execution on a single device is preferable. However, it's important to note the distinction between low-cost resource-constrained devices, where collaborative approaches can be beneficial due to resource limitations, and more modern computing platforms with embedded CNN acceleration, for which collaborative approaches may not provide significant benefits. For instance, low-cost resource-constained devices include devices like Raspberry Pi models, Arduino boards, and budget smartphones, which typically lack specialized hardware accelerators for tasks like CNN inference. On the other hand, modern computing platforms with embedded CNN acceleration, such as high-end smartphones like the iPhone Pro series or Samsung Galaxy S series, edge computing devices like the NVIDIA Jetson series, and AI-enabled cameras or smart sensors are specifically designed to efficiently handle CNN computations on a single device without the need for collaborative computing. However, this is often not the case for many older, resource-constrained devices in the environment, which typically have lower CPU frequencies and less energy-efficient execution. Moreover, most real-time CNN-driven applications require execution completion within several milliseconds, making collaborative approaches still advantageous in specific scenarios.

7.5 Summary

In conclusion, to determine the optimal approach for different configurations, Table 7.3 is generated. This table provides insights into the most energy-efficient execution, denoted as either *Col* for collaborative execution

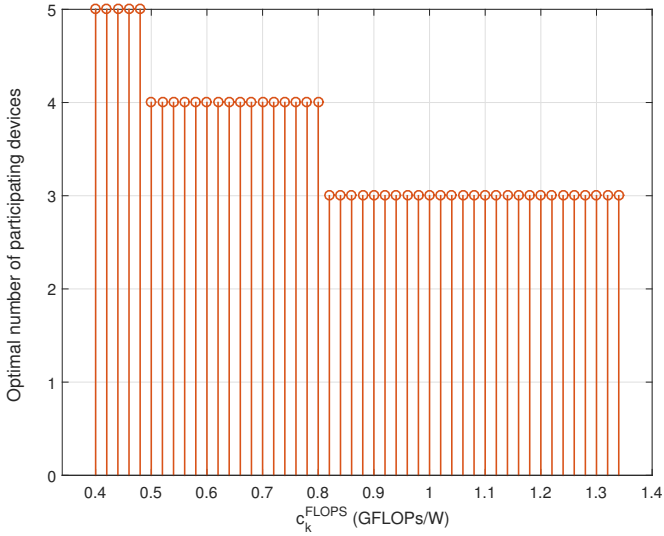


FIGURE 7.5: Optimal number of participating devices vs. c_k^{FLOPS} for $f_k^{\text{CPU}} = 0.9$ GHz, $\forall k$

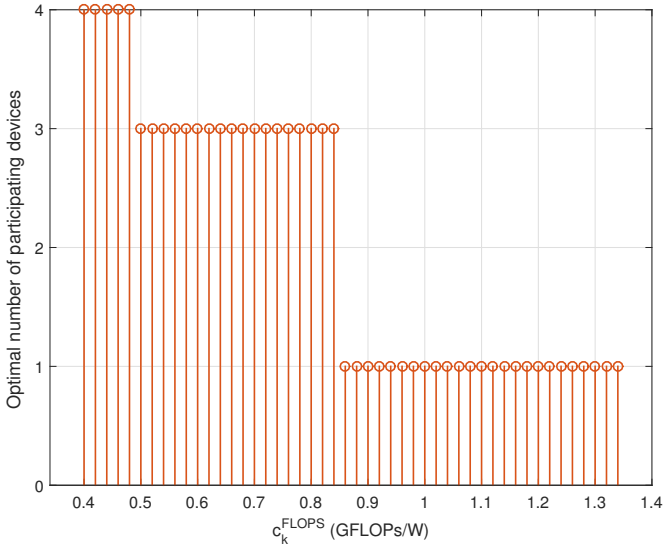


FIGURE 7.6: Optimal number of participating devices vs. c_k^{FLOPS} for $f_k^{\text{CPU}} = 1.5$ GHz, $\forall k$

TABLE 7.3: The most energy efficient execution for each configuration

Latency	**Computing Capability**		
	Low	Mixed	High
Low	*Col*	*Col*	*Col*
Moderate	*Col*	*Col*	*Non-col*
High	*Col*	*Col*	*Non-col*

or *Non-col* for non-collaborative execution, across all combinations of maximum allowed latency and computing capability of the participating devices. The initial goal of this research was to leverage communication opportunities for efficient computation and design collaborative CNN execution scenarios suitable for latency-critical applications on various resource-constrained devices. It corresponds to the low latency and low computing capability case in Table 7.3, which is consistent to our target scenario. Additionally, the collaborative approach can be mostly preferred if the system does not have a major amount of high computing capability devices, even if the maximum latency given to the system is high. However, in scenarios where comparatively higher computing capability devices are incorporated into the system, it could be more favorable to execute the CNN model individually.

Chapter 8

Conclusion and Perspectives

In this thesis, we have undertaken an extensive exploration of collaborative convolutional neural network (CNN) inference within resource-constrained environments. Our research comprises four distinct studies, each contributing innovative approaches to optimize the inference of CNN models through various forms of parallelism and communication strategies. These contributions collectively represent a significant advancement in the field of collaborative CNN inference.

Our first contribution introduces several schemes for efficient distributed collaborative CNN inference, allowing us to harness the computing resources of resource-constrained devices (RCDs) without relying solely on powerful servers. We then delve into various parallelism strategies, including data parallelism, model parallelism, and hybrid parallelism, tailored to specific scenarios, thereby enhancing energy efficiency and reducing communication overhead.

Furthermore, our research has led to the development of joint optimization frameworks, which take into account communication, computation, and workload distribution parameters. This holistic approach minimizes energy consumption and provides a well-rounded solution for collaborative CNN inference. We have also advocated for decentralized system architectures that leverage the computing resources of participating devices, promoting scalability and reliability even in challenging deployment scenarios.

Our work also involves formulating convex optimization problems with analytical solutions, leveraging the primal-dual decomposition technique.

These solutions offer practical and scalable means of achieving optimal parameter settings in resource-constrained collaborative CNN inference. Finally, our fourth study focuses on addressing interference challenges, aiming to improve the robustness of collaborative CNN inference in scenarios with potential interference.

The significance of these contributions lies in their potential to revolutionize how we execute CNN models on devices with limited resources. In an era where CNN applications expand into wireless communication and edge intelligence contexts, our research provides invaluable insights into the critical issues of energy efficiency.

Moreover, our investigation into the energy efficiency parameter in Chapter 7 and its implications for collaborative execution sheds light on the optimal conditions for leveraging collaborative inference. By identifying the threshold where collaborative execution becomes less energy-efficient, we offer guidance on when to prefer non-collaborative approaches instead of collaborative execution, particularly on modern computing platforms with embedded CNN acceleration. This insight emphasizes the importance of considering device capabilities and the suitability of collaborative strategies, especially in scenarios with low-cost generic computing platforms versus modern platforms with embedded CNN acceleration.

While this thesis represents a significant step forward in the field of collaborative CNN inference, offering efficient, scalable, and energy-conscious solutions for resource-constrained environments, there are promising avenues for future research and development.

One area that warrants further exploration is the investigation of novel parallelism techniques and optimization strategies tailored to specific types of neural networks beyond CNNs. Adapting collaborative inference schemes to accommodate recurrent neural networks (RNNs) and transformer architectures could extend the applicability of our proposed methods to a broader range of artificial intelligence (AI) models.

Additionally, the implementation and evaluation of the proposed collaborative CNN inference schemes on real-world applications and diverse datasets remain an essential next step. This empirical validation will not only verify the effectiveness of the proposed approaches but also uncover potential challenges and optimizations required for practical deployment.

Furthermore, considering the dynamic nature of edge environments, investigating adaptive mechanisms for workload distribution and parameter optimization could enhance the robustness and adaptability of collaborative CNN inference systems. Exploring reinforcement learning techniques

for dynamically adjusting collaboration strategies in response to changing resource availability could be an interesting idea for future research.

Interdisciplinary collaboration with experts in communication networks, hardware design, and system security could enrich the research landscape. Investigating the intersection of collaborative CNN inference and secure, privacy-preserving communication protocols would be particularly relevant, addressing concerns related to data privacy and security in distributed AI systems.

Finally, as the field of AI continues to evolve, exploring the integration of federated learning approaches within the collaborative edge intelligence (EI) framework could offer novel solutions for on-device deep learning inference while respecting data privacy constraints.

In conclusion, the future work outlined above represents a roadmap for advancing the state-of-the-art in collaborative on-device deep learning inference. By exploring these avenues, researchers can contribute to the development of more efficient, scalable, and adaptive solutions for AI applications deployed on resource-constrained devices, furthering the impact of collaborative edge intelligence.

We extend our gratitude to all who have contributed, collaborated, and supported this research journey, and we eagerly anticipate the future developments and applications that will arise from this work.

Appendix A

Solution of Internal Optimization Problem in Chapter 3

According to the Lagrange duality theory, we have the following partial Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathbf{x}_{n,k}, d_{1,k}^{\text{row}}, t_n, \beta_{n,k}) = & \frac{3^6 \kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{1,k}^{\text{row}})^3}{(2^{n-1})^3 (t_{n,k}^{\text{LOC}})^2} \\ & + I_{n,k}^{\text{a}} (E_{n,k}^{\text{EXC}_{\text{RF}_\text{a}}} + t_{n,k}^{\text{EXC}_\text{a}} P_k^c) \\ & + I_{n,k}^{\text{b}} (E_{n,k}^{\text{EXC}_{\text{RF}_\text{b}}} + t_{n,k}^{\text{EXC}_\text{b}} P_k^c) \\ & + \beta_{n,k} (t_{n,k}^{\text{LOC}} + I_{n,k}^{\text{a}} t_{n,k}^{\text{EXC}_\text{a}} + I_{n,k}^{\text{b}} t_{n,k}^{\text{EXC}_\text{b}} - t_n)\end{aligned} \tag{A.1}$$

where $\mathbf{x}_{n,k} = \{t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{EXC}_{\{\text{a,b}\}}}, E_{n,k}^{\text{EXC}_{\text{RF}_{\{\text{a,b}\}}}}\}$ and $\beta_{n,k} \geq 0$ is the Lagrange multiplier associated with (3.23). The dual function is defined as

$$\begin{aligned} g(d_{1,k}^{\text{row}}, t_n, \beta_{n,k}) = & \min_{\mathbf{x}_{n,k}} \quad \mathcal{L}(\mathbf{x}_{n,k}, d_{1,k}^{\text{row}}, t_n, \beta_{n,k}) \\ & \text{s.t.} \quad (3.17), (3.20), (3.31), \\ & \quad 0 \leq E_{n,k}^{\text{EXC}_{\text{RF}_{\{\text{a,b}\}}}}, \\ & \quad 0 \leq t_{n,k}^{\text{LOC}}, t_{n,k}^{\text{EXC}_{\{\text{a,b}\}}} \leq t_n \end{aligned} \tag{A.2}$$

where the problem can be decomposed into three sub-problems. The first part is the local computation optimization problem, which is expressed as

$$\begin{aligned} \min_{t_{n,k}^{\mathrm{LOC}}} \quad & \frac{3^6 \kappa_k c_k^3 (d_n^{\mathrm{col}})^3 (d_{1,k}^{\mathrm{row}})^3}{(2^{n-1})^3 (t_{n,k}^{\mathrm{LOC}})^2} + \beta_{n,k} t_{n,k}^{\mathrm{LOC}} \\ \text{s.t.} \quad & \frac{9 c_k d_n^{\mathrm{col}} d_{1,k}^{\mathrm{row}}}{2^{n-1} \nu_k^{\max}} \leq t_{n,k}^{\mathrm{LOC}} \leq t_n \end{aligned} \tag{A.3}$$

The second part is the exchange communication with the neighbor below (assuming that $I_{n,k}^{\mathrm{b}} = 1$ if there is any neighbor below),

$$\begin{aligned} \min_{t_{n,k}^{\mathrm{EXC_b}}, E_{n,k}^{\mathrm{EXC_{RF_b}}}} \quad & E_{n,k}^{\mathrm{EXC_{RF_b}}} + t_{n,k}^{\mathrm{EXC_b}} (P_k^c + \beta_{n,k}) \\ \text{s.t.} \quad & (3.20), (3.31), \\ & 0 \leq E_{n,k}^{\mathrm{EXC_{RF_b}}}, \\ & 0 \leq t_{n,k}^{\mathrm{EXC_b}} \leq t_n \end{aligned} \tag{A.4}$$

Finally, the exchange communication problem with the neighbor above can be solved just like (A.4).

A.0.1 Solution of Problem (A.3)

The Lagrangian regarding this problem is

$$\begin{aligned} \mathcal{L}_{n,k}^{①} = & \frac{3^6 \kappa_k c_k^3 (d_n^{\mathrm{col}})^3 (d_{1,k}^{\mathrm{row}})^3}{(2^{n-1})^3 (t_{n,k}^{\mathrm{LOC}})^2} + \beta_{n,k} t_{n,k}^{\mathrm{LOC}} \\ & + \gamma_{n,k}^{①} \left(\frac{9 c_k d_n^{\mathrm{col}} d_{1,k}^{\mathrm{row}}}{2^{n-1} \nu_k^{\max}} - t_{n,k}^{\mathrm{LOC}} \right) + \gamma_{n,k}^{②} \left(t_{n,k}^{\mathrm{LOC}} - t_n \right) \end{aligned} \tag{A.5}$$

where $\gamma_{n,k}^{①}, \gamma_{n,k}^{②} \geq 0$ are the Lagrange multipliers. With respect to the variable $t_{n,k}^{\mathrm{LOC}}$, we have the following KKT condition:

$$\frac{\partial \mathcal{L}_{n,k}^{①}}{\partial t_{n,k}^{\mathrm{LOC}}} = -\frac{2\, 3^6 \kappa_k c_k^3 (d_n^{\mathrm{col}})^3 (d_{1,k}^{\mathrm{row}})^3}{(2^{n-1})^3 (t_{n,k}^{\mathrm{LOC}})^3} + \beta_{n,k} - \gamma_{n,k}^{①} + \gamma_{n,k}^{②} = 0 \tag{A.6}$$

where the complementary slackness conditions are

$$\gamma_{n,k}^{①}\left(\frac{9c_k d_n^{\text{col}} d_{1,k}^{\text{row}}}{2^{n-1}\nu_k^{\max}} - t_{n,k}^{\text{LOC}}\right) = 0 \tag{A.7}$$

$$\gamma_{n,k}^{②}\left(t_{n,k}^{\text{LOC}} - t_n\right) = 0 \tag{A.8}$$

We can find the optimal value of $t_{n,k}^{\text{LOC}}$ as follows:

- $t_{n,k}^{\text{LOC}} = \frac{9c_k d_n^{\text{col}} d_{1,k}^{\text{row}}}{2^{n-1}\nu_k^{\max}} < t_n \Rightarrow \gamma_{n,k}^{①} > 0, \gamma_{n,k}^{②} = 0$:

  RCD k works at its highest CPU frequency. We substitute $t_{n,k}^{\text{LOC}}$ into (A.6):
  $\Rightarrow \gamma_{n,k}^{①} = -2\kappa_k(\nu_k^{\max})^3 + \beta_{n,k} > 0 \Rightarrow \beta_{n,k} > 2\kappa_k(\nu_k^{\max})^3$.

- $\frac{9c_k d_n^{\text{col}} d_{1,k}^{\text{row}}}{2^{n-1}\nu_k^{\max}} < t_{n,k}^{\text{LOC}} < t_n \Rightarrow \gamma_{n,k}^{①} = \gamma_{n,k}^{②} = 0$:

  RCD k works with the CPU frequency that is lower than the maximum CPU frequency. From (A.6), $t_{n,k}^{\text{LOC}} = \frac{9c_k d_n^{\text{col}} d_{1,k}^{\text{row}}}{2^{n-1}}\left(\frac{2\kappa_k}{\beta_{n,k}}\right)^{1/3}$ and we substitute $t_{n,k}^{\text{LOC}}$ into the inequality:
  $\Rightarrow \frac{2\ 3^6\kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{1,k}^{\text{row}})^3}{(2^{n-1})^3 (t_n)^3} < \beta_{n,k} < 2\kappa_k(\nu_k^{\max})^3$.

- $t_{n,k}^{\text{LOC}} = t_n \Rightarrow \gamma_{n,k}^{①} = 0, \gamma_{n,k}^{②} > 0$:

  We substitute $t_{n,k}^{\text{LOC}}$ into (A.6):
  $\Rightarrow \gamma_{n,k}^{②} = \frac{2\ 3^6\kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{1,k}^{\text{row}})^3}{(2^{n-1})^3 (t_n)^3} - \beta_{n,k} > 0 \Rightarrow \beta_{n,k} < \frac{2\ 3^6\kappa_k c_k^3 (d_n^{\text{col}})^3 (d_{1,k}^{\text{row}})^3}{(2^{n-1})^3 (t_n)^3}$.

Finally, we collect these findings and obtain the following result for the optimal value of $t_{n,k}^{\text{LOC}}$:

$$(t_{n,k}^{\text{LOC}})^* = \begin{cases} \frac{m_{n,k}}{\nu_k^{\max}} & \beta_{n,k} \geq 2\kappa_k(\nu_k^{\max})^3 \\ m_{n,k}\left(\frac{2\kappa_k}{\beta_{n,k}}\right)^{1/3} & \frac{2\kappa_k(m_{n,k})^3}{(t_n)^3} < \beta_{n,k} < 2\kappa_k(\nu_k^{\max})^3 \\ t_n & \beta_{n,k} \leq \frac{2\kappa_k(m_{n,k})^3}{(t_n)^3} \end{cases} \tag{A.9}$$

A.0.2 Solution of Problem (A.4)

The Lagrangian for this problem is

$$\begin{aligned}\mathcal{L}^{②}_{n,k} &= E^{\mathrm{EXC_{RF_b}}}_{n,k} + t^{\mathrm{EXC_b}}_{n,k}\,(P^c_k + \beta_{n,k}) \\ &+ \mu^{①}_{n,k}(d^{\mathrm{col}}_n - t^{\mathrm{EXC_b}}_{n,k} r^{\mathrm{EXC_b}}_{n,k}) - \mu^{②}_{n,k} E^{\mathrm{EXC_{RF_b}}}_{n,k} \\ &+ \mu^{③}_{n,k}(E^{\mathrm{EXC_{RF_b}}}_{n,k} - t^{\mathrm{EXC_b}}_{n,k} p^{\max}_k) + \mu^{④}_{n,k}(t^{\mathrm{EXC_b}}_{n,k} - t_n)\end{aligned} \tag{A.10}$$

where $\mu^{①}_{n,k}, \mu^{②}_{n,k}, \mu^{③}_{n,k}, \mu^{④}_{n,k} \geq 0$ are the Lagrange multipliers. We have the following KKT conditions:

$$\frac{\partial \mathcal{L}^{②}_{n,k}}{\partial E^{\mathrm{EXC_{RF_b}}}_{n,k}} = 1 - \mu^{②}_{n,k} + \mu^{③}_{n,k} - \mu^{①}_{n,k} \frac{h^{\mathrm{EXC_b}}_{n,k}}{N_0 + \frac{h^{\mathrm{EXC_b}}_{n,k}}{B} \frac{E^{\mathrm{EXC_{RF_b}}}_{n,k}}{t^{\mathrm{EXC_b}}_{n,k}}} = 0 \tag{A.11}$$

and

$$\begin{aligned}\frac{\partial \mathcal{L}^{②}_{n,k}}{\partial t^{\mathrm{EXC_b}}_{n,k}} &= P^c_k + \beta_{n,k} - \mu^{③}_{n,k} p^{\max}_k + \mu^{④}_{n,k} - \mu^{①}_{n,k} r^{\mathrm{EXC_b}}_{n,k} \\ &+ \mu^{①}_{n,k} \frac{\frac{h^{\mathrm{EXC_b}}_{n,k}}{N_0} \frac{E^{\mathrm{EXC_{RF_b}}}_{n,k}}{t^{\mathrm{EXC_b}}_{n,k}}}{1 + \frac{h^{\mathrm{EXC_b}}_{n,k}}{BN_0} \frac{E^{\mathrm{EXC_{RF_b}}}_{n,k}}{t^{\mathrm{EXC_b}}_{n,k}}} = 0\end{aligned} \tag{A.12}$$

where the complementary slackness conditions are

$$\mu^{①}_{n,k}(d^{\mathrm{col}}_n - t^{\mathrm{EXC_b}}_{n,k} r^{\mathrm{EXC_b}}_{n,k}) = 0 \tag{A.13}$$

$$\mu^{②}_{n,k} E^{\mathrm{EXC_{RF_b}}}_{n,k} = 0 \tag{A.14}$$

$$\mu^{③}_{n,k}(E^{\mathrm{EXC_{RF_b}}}_{n,k} - t^{\mathrm{EXC_b}}_{n,k} p^{\max}_k) = 0 \tag{A.15}$$

$$\mu^{④}_{n,k}(t^{\mathrm{EXC_b}}_{n,k} - t_n) = 0 \tag{A.16}$$

We can find the optimal values of $E^{\mathrm{EXC_{RF_b}}}_{n,k}$ and $t^{\mathrm{EXC_b}}_{n,k}$ as follows:

- $E^{\mathrm{EXC_{RF_b}}}_{n,k} = 0, 0 < t^{\mathrm{EXC_b}}_{n,k} < t_n \Rightarrow \mu^{②}_{n,k} > 0$, $\mu^{③}_{n,k} = 0$ and $\mu^{④}_{n,k} = 0$:
  By using (A.11), $\mu^{②}_{n,k} = 1 - \mu^{①}_{n,k} \frac{h^{\mathrm{EXC_b}}_{n,k}}{N_0} > 0 \Rightarrow \mu^{①}_{n,k} < \frac{N_0}{h^{\mathrm{EXC_b}}_{n,k}}$.

- $0 < E_{n,k}^{\mathrm{EXC_{RF_b}}} < t_{n,k}^{\mathrm{EXC_b}} p_k^{\max} \Rightarrow \mu_{n,k}^{②} = \mu_{n,k}^{③} = 0$:

  From (A.11), $\mu_{n,k}^{①} = \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{E_{n,k}^{\mathrm{EXC_{RF_b}}}}{B t_{n,k}^{\mathrm{EXC_b}}} > 0$. We obtain the following relation:

  $$\frac{E_{n,k}^{\mathrm{EXC_{RF_b}}}}{t_{n,k}^{\mathrm{EXC_b}}} = B(\mu_{n,k}^{①} - \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}}) \tag{A.17}$$

  Additionally, because the multiplier $\mu_{n,k}^{①}$ is greater than zero, i.e. $\mu_{n,k}^{①} > 0$, the related constraint should be satisfied with equality to obey the complementary slackness condition in (A.13) so that the following equation occurs:

  $$d_n^{\mathrm{col}} = t_{n,k}^{\mathrm{EXC_b}} B \ln(1 + \frac{h_{n,k}^{\mathrm{EXC_b}}}{BN_0} \frac{E_{n,k}^{\mathrm{EXC_{RF_b}}}}{t_{n,k}^{\mathrm{EXC_b}}}) \tag{A.18}$$

  which means that the exchange communication is performed at the achievable data rate. By substituting (A.17) into (A.18), we can find an expression for $t_{n,k}^{\mathrm{EXC_b}}$ as follows:

  $$t_{n,k}^{\mathrm{EXC_b}} = \frac{d_n^{\mathrm{col}}}{B \ln(\frac{h_{n,k}^{\mathrm{EXC_b}}}{N_0} \mu_{n,k}^{①})} \tag{A.19}$$

  where due to the constraint $t_{n,k}^{\mathrm{EXC_b}} \leq t_n$, the multiplier $\mu_{n,k}^{①}$ should satisfy the following condition:

  $$\mu_{n,k}^{①} \geq \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} e^{\frac{d_n^{\mathrm{col}}}{Bt_n}} \tag{A.20}$$

  Furthermore, from the definition $0 < \frac{E_{n,k}^{\mathrm{EXC_{RF_b}}}}{t_{n,k}^{\mathrm{EXC_b}}} = B(\mu_{n,k}^{①} - \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}}) < p_k^{\max}$, we have

  $$\frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} < \mu_{n,k}^{①} < \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \tag{A.21}$$

- $E_{n,k}^{\mathrm{EXC_{RF_b}}} = t_{n,k}^{\mathrm{EXC_b}} p_k^{\max} \Rightarrow \mu_{n,k}^{②} = 0, \mu_{n,k}^{③} > 0$:
  From (A.11), we have

$$\mu_{n,k}^{③} = \mu_{n,k}^{①} \frac{h_{n,k}^{\mathrm{EXC_b}}}{N_0 + \frac{h_{n,k}^{\mathrm{EXC_b}}}{B} p_k^{\max}} - 1 > 0 \tag{A.22}$$

  where if $\mu_{n,k}^{①} = 0$, the multiplier $\mu_{n,k}^{③}$ is equal to -1, which is infeasible. Therefore, the multiplier $\mu_{n,k}^{①}$ should satisfy $\mu_{n,k}^{①} > 0$ and to obey the complementary slackness condition in (A.13), we have the following equation:

$$d_n^{\mathrm{col}} = t_{n,k}^{\mathrm{EXC_b}} B \ln(1 + \frac{h_{n,k}^{\mathrm{EXC_b}}}{BN_0} p_k^{\max}) \tag{A.23}$$

  Thus, we have the following expression for $t_{n,k}^{\mathrm{EXC_b}}$:

$$t_{n,k}^{\mathrm{EXC_b}} = \frac{d_n^{\mathrm{col}}}{B \ln(1 + \frac{h_{n,k}^{\mathrm{EXC_b}}}{BN_0} p_k^{\max})} = \frac{d_n^{\mathrm{col}}}{r_{n,k}^{\mathrm{EXC_b}}(p_k^{\max})} \tag{A.24}$$

  and from (A.22), the following constraint needs to be satisfied:

$$\mu_{n,k}^{①} > \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \tag{A.25}$$

Finally, the optimal values for the optimization parameters $p_{n,k}^{\mathrm{EXC_{RF_b}}}$ and $t_{n,k}^{\mathrm{EXC_b}}$ are obtained as

$$(p_{n,k}^{\mathrm{EXC_b}})^* = \begin{cases} 0 & \mu_{n,k}^{①} \leq \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} \\ B\left(\mu_{n,k}^{①} - \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}}\right) & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} < \mu_{n,k}^{①} < \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \\ p_k^{\max} & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \leq \mu_{n,k}^{①} \end{cases} \tag{A.26}$$

and

$$(t_{n,k}^{\mathrm{EXC_b}})^* = \begin{cases} t_n & \mu_{n,k}^{①} \leq \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} e^{\frac{d_n^{\mathrm{col}}}{Bt_n}} \\ \frac{d_n^{\mathrm{col}}}{B \ln\left(\frac{h_{n,k}^{\mathrm{EXC_b}}}{N_0} \mu_{n,k}^{①}\right)} & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} e^{\frac{d_n^{\mathrm{col}}}{Bt_n}} < \mu_{n,k}^{①} < \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \\ \frac{d_n^{\mathrm{col}}}{B \ln\left(1+\frac{h_{n,k}^{\mathrm{EXC_b}}}{BN_0} p_k^{\max}\right)} & \frac{N_0}{h_{n,k}^{\mathrm{EXC_b}}} + \frac{p_k^{\max}}{B} \leq \mu_{n,k}^{①} \end{cases} \tag{A.27}$$

Appendix B

Solution of External Optimization Problem in Chapter 3

We have the following partial Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathbf{y}_{n,k}, \mathbf{z}_{n,k}) = &\sum_{\substack{k=1\\k\neq\tilde{k}}}^{K}(E_k^{\mathrm{FRD_{RF}}} + t_k^{\mathrm{FRD}} P_{\tilde{k}}^{c}) \\ &+ \sum_{k=1}^{K}\sum_{n=1}^{N^{\mathrm{LAY}}}[cons_{n,k}^{①}(d_{1,k}^{\mathrm{row}})^3 - \beta_{n,k}t_n + \gamma_{n,k}^{①} cons_{n,k}^{②} d_{1,k}^{\mathrm{row}}] \\ &+ \lambda(d^{\mathrm{row}} - \sum_{k=1}^{K} d_{1,k}^{\mathrm{row}}) + \phi(\sum_{\substack{k=1\\k\neq\tilde{k}}}^{K} t_k^{\mathrm{FRD}} + \sum_{n=1}^{N^{\mathrm{LAY}}} t_n - \tau) \\ &+ \sum_{\substack{k=1\\k\neq\tilde{k}}}^{K}\theta_k[d^x(d_{1,k}^{\mathrm{row}} + 1 + I_{k,1}^{\mathrm{a}} I_{k,1}^{\mathrm{b}}) - t_k^{\mathrm{FRD}} r_k^{\mathrm{FRD}}]\end{aligned} \tag{B.1}$$

where $\mathbf{y}_{n,k} = \{E_k^{\mathrm{FRD_{RF}}}, t_k^{\mathrm{FRD}}, d_{1,k}^{\mathrm{row}}, t_n\}$ and $\mathbf{z}_{n,k} = \{\beta_{n,k}, \gamma_{n,k}^{①}, \theta_k, \lambda, \phi\}$. The constant terms in the Lagrangian can be expressed as

$$cons_{n,k}^{①} = \frac{3^6 \kappa_k c_k^3 (d_n^{\mathrm{col}})^3}{(2^{n-1})^3((t_{n,k}^{\mathrm{LOC}})^*)^2}, \quad cons_{n,k}^{②} = \frac{9 c_k d_n^{\mathrm{col}}}{2^{n-1} v_k^{\max}} \tag{B.2}$$

where $(t_{n,k}^{\mathrm{LOC}})^*$ is the optimal local computation time found in the internal optimization problem of layer-pack n operation in RCD k. Also, $\phi, \theta_k \geq 0$ and $\lambda \in \Re$ are the Lagrange multipliers. The dual function is given as

$$\begin{aligned} g(\mathbf{z}_{n,k}) = \min_{\mathbf{y}_{n,k}} \quad & \mathcal{L}(\mathbf{y}_{n,k}, \mathbf{z}_{n,k}) \\ \text{s.t.} \quad & 0 \leq E_k^{\mathrm{FRD_{RF}}} \leq t_k^{\mathrm{FRD}} p_{\widetilde{k}}^{\max} \\ & 0 \leq t_k^{\mathrm{FRD}}, t_n \leq \tau \\ & 0 \leq d_{1,k}^{\mathrm{row}} \leq d^{\mathrm{row}} \end{aligned} \tag{B.3}$$

The problem can be decomposed into three sub-problems. The first problem is

$$\begin{aligned} \min_{t_n} \quad & t_n(\phi - \sum_{k=1}^{K} \beta_{n,k}) \\ \text{s.t.} \quad & 0 \leq t_n \leq \tau, \end{aligned} \tag{B.4}$$

while the second problem is

$$\begin{aligned} \min_{d_{1,k}^{\mathrm{row}}} \quad & \sum_{k=1}^{K} (d_{1,k}^{\mathrm{row}})^3 \sum_{n=1}^{N^{\mathrm{LAY}}} cons_{n,k}^{①} + \sum_{k=1}^{K} d_{1,k}^{\mathrm{row}} \sum_{n=1}^{N^{\mathrm{LAY}}} \gamma_{n,k}^{①} cons_{n,k}^{②} \\ & - \sum_{k=1}^{K} \lambda \, d_{1,k}^{\mathrm{row}} + \sum_{\substack{k=1 \\ k \neq \widetilde{k}}}^{K} \theta_k d^x d_{1,k}^{\mathrm{row}} \\ \text{s.t.} \quad & 0 \leq d_{1,k}^{\mathrm{row}} \leq d^{\mathrm{row}} \end{aligned} \tag{B.5}$$

and the last problem is

$$\begin{aligned} \min_{E_k^{\mathrm{FRD_{RF}}}, t_k^{\mathrm{FRD}}} \quad & E_k^{\mathrm{FRD_{RF}}} + t_k^{\mathrm{FRD}}(P_{\widetilde{k}}^c + \phi) - \theta_k t_k^{\mathrm{FRD}} r_k^{\mathrm{FRD}} \\ \text{s.t.} \quad & 0 \leq E_k^{\mathrm{FRD_{RF}}} \leq t_k^{\mathrm{FRD}} p_{\widetilde{k}}^{\max}, \\ & 0 \leq t_k^{\mathrm{FRD}} \leq \tau \end{aligned} \tag{B.6}$$

B.0.1 Solution of Problem (B.4)

The Lagrangian regarding this problem is

$$\mathcal{L}_n^{③} = t_n(\phi - \sum_{k=1}^{K} \beta_{n,k}) + \epsilon_n^{①}(t_n - \tau) - \epsilon_n^{②} t_n \tag{B.7}$$

where $\epsilon_n^{①}, \epsilon_n^{②} \geq 0$ are the Lagrange multipliers and we have the following KKT condition with respect to the variable t_n:

$$\frac{\partial \mathcal{L}_n^{③}}{\partial t_n} = \phi - \sum_{k=1}^{K} \beta_{n,k} + \epsilon_n^{①} - \epsilon_n^{②} = 0 \tag{B.8}$$

where the complementary slackness conditions are

$$\epsilon_n^{①}(t_n - \tau) = 0 \tag{B.9}$$

$$\epsilon_n^{②} t_n = 0 \tag{B.10}$$

We can find the optimal value of t_n as

$$(t_n)^* = \begin{cases} 0 & \phi > \sum\limits_{k=1}^{K} \beta_{n,k} \\ (0, \tau) & \phi = \sum\limits_{k=1}^{K} \beta_{n,k} \\ \tau & \phi < \sum\limits_{k=1}^{K} \beta_{n,k} \end{cases} \tag{B.11}$$

B.0.2 Solution of Problem (B.5)

The problem (B.5) can be modified as

$$\begin{aligned} \min_{d_{1,k}^{\text{row}}} \quad & (d_{1,k}^{\text{row}})^3 \Big(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①} \Big) - d_{1,k}^{\text{row}} \psi_k \\ \text{s.t.} \quad & 0 \leq d_{1,k}^{\text{row}} \leq d^{\text{row}} \end{aligned} \tag{B.12}$$

where

$$\psi_k = \begin{cases} \lambda - \sum\limits_{n=1}^{N^{\text{LAY}}} \gamma_{n,k}^{①} cons_{n,k}^{②} & k = \widetilde{k} \\ \lambda - \sum\limits_{n=1}^{N^{\text{LAY}}} \gamma_{n,k}^{①} cons_{n,k}^{②} - \theta_k d^x & k \neq \widetilde{k} \end{cases} \tag{B.13}$$

The Lagrangian for (B.12) is

$$\mathcal{L}_k^{④} = (d_{1,k}^{\text{row}})^3 \Big(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①} \Big) - d_{1,k}^{\text{row}} \psi_k + \delta_k^{①} (d_{1,k}^{\text{row}} - d^{\text{row}}) - \delta_k^{②} d_{1,k}^{\text{row}} \tag{B.14}$$

where $\delta_k^{①}, \delta_k^{②} \geq 0$ are the Lagrange multipliers. We have the following KKT condition with respect to the variable $d_{1,k}^{\text{row}}$:

$$\frac{\partial \mathcal{L}_k^{④}}{\partial d_{1,k}^{\text{row}}} = 3(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①})(d_{1,k}^{\text{row}})^2 - \psi_k + \delta_k^{①} - \delta_k^{②} = 0 \tag{B.15}$$

where the complementary slackness conditions are

$$\delta_k^{①}(d_{1,k}^{\text{row}} - d^{\text{row}}) = 0 \tag{B.16}$$

$$\delta_k^{②} d_{1,k}^{\text{row}} = 0 \tag{B.17}$$

We can find the optimal value of $d_{1,k}^{\text{row}}$ as follows:

- $d_{1,k}^{\text{row}} = 0 \Rightarrow \delta_k^{①} = 0, \delta_k^{②} > 0$:
  From (B.15), $\delta_k^{②} = -\psi_k > 0 \Rightarrow \psi_k < 0$
- $0 < d_{1,k}^{\text{row}} < d^{\text{row}} \Rightarrow \delta_k^{①} = \delta_k^{②} = 0$:
  From (B.15), $d_{1,k}^{\text{row}} = \sqrt{\frac{\psi_k}{3(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①})}}$ and we have the following inequality by substituting $d_{1,k}^{\text{row}}$ expression into (B.15):
  $0 < \psi_k < 3(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①})(d^{\text{row}})^2$
- $d_{1,k}^{\text{row}} = d^{\text{row}} \Rightarrow \delta_k^{①} > 0, \delta_k^{②} = 0$:
  From (B.15), $\delta_k^{①} = \psi_k - 3(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①})(d^{\text{row}})^2 > 0$
  $\Rightarrow \psi_k > 3(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①})(d^{\text{row}})^2$

We obtain the optimal value of $d_{1,k}^{\text{row}}$ as

$$(d_{1,k}^{\text{row}})^* = \begin{cases} 0 & \psi_k \leq 0 \\ \sqrt{\frac{\psi_k}{3\left(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①}\right)}} & 0 < \psi_k < 3\left(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①}\right)(d^{\text{row}})^2 \\ d^{\text{row}} & \psi_k \geq 3\left(\sum_{n=1}^{N^{\text{LAY}}} cons_{n,k}^{①}\right)(d^{\text{row}})^2 \end{cases} \tag{B.18}$$

B.0.3 Solution of Problem (B.6)

This problem is similar to the exchange communication problem in Section A.0.2 of Appendix A with some exceptions. The Lagrangian of this

problem is

$$\begin{aligned}\mathcal{L}_k^{⑤} = E_k^{\mathrm{FRD_{RF}}} + t_k^{\mathrm{FRD}}(P_{\tilde{k}}^c + \phi) - \theta_k t_k^{\mathrm{FRD}} r_k^{\mathrm{FRD}} - \xi_k^{①} E_k^{\mathrm{FRD_{RF}}} \\ + \xi_k^{②}(E_k^{\mathrm{FRD_{RF}}} - t_k^{\mathrm{FRD}} p_{\tilde{k}}^{\max}) - \xi_k^{③} t_k^{\mathrm{FRD}} + \xi_k^{④}(t_k^{\mathrm{FRD}} - \tau)\end{aligned} \tag{B.19}$$

where $\xi_k^{①}, \xi_k^{②}, \xi_k^{③}, \xi_k^{④} \geq 0$ are the Lagrange multipliers and we have the following KKT conditions with respect to the variables $E_k^{\mathrm{FRD_{RF}}}$ and t_k^{FRD}:

$$\frac{\partial \mathcal{L}_k^{⑤}}{\partial E_k^{\mathrm{FRD_{RF}}}} = 1 - \xi_k^{①} + \xi_k^{②} - \theta_k \frac{h_k^{\mathrm{FRD}}}{N_0 + \frac{h_k^{\mathrm{FRD}}}{B}\frac{E_k^{\mathrm{FRD_{RF}}}}{t_k^{\mathrm{FRD}}}} = 0 \tag{B.20}$$

and

$$\begin{aligned}\frac{\partial \mathcal{L}_k^{⑤}}{\partial t_k^{\mathrm{FRD}}} =& P_{\tilde{k}}^c + \phi - \xi_k^{②} p_{\tilde{k}}^{\max} - \xi_k^{③} + \xi_k^{④} - \theta_k r_k^{\mathrm{FRD}} \\ &+ \theta_k \frac{\frac{h_k^{\mathrm{FRD}}}{N_0}\frac{E_k^{\mathrm{FRD_{RF}}}}{t_k^{\mathrm{FRD}}}}{1 + \frac{h_k^{\mathrm{FRD}}}{BN_0}\frac{E_k^{\mathrm{FRD_{RF}}}}{t_k^{\mathrm{FRD}}}} = 0\end{aligned} \tag{B.21}$$

where the complementary slackness conditions are

$$\xi_k^{①} E_k^{\mathrm{FRD_{RF}}} = 0 \tag{B.22}$$

$$\xi_k^{②}(E_k^{\mathrm{FRD_{RF}}} - t_k^{\mathrm{FRD}} p_{\tilde{k}}^{\max}) = 0 \tag{B.23}$$

$$\xi_k^{③} t_k^{\mathrm{FRD}} = 0 \tag{B.24}$$

$$\xi_k^{④}(t_k^{\mathrm{FRD}} - \tau) = 0 \tag{B.25}$$

By following the same path with the exchange communication problem solution in Section A.0.2 of Appendix A, we can find the optimal values of $(p_k^{\mathrm{FRD}})^* = (E_k^{\mathrm{FRD_{RF}}})^*/(t_k^{\mathrm{FRD}})^*$ and $(t_k^{\mathrm{FRD}})^*$ as follows:

$$(p_k^{\mathrm{FRD}})^* = \begin{cases} 0 & \theta_k \leq \frac{N_0}{h_k^{\mathrm{FRD}}} \\ B\left(\theta_k - \frac{N_0}{h_k^{\mathrm{FRD}}}\right) & \frac{N_0}{h_k^{\mathrm{FRD}}} < \theta_k < \frac{N_0}{h_k^{\mathrm{FRD}}} + \frac{p_{\tilde{k}}^{\max}}{B} \\ p_{\tilde{k}}^{\max} & \frac{N_0}{h_k^{\mathrm{FRD}}} + \frac{p_{\tilde{k}}^{\max}}{B} \leq \theta_k \end{cases} \tag{B.26}$$

and

$$(t_k^{\mathrm{FRD}})^* = \begin{cases} 0 & \theta_k < cons_k^{③} \\ (0,\tau) & \theta_k = cons_k^{③} \\ \tau & \theta_k > cons_k^{③} \end{cases} \tag{B.27}$$

Appendix C

Solution of Internal Optimization Problem in Chapter 4

By using the Lagrange duality theory, we have the following Lagrangian for (4.26):

$$\begin{aligned}\mathcal{L}(\mathbf{x}_3, d_{n,k}^{\mathrm{ch}}, t_n^{\mathrm{BC}}, t_n, \beta_{n,k}) = & \frac{36^3 (d^{\mathrm{row}})^3 (d^{\mathrm{col}})^3 \kappa_k c_k^3}{64^n} \frac{(d_{n,k}^{\mathrm{ch}})^3}{(t_{n,k}^{\mathrm{LOC}})^2} \\ & + (E_{n,k}^{\mathrm{Agg_{RF}}} + t_{n,k}^{\mathrm{Agg}} P_k^c) + \beta_{n,k}(t_n^{\mathrm{BC}} + t_{n,k}^{\mathrm{LOC}} + t_{n,k}^{\mathrm{Agg}} - t_n)\end{aligned} \tag{C.1}$$

where $\beta_{n,k} \geq 0$ is the Lagrange multiplier associated with (4.14). Then, the dual function is

$$\begin{aligned} g(d_{n,k}^{\mathrm{ch}}, t_n^{\mathrm{BC}}, t_n, \beta_{n,k}) = \min_{\mathbf{x}_3} \quad & \mathcal{L}(\mathbf{x}_3, d_{n,k}^{\mathrm{ch}}, t_n^{\mathrm{BC}}, t_n, \beta_{n,k}) \\ \text{s.t.} \quad & (4.8), (4.11), (4.23), \\ & 0 \leq E_{n,k}^{\mathrm{Agg_{RF}}}, \\ & 0 \leq t_{n,k}^{\mathrm{LOC}}, t_{n,k}^{\mathrm{Agg}} \leq t_n \end{aligned} \tag{C.2}$$

where we can solve this problem by separating into two sub-problems: one for local computation and one for aggregation.

The aggregation problem for $n \in [N^{\mathrm{LAY}}]$ and $k \in [K_n] - \{\widetilde{k}\}$ is

$$\begin{aligned} \underset{E_{n,k}^{\mathrm{Agg_{RF}}}, t_{n,k}^{\mathrm{OT}}}{\min.} \quad & E_{n,k}^{\mathrm{Agg_{RF}}} + t_{n,k}^{\mathrm{Agg}}(P_k^c + \beta_{n,k}) \\ \text{s.t.} \quad & (4.11), (4.23), \\ & 0 \leq E_{n,k}^{\mathrm{Agg_{RF}}}, \\ & 0 \leq t_{n,k}^{\mathrm{Agg}} \leq t_n \end{aligned} \tag{C.3}$$

while the local computation problem for $n \in [N^{\mathrm{LAY}}]$ and $k \in [K_n]$ is

$$\begin{aligned} \underset{t_{n,k}^{\mathrm{LOC}}}{\min.} \quad & \frac{36^3 (d^{\mathrm{row}})^3 (d^{\mathrm{col}})^3 \kappa_k c_k^3}{64^n} \frac{(d_{n,k}^{\mathrm{ch}})^3}{(t_{n,k}^{\mathrm{LOC}})^2} + \beta_{n,k} t_{n,k}^{\mathrm{LOC}} \\ \text{s.t.} \quad & 36 \frac{c_k}{\nu_k^{\max}} \frac{d^{\mathrm{row}} d^{\mathrm{col}}}{4^n} d_{n,k}^{\mathrm{ch}} \leq t_{n,k}^{\mathrm{LOC}} \leq t_n \end{aligned} \tag{C.4}$$

By following the Lagrange duality theory, we can find the solutions of the problems (C.3) and (C.4) as

$$(t_{n,k}^{\mathrm{Agg}})^* = \begin{cases} t_n & (\mu_{n,k}^{\mathrm{Agg}})' \leq 0 \\ \dfrac{4^{-n} d^{\mathrm{row}} d^{\mathrm{col}}}{B \ln\left(1 + \frac{h_{n,k}^{\mathrm{Agg}}}{B N_0 l(R_{n,k}^{\mathrm{Agg}})} (\mu_{n,k}^{\mathrm{Agg}})'\right)} & 0 < (\mu_{n,k}^{\mathrm{Agg}})' < p_k^{\max} \\ \dfrac{4^{-n} d^{\mathrm{row}} d^{\mathrm{col}}}{B \ln\left(1 + \frac{h_{n,k}^{\mathrm{Agg}}}{B N_0 l(R_{n,k}^{\mathrm{Agg}})} p_k^{\max}\right)} & p_k^{\max} \leq (\mu_{n,k}^{\mathrm{Agg}})' \end{cases} \tag{C.5}$$

$$(p_{n,k}^{\mathrm{Agg}})^* = \begin{cases} 0 & (\mu_{n,k}^{\mathrm{Agg}})' \leq 0 \\ (\mu_{n,k}^{\mathrm{Agg}})' & 0 < (\mu_{n,k}^{\mathrm{Agg}})' < p_k^{\max} \\ p_k^{\max} & p_k^{\max} \leq (\mu_{n,k}^{\mathrm{Agg}})' \end{cases} \tag{C.6}$$

and

$$(t_{n,k}^{\mathrm{LOC}})^* = \begin{cases} \frac{m_{n,k} d_{n,k}^{\mathrm{ch}}}{\nu_k^{\max}} & \beta'_{n,k} \leq \frac{m_{n,k} d_{n,k}^{\mathrm{ch}}}{\nu_k^{\max}} \\ \beta'_{n,k} & \frac{m_{n,k} d_{n,k}^{\mathrm{ch}}}{\nu_k^{\max}} < \beta'_{n,k} < t_n \\ t_n & \beta'_{n,k} \geq t_n \end{cases} \tag{C.7}$$

where $(\mu_{n,k}^{\mathrm{Agg}})' = B(\mu_{n,k}^{\mathrm{Agg}} - \frac{N_0 l(R_{n,k}^{\mathrm{Agg}})}{h_{n,k}^{\mathrm{Agg}}})$, $\beta'_{n,k} = m_{n,k} d_{n,k}^{\mathrm{ch}} (\frac{\beta_{n,k}}{2\kappa_k})^{-1/3}$ and $m_{n,k} = 36 d^{\mathrm{row}} d^{\mathrm{col}} c_k / 4^n$ are defined for easier notation where $\mu_{n,k}^{\mathrm{Agg}}$ is the Lagrange

multiplier associated with (4.11). The problem solutions are similar to the ones in Appendices A and B.

For aggregation stage, the interpretation of the results are as follows:

- The first case $(\mu_{n,k}^{\text{Agg}})' \leq 0$ means there is no aggregation stage for RCD k, so it does not participate to layer-pack n operation.
- The second case $0 < (\mu_{n,k}^{\text{Agg}})' < p_k^{\max}$ indicates that RCD k participates to layer-pack n operation with the RF transmit power lower than the maximum $p_k^{\max}$.
- The third case $p_k^{\max} \leq (\mu_{n,k}^{\text{Agg}})'$ is the one where RCD k participates to layer-pack n operation with the maximum RF transmit power $p_k^{\max}$.

For local computation, the resulting optimal solution $(t_{n,k}^{\text{LOC}})^*$ can be explained as

- In the first case $\beta'_{n,k} \leq \frac{m_{n,k} d_{n,k}^{\text{ch}}}{\nu_k^{\max}}$, CPU of RCD k works at its maximum frequency during the local computation of layer-pack n operation.
- In the second case $\frac{m_{n,k} d_{n,k}^{\text{ch}}}{\nu_k^{\max}} < \beta'_{n,k} < t_n$, CPU of RCD k works at the frequency lower than the maximum allowed frequency $\nu_k^{\max}$.
- In the last case $\beta'_{n,k} \geq t_n$, the entire time is allocated to the local computation which is feasible for the master RCD but infeasible for the slave RCDs.

Appendix D

Solution of External Optimization Problem in Chapter 4

Given the optimal parameters of the internal optimization problem, we have the following Lagrangian for (4.27):

$$\begin{aligned}\mathcal{L}(\mathbf{x}_4, \beta_{n,k}, \lambda_n, \phi) = &\sum_{n=1}^{N^{\mathrm{LAY}}} (E_n^{\mathrm{BC_{RF}}} + t_n^{\mathrm{BC}} P_{\tilde{k}}^{c}) \\ &+ \sum_{n=1}^{N^{\mathrm{LAY}}} \sum_{k=1}^{K_n} \left[\frac{\kappa_k (m_{n,k})^3}{((t_{n,k}^{\mathrm{LOC}})^*)^2} (d_{n,k}^{\mathrm{ch}})^3 + \beta_{n,k} (t_n^{\mathrm{BC}} - t_n) \right] \\ &+ \sum_{n=1}^{N^{\mathrm{LAY}}} \lambda_n (d_n^{\mathrm{ch}} - \sum_{k=1}^{K_n} d_{n,k}^{\mathrm{ch}}) + \phi (\sum_{n=1}^{N^{\mathrm{LAY}}} t_n - \tau)\end{aligned} \tag{D.1}$$

where $\lambda_n \in \Re$ and $\phi \geq 0$ are the Lagrange multipliers associated with (4.13) and (4.16), respectively. The dual function is given as

$$\begin{aligned} g(\beta_{n,k}, \lambda_n, \phi) = \min_{\mathbf{x}_4} \quad & \mathcal{L}(\mathbf{x}_4, \beta_{n,k}, \lambda_n, \phi) \\ \text{s.t.} \quad & (4.4), (4.20), \\ & 0 \leq d_{n,k}^{\mathrm{ch}}, t_n, t_n^{\mathrm{BC}}, E_n^{\mathrm{BC_{RF}}} \end{aligned} \tag{D.2}$$

The external optimization problem can be decomposed into three sub-problems where the first problem is

$$\begin{aligned} \min_{t_n} \quad & t_n(\phi - \sum_{k=1}^{K_n} \beta_{n,k}) \\ \text{s.t.} \quad & 0 \leq t_n \leq \tau \end{aligned} \tag{D.3}$$

while the second problem is defined as

$$\begin{aligned} \min_{d_{n,k}^{\text{ch}}} \quad & \frac{\kappa_k (m_{n,k})^3}{((t_{n,k}^{\text{LOC}})^*)^2} (d_{n,k}^{\text{ch}})^3 - \lambda_n d_{n,k}^{\text{ch}} \\ \text{s.t.} \quad & 0 \leq d_{n,k}^{\text{ch}} \leq d_n^{\text{ch}} \end{aligned} \tag{D.4}$$

and the third problem is

$$\begin{aligned} \min_{E_n^{\text{BC}_{\text{RF}}}, t_n^{\text{BC}}} \quad & E_n^{\text{BC}_{\text{RF}}} + t_n^{\text{BC}} (P_{\tilde{k}}^c + \sum_{k=1}^{K_n} \beta_{n,k}) \\ \text{s.t.} \quad & (4.4), (4.20), \\ & 0 \leq E_n^{\text{BC}_{\text{RF}}}, \\ & 0 \leq t_n^{\text{BC}} \leq t_n \end{aligned} \tag{D.5}$$

We find the optimal parameters as

$$(t_n)^* = \begin{cases} 0 & \phi > \sum_{k=1}^{K_n} \beta_{n,k} \\ (0, \tau) & \phi = \sum_{k=1}^{K_n} \beta_{n,k} \\ \tau & \phi < \sum_{k=1}^{K_n} \beta_{n,k} \end{cases} \tag{D.6}$$

$$(d_{n,k}^{\text{ch}})^* = \begin{cases} 0 & \lambda_n' \leq 0 \\ \lambda_n' & 0 < \lambda_n' < d_n^{\text{ch}} \\ d_n^{\text{ch}} & \lambda_n' \geq d_n^{\text{ch}} \end{cases} \tag{D.7}$$

$$(t_n^{\mathrm{BC}})^* = \begin{cases} t_n & (\mu_n^{\mathrm{BC}})' \leq 0 \\ \dfrac{4^{1-n} d_y d_x}{B \ln\left(1+\frac{\min_k\left(h_{n,k}^{\mathrm{BC}}/l(R_{n,k}^{\mathrm{BC}})\right)}{BN_0}(\mu_n^{\mathrm{BC}})'\right)} & 0 < (\mu_n^{\mathrm{BC}})' < p_{\tilde{k}}^{\max} \\ \dfrac{4^{1-n} d_y d_x}{B \ln\left(1+\frac{\min_k\left(h_{n,k}^{\mathrm{BC}}/l(R_{n,k}^{\mathrm{BC}})\right)}{BN_0} p_{\tilde{k}}^{\max}\right)} & p_{\tilde{k}}^{\max} \leq (\mu_n^{\mathrm{BC}})' \end{cases} \tag{D.8}$$

$$(p_n^{\mathrm{BC}})^* = \begin{cases} 0 & (\mu_n^{\mathrm{BC}})' \leq 0 \\ (\mu_n^{\mathrm{BC}})' & 0 < (\mu_n^{\mathrm{BC}})' < p_{\tilde{k}}^{\max} \\ p_{\tilde{k}}^{\max} & p_{\tilde{k}}^{\max} \leq (\mu_n^{\mathrm{BC}})' \end{cases} \tag{D.9}$$

where $\lambda_n' = (t_{n,k}^{\mathrm{LOC}})^* \sqrt{\frac{\lambda_n}{3\kappa_k (m_{n,k})^3}}$, $(\mu_n^{\mathrm{BC}})' = B(\mu_n^{\mathrm{BC}} - \frac{N_0}{\min_k\left(h_{n,k}^{\mathrm{BC}}/l(R_{n,k}^{\mathrm{BC}})\right)})$ and μ_n^{BC} is the Lagrange multiplier associated with (4.4). The interpretation of broadcasting parameters is similar to the one for aggregation in Appendix C while for t_n, since it is not feasible to have $t_n = 0$ or $t_n = \tau$, the equation $\phi = \sum_{k=1}^{K_n} \beta_{n,k}$ needs to be satisfied. For $d_{n,k}^{\mathrm{ch}}$, the optimal value is λ_n'. It cannot be either 0 if RCD k participates to layer-pack n operation, or d_n^{ch} since the proposed scheme is a distributed collaborative system.

Bibliography

[1] H. Wu, Z. Zhang, C. Guan, K. Wolter and M. Xu, "Collaborate Edge and Cloud Computing With Distributed Deep Learning for Smart City Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099-8110, Sept. 2020.

[2] F. Wang, M. Zhang, X. Wang, X. Ma and J. Liu, "Deep Learning for Edge Computing Applications: A State-of-the-Art Survey," *IEEE Access*, vol. 8, pp. 58322-58336, 2020.

[3] X. Ran, H. Chen, X. Zhu, Z. Liu and J. Chen, "DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, USA, pp. 1421-1429, 2018.

[4] X. Tang, X. Chen, L. Zeng, S. Yu and L. Chen, "Joint Multiuser DNN Partitioning and Computational Resource Allocation for Collaborative Edge Intelligence," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9511-9522, June 2021.

[5] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655-1674, Aug. 2019.

[6] K. Yang, Y. Shi and Z. Ding, "Data Shuffling in Wireless Distributed Computing via Low-Rank Optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3087-3099, June 2019.

[7] R. Hadidi, J. Cao, M. S. Ryoo and H. Kim, "Toward Collaborative Inferencing of Deep Neural Networks on Internet-of-Things Devices," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4950-4960, June 2020.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv:1510.00149, [Online]*, 2015, Available: https://arxiv.org/abs/1510.00149.

[9] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*, pp. 389–400, 2018.

[10] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[11] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating While Computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[12] L. Guan, X. Ke, M. Song and J. Song, "A Survey of Research on Mobile Cloud Computing," *10th IEEE/ACIS International Conference on Computer and Information Science*, Sanya, China, pp. 387-392, 2011.

[13] X. Cao, F. Wang, J. Xu, R. Zhang and S. Cui, "Joint Computation and Communication Cooperation for Energy-Efficient Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4188-4200, June 2019.

[14] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas and S. Khaf, "A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing," *IEEE Access*, vol. 7, pp. 149623-149633, 2019.

[15] C. -Y. Yang, J. -J. Kuo, J. -P. Sheu and K. -J. Zheng, "Cooperative Distributed Deep Neural Network Deployment with Edge Computing," *ICC 2021 - IEEE International Conference on Communications*, Montreal, QC, Canada, pp. 1-6, 2021.

[16] E. El Haber, T. M. Nguyen and C. Assi, "Joint Optimization of Computational Cost and Devices Energy for Task Offloading in Multi-Tier Edge-Clouds," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3407-3421, May 2019.

[17] J. Du, M. Shen and Y. Du, "A Distributed In-Situ CNN Inference System for IoT Applications," *IEEE 38th International Conference on Computer Design (ICCD)*, Hartford, CT, USA, pp. 279-287, 2020.

[18] Y. Shi, K. Yang, T. Jiang, J. Zhang and K. B. Letaief, "Communication-Efficient Edge AI: Algorithms and Systems," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2167-2191, 2020.

[19] "Mobile-edge computing-Introductory technical white paper," White Paper, ETSI, Sophia Antipolis, France, Sep. 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf.

[20] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

[21] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," *10th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India, pp. 1-8, 2016.

[22] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757-6779, 2017.

[23] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[24] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[25] Multi-Access Edge Computing. Accessed: Jun. 30, 2019. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing.

[26] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869-904, 2020.

[27] 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," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[28] E. Cui, W. Zhang, D. Yang, W. Wu and F. Lyu, "Resource-Efficient DNN Training and Inference for Heterogeneous Edge Intelligence in 6G," *IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Haikou, Hainan, China, pp. 837-845, 2021.

[29] 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 Journal*, vol. 7, no. 8, pp. 7457-7469, Aug. 2020.

[30] 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.

[31] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[32] A. E. Eshratifar, M. S. Abrishami and M. Pedram, "JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565-576, Feb. 2021.

[33] S. Teerapittayanon, B. McDanel and H. T. Kung, "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices," *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, pp. 328-339, 2017.

[34] 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, Jan. 2020.

[35] C. Hu, W. Bao, D. Wang and F. Liu, "Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Paris, France, pp. 1423-1431, 2019.

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

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

[38] 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 Transactions on Networking*, vol. 29, no. 2, pp. 595-608, April 2021.

[39] C. -C. Hsu, C. -K. Yang, J. -J. Kuo, W. -T. Chen and J. -P. Sheu, "Cooperative Convolutional Neural Network Deployment over Mobile Networks," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, pp. 1-7, 2020.

[40] L. Zeng, E. Li, Z. Zhou and X. Chen, "Boomerang: On-Demand Cooperative Deep Neural Network Inference for Edge Intelligence on the Industrial Internet of Things," *IEEE Network*, vol. 33, no. 5, pp. 96-103, Sept.-Oct. 2019.

[41] W. He, S. Guo, S. Guo, X. Qiu and F. Qi, "Joint DNN Partition Deployment and Resource Allocation for Delay-Sensitive Deep Learning Inference in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9241-9254, Oct. 2020.

[42] F. M. Campos de Oliveira and E. Borin, "Partitioning Convolutional Neural Networks for Inference on Constrained Internet-of-Things Devices," *30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Lyon, France, pp. 266-273, 2018.

[43] T. Mohammed, C. Joe-Wong, R. Babbar and M. D. Francesco, "Distributed Inference Acceleration with Adaptive DNN Partitioning and Offloading," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Toronto, ON, Canada, pp. 854-863, 2020.

[44] C. Shi, L. Chen, C. Shen, L. Song and J. Xu, "Privacy-Aware Edge Computing Based on Adaptive DNN Partitioning," *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, pp. 1-6, 2019.

[45] C. Dong, S. Hu, X. Chen and W. Wen, "Joint Optimization With DNN Partitioning and Resource Allocation in Mobile Edge Computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 3973-3986, Dec. 2021.

[46] F. Xue, W. Fang, W. Xu, Q. Wang, X. Ma and Y. Ding, "EdgeLD: Locally Distributed Deep Learning Inference on Edge Device Clusters," *IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Yanuca Island, Cuvu, Fiji, pp. 613-619, 2020.

[47] H. -J. Jeong, H. -J. Lee, K. Yong Shin, Y. Hwan Yoo and S. -M. Moon, "PerDNN: Offloading Deep Neural Network Computations

to Pervasive Edge Servers," *IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, Singapore, Singapore, pp. 1055-1066, 2020.

[48] 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.

[49] L. Lockhart, P. Harvey, P. Imai, P. Willis and B. Varghese, "Scission: Performance-driven and Context-aware Cloud-Edge Distribution of Deep Neural Networks," *IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, Leicester, UK, pp. 257-268, 2020.

[50] M. Gao, R. Shen, L. Shi, W. Qi, J. Li and Y. Li, "Task Partitioning and Offloading in DNN-Task Enabled Mobile Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2435-2445, April 2023.

[51] E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Collaborative On-Device CNN Execution Considering Model Parallelism for Latency-Critical Applications," *IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Toronto, ON, Canada, pp. 1-7, 2023.

[52] E. Kilcioglu, I. Stupia and L. Vandendorpe, "A Relay-Assisted Communication Scheme for Collaborative On-Device CNN Execution Considering Hybrid Parallelism," *IEEE Access*, vol. 11, pp. 99397-99412, 2023.

[53] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

[54] W. Y. B. Lim *et al.*, "UAV-Assisted Communication Efficient Federated Learning in the Era of the Artificial Intelligence of Things," *IEEE Network*, vol. 35, no. 5, pp. 188-195, Sept./Oct. 2021.

[55] Z. Qu *et al.*, "Partial Synchronization to Accelerate Federated Learning Over Relay-Assisted Edge Networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4502-4516, Dec. 2022.

[56] Z. Lin, H. Liu and Y. -J. A. Zhang, "Relay-Assisted Cooperative Federated Learning," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7148-7164, Sept. 2022.

[57] S. Feng, D. Niyato, P. Wang, D. I. Kim and Y. -C. Liang, "Joint Service Pricing and Cooperative Relay Communication for Federated Learning," *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Atlanta, GA, USA, pp. 815-820, 2019.

[58] M. S. Al-Abiad, M. Z. Hassan and M. J. Hossain, "Energy-Efficient Resource Allocation for Federated Learning in NOMA-Enabled and Relay-Assisted Internet of Things Networks," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 24736-24753, Dec. 2022.

[59] S. H. Alsamhi *et al.*, "Drones' Edge Intelligence Over Smart Environments in B5G: Blockchain and Federated Learning Synergy," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 295-312, March 2022.

[60] X. Zhang, R. Chen, J. Wang, H. Zhang and M. Pan, "Energy Efficient Federated Learning over Cooperative Relay-Assisted Wireless Networks," *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Rio de Janeiro, Brazil, pp. 179-184, 2022.

[61] P. Li, Y. Zhong, C. Zhang, Y. Wu and R. Yu, "FedRelay: Federated Relay Learning for 6G Mobile Edge Intelligence," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 5125-5138, April 2023.

[62] C. Zhang, P. Patras and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224-2287, 2019.

[63] G. Hinton, O. Vinyals and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531, [Online]*, 2015, Available: https://arxiv.org/abs/1503.02531.

[64] S. Teerapittayanon, B. McDanel and H.T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks," *arXiv:1709.01686, [Online]*, 2017, Available: https://arxiv.org/abs/1709.01686.

[65] A. Zniber, O. Karrakchou and M. Ghogho, "Dynamic Early Exiting Predictive Coding Neural Networks," *arXiv:2309.02022, [Online]*, 2023, Available: https://arxiv.org/abs/2309.02022.

[66] J. Mao *et al.*, "MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs," *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Irvine, CA, pp. 751-756, 2017.

[67] D. P. Palomar and Mung Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439-1451, Aug. 2006.

[68] M. Mishra, "Convolutional Neural Networks, Explained," *towardsdatascience.com*, Aug. 26, 2020.

[69] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," *towardsdatascience.com*, Dec. 15, 2018.

[70] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming," Sept. 2013, Available: http://cvxr.com/cvx/.

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

[72] V. Sze, Y. Chen, T. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.

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

[74] A. Paris, H. Mirghasemi, I. Stupia and L. Vandendorpe, "Leveraging User-Diversity in Energy-Efficient Edge-Facilitated Collaborative Fog Computing," *IEEE Access*, vol. 9, pp. 95636-95650, 2021.

[75] X. Cao, F. Wang, J. Xu, R. Zhang and S. Cui, "Joint Computation and Communication Cooperation for Energy-Efficient Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4188-4200, June 2019.

[76] N. Janatian, I. Stupia and L. Vandendorpe, "Optimal Offloading Strategy and Resource Allocation in SWIPT-based Mobile-Edge Computing Networks," *15th International Symposium on Wireless Communication Systems (ISWCS)*, Lisbon, pp. 1-6, 2018.

[77] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, Nov. 2012.

[78] F. Mattsson, "Estimating energy consumption of Wifi transceiver circuits on a single board

computer," 2021, Available: https://www.diva-portal.org/smash/get/diva2:1604008/FULLTEXT01.pdf.

[79] "The GFLOPS/W of the various machines in the VMW Research Group," Available: https://web.eece.maine.edu/ vweaver/group/green_machines.html.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Int. Conf. Neural Inf. Process. Syst., pp. 1097–1105, 2012.