COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Signal, Image, Vision*

Par
Quentin MILOT

Precision refinement acceleration using statistical methods for quality estimation

Thèse présentée et soutenue à Rennes, le 17 Décembre 2025
Unité de recherche : IETR
Thèse N° : 25ISAR 31 D25 - 31

Rapporteur·trice·s avant soutenance :

Alberto BOSIO	Professeur des Universités, Ecole Centrale de Lyon
Fabienne JEZEQUEL	Maître de Conférences, HDR, Université Panthéon-Assas

Composition du Jury :

Président·e :	Florent DE DINECHIN	Professeur des Universités, INSA Lyon
Examinateur·trice·s :	Alberto BOSIO	Professeur des Universités, Ecole Centrale de Lyon
	Florent DE DINECHIN	Professeur des Universités, INSA Lyon
	Fabienne JEZEQUEL	Maître de Conférences, HDR, Université Panthéon-Assas
	Thibault HILAIRE	Professeur des Universités, Sorbonne Université
Dir. de thèse :	Daniel MENARD	Professeur des Universités, INSA Rennes
Enc. de thèse :	Mickaël DARDAILLON	Maître de Conférences, INSA Rennes

Acknowledgement

I would like to thank my thesis supervisors, Daniel Menard and Mickaël Dardaillon, for their support and guidance throughout my three years of PhD study.

I would also like to thank my reviewers, Alberto Bosio and Fabienne Jezquel, for taking the time to review these documents, as well as the members of my jury, Florent De Dinechin and Thibault Hilaire.

I would also like to thank Eva Giboulot for her help and advice on the stratified sampling and importance sampling methods.

Furthermore, I would like to thank Salomé Guinaudeau, L. Koulinski and my friends and family for their support over the past three years.

Finally, I would like to thank everyone in the EII department and the VAADER research team for my five years as an INSA Rennes student and three years as a PhD student.

I consider this work to be a collective effort, which would not have been possible without any of the people mentioned above.

Table of Contents

1 Introduction **9**

1.1 General Context 9

1.1.1 Electronics and Sustainability 9

1.1.2 Larger scaling computations 10

1.1.3 Solutions for computation reduction 12

1.2 Objectives and Contributions 13

1.3 Outline 15

I Background **17**

2 Approximate computing **18**

2.1 Introduction 18

2.2 Approximate computing domains 18

2.2.1 Computation level 18

2.2.2 Hardware level 19

2.2.3 Data level 20

2.3 Data Representation 21

2.3.1 Floating-point 21

2.3.2 Fixed-point 24

2.3.3 Other representations 25

2.4 Conclusion 28

3 Precision Optimization **30**

3.1 Introduction 30

3.2 Quantization error estimation 30

3.2.1 Analytical Estimation 31

3.2.2 Estimation by Simulation 34

3.3 Design space exploration for quantization 35

3.3.1 Heuristic optimization for quantization 36

3.3.2 Machine learning quantization optimization strategy 40
3.4 Conclusion 42

II Contributions 43

4 Precision refinment for custom floating-point 44
4.1 Introduction 44
4.2 Custom floating-point 45
4.2.1 Custom floating-point description 45
4.2.2 Existing frameworks for wordlength optimization 46
4.2.3 Custom floating-point emulation 47
4.2.4 Exponent and Mantissa correlation 47
4.2.5 Range Analysis 51
4.3 Proposed Method 52
4.3.1 Custom floating-point flowchart 52
4.3.2 Global Wordlength Optimization strategies 57
4.4 Experimental results 61
4.4.1 IIR filter 61
4.4.2 Kalman filter 61
4.4.3 Squeezenet CNN 61
4.4.4 Results 62
4.5 Conclusion 65

5 Stratified sampling for precision optimization 68
5.1 Introduction 68
5.2 Confidence interval with Monte-Carlo simulations 69
5.3 Proposed method based on Stratified Sampling 71
5.3.1 Stratified Sampling concept 71
5.3.2 Subspace selection 73
5.4 Metric for grouping inputs 74
5.4.1 Specific metrics for classification problems 74
5.4.2 Difficulty 78
5.5 Experimental results 79
5.5.1 Comparison of the metric for input grouping 79

5.5.2 Comparison on a complete optimization 81
5.6 Conclusion 84

6 Importance Sampling for precsision optimization 86
6.1 Introduction 86
6.2 Importance Sampling 87
6.3 Probability Density Function selection 87
6.3.1 Simulation using Gaussian distribution 88
6.3.2 Simulation using Zipf distribution 90
6.3.3 Curse of High dimensionality 90
6.4 Adaptive Importance Sampling 91
6.5 Experimental results 92
6.6 Conclusion 93

7 Conclusion and future works 95
7.1 Thesis contributions 95
7.2 Future works 98
7.2.1 Short term perspectives 98
7.2.2 Long term perspectives 99

A French summary 103
A.1 Contexte Général 103
A.1.1 Électronique et développement durable 103
A.1.2 Calculs à plus grande échelle 104
A.1.3 Solutions pour la réduction des calculs 106
A.2 Objectifs et Contributions 107
A.3 Plan 109

Bibliography 111

Chapter 1

Introduction

1.1 General Context

1.1.1 Electronics and Sustainability

The evolving climate conditions around the world require us to rethink our entire production process. Some laws and norms have been established to address these requirements. For example, at national level, France's 2015 Energy Code (Article L100-4) [Fra23] calls for a 40% reduction in Greenhouse Gas (GhG) emissions by 2030 and, at European level the European Union set a goal of achieving carbon neutrality by 2050 [dév25].

To meet these criteria, every production sector needs to be considered. According to [Com25], the Information and Communication Technologies (ICT) sector's impact, from production to end of life, represents 4% of total GhG emissions. Although this number seems small, each sector must reduce its impact in order to achieve the desired reduction. For this reason, Life Cycle Assessment (LCA) [HS02] research has been developed over the years for electronic devices [Boy11]. This analysis takes into account the five stages of a product's life: 1) Resource extraction; 2) Manufacturing; 3) Transport; 4) Use; 5) End of life.

This method allows determining the impact of each life cycle stage and where to focus effort for a global optimization. For example, in [Lou+23], the authors provide an LCA of a Dell computer used for five years. Figure 1.1 presents the GhG emissions expressed in kilograms of CO2 equivalent (kgCO2eq) for each stage of the computer's LCA. Most of the emissions are due to the components of the Dell computer with 248.17 kgCO2eq. The second-most significant source of emissions is electricity consumption during its lifetime with 29.79 kgCO2eq.

For each LCA stage, several approaches can be envisaged [Pel+24] to reduce the environmental impact. For example, increasing the recyclability of a product reduces its impact at the end of its life. Increasing the product's lifetime reduces its impact during use

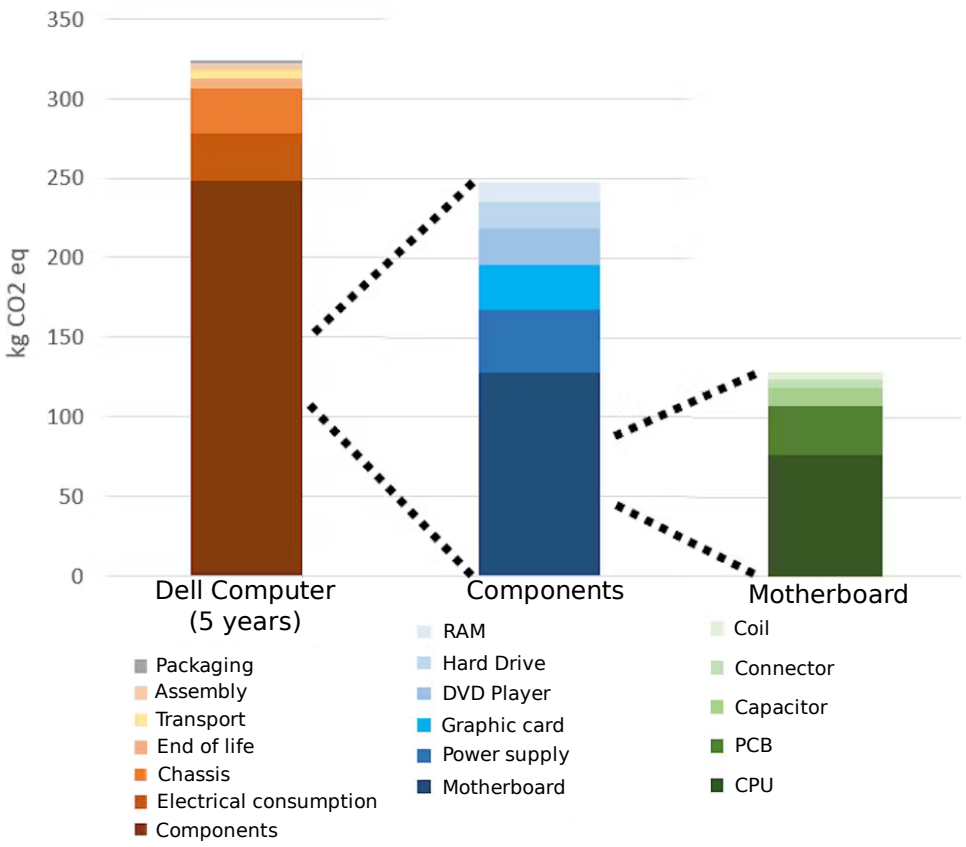

Figure 1.1 – Emissions in CO2 equivalents for a Dell computer over 5 years [Lou+23](extracted and translated from [Pel+24]).

and overall. Finally, reducing a product's energy consumption reduces its impact during usage. Nevertheless, these recommendations go against the current trend of computing and electronic applications, as presented in the next section.

1.1.2 Larger scaling computations

Over the years, the complexity of numerous applications has increased significantly. This increase in complexity can be explained by the greater required performances of these applications and the increasingly complex tasks they perform. A good example of this evolution is the growth of the machine learning field. Figure 1.2, extracted from [Bia+18], represents the accuracy of convolutional neural networks (CNNs) according to their number of parameters and floating-point operations (FLOPs). A tendency for an increase in size to gain accuracy can be observed.

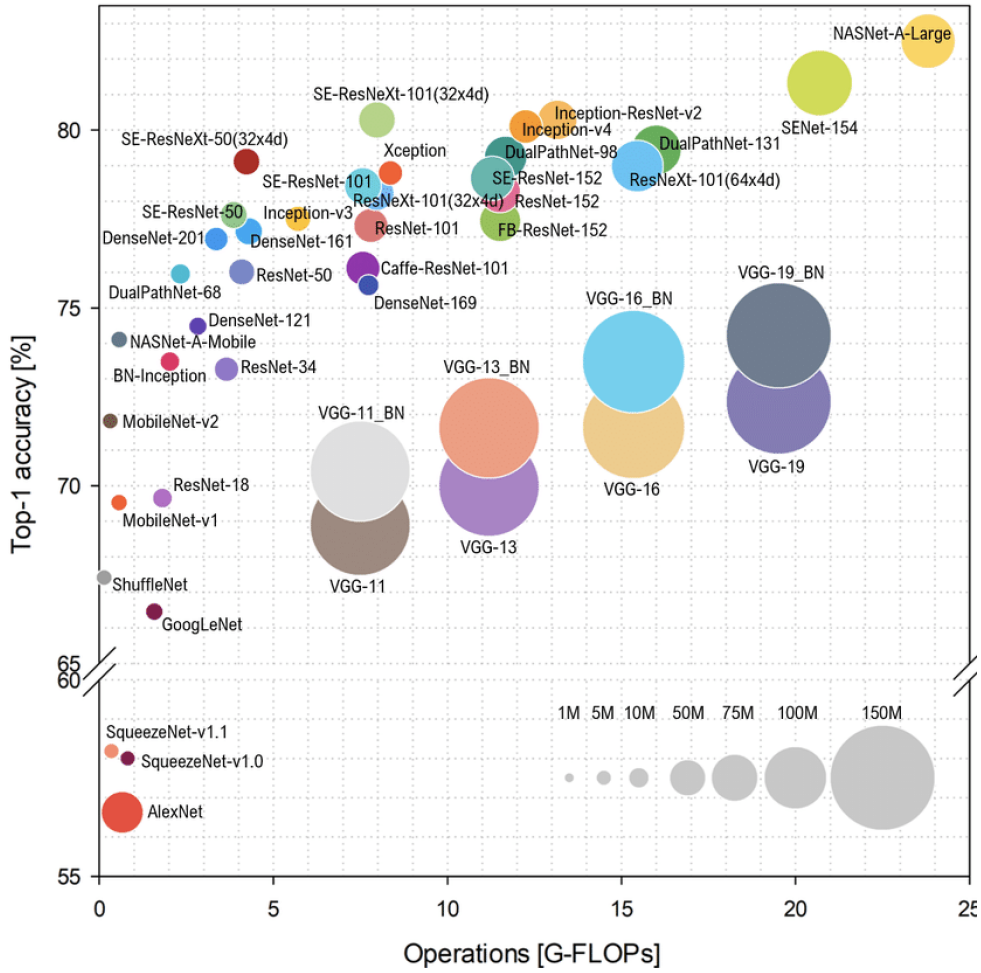

Figure 1.2 – Diagram of comparison of different CNN according to number of Operation and accuracy [Bia+18].

Another good example is radio astronomy. For their study purposes, the increased resolution of radio telescopes leads to an increase in the volume of data produced. This makes the field look for energy-efficient optimization [Cor+22].

This tendency to increase goes against the climate recommendations defined in Section 1.1.1. This tendency also incurs significant costs for the company. For these reasons, solutions have been explored to reduce application computation, as presented in the next section.

1.1.3 Solutions for computation reduction

In [Eec23], the authors present a formula to synthesize global emissions from human sources in electronics usage, F_{op}, according to the number of chips or dies produced, C; the energy efficiency (energy consumption per chip during their lifetime),$\frac{E}{C}$; and the carbon intensity, $\frac{F}{E}$. This gives the following equation:

$$F_{op} = C \times \frac{E}{C} \times \frac{F}{E} \tag{1.1}$$

This equation offers several ways to reduce the impact of an application. The first is reducing the number of chips or dies produced, C. This parameter is mostly a political decision and depends on the other two parameters. Indeed, a reduction in energy consumption often leads to what is called the rebound effect. The rebound effect is an increase in production linked to a cost reduction. However, it is still important to reduce the other variables to minimize the global impact if the number of chips or dies remains the same. The carbon intensity $\frac{F}{E}$ depends on the country and the way electricity is produced. The energy efficiency, $\frac{E}{C}$, is more related to the field of electronic engineering. There are multiple methods to improve the energy efficiency, by optimizing the software and the hardware.

More specifically, in this research work, the domain of approximate computing has been explored. Approximate computing involves adjusting the quality tolerance of an application to reduce energy consumption and accelerate processing. We explored this method due to its potential to be generalized for many applications through automated tools. The concept of approximate computing has been extensively studied during the two last decades and has been materialized by commercial Computer Aided Design (CAD) tools.

For example, WeDoLow, a French company founded by Justine Bonnot, and exploiting more than 20 years of research at institutions like IETR and Inria, aims to provide an optimization framework to improve the efficiency and reduce the implementation cost of C/C++ embedded systems [WeD]. They focus on multiple axes, such as implementing SIMD optimization, realizing polynomial approximation, optimizing vector size for memory reduction, and quantifying variables. Their tool performs static and dynamic analyses of the code to determine potential optimizations and the range of variables.

As a partner for this thesis, WeDoLow's analytic tools were used on the application to perform the code analysis and transformation for the optimization process. The ana-

lytic tool used is a source-to-source compiler. Their tools aim to be integrated into the workflows of other companies that develop applications, with the goal of increasing application performances and reducing their energy consumption. These tools have been used for fixed-point optimization and will be used for our benchmarking and contributions. A crucial consideration, for this kind of CAD tool, is the optimization time. This optimization time must be reasonable, so that the tool is fully adopted by developers. This thesis will explore methods to reduce this optimization time.

1.2 Objectives and Contributions

This thesis focuses on developing tools to reduce the implementation cost of generic applications, with a particular attention on energy consumption. The development of these tools is based on the idea of reducing the energy consumption of chips over their lifetime. This refers to the energy efficiency $\frac{E}{C}$ in equation 1.1. Proposing complete generic tools makes it easier to implement energy consumption reduction in a product's entire workflow. Integrating them into widely used tools, such as continuous integration, could lead to widespread adoption in the industry. However, generic tools limit the used methodology. Not every approximate computing method can be transformed into a generic one. For example, the method of reducing frequency or operation voltage to reduce energy costs, as described in [MYD20], is architecture-dependent. In this context, this thesis focuses on quantization and precision refinement.

The precision refinement process aims at optimizing the data wordlength. Reducing the data wordlength or modifying the type of representation can decrease the energy consumption [Jou+21]. However, this comes with the drawback of introducing an error between the real value and the represented value, which can lead to a quality degradation at the application output. The impact of any modification on this quality degradation has been studied. Some tools [Bon+22] have been developed using a mathematical representation to determine the propagation of errors due to quantization in an algorithm and its impact on the entire application. These methods are highly effective in estimating quality degradation. However, these methods are difficult to extend to unsmooth operations (non differentiable operations). These kinds of operations are becoming increasingly prevalent, especially with the rise of machine learning applications. This thesis focuses on developing generic tools to evaluate the quantization impact with Monte-Carlo simulations.

Simulation-based approaches involve comparing a golden reference model with its

quantized version [Bon+22]. By applying the same input to both models, a comparison can be made between the two versions of the model output. By performing this comparison with multiple values for the inputs, a tendency can be identified to represent the quantization degradation. While this method is more generalizable, obtaining confidence in the estimation requires many values for the inputs and leads to a long simulation time. This long simulation time is an issue, especially when scaled to a complete optimization process which explore the wordlength design space.

To optimize an application using simulation, it is important to explore a large number of configurations, N_C. Each configuration requires to test N_S inputs, and each simulation takes a certain amount of time, t_S. The total optimization time, t_{opt}, can then be expressed as follows :

$$t_{opt} = N_C \times N_S \times t_s \tag{1.2}$$

Reducing each of these parameters will reduce the optimization time. The parameter N_C depends on the method used to explore the design space. This depends on the algorithm selected for the optimization and the number of variables to optimize. The parameter N_S depends on how quality is estimated and the desired level of confidence in the estimation. The parameter t_s depends on the method used to simulate the impact of a limited precision. The goal of this thesis is to accelerate the optimization process for precision refinement. The parameter t_s has been explored in the literature [SIE22; DK24], but its scope is more related to compilation tools and synthesis. This thesis will focus on exploring N_S, the number of input samples and N_C, the number of configurations.

In this thesis, to accelerate the optimization process for precision refinement, the following contributions have been proposed.

Precision refinement for custom floating-point. For the first contribution, we propose an approach that optimizes data wordlength for applications using custom floating-point datatype. This approach determines the wordlength of the exponent and mantissa for each data. This approach reduces the number of configurations to be explored, N_C by analyzing the data dynamic range. From this analysis, a boundary is determined for the exponent wordlength. Then, the optimization problem is decomposed in four optimization strategies. These strategies distribute the optimization between reducing the exponent and mantissa wordlengths. These strategies has been tested on three different applications to estimate the tradeoff between the implementation cost and the quality.

Stratified sampling to accelerate quality degradation evaluation. The two other contributions explore statistical tools to reduce the number of samples N_S used to evaluate the quality degradation due to quantization. The second contribution exploits stratified sampling method. This method groups samples with similar behavior, to reduce the global variance of the estimated quality for the configuration. With some preprocessing analysis, this method allows us to reduce the number of samples required for a similar confidence interval by an order of magnitude.

Importance sampling to accelerate quality degradation evaluation. Like the previous contribution, this third contribution aims at reducing the number of samples N_S used to evaluate the quality degradation due to quantization. This contribution is based on importance sampling, which modifies the law used to draw samples in order to draw more informative samples.

1.3 Outline

This thesis is divided into two parts. The first part provides the necessary background information. Chapter 2 presents the concepts of approximate computing and precision in detail. We present methods from the literature for the three levels of approximate computing: the hardware level, the computing level, and the data level. Then, we focus more on the data level by describing data representation methods. Chapter 3 presents various tools for precision optimization. In this chapter, we present two ways to estimate quantization error: analytical estimation and simulation-based estimation.

The second part of the thesis presents the contributions proposed during this research work. In chapter 4, the first contribution is presented. The proposed approach that optimizes data wordlength for applications using custom floating-point precision datatype is described. After the presentation of the custom floating-point characteristics, the proposed method is explained and the four optimization strategies to determine the exponent and mantissa wordlength are detailed. Finally, the experimental results are provided and the performances of the different strategies are compared. This contribution has been published first at DASIP 2024 and then extended and submitted in Journal of System Architecture.

In chapter 5 the second contribution is presented. The proposed method based on stratified sampling reduces the global variance of the output quality estimation by group-

ing samples with similar behavior. After the presentation of the confidence interval on quality estimation in the context of Monte-Carlo simulations, the proposed method based on stratified sampling is described. Different metrics for grouping input data are evaluated. Finally, the experimental results are provided and the efficiency of the proposed approach is evaluated for the different input grouping metrics. This contribution has been published in Design & Diagnostics of Electronic Circuits & Systems (DDECS) 2025.

In chapter 6 the third contribution is presented. The proposed method based on importance sampling modify the law to draw samples in order to draw more informative samples. After the presentation of the proposed method, the different Probability Density Function selected to modify the law to draw samples are detailed. Then an Adaptive Importance Sampling Algorithm is proposed to estimate the quality. Finally, the experimental results are provided and the performances of the proposed approaches are evaluated.

Finally, in Chapter 7, we conclude this thesis and propose future perspectives.

Part I

Background

Chapter 2

Approximate computing

2.1 Introduction

The increasing complexity of digital systems requires their optimization in order to reduce implementation costs. Numerous modern applications process data and the accuracy of these application output is a interesting lever for reducing the implementation cost. Approximate computing is a paradigm that modifies the execution of an algorithm to produce outputs that are inaccurate but yet usable. Using these outputs provides the benefits of lower implementation cost through faster execution time, lower memory usage and reduced energy consumption.

In the first section of this chapter, a non-exhaustive list of approximate computing techniques are presented. Approximate computing can be considered at three levels : the computation level, the hardware level and the data level.

The second section of this chapter focus on the data level of approximate computing. The two commonly used arithmetic corresponding to floating-point and fixed point are described. Then, the posit and logarithmic arithmetic are briefly presented.

2.2 Approximate computing domains

2.2.1 Computation level

Approximate computing at computation level involves modifying an algorithm itself to reduce its complexity by decreasing the number of processing operations and memory exchanges. This modification comes in two forms: Computation Skipping and Computation Approximation.

Computation Skipping involves omitting parts of a program. A processing may be removed if it refines the output and does not alter the nature of that output. Two granularity levels can be considered for omitting parts of a program. At fine grained, a part of the

computation, corresponding to one up to several expressions, is skipped like in the Loop Perforation [Sid+11] and Early Termination [Zha+14]. Loop Perforation is the concept of executing only a subset of the iterations in loops. The number of skipped iterations is defined by a parameter called the perforation rate. Early Termination [Zha+14] ends a computation process before reaching its end. This technique is well adapted for iterative refinement algorithms in which an iterative method is used to improve the output algorithm accuracy at each iteration. This approach involves determining the convergence of a computation loop using techniques such as gradient descent. If convergence has been achieved, the loop will be terminated early and set to that value. At coarse grained , this method can be extended to skip an entire processes or task in complex applications and referred to Block Skipping [Bon+22].

Computation Approximation involves replacing complex computations with faster and simpler alternatives that approximate the results. One method for this is Fuzzy Memoization, which stores the results of computations in a Lookup Table (LUT) to reuse them later [Sur+15]. Reading from a LUT table is faster and less costly in terms of power than performing multiple calculations of the same computation. Furthermore, by masking the N least significant bits from the operands, groups of operands with similar results can be assigned to the same LUT address to accelerate and further reduce costs. Another method is Mathematical Approximation, which involves implementing simpler mathematical functions in place of more complex ones [BNM16]. An example consists of using a polynomial function with predefined coefficients stored in an LUT instead of a more sophisticated function, as the polynomial function is faster than its more complex counterpart. In [DT05], the authors present generic tools for exploiting multipartite tables. This method is based on the bipartite method [SM95], which consists of approximating a function with affine segments. For their generic tool, the authors defined a design space exploration that will be examined by dividing it into the least and most significant subwords. These sub-words will be stored in tables of variable size to decompose the desired function.

2.2.2 Hardware level

Approximate computing at the hardware level involves using circuits with lower energy consumption or faster execution with known inaccuracy. It exists in three forms: Approximate Processing Units, Circuit Parameters Modification, and Unreliable Storage.

Approximate Processing Units consist of using dedicated units that are known to be

faster than their counterparts. Their increased speed comes from known inaccuracies, such as Inexact Adders. These adders reduce the length of the critical carry chain path in adders to speed up the addition process [Gup+11].

Circuit Parameters Modification involves altering parameters such as frequency or operation voltage to reduce energy costs. Systems often operate at constructor-recommended voltages and frequencies, but adjusting critical voltages can lead to reduced power consumption while still producing accurate results. The drawbacks include slower computation times and inaccuracies. Additionally, manipulating these parameters may create cybersecurity vulnerabilities for attackers [MYD20].

Unreliable Storage refers to using a memory system with a known tendency to decay. The parameters, such as supply voltage [Fru+15] or refresh rate [Rah+14], can be adjusted according to the operation parameters of the memory system to use them effectively.

2.2.3 Data level

Data level approximate computing consists in exploring the amount of data used for the computation or how to represent them. It exists in three forms : Reduction of the number of data, Less up-to-date data and Precision Optimization.

The number of data reduction involves decreasing the volume of data to be processed. This can be done in the space domain (by down-scaling the data *i.e.* reducing the number of data in each dimension) or in the time domain (by down-sampling the data *i.e.* reducing the sampling rate) [ACN14]. This method is particularly used in signal, image, and video processing.

Less up-to-date data consists of using outdated data. This reduces the amount of data used by a processing platform. Depending on the application, relaxing synchronization with outdated data in parallel programs can lead to significant reduction in computation time and energy consumption. In [Ren+12] the authors demonstrate that the K-means algorithm on an IBM Power 7 spends 90% of the computation time on synchronization. Using the method described in [KAK18], approximate computing selection allows one to avoid critical error synchronization, increasing parallelization at the cost of a small loss in accuracy.

Precision optimization consists in modifying the data type or the data wordlength used for the data representation. The use of specific data types can accelerate the computation or consume less energy. The reduction of the wordlength also leads to a reduction in energy and a speed up in the computation. For example, an 8 bits adder using fixed-point rep-

resentation costs 70% less picojoules per operation than a 32 bits adder using fixed-point representation, which itself costs 75% less picojoules per operation than its floating-point counterpart on Tensor Processing Unit designed with a 45 nm technology node [Jou+21]. However, both of these parameters modify the way the data are represented and can then lead to inaccuracy in the final results.

With the explosion of the number of parameters in applications, the reduction brought by a modification on precision optimization is even more significant. Developing tools to allow an easy optimization would allow to disseminate these methods and to massively reduce the energy consumption from electronic applications. This is why this approximate computing form is the center of this thesis and will be developed in future parts.

2.3 Data Representation

To achieve precision optimization, it is important to understand how data are represented within an application. Data representation can be done according to multiple standards. In this section, we present the two most common representations: floating-point and fixed-point, as well as some less common representations.

2.3.1 Floating-point

The floating-point representation aims at approximating real number by combining a significand commonly known as mantissa and an explicit scale factor which is a power of the base. The base 2 is considered. The data representation consists of three parameters corresponding to a sign part, an exponent part, and a mantissa part (see Figure 2.1).

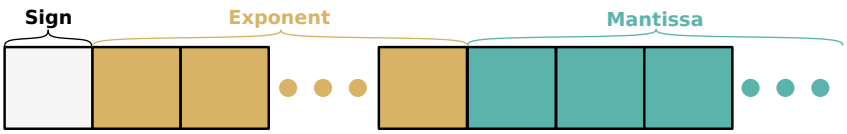

Figure 2.1 – Floating-point representation

The sign part S indicates whether a number is positive (0) or negative (1). The exponent part E represents the closest power of 2 for the number being represented, with an intrinsic bias. This exponent part is represented with W_E bits. This bias B allows encoding only positive exponent for the representation of both positive and negative exponents. Finally, the mantissa part M selects the closest representable number within the selected

power of 2. This mantissa part is represented with W_M bits. These three parts enable to represent a number using the following formula :

$$X = (-1)^S(1 + M) \times 2^{E-B} \tag{2.1}$$

However, there can be multiple combinations of mantissa and exponent that lead to a similar result. For example, $3 = (1 + 0.5) \times 2^1 = (1 + 2) \times 2^0$. The combinations (M = 0.5, E = 2) and (M = 2, E = 0) produce the same result. To avoid wasting bits in the configuration used to represent a number, a standard has been established for the exponent and mantissa. The most common standard is the IEEE 754 standard [08], which limits the mantissa value to the interval $[0; 1[$ and uses the formula given in equation 2.1 for number in the range $[-2^{E_{\max}-B-1}; -2^{-(E_{\max}-B-2)}] \cup [2^{-(E_{\max}-B-2)}; 2^{E_{\max}-B-1}]$ with $E_{\max} = 2^{W_E}$. This standard also defines special cases outside the defined range using certain bit configurations.

The first type are numbers called subnormals, which are in the range $]-2^{-(E_{\max}-B-2)}; 2^{-(E_{\max}-B-2)}[$. These numbers are represented by setting all the exponent bits to zero and using the mantissa bits without the implicit addition of one described in equation 2.1. This method allows increasing the range for numbers close to 0 with a small absolute error given by the following expression

$$X = (-1)^S \times M \times 2^{E-B} \tag{2.2}$$

The two other special numbers are $\pm\infty$ and Not a Number (NaN). The first ones are represented with every exponent bit set to 1 and every mantissa bit set to 0. The sign bits allow to select between $+\infty$ and $-\infty$. The NaN is represented with every exponent bit set to 1 and at least one mantissa bit set to 1. A summary of the types of values is represented in table 2.1.

Table 2.1 – Exponent and mantissa according to the types of values to represent for the IEEE754 norms [08]

Values	**Exponent**	**Mantissa**
0	0	0
Subnormal values	0	$\neq 0$
Normalized values	$[1; 2^{W_E} - 2]$	$[0; 2^{W_M} - 1]$
$\pm\infty$	$2^{W_E} - 1$	0
NaN	$2^{W_E} - 1$	$\neq 0$

For easy implementation in processors, the word length for each part has been normalized. In table 2.2, we describe the characteristics of common formats.

Table 2.2 – Floating-point types wordlength [08]

Type	Wordlength	Sign	Mantissa	Exponent	Bias
float 256	256	1	236	19	262143
float 128	128	1	112	15	16383
float 64	64	1	52	11	1023
float 32	32	1	23	8	127
TF32	18	1	10	8	127
float 16	16	1	9	5	15
bfloat 16	16	1	7	8	127

The most common types are single and double precision encoded on 32 and 64 bits, respectively. These types provide significant accuracy over large ranges without requiring any consideration for quantization from developers in the context of most of the signal and image processing. As a result, they are implemented in most modern processors and GPUs. However, these types can be oversized for the data used in embedded applications, leading to significant memory usage and slower applications. This is why smaller normalized floating-point data types have been introduced. The half precision FP16, introduced in the IEEE standard in 2008, is 16-bit floating point number with 5 bits for the exponent and 10 bits for the mantissa. An 8-bit version is currently under debate between two representations: 4 bits for the exponents or 3 bits for the exponent [Mic+22; Aa22].

With the development of machine learning tools, new floating-point representations have been developed. One such adaptation is the Brain Floating-point (bfloat16), proposed by Google and based on single precision representation. It assumes that the range is larger than the mantissa description and thus shortens the mantissa bits to reduce memory and power consumption while making it more suitable for use on GPUs. Another type is TensorFloat-32 (TF32), a combination of FP16's mantissa and bfloat16's exponent, allowing for a range as significant as bfloat16 while maintaining similar precision to FP16. This format was presented by NVIDIA and is primarily used for intermediate computations on GPUs rather than end-user applications.

2.3.2 Fixed-point

The fixed-point representation aims at approximating real number through the product of a integer number with an implicit scale factor which is a power of two. Thus, the number is made-up of an integer part and a fractional part as illustrated in Figure 2.2. The binary-point position is implicitly defined by the developer. The developer is in charge of inserting bit shifts to perform correctly operations between data having different fixed-point formats. A sign bit is used to represent positive or negative values using two's complement notation.

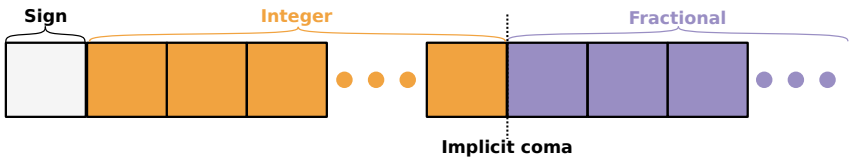

Figure 2.2 – Fixed-point representation

Contrary to what its name could imply, the fixed-point representation doesn't have a fixed format and does not belong to a standard. It requires adapting the integer size I and fractional size F to the values that must be represented. Every natural number in the range $[0, 2^I - 1]$ for a non signed format, or $[2^{-I-1}, 2^{I-1}]$ for a signed format, can be represented. The fractional part allows to represent real numbers in this range with a maximal absolute error Δ_m between the real value and the represented value of 2^{-F}. In Figure 2.3, the absolute error Δ is represented according to the value for two different fixed-point formats. It shows that the maximum absolute error does not evolve with the value being represented.

The most common wordlengths for the sum $I+F$ are the predefined integer wordlengths used in GPUs and processors, such as INT4, INT8, INT16, INT32, and their unsigned counterparts.

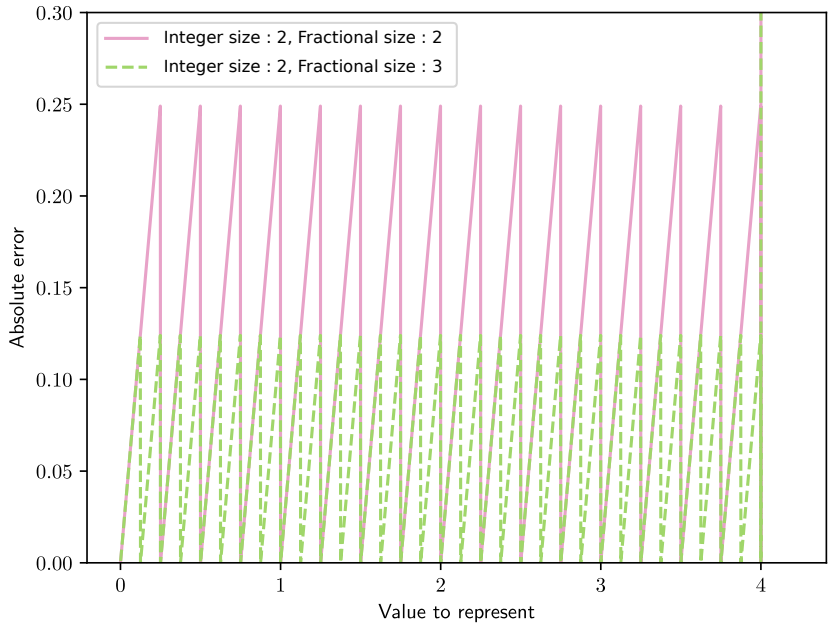

Figure 2.3 – Absolute error Δ of a value according to the value represented and the fractional size using truncation

Fixed-point has the primary advantages of being faster and requiring less energy per operation than floating-point with a similar word length, as illustrated in [Jou+21]. Furthermore, due to its structure, it can be used with large computational devices, such as GPUs.

Quantizing an application to a fixed-point format allows one to obtain an application that is even quicker and cheaper while maintaining decent quality. This is why many quantization tools use fixed-point refinement, such as TensorFlow [Mig17] or HAWQV3 [Yao+21].

2.3.3 Other representations

Other representations have been proposed in the literature over the years to fit more specific use cases or accelerate applications. In this section, we present two representations : the posit and logarithmic arithmetic.

Posit

Posit is a type designed as an alternative to IEEE 754 standard floating-point numbers [GY17]. It is designed to provide a more rigorous and mathematical representation, although it comes at the cost of increased execution time and storage requirements

The Posit representation consists of four parts :

— The sign bit, p which allows for the representation of both positive and negative numbers using the 2's complement of the entire number.
— The regime, a variable-length part consisting of a series of 0s and 1s. The regime is defined until a different type of bit (opposite regime bits) is represented. For example, a sequence of two 0s followed by a 1 represents a regime of length $m = 2$. The size of this regime es defined a value called $useed = 2^{2^{es}}$ The regime selects a value corresponding to $useed^k$, where $k = -m$ if the regime consists of 0s and $k = m - 1$ if it consists of 1s.
— The exponent part e allows for subdivisions within the number represented by the regime part. The maximum size of this exponent part is defined as es, but its final size depends on the remaining bits left by the regime part.
— The fractional part f is similar to the mantissa part in floating-point representation and allows for selection inside the power of two selected by the regime part and the exponent part. It is encoded as an integer that will be divided by the *useed* of the number, with an implicit addition of one.

This four parts allows to represent a number X according to the following formula :

$$X = (-1)^p \times useed^k \times 2^e \times f \tag{2.3}$$

An example of the regime and exponent can be seen in Figure 2.4. A representation of a number can be found in Figure 2.5.

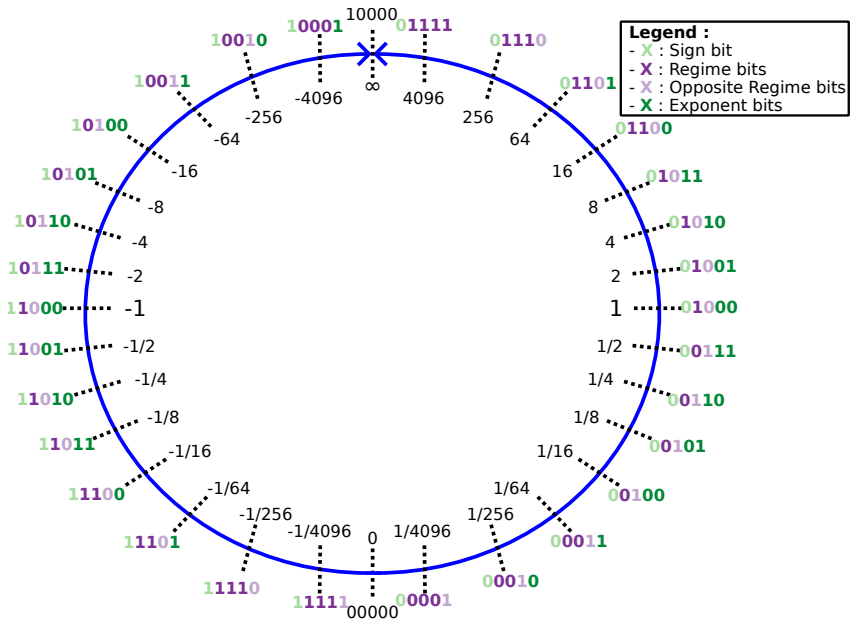

Figure 2.4 – Posit construction with two exponent bits for a word of size 5 [GY17]

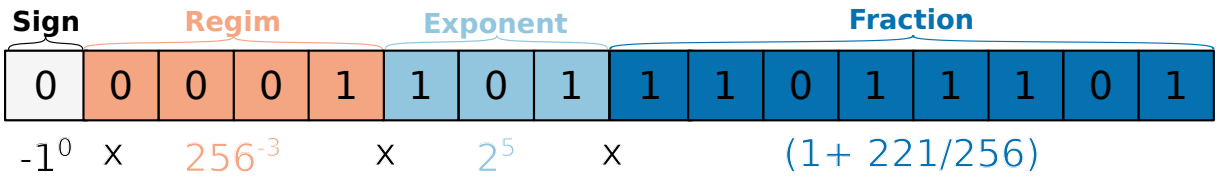

Figure 2.5 – Posit bit string and mathematical formula example [GY17]

At the time of writing this document, Posit is not widely used. The authors in [De +19] discuss both the advantages and disadvantages of Posit, which may explain its marginality. The advantage is that it can represent more numbers, providing greater accuracy for values close to 1. However, its trade-off for this benefit is reduced accuracy for very large values. Nevertheless, it still offers a larger range than floating-point for similar precision. Moreover, unlike floating-point representation, Posit maintains consistency in summations. The disadvantages include some inexactitude in multiplication, such as rounding errors or multiplicative cancellations.

Logarithmic Arithmetic

The Logarithmic Numerical Systems (LNS) is a representation structured for application in signal processing and neural networks [CP13; Nev+21]. LNS represents a number

in two parts: the base-2 logarithm of the number L_X and the sign of the number S_X, using the following formula:

$$L_X = \lfloor \log_2(x) \rfloor, S_X = sign(x) \tag{2.4}$$

This method allows to represent an important number of possible values. However, it cannot represent numbers between the powers of 2, which leads to an important absolute error and relative error. This method is then really adapted to systems that want to classify huge ranges of values or values that can be translated using log, such as in signal processing.

This method of representation also has the advantage of making multiplication easier according to the logarithm properties. For a number $Z = X \times Y$, represented in LNS by the numbers L_Z, L_X and L_Y, with a sign S_Z, S_X and S_Y, the values L_Z and S_Z can be determined as follows:

$$L_Z = L_X + L_Y, S_Z = S_X \oplus S_Y \tag{2.5}$$

However, the addition is more complexe due to the logarithm properties. The convention for the addition of two logarithms numbers is defined as follows: For a number $Z = X + Y$, with $|X| > |Y|$:

$$L_Z = L_X + \log_2(1 + 2^{L_Y - L_X}), S_Z = S_X, \tag{2.6}$$

This limitation on basic operations makes it really hard to use in most applications. This is why this representation is really restrained in its usage.

2.4 Conclusion

Approximate computing is a paradigm for reducing costs and accelerating applications by potentially reducing their quality. This can be done at multiple levels: computation, hardware, or data. Due to the explosion in the number of data in real applications over the years, the data-level approach has been widely explored in the literature.

Data-level approximate computing explore the way data are represented in an application. There are two major ways to represent data: floating-point and fixed-point. The advantage of the floating-point representation is its ability to represent a wide range of numbers, but it sacrifices speed and introduces potential absolute error with real values.

The fixed-point representation's strengths are speed and constant absolute error, but this arithmetic leads to a smaller range.

Other ways of representing data exist. In this chapter, we have presented two methods: the posit and logarithmic arithmetic. These two examples illustrate two alternatives to floating-point and fixed-point arithmetic. They are either not as efficient as their competitors or too specific to be used in generic applications. For this reason, wordlength optimization are primarily developed for floating-point and fixed-point arithmetic.. In the next chapter, we will develop strategies to optimize precision for an application.

Chapