electronics

MDPI

Article

Impact of Input Data on Intelligence Partitioning Decisions for IoT Smart Camera Nodes

Isaac Sánchez Leal [1,*], Irida Shallari [1], Silvia Krug [1,2], Axel Jantsch [1,3] and Mattias O'Nils [1]

[1] Department of Electronics Design, Mid Sweden University, Holmgatan 10, 851 70 Sundsvall, Sweden; irida.shallari@miun.se (I.S.); silvia.krug@imms.de (S.K.); axel.jantsch@tuwien.ac.at (A.J.); mattias.onils@miun.se (M.O.)

[2] System Design Department, IMMS Institut für Mikroelektronik und Mechatronik-Systeme Gemeinnützige GmbH (IMMS GmbH), Ehrenbergstraße 27, 98693 Ilmenau, Germany

[3] Institute of Computer Technology, TU Wien (Vienna University of Technology), Gusshausstrasse 27-29/384, 1040 Vienna, Austria

* Correspondence: isaac.sanchezleal@miun.se

Abstract: Image processing systems exploit image information for a purpose determined by the application at hand. The implementation of image processing systems in an Internet of Things (IoT) context is a challenge due to the amount of data in an image processing system, which affects the three main node constraints: memory, latency and energy. One method to address these challenges is the partitioning of tasks between the IoT node and a server. In this work, we present an in-depth analysis of how the input image size and its content within the conventional image processing systems affect the decision on where tasks should be implemented, with respect to node energy and latency. We focus on explaining how the characteristics of the image are transferred through the system until finally influencing partition decisions. Our results show that the image size affects significantly the efficiency of the node offloading configurations. This is mainly due to the dominant cost of communication over processing as the image size increases. Furthermore, we observed that image content has limited effects in the node offloading analysis.

Keywords: camera node optimization; intelligence partitioning; input changes; inter-task data amount; IoT; WVSN

Citation: Leal, I.S.; Shallari, I.; Krug, S.; Jantsch, A.; O'Nils, M. Impact of Input Data on Intelligence Partitioning Decisions for IoT Smart Camera Nodes. *Electronics* **2021**, *10*, 1898. https://doi.org/10.3390/electronics10161898

Academic Editors: Abdellah Touhafi and Gianluca Cornetta

Received: 10 July 2021
Accepted: 3 August 2021
Published: 7 August 2021

1. Introduction

Internet of Things (IoT) applications based on smart and distributed cameras have gained considerable attention in the last decades and become very popular. There are numerous applications that require the integration of Wireless Vision Sensor Networks (WVSN), such as traffic monitoring in smart cities [1], security and inspection during production in Industry 4.0 [2,3], the diagnosis of diseases in healthcare [4] and self-driving cars in the automotive sector [5]. The continuous advancement in the different sectors means that more and more applications require greater speed, greater autonomy, more data and a smaller system size. As a consequence, WVSN systems must have low latency and low power consumption.

The demands on WVSN make integrating image processing systems in IoT challenging due to the main characteristics and somehow contrasting nature of image processing and IoT. On the one hand, we have image processing systems that rely on high data volumes and require significant processing capabilities while, on the other hand, we have the IoT, relying on wireless communication and developed with focus on low data volume applications. Thus, the contrasting nature of the two defines the core challenges in designing image processing-based IoT systems.

To understand how the amount of data affects the processing in the camera node, we need to analyze the relationship between the application requirements and the node

constraints. For example, the smaller size requirement (to reduce weight in drones or hide its location in vehicles) affects all camera components but especially the battery [6]; for this reason, the autonomy of the camera could be compromised. In addition, a low latency requirement in the IoT system requires a general increase in speed at the node, affecting both the processing and communication parts and resulting in an overall increase of energy consumption. Furthermore, the requirement of handling more data means using more memory and increment in both computational and communication load, thereby affecting both energy and latency at the node. Therefore, the smart camera node is undoubtedly a crucial component in an IoT system with the data captured and processed by the system a main factor affecting processing and communication requirements.

Image processing systems consist of a series of image processing tasks that are typically sequential, and the initial input image is reduced as it progresses through the image processing pipeline. As a result, the data volume is changing through the processing stages, which has a significant importance in an IoT node if we consider node offloading. From a node offloading perspective, the performance of an IoT node is defined by the inter-dependencies of the processing and the communication component. In our previous work, we have introduced our node offloading method, intelligence partitioning [7], which relies on the computational and communication inter-dependencies to extract the optimal partition point as a trade-off between energy efficiency and latency. Throughout that analysis, we considered the input image as predefined; hence, its characteristics such as size and content were not included in the performance analysis while searching for the optimum partition point. The aim of this paper is to provide an analysis on how the change of the input image in terms of size and content affects the choice of the optimal partition point for node. To address this, we provide an analysis based on a traditional image processing system which we have implemented on a RaspberryPi and measured the energy consumption for different partition and input image configurations. Our aim is to achieve a better understanding of how to optimize data-intensive IoT nodes. Thus, we provide a method to study a broader design space for image processing systems with intelligence partitioning which enables designers to choose appropriate input options.

As a motivating example, we propose an industrial biscuit inspection application. The system to analyze combines IoT technology with computer vision; in this way, the factory benefits from the advantages offered by both systems. On the one hand, IoT technology makes the vision system able to work wireless. It also provides the factory with the necessary flexibility to carry out future adaptations of the production system, avoiding rewiring and costly installations. On the other hand, the vision system is responsible for detecting cookies that do not meet the quality criteria (size and shape) and providing their coordinates. The camera node must capture the images of the biscuits that are transported by a conveyor belt at a certain speed, so image processing time is vital. In addition, the autonomy of the system is important to avoid production stoppages due to battery changes. To meet the latency and power demands, we decided to apply the system partition, where the challenge is to find the optimal partition point that guarantees low latency and low power consumption. Our hypothesis is based on the fact that the characteristics of the image (size and content) will have a non-linear effect on the latency and power of the system, thus affecting partition decisions. Through the study of this application, we intend to understand and explain the importance of image size in partitioning decisions, which together with the type of processing platform, type of communication and type of optimization, will influence the latency and energy in each of the partitions. The challenges found in the motivating example (reducing latency and power consumption at the node) are currently found in most wireless vision systems, where limited node resources force us to adopt creative design solutions (such as partitions).

The article is structured as follows. First, we analyze how the inter-task data amount varies along the processing chain with respect to changes in the image. Next, we analyze the dependency between processing time of a task and data to handle by this task to see how changes in the amount of data affect the processing time. Then, we study and compare

the latency and energy behaviors of the processing and communication components of each partition point in an example image processing system. Finally, we look at how changes in the image affect the optimal partitioning solutions based on the energy and latency optimization objectives at the node.

2. Related Work

The integration of image processing systems in the context of IoT continues to pose challenges. Energy and latency are often primarily responsible for the current research effort [8–12] to contribute to the overall improvement of these systems. Smart cameras with integrated image analysis capabilities are among the most challenging IoT nodes. The data intensive nature of such systems can impose high energy consumption and long delays in the wireless communication and the processing tasks. Thus, deploying low-latency/energy data-intensive IoT nodes is a challenging task.

To address this, several approaches have been presented, relying on fine-tuned implementations to improve the node latency and energy. Abas et al. [13] achieved energy efficiency in the node by adapting the activity of the camera to the amount of remaining battery charge. In addition, the system only records and transmits information when events of interest are detected. Anagnostou et al. [14] presented an energy aware hardware activation scheduler that activates functions as the energy is available. Qurabat et al. [15] improved the energy efficiency proposing data reduction applied at two levels of network design: the sensor nodes and the gateway. Dai et al. [16] managed to reduce latency at the edge nodes by improving a low latency object detection algorithm. In the recent years, the techniques and technologies based on Artificial Intelligence (AI) are being imposed in the IoT field. Mohammadi et al. explained in their survey [17] how Deep Learning (DL) is already being applied in each layer of the IoT systems (Device, Edge/Fog and Cloud). In this survey, the authors highlighted four methods and technologies to incorporate DL on IoT devices. From the point of view of computational efficiency, the Network Compression and Accelerators stand out. Regarding the energy perspective, the Approximate Computing and Tinymote with DL are the most efficient. These works propose solutions for the efficiency of the current node, but we want to address the problem at the system design level, where we would take into account the amount of data processed and communicated to achieve an advantage in terms of latency and energy.

During the past decade, other works proposed node offloading by distributing the processing tasks between the node and a server. This approach opens up a new design exploration dimension where image processing may be local, partially local or totally remote. In their work, Pinto et al. [18] analyzed three scenarios to find out when it is best to send data and when to process them locally. Khursheed et al. [19] investigated the partition between hardware (Field-Programmable Gate Array or FPGA) and software (micro-controller) but also between local and central processing in order to achieve an optimal partition point that guarantees a minimum energy consumption in the camera node. Imran et al. [20] descreased the execution time by parallelizing tasks in a FPGA. In their work, they also considered the partition of the tasks in order to distribute the processing load between the node and the server. In [21], Motlagh et al. proposed an Unmanned Aerial Vehicle (UAV) as an IoT node where the energy savings are achieved by moving local processing (on-board processing) to a Multi-Access Edge Computing (MEC) node. Zhao et al. [22] presented a system partition approach based on Convolutional Neural Networks (CNNs) by distributing the convolutional layers that contribute greatly to inference latency. Although these works take into account processing scenarios, they do not consider the effects on latency and energy due to changes in the system input.

The aforementioned works offer solutions that undoubtedly contribute to the improvement and optimization of the camera node. However, these works do not consider changes in the image in their analysis. From a data point of view, an image can vary with respect to two dimensions: size and content. The size refers to the resolution of the image, which directly influences the number of pixels to be processed. Content refers to features in the

image that are relevant to the application. For example, in an object identification task, the number of objects in an image might affect the processing and communication load, subsequently affecting the efficiency of the optimization methods proposed.

Many authors have studied the effect that changes in image size and content have on their systems. In [23], Wang et al. studied an image content weighting methodology in order to improve image quality assessment (IQA) algorithms. Fookes et al. [24] analyzed the impact of image resolution on facial recognition performance showing that the methodology proposed for super-resolution improves the recognition of low-resolution and noisy facial images. In [25], Yan et al. developed a model to map pedestrians at different resolutions. This way, the authors managed to reduce the average rate of false positives in the context of detection in traffic scenes. In the work presented by Alhilal et al. [26], a low complexity vision system was designed for the identification of objects in WVSN. The proposed architecture is characterized by low power processing, thus achieving the energy efficiency of the node. In [27], Gu et al. investigated the influence of viewing distance and image resolution on IQA performance. Ur Rehman et al. focused on efficient image delivery based on object detection [28]. Their work proposes a new object detection model to reduce false transmission of images and transmitting only image segments instead of complete images. The results showed considerable energy savings in the camera node compared to more current techniques. Romic et al. [29] evaluated the performance of a stairs detection system based on cameras in function of distinct image resolutions. The results showed that the selection of an optimal resolution is essential to achieve the trade-off between precision and processing speed. Yazidi et al. faced the challenge of data growth from the IoT and the generation of Big Data by different platforms [30]. The authors conducted a data size latency sensitivity study in order to measure the performance of the Apache Spark framework (Spark is a fast parallel processing system in the Big Data environment) and evaluate the computational complexity. Despite the existence of studies related to system input changes, none of them addresses the problem of the impact of size and content of the input images on both the latency and energy at the node. However, the fact that there is a great research effort to cover the effect of input image changes on the implementations highlights its importance.

Summarizing, none of the related works studied the effect that changes in the input image could have on design decisions related to system deployment. Specifically, design decisions related to system deployment would be affected by how changes in the image are transferred through the system to affect energy and latency at different partitions. In this work, we attack the problem of camera node optimization through intelligence partitioning [7]. Intelligence partitioning targets finding a suitable cut-off point that optimizes either latency or energy or both considering processing and communication in the sensor node (Figure 1).

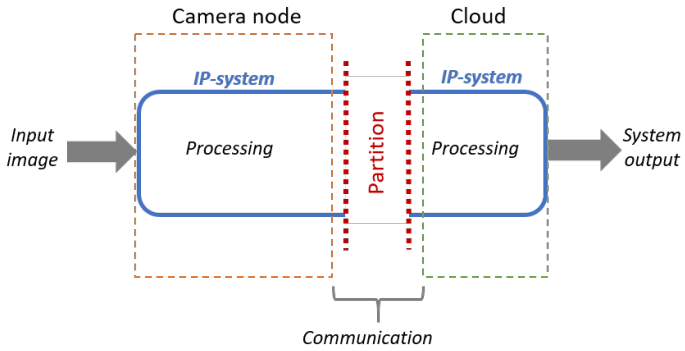


Figure 1. Intelligence partitioning.

In this article, we rely on the previously introduced intelligence partitioning method, but we provide a more elaborate view of the problem, including variations in the input image size and content. This is because image size and content could have an impact on the inter-task data amount during the processing chain, affecting the workload of the

processing tasks and the node efficiency. Therefore, we explore the use of intelligence partitioning as a methodology for the camera node optimization, this way contributing to the problem of implementing data-intensive, low-latency/energy camera nodes.

3. Theory

3.1. Intelligence Partitioning/Node Offloading

Intelligence partitioning is a method developed by Shallari et al. to improve the energy efficiency of smart sensor nodes by analyzing the trade-off between processing and communication [7]. It focuses on the prospective energy consumption variation due to partitioning the processing trail at any given point, and allocating the processing tasks between the sensor node and a remote processing unit. By taking into consideration a variety of partition configurations and wireless communication technologies, it provides insight into the inter-effects of processing and communication in the overall energy consumption of the sensor node. However, the current approach only provides a partial view of the problem, because in the analysis of the optimal partitioning point, the size of the input data and subsequently the inter-task data amount between the processing tasks are considered fixed.

The system inter-task data amount during processing could vary for two reasons. The first concerns the content of the image. There are processing tasks that could be sensitive to the content of the image. This is because normally the objective of an image processing system is to detect objects in the image for subsequent counting, identification, classification, etc. Therefore, there should be variations in the data amounts during processing that result from the outputs of the system tasks that are dependent on the image content. The second concerns the initial stages of design, where the image size must be selected. This decision directly affects the data amount that the system must process and therefore should affect the inter-task data amount during processing too.

Because of the close relationship between data, time and energy, we think that data are the common factor in node constraints. For this reason, we believe that the choice of system input size could have an effect on subsequent partitioning decisions.

3.2. Communication Model

Intelligence partitioning assumes that several tasks can be executed in different locations. As a result, data have to be exchanged between these locations depending on the partitioning point. However, the energy and delay resulting from this transfer depend on the inter-task data amount to exchange at each task as well as the chosen communication technology. Several technologies have been discussed in the context of IoT or smart cameras.

In order to evaluate multiple technologies and their impact on the partitioning depending on the data change, we employ a framework that models the data transfer of various IoT commuication technologies. Krug and O'Nils [31] introduced a modular framework that allows us to evaluate and select the most suitable communication technologies for our system. The models are implemented in Matlab and calculate the latency and energy per data transfer for several communication technologies. The framework covers the functional level of sensing which, in our work, corresponds to the camera node and is thus viable for this task.

To calculate the energy and delay, Krug and O'Nils considered the communication technology, the resulting protocol-specific timing based on the data amount to transfer and corresponding real hardware transceivers. The amount of data to transfer is used to determine the number of packets to be sent by the transceiver and thus determines its activity. The energy consumption then depends on the resulting duration of each activity as well as the corresponding power consumption of the selected hardware. As a result, the models are able to provide the communication cost for an arbitrary data amount.

In order to observe the impact of the communication component at the partion points, in this study, we chose a subset of models. We analyze an image processing system with

a relatively high data amount to transfer compared to traditional IoT use cases. Due to this, we selected the following communication technologies that are able to handle this data amount: Bluetooth 5.0, 802.11n (Wi-Fi), LTE Cat.4, and LTE Cat.1. All technologies are suitable for higher data rates and are used for smart camera applications, where LTE Cat.4 corresponds to traditional smartphone type communication. Other popular low power communication technologies such as LoRa were not considered in this study as they are not able to handle large data amounts required to send images or intermediate data. For these technologies, the partitioning results in complete in-node processing always.

4. Methodology

We focus on sequential processing systems because the partitioning cut between the sensor node and the cloud server has to be a single, directed vertex. More general architectures can be transformed into an acyclic, sequential system by collapsing cycles and fork-join structures into single processing nodes. Thus, our work is also applicable to general processing architectures that can be transformed in this way into an sequential architecture, which can be done if the overall processing algorithm has a single input and a single output. Thus, our method is general but limited by the fact that a partitioning cut can only be applied to directed vertices that separate the architecture graph into two otherwise disconnected sub-graphs.

4.1. Optimization Problem

In an IoT system, the data are captured by the devices and sent to the cloud for analysis, processing or both. Once the cloud computes a solution, the data could be sent back to the same node or an actuator in order to achieve the system objective. In this article, we limit the scope of the optimization problem to the sensor node, assuming it does not expect data back. Therefore, we analyze both the time and the energy in order to optimize and offload the node.

As mentioned before, image processing systems usually involve a set of smaller tasks t_i that form a complex processing function F:

$$F = \{t_1, t_2, \cdots, t_N\}. \tag{1}$$

In a distributed system, a specific task t_i of the function F is not bound to a specific geographical location. Therefore, the actual execution of tasks is location independent [7]. Each task can be mapped to any node in the system: the camera node, a cloud server, or an inter-task fog computational resource. Formally, the distributed function F has its functionality distributed between node and cloud so that,

$$\begin{aligned} F &= f_{Node} \cup f_{Cloud} \text{ and} \\ \emptyset &= f_{Node} \cap f_{Cloud} \end{aligned} \tag{2}$$

where the subsets f_{Node} and f_{Cloud} are the different clusters of tasks composing the function F that are executed at the receptive location. Because of this, both the latency and the energy of the tasks are also given in different entities. The mapping of the computational and communication load of a node between the different computational resources (Figure 2) is defined as intelligence partitioning, $\Im$:

$$\Im(F) = \begin{cases} \{f_{Node}, f_{Cloud}\} \\ \{L_{P_{Node}}, L_{C_{Node \to Cloud}}\} \\ \{E_{P_{Node}}, E_{C_{Node \to Cloud}}\} \end{cases} \tag{3}$$

where $L_{P_{Node}}$ and $E_{P_{Node}}$ are the node latency and energy due to processing and $L_{C_{Node \to Cloud}}$ and $E_{C_{Node \to Cloud}}$ are the latency and energy due to communication from the node to the cloud.

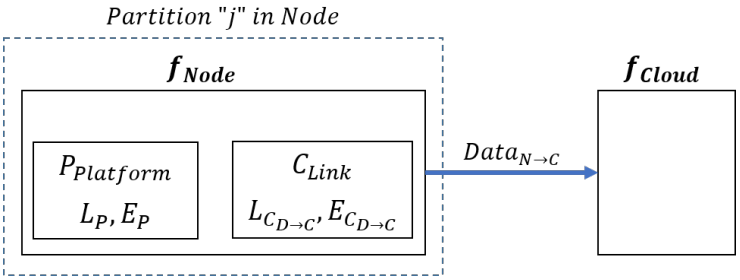


Figure 2. Computational and communication load of a node.

Both latency and energy have a direct relationship with the data amount. The data transfer (D_i) between the processing tasks in the node make up the D function,

$$D = \{D_0, D_1, \cdots, D_N\} \tag{4}$$

where D_0 is the input image, D_1,D_2, ... are the data between the processing tasks, and D_N is the system output data. The data amount transferred between two computational layers can be expressed as,

$$D_{Node\to Cloud} = D_j \tag{5}$$

where j is the position of the system partition cut.

Our optimization problem is focused on the latency and energy; therefore, system partitioning must allow a reduction of one or both of these. The latency in the node (L_{Node}), in a specific partition cut, depends on two components, the computation and communication latencies. The computation latency (L_P) can be defined as the accumulated processing time of the tasks (t) that make up the system from its start to the partition cut. The computational latency depends on the processing platform (P_P) but also on the data (D_i) generated by the processing algorithms. The communication latency (L_C) refers to the time to transfer data from the partition point (D_j) and it depends on the intrinsic characteristics of communication technology (C_C). The node latency (L_{Node}) is derived as,

$$L_{Node} = \sum_{i=1}^{j}(L_P(D_{i-1}, t_i, P_P)) + L_C(D_j, C_C) \tag{6}$$

where D_{i-1} are input to t_i and D_i is the output. L_P and L_C are the measurement or estimation function for the processing and communication latency, respectively. Like the latency, the energy consumption in the node until the partition cut has two components, the computation and communication energies. Therefore, the node energy (E_{Node}) is derived as

$$E_{Node} = \sum_{i=1}^{j}(E_P(D_{i-1}, t_i, P_P)) + E_C(D_j, C_C). \tag{7}$$

In Section 5.3.3, we will consider three specific objective functions for our study case in order to minimize once latency (L_{Node}), once energy (E_{Node}) only, and once energy (E_{Node}) under delay constraints ($MaxDelayConstrain$).

4.2. Method for Analyzing Algorithm Image Sensitivity

According to Equations (6) and (7), the optimal partition points are data-dependent. This means that both the latency and the energy of the partitions will vary based on the inter-task data amount processed before the partition cut and the data amount transmitted over the link in the partition cut.

The inter-task data amount through the system varies for two reasons that are closely related. The first is due to the dependence of the data amount between processing tasks. Since being the image introduced into the system, it undergoes a series of changes that lead to a reduction of data. This reduction is due to the fact that each task exerts a reducing action

that will affect the inter-task data amount to a lesser or greater extent. The second is due to changes in the image. Normally, during the execution of the application, the image changes in content, which refers to the number of objects. The objective of a processing system is usually the recognition of objects for their subsequent labeling, counting, classification, etc. So, there are processing tasks that would be sensitive to these changes in the image. Because of this, a change in the number of objects could affect the data amounts in the outputs of these tasks, producing variations in the inter-task data amount during the processing chain. Another change of the image refers to its size. This type of change would also have an effect on the data amount throughout the system. This is because, at the beginning of the system, the tasks are in charge of processing pixels in order to isolate relevant information for the application. Because of this, the data amount in the output will change relative to the size of the image. Although normally the size of the input image does not vary during the execution of the application, it would play a fundamental role in the design of the system. This is because the number of objects is related to the resolution of the image, so a higher resolution (larger image size) would allow increasing the number of objects per image. Conversely, a large image size could affect communication and processing times, thereby affecting the partition latency and energy. Therefore, we focus on two aspects of the image: its size and its content in terms of the number of objects visible in the image.

We have analyzed the image sensitivity of the tasks by observing their outputs while introducing multiple images into the system, with various sizes and number of objects. We expect to see that the output of an image-sensitive processing task could vary based on the input image size, its content, or both. However there could also be non-image sensitive tasks in image processing systems. Accordingly, each algorithm has a characteristic image sensitivity and that causes variations in the inter-task data amount, which in turn could play a role in partitioning decisions.

4.3. Processing Time Behavior Analysis

The variation of inter-task data amount during the processing chain means that the data in the inputs of the algorithms (D_{i-1}) varies due to the dependency between the tasks. These changes should affect the time it takes to execute each of the system tasks. The processing time is an important aspect at a partition point because of its direct impact on overall latency and energy (Equations (6) and (7)). For this reason, the processing time behavior of the tasks with respect to the data amount could be another factor that influences the best partitioning solutions. We have analyzed the processing time behavior by measuring the time it takes to process the input data in each of the system tasks, which varies depending on the input image and the data amount dependency between the previous tasks. We are interested in observing how the processing task times behave to analyze how they are influenced by the changes in the input image (Table 1). The variation of the processing times will contribute in the partitions by increasing or decreasing their delay based on the data amounts.

4.4. Partitioning Depending on System Architecture

In this work, we analyze the latency and energy of the partitions at different points in the processing chain. To do this, we first analyze the tasks of the system separately, extracting the amount of data at each tasks input and output as well as the related processing time and energy. The partition points are dependent on the type of system architecture which can be sequential or parallel. In a purely sequential system, the data amounts, latencies, and energies of the partition points correspond to the inputs and outputs of the processing tasks. This would greatly simplify the analysis of partitions. However, in a parallel architecture, neither the data amount, nor the latencies, nor the energies of the partition points have to correspond to the inputs or outputs of the processing tasks. Then, the analysis of the partitions in a parallel architecture would require a previous serialization process.

Table 1. Processing tasks, data out and processing time behaviors due to changes in image.

#	Output Type	Changes in Data Out Due to:		Processing Time
		Δ Img. Size	Δ Img. Objects	Behavior
t1	Color space YCbCr	ΔLinear	Constant	$\Delta Linear_{Size}$
t2	Y channel	ΔLinear	Constant	$\Delta Linear_{Size}$
t3	Array with 256 elements	Constant	Constant	$\Delta Linear_{Size}$
t4	Binary image without background	ΔLinear	Constant	$\Delta Linear_{Size}$
t5	Binary image with detected regions	ΔLinear	Constant	$\Delta Linear_{Size}$
t6	Binary image with filled regions	ΔLinear	Constant	$\Delta Linear_{Size}$
t7	Binary image without particles	ΔLinear	Constant	$\Delta Linear_{Size}$
t8	Non-binary labels image	ΔLinear	Constant	$\Delta Linear_{Size}$
t9	2D features matrix	Constant	ΔLinear	$\Delta Linear_{Size,Objects}$
t10	2D coordinates matrix	Constant	ΔLinear	$\Delta Linear_{Objects}$

The serialization process consists of grouping the tasks in parallel forming compound tasks. The time of the compound task is the maximum time between the tasks that compose it and the energy is the sum of the energies of all the tasks compounding it. The data amount in the output of the compound task is the result of summing the data in the outputs of the tasks that compose it. Once the parallel architecture is serialized, the data amounts, latencies, and energies between the compound tasks will correspond to specific points in the processing chain.

Our application case is a serial architecture from a time and energy point of view, but there is a parallel data stream where data are transferred between non-consecutive tasks (Channel Separation and Image Segmentation). For this reason, we only apply serialization from a data perspective. Figure 3 shows the system flow diagram where we have marked the partition points from 1 to 10. Partition 1 ($p1$) corresponds to the system input data, and number 10 ($p10$) to the output. We have observed that the data amount, time and energy of the Image Histogram is too small when compared to the previous task (Channel Separation) and the latter one (Image Segmentation). Due to this, the partition point after Image Histogram will be practically the same (almost same latency and energy) as the partition point after Channel Separation. Then, the time and energy in Image Histogram will be accumulated in the next partition, that is in the output of Image Segmentation.

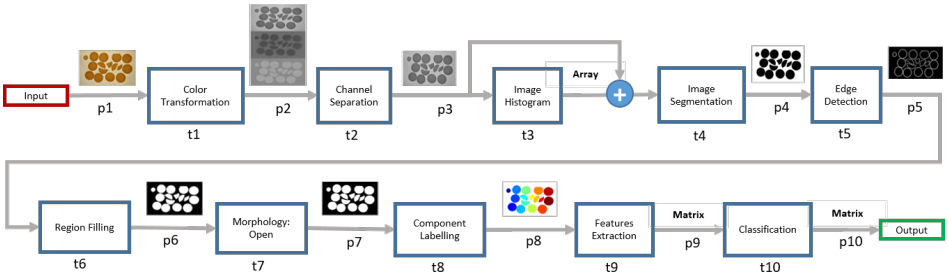


Figure 3. System flow diagram with numbers indicating the partition points to analyze.

5. Results

We have carried out an in-depth analysis of the latency and energy of different partitions. We have first designed the system and a dataset for the purpose of the study. We then ran the system for each of the dataset images and extracted data amounts and execution times for each of the system tasks during processing. Finally, we set the optimization objectives and analyzed the impact of varying the input image at various partition points in the system.

5.1. Application Case

Our case study is a biscuit inspection system. This system is responsible for analyzing images and identifying whether biscuits are acceptable or not, depending on the visual biscuit characteristics. The system is made up of ten processing tasks: Color Transformation, Channel Separation, Image Histogram, Image Segmentation, Edge Detection, Image Region Filling, Morphology, Component Labeling, Features Extraction and Classification. Figure 3 shows the system flow diagram, and the data types of the output from each processing task are found in Table 1.

The primary input is an RGB image, which has a certain size and number of objects. Then, the image is transformed from RGB to YCbCr color space. The Channel Separation task receives the YCbCr image and separates the "Y" channel. The Image Histogram task then uses this channel to compute a histogram, which is used for the next task. Image Segmentation has two inputs, a 1D array coming from Image Histogram and the image from Channel Separation. Initially, this task uses the 1D array to define the intensity level that separates the background from the objects of interest; it computes a global threshold. Sequentially, the threshold is used then to convert the input image into a binary image without background (2D binary matrix). Next, we introduce the binary image into the Edge Detection task, which applies a Sobel operator [32] to detect the edges of the objects. The output of this task is a binary image showing the edges of the objects (2D binary matrix). Next, we fill the objects regions by applying a Region Filling task. The output is a binary image (2D binary matrix) with the regions defined in white. The Morphology task captures the binary image with filled regions and applies the Open Operator [33] to remove the image artifacts (biscuits crumbs). The Morphology results in an image whose objects are only biscuits (2D binary matrix). The Component Labeling task is responsible for differentiating the regions of the image by assigning distinct numerical labels to each of the regions. Therefore, the output is a 2D matrix of non-binary labels. Next, the Feature Extraction task analyzes the label matrix to extract the characteristics of the objects, as their centroid, perimeter and eccentricity. Its output is a matrix where each row corresponds to each object and the columns indicate the features. Finally, the Classification task analyzes the features matrix in order to separate the good cookies from the bad ones. The system output is a 2D matrix with the coordinates of the cookies and an additional bit indicating their quality.

5.2. Experiment Description

The scope is to analyze the impact that variations in the input image could cause to the optimal partition point, shifting it as a result of image size and content variations. Due to the configuration of the camera layout for frame capture, the maximum number of objects is limited to 18. Thus, we consider 10 different images containing 0, 2, 4, 6, 8, 12, 14, 16 and 18 objects, respectively. Similarly, we consider ten different image sizes between 292×436 and 2848×4288 pixels. Hence, 100 different images are used in the experiments, as illustrated in Figure 4.

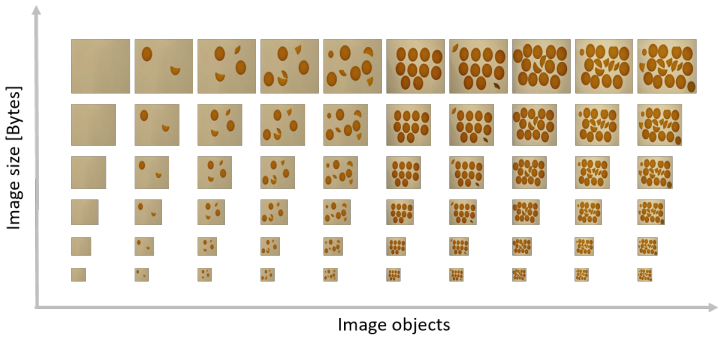


Figure 4. Sixty images of the 100-image dataset.

As shown in Equations (6) and (7), both latency and energy at a partition point have two components: processing and communication. The processing latency and energy at a partition point is the sum of all the latencies and energies of the processing tasks until the partition point; hence, all these tasks are implemented in the smart sensor node. Instead, the communication latency and energy are defined by the resulting data size at any given partition point. Therefore, the latency and energy at a partition point is the cumulative processing time plus the communication time of the serialized system.

The image processing tasks in Figure 3 were implemented on a Raspberry Pi 2 model B relying on the OpenCV libraries. We measured the power consumption for each image processing task with on the UNI-T tester [34] and the accuracy is in the range of 10 mA and 10 mV. Furthermore, we used the gettimeofday() function to record the execution time in the range of microseconds for each image processing task. Based on these measurements, we calculated the processing energy consumption for different partitioning configurations.The measured variation in computational latency was less than 5% and both energy consumption and latency measurements were obtained as an average of 10 measurements for each input image and processing configuration. In our analysis, we have also used four wireless communication technologies: BLE 5, 802.11n, LTE Cat.1 and LTE Cat. 4. For each data size resulting from the 10 input images and the inter-task data sizes, we have calculated the latency and energy consumption based on the communication model described in Section 3.2. First, we have extracted the data amounts based on the sizes of the images and their content (Section 5.3.1). Then, we have used the communication model described in Section 3.2 to generate both the time and the energy to communicate such data.

5.3. Experimental Results

This section shows the results obtained through the experiments where we analyze and characterize the processing and communication, pose the optimization problem and finally study how the best partitioning solutions are affected.

5.3.1. Analysis of Inter-Task Data Amount

Table 1 summarizes how the data amount and the processing time of different tasks depends on image size and number of objects.

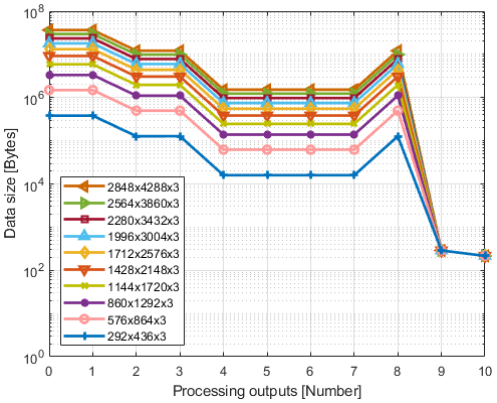
From the data amount perspective, we see that outputs of tasks 1, 2, 4, 5, 6, 7 and 8 are just sensitive to the change of the image size, while tasks 9 and 10 are sensitive to the number of objects. Task 3 is not sensitive to size or objects.

Figure 5 depicts the result of the inter-task data amount experiment and shows the impact of image size and numbers of objects on the data amounts inside the processing pipeline (Figure 5). We can see that the size of the primary input image has a much greater effect than the number of objects in the image. Only tasks 9 and 10 are sensitive to the number of objects.

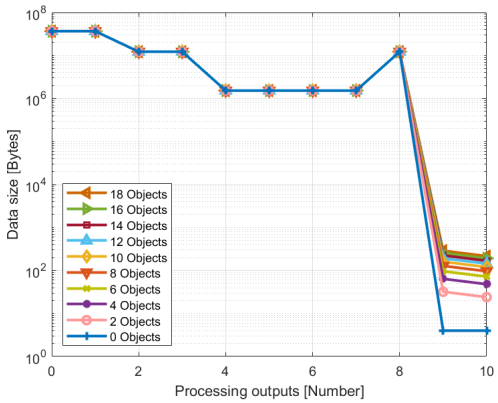
5.3.2. Characterization of Processing and Communication

Table 1 shows the execution time of the tasks is also affected by the size (from task 1 to 9) and the objects (tasks 9 and 10) of the input image. The change in the execution time of each task grows linearly with respect to its input. We see that the time in task 9 varies by both image size and content, this is because Features Extraction analyzes pixels to provide an output based on objects.

Figure 6 depicts the result of the Raspberry experiment and shows the impact of image size and numbers of objects on the processing time of individual processing tasks during the processing pipeline.

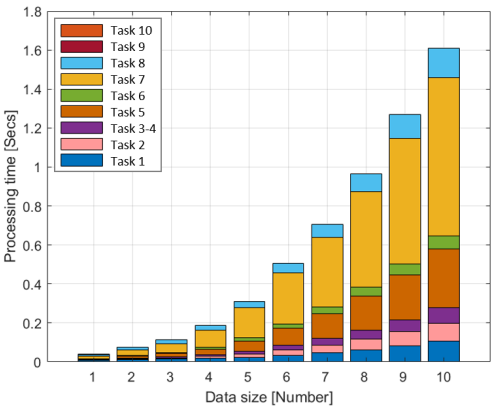


(**a**) Data amounts with respect to image size for an image with 18 objects.

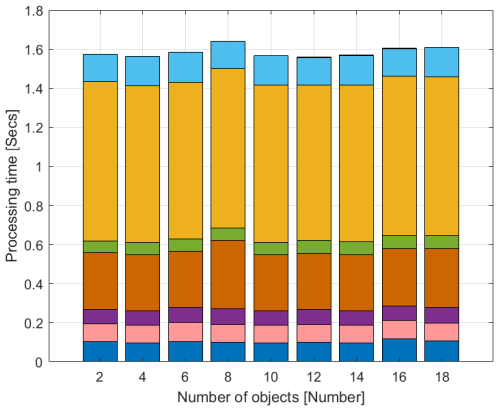


(**b**) Data amounts with respect to image objects for an image size of 2848 × 4288 × 3.

Figure 5. Data amounts along the system during the processing chain.



(**a**) Processing time with respect to image size for an image with 18 objects.



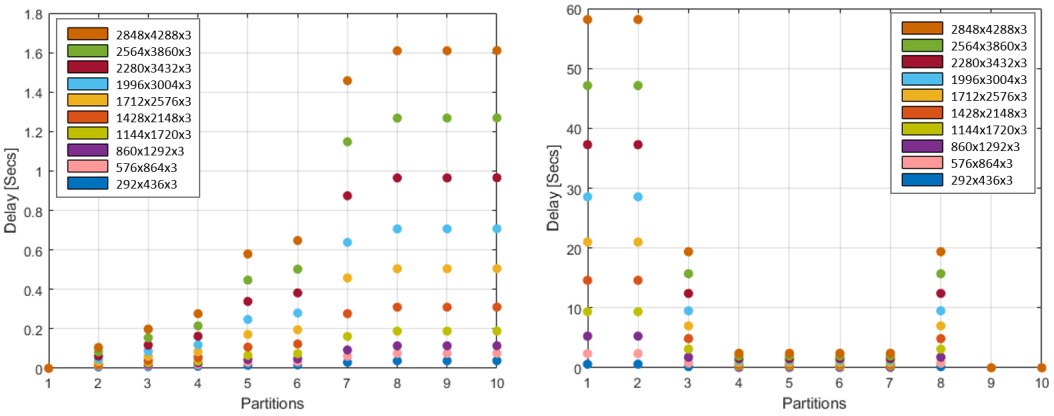
(**b**) Processing time with respect to image objects for an image size of 2848 × 4288 × 3.

Figure 6. Processing times of tasks from the Raspberry implementation.

Changes in the inter-task data amount and processing times during processing lead us to think that partition points might also be affected. Both latency and energy at a partition point depend on two components: processing and communication (Equations (6) and (7)), which vary differently depending on the size of the image and the number of objects. Figures 7–10 break the analysis into processing and communication, considering delay or energy.

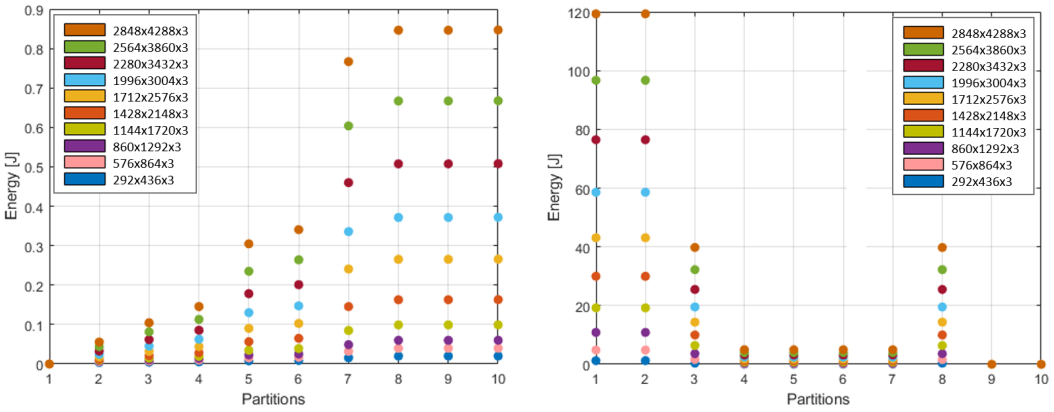
We see that the processing times and energy incurred by tasks 9 and 10 are so small when compared to the other tasks, that their effect is hardly noticeable in Figures 7–10.

Figure 9a shows the total latency of the partitions that results when summing the processing and communication components of Figure 7a,b. However, Figure 9b shows the total energy of the partitions that results from summing the processing and communication components of Figure 8a,b. Both processing and communication strongly depend on the image size, and communication dominates both latency and energy.



(**a**) Proccesing latency with Raspberry. (**b**) Communication latency with LTE Cat.1.

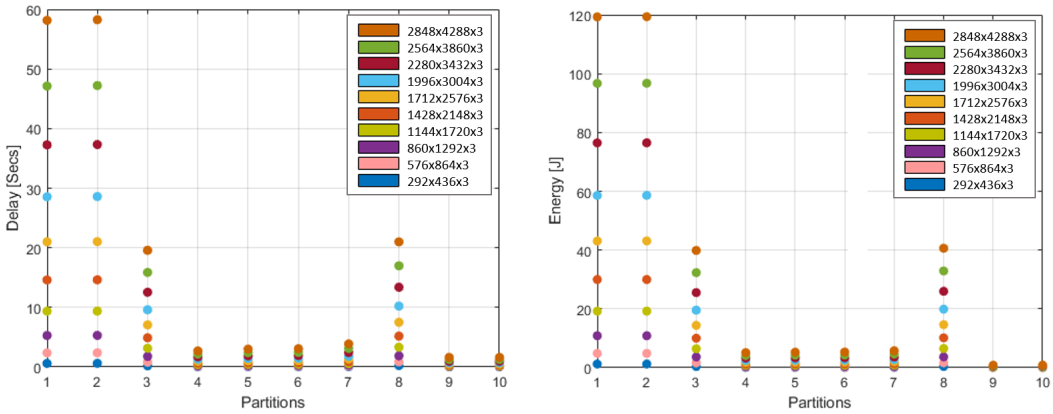
Figure 7. Latency per partition along the system as a function of the image size for an image with 18 objects. The processing latency of partition j is the sum of the processing latencies of all tasks 1 to j. The communication latency of partition j is the latency of communicating the output data of task j over the wireless communication link.



(**a**) Proccesing energy with Raspberry. (**b**) Communication energy with LTE Cat.1.

Figure 8. Energy per partition along the system as a function of the image size for an image with 18 objects. What was mentioned in the caption of Figure 7 also applies to energy.

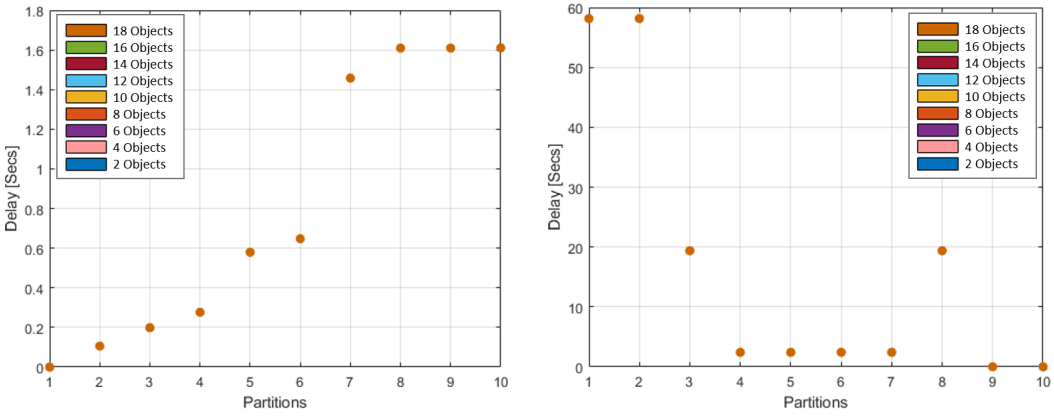
The number of objects affects the processing latency only for partitions 9 and 10 (Figure 10), where this effect is imperceptible in the plot as it is too small. Latency due to object number is independent of the image size; therefore, it becomes more important as we reduce the size. However, the effect of the number of objects on latency is very small, even for small images. Similarly, the effect of the number of objects on the processing energy is negligible. Therefore, we decided to ignore this effect in the remainder of our analysis.



(**a**) Latency of the partitions based on Equation (6).

(**b**) Energy of the partitions based on Equation (7).

Figure 9. Total latency and energy per partition along the system as a function of the image size for an image with 18 objects, using Raspberry for processing and LTE Cat.1 for communication.



(**a**) Proccessing latency with Raspberry.

(**b**) Communication latency with LTE Cat.1.

Figure 10. Latencies of the partitions along the system as a function of the number of objects for an image size of 2848 × 4288 × 3.

5.3.3. Optimization problem

As the partitioning decision involves a trade-off between energy and latency, we can formulate different optimization problems.

Minimizing latency:

$$L_{P_{Node}} + L_{C_{Node \rightarrow Cloud}} \rightarrow Min \tag{8}$$

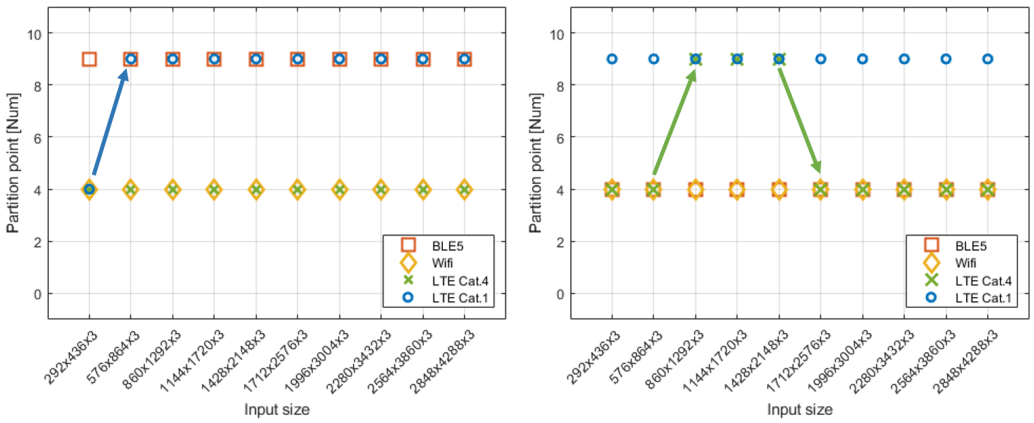
Minimizing energy:

$$E_{P_{Node}} + E_{C_{Node \rightarrow Cloud}} \rightarrow Min \tag{9}$$

Minimizing energy under a latency constraint:

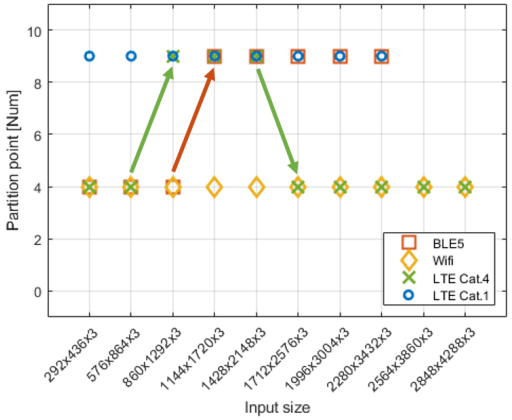
$$\begin{aligned} L_{P_{Node}} + L_{C_{Node \rightarrow Cloud}} &\leq MaxDelayConstraint \\ E_{P_{Node}} + E_{C_{Node \rightarrow Cloud}} &\rightarrow Min \end{aligned} \tag{10}$$

Figure 11 shows the best partitioning solutions for each optimization goal. Each of the figures present four different communication technologies (BLE5, Wifi, LTE Cat.4 and Cat.1) and a single processing platform (Raspberry). The graphs show the optimal partitions for the different image sizes, and we note that the optimal point is different for different image sizes, at least for some communication technologies.



(**a**) Optimal partitions for minimum latency.

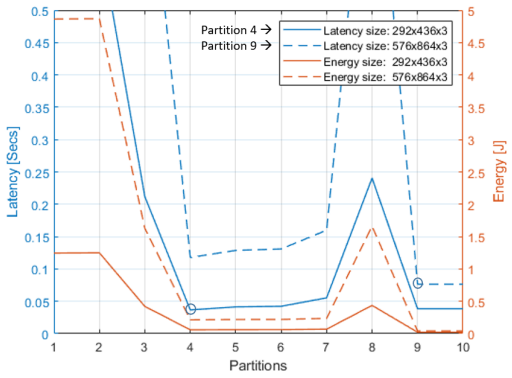
(**b**) Optimal partitions for minimum energy.



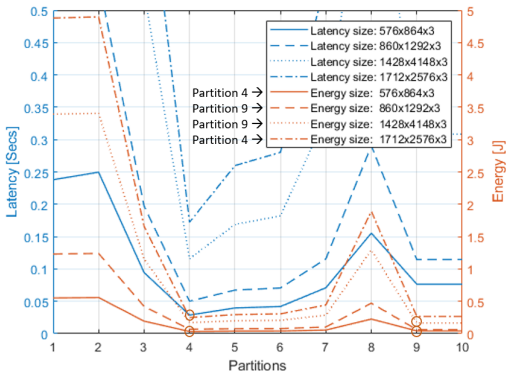
(**c**) Optimal partitions for minimum energy under latency constraint.

Figure 11. Partitioning solutions for three different objective functions, four different communication protocols and ten different image sizes.

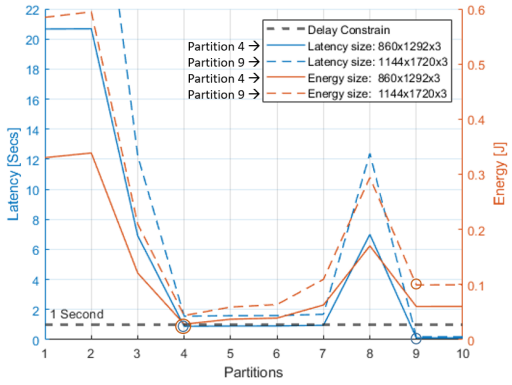
In Figure 12, we show the times and energies of the partitions as a function of the image sizes in order to observe how the change in the partitioning solutions occurs. Then, the cases in Figure 12a–c correspond to the cases seen in Figure 11. However, as the case of LTE Cat.4 is repeated in Figure 11b,c, Figure 12c only shows the case for BLE5 communication. It is by applying the optimization functions to each of these graphs when we observe the changes in the best partitioning solutions. We have marked the partitioning solutions with circles that meet the optimization objective.



(**a**) Latency minimization for LTE Cat.1



(**b**) Energy minimization for LTE Cat.4



(**c**) Energy minimization under a delay constraint for BLE5

Figure 12. Latency and energy for three communication technologies and all possible partitions. The optimal partition points are marked with circles.

6. Discussion

6.1. Impact on System-Wide Energy and Latency

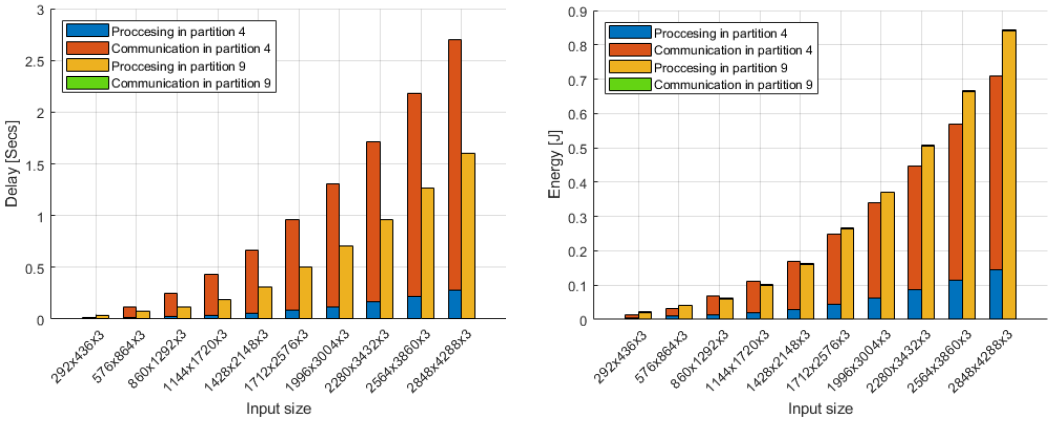
The intelligence partitioning aims at finding the optimal system partition point with respect to system latency and node energy. For that reason, we analyze the partition points from the time and energy perspectives, according to the objective functions in Equations (8)–(10).

We have observed that changing the size of the input image leads to a chain of effects that ultimately alter the latencies and energies of the system. First, changes in image size affect the inter-task data amount along the system during the processing chain (Figure 5). This is due to some processing tasks are sensitive to image size, its content, or both (Table 1). The data amount at the partitioning point directly affects communication latency and energy, and consequently, they are data-dependent (Figures 7b and 8b). In turn, the latency is also affected due to its dependency on the input data of the processing tasks (Figure 7a). Finally, due to the relationship between time and energy in the processing platform, the processing energy varies with processing time (Figure 8a).

6.2. Impact on Optimal Partition Point Selection

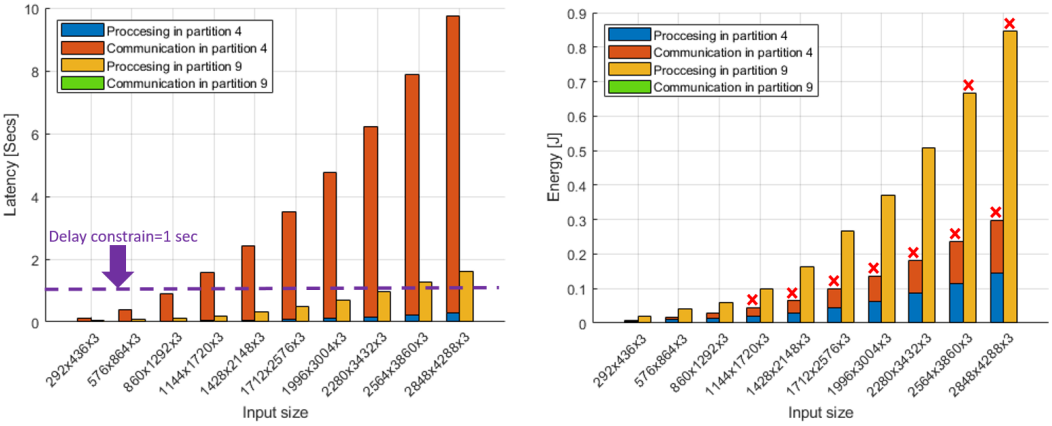
Figure 11 presents three cases where the partitioning solutions change due to changes in the image size. The partitioning solution change could be related to how the contribution of the processing and communication components influences partition time and energy. In Figures 7 and 8, we see that the processing and communication behaviors at the partition points are completely different. For this reason, in early partitions, where the data amount in partition points is greater (Figure 5), there would be a small accumulated processing

time, but a lot of communication. In addition, near the end of the system, there are small data amounts to be sent, but a lot of accumulated processing time. Due to this difference in data dependency, the best partitioning solution in latency, energy or both may vary with respect to the size of the image. To demonstrate this, Figures 13 and 14 show the contribution of the processing and communication of partitions 4 and 9 depending on the latency and energy in function of the image size.



(**a**) With communication technology LTE Cat.1 (**b**) With communication technology LTE Cat.4

Figure 13. Contribution of the processing and communication components to the time and energy of partitions 4 and 9 depending on the image size.



(**a**) Latency constraint with communication technology BLE5 (**b**) Minimum energy under the latency constraint with communication technology BLE5

Figure 14. Contribution of the processing and communication components to the time and energy of partitions 4 and 9 depending on the image size.

Figures 13a,b and 14 correspond to the cases presented in Figure 12a–c. The time and energy of communication of partition 9 is not visible because the data size in partition 9 is very small, only reaching a maximum of 288 Bytes in comparison to 1.53 MBytes in partition 4. So, the time used to send data in partition 9 is around microseconds, which is much lower than partition 4, which is in seconds. The same observation applies to communication energy.

From Figure 13a,b, we see that partition 4 is dominated by communication, while partition 9 is dominated by processing, because, in partition 4, there is a large data amount to be sent, requiring more time and energy compared to the processing part up to that point. However, the processing dominance of partition 9 is due to the small data amount to be sent, but in the final stage of the processing, a lot of processing time and energy has been spent.

Figure 14a shows the latency perspective, with the latency constraint marked with a purple line (1 s). We note that due to communication dominance, partition 4 only meets the time constraint from size 292 × 436 × 3 to 1144 × 1720 × 3. However, partition 9 meets the time constraint of sizes 292 × 436 × 3 to 2280 × 3432 × 3.

Figure 14b shows the energy perspective. This time we have marked the partition points that do not meet the delay constraint with an "x". This way, we see that although a lot of processing energy is expended in partition 9, due to the delay constraint, this is the best partitioning solution for size 1144 × 1720 × 3 to size 2280 × 3432 × 3.

Both the processing and communication time and energy are dependent on the data, so the best partitioning solutions are closely related to the image size. Hence, the best partitioning solution is different for different image sizes. In addition, the type of optimization problem also influences the optimal solution.

6.3. How to Make Partition Decisions Based on Image Size?

In this work, we have analyzed the impact of the size and content of the image on the deployment of system tasks. We have shown that the size of the image influences partitioning decisions and studied why, which is our main contribution. We have created this section so that designers can take advantage of this knowledge and thus apply it, to achieve a more effective node offload.

Our study shows that the best implementation depends on the details of the application requirements (such as minimum image resolution, power and energy budgets), and the communication technology and protocol used. Because the optimal partitioning choice is sensitive to all these parameters, we advocate a systematic and quantitative exploration of the design space (Figure 15).

Partitioning analysis:

1. Select or create a typical image of the application in question. Subsequently, scale the image in order to have a dataset with different resolutions.
2. Run the system for each of the images in the dataset and:
   - Measure the amount of data at the outputs of each of the tasks in the system;
   - Measure the execution time and processing energy at the outputs of each of the tasks in the system;
   - Measure the communication time and energy at the outputs of each of the tasks in the system employing the communication framework [31].
3. If the system is not purely sequential, serialize and assign the times, energies and amounts of data to the composite tasks.
4. Apply Equations (6) and (7) to calculate the time and energy of the partitions.
5. Apply the optimization objective in order to find the image size that allows the optimal partitioning point.

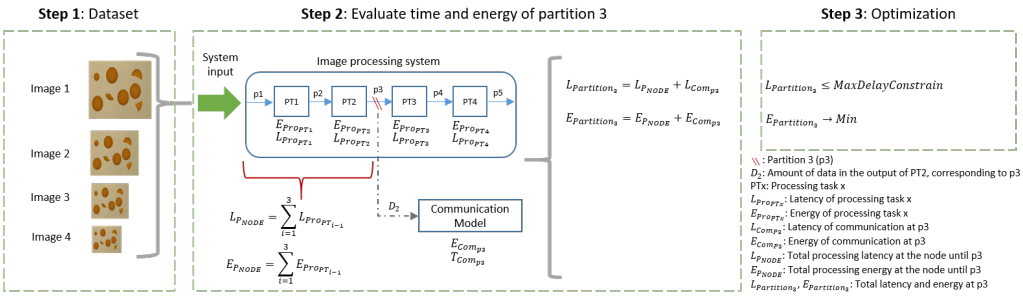


Figure 15. Example of partitioning analysis. In this example, we evaluate partition 3 with respect to a set of four images which vary in size. The partition analysis must be done on all possible partition points of the system.

To summarize, the latency and energy consumption for both processing and communication are dependent on the image size and the inter-task data amount, which define where would be the most optimal partitioning point for a given scenario. These results prove that the inter-task data amount has a major importance in the design space exploration of data intensive IoT nodes, as the one introduced in our recent work [35].

7. Conclusions

We have applied intelligence partitioning with the purpose of offloading an image processing-based IoT node with respect to node energy and to reduce latency. This work shows that the optimal partitioning point depends heavily on the selection of the input image size, but is hardly affected by the number of objects in the image. In addition, the optimal partitioning point also depends on the specific objective function.

The reason why size affects partitioning points is due to two aspects. The first is that, during processing, there are more tasks processing pixels than processing data referring to objects. Because of this, the cumulative effect of pixel processing is much larger than processing dedicated to objects making size more relevant. The second aspect concerns the communication of data at the point of partitioning. We have found that data communication dominates over processing in both energy and time resource consumption at the node. In this way, the relevance of image size is enhanced because the largest amounts of data to be sent are found in the partitions made at the beginning of the processing stage. In addition, due to the reduction effect of the algorithms, the amount of data to be processed in the object processing stage is much lower than in the pixel stage, so sending data is less costly, making objects less important.

Author Contributions: Conceptualization, I.S.L., S.K. and M.O.; methodology, I.S.L.; software, I.S.L., S.K. and I.S.; validation, I.S.L., M.O., S.K., A.J. and I.S.; formal analysis, I.S.L., M.O., A.J. and S.K.; investigation, I.S.L.; resources, M.O.; data curation, I.S.L. and I.S.; writing—original draft preparation, I.S.L., M.O., S.K., A.J. and I.S.; writing—review and editing, I.S.L., M.O., S.K., A.J. and I.S.; visualization, I.S.L.; supervision, M.O., S.K. and A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Swedish Knowledge Foundation grant Research profile NIIT.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
WVSN	Wireless Vision Sensor Network
AI	Artificial Intelligence
DL	Deep Learning
FPGA	Field-Programmable Gate Array
MEC	Multi-Access Edge Computing
CNNs	Convolutional Neural Networks
IQA	Image Quality Assessment
LTE	Long Term Evolution
BLE5	Bluetooth Low Energy version 5
WiFi	IEEE 802.11 wireless LAN

References

1. Nguyen Gia, T.; Peña Queralta, J.; Westerlund, T. 16—Exploiting LoRa, edge, and fog computing for traffic monitoring in smart cities. In *LPWAN Technologies for IoT and M2M Applications*; Academic Press: Cambridge, MA, USA, 2020; pp. 347–371. [CrossRef]
2. Grazia Gnoni, M.; Angelo Bragatto, P.; Francesca Milazzo, M.; Setola, R. Integrating IoT technologies for an "intelligent" safety management in the process industry. *Procedia Manuf.* **2020**, *42*, 511–515. [CrossRef]

3. Židek, K.; Piteľ, J.; Adámek, M.; Lazorík, P.; Hošovský, A. Digital Twin of Experimental Smart Manufacturing Assembly System for Industry 4.0 Concept. *Sustainability* **2020**, *12*, 3658. [CrossRef]
4. Muthu, B.; Sivaparthipan, C.B.; Manogaran, G.; Sundarasekar, R.; Kadry, S.; Shanthini, A.; Dasel, A. IOT based wearable sensor for diseases prediction and symptom analysis in healthcare sector. *Peer-to-Peer Netw. Appl.* **2020**, *42*, 511–515. [CrossRef]
5. Maaz, M.; Mohammed, S. IoT programming to Develop Self Driving Robotics Car using OpenCV and Other Emerging Technologies. *TechRxiv* **2020**. [CrossRef]
6. Blaauw, D.; Sylvester, D.; Dutta, P.; Lee, Y.; Lee, I.; Bang, S.; Kim, Y.; Kim, G.; Pannuto, P.; Kuo, Y.-S.; et al. IoT design space challenges: Circuits and systems. In Proceedings of the Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers, Honolulu, HI, USA, 9–12 June 2014; pp. 1–2. [CrossRef]
7. Shallari, I.; Krug, S.; O'Nils, M. Communication and computation inter effects in people counting using intelligence partitioning. *J. Real-Time Image Process.* **2019**, *17*, 1869–1882. [CrossRef]
8. Afzal, B.; Alvi, S.A.; Shah, G.A.; Mahmood, W. Energy efficient context aware traffic scheduling for IoT applications. *Ad Hoc Netw.* **2017**, *62*, 101–115. [CrossRef]
9. Veeramanikandan, M.; Sankaranarayanan, S. Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction. *J. Parallel Distrib Comput.* **2019**, *127*, 18–27.
10. Nsiah, K.A.; Amjad, Z.; Sikora, A.; Hilt, B.; Lauffenburger, J. Latency Reduction Techniques for NB-IoT Networks. In Proceedings of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; Volume 1, pp. 478–482. [CrossRef]
11. Duy La, Q.; Ngo, M.V.; Dinh, T.Q.; Quek, T.Q.; Shin, H. Enabling intelligence in fog computing to achieve energy and latency reduction. *Digit. Commun. Netw.* **2019**, *5*, 3–9.
12. Zeadally, S.; Karim Shaikh, F.; Talpur, A.; Sheng, Q.Z. Design architectures for energy harvesting in the Internet of Things. *Renew. Sustain. Energy Rev.* **2020**, *128*, 109901. [CrossRef]
13. Abas, K.; Obracz, K.; Miller, L. Solar-powered, wireless smart camera network: An IoT solution for outdoor video monitoring. *Comput. Commun.* **2018**, *118*, 217–233. [CrossRef]
14. Anagnostou, P.; Gomez, A.; Hager, P.; Fatemi, H.; de Gyvez, J.P.; Thiele, L.; Benini, L. Torpor: A Power-Aware HW Scheduler for Energy Harvesting IoT SoCs. In Proceedings of the 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Platja d'Aro, Spain, 2–4 July 2018; pp. 54–61. [CrossRef]
15. Al-Qurabat, A.K.M.; Abou Jaoude, C.; Idrees, A.K. Two Tier Data Reduction Technique for Reducing Data Transmission in IoT Sensors. In Proceedings of the 15th International Wireless Communications Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 168–173. [CrossRef]
16. Dai, C.; Liu, X.; Chen, W.; Lai, C.F. A Low-Latency Object Detection Algorithm for the Edge Devices of IoV Systems. *IEEE Trans. Veh. Technol.* **2020**, *69*, 11169–11178. [CrossRef]
17. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2923–2960. [CrossRef]
18. Pinto, A.; Zhang, Z.; Dong, X.; Velipasalar, S.; Vuran, M.C.; Gursoy, M.C. Energy Consumption and Latency Analysis for Wireless Multimedia Sensor Networks. In Proceedings of the 2010 IEEE Global Telecommunications Conference GLOBECOM, Miami, FL, USA, 6–10 December 2010; pp. 1–5.
19. Khursheed, K.; Imran, M.; Malik, A.W.; O'Nils, M.; Lawal, N.; Thörnberg, B. Exploration of Tasks Partitioning between Hardware Software and Locality for a Wireless Camera Based Vision Sensor Node. In Proceedings of the Sixth International Symposium on Parallel Computing in Electrical Engineering, Luton, UK, 3–7 April 2011; pp. 127–132. [CrossRef]
20. Imran, M.; Shahzad, K.; Ahmad, N.; O'Nils, M.; Lawal, N.; Oelmann, B. Energy-Efficient SRAM FPGA-Based Wireless Vision Sensor Node: SENTIOF-CAM. *IEEE Trans. Circ. Syst. Video Technol.* **2014**, *24*, 2132–2143. [CrossRef]
21. Motlagh, N.H.; Bagaa, M.; Taleb, T. UAV-Based IoT Platform: A Crowd Surveillance Use Case. *IEEE Commun. Mag.* **2017**, *55*, 128–134. [CrossRef]
22. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst.* **2018**, *37*, 2348–2359. [CrossRef]
23. Wang, Z.; Li, Q. Information Content Weighting for Perceptual Image Quality Assessment. *IEEE Trans. Image Process.* **2011**, *20*, 1185–1198. [CrossRef]
24. Fookes, C.; Lin, F.; Chandran, V.; Sridharan, S. Evaluation of image resolution and super-resolution on face recognition performance. *J. Vis. Commun. Image Represent.* **2012**, *23*, 75–93. [CrossRef]
25. Yan, J.; Zhang, X.; Lei, Z.; Liao, S.; Li, S.Z. Robust Multi-resolution Pedestrian Detection in Traffic Scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013.
26. Alhilal, M.S.; Soudani, A.; Al-Dhelaan, A. Low power scheme for image based object identification in wireless multimedia sensor networks. In Proceedings of the 2014 International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, Morocco, 14–16 April 2014; pp. 927–932. [CrossRef]
27. Gu, K.; Liu, M.; Zhai, G.; Yang, X.; Zhang, W. Quality Assessment Considering Viewing Distance and Image Resolution. *IEEE Trans. Broadcasting* **2015**, *61*, 520–531. [CrossRef]
28. Ur Rehman, Y.A.; Tariq, M.; Sato, T. A Novel Energy Efficient Object Detection and Image Transmission Approach for Wireless Multimedia Sensor Networks. *IEEE Sens. J.* **2016**, *16*, 5942–5949. [CrossRef]

29. Romić, K.; Galić, I.; Leventić, H. Influence of the input image resolution on the staircase detection. In Proceedings of the International Symposium ELMAR, Zadar, Croatia, 12–14 September 2016; pp. 177–180. [CrossRef]
30. Yazidi, A.E.; Lahcen Hasnaoui, M.; Azizi, M.S. Sensitivity analysis of latency to data size in Spark environment. In Proceedings of the IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Kenitra, Maroc, 2–3 December 2020; pp. 1–5. [CrossRef]
31. Krug, S.; O'Nils, M. Modeling and comparison of delay and energy cost of IoT data transfers. *IEEE Access* **2019**, *7*, 58654–58675. [CrossRef]
32. Jin-Yu, Z.; Yan, C.; Xian-Xiang, H. Edge detection of images based on improved Sobel operator and genetic algorithms. In Proceedings of the IEEE International Conference on Image Analysis and Signal Processing, Linhai, China, 11–12 April 2009; pp. 31–35. [CrossRef]
33. Khosravy, M.; Gupta, N.; Marina, N.; Sethi, I.; Asharif, M. Morphological Filters: An Inspiration from Natural Geometrical Erosion and Dilation. In *Nature-Inspired Computing and Optimization*; Modeling and Optimization in Science and Technologies; Springer: Cham, Switzerland, 2017; Volume 10. [CrossRef]
34. UNI-TREND TECHNOLOGY. UNI-T. Available online: https://www.uni-trend.com/meters/html/product/tyyq/Borescope/UT658/UT658DUAL.html (accessed on 6 May 2021).
35. Shallari, I.; Leal, I.S.; Krug, S.; Jantsch, A.; O'Nils, M. Design Space Exploration for an IoT Node: Trade-Offs in Processing and Communication. *IEEE Access* **2021**, *9*, 65078–65090. [CrossRef]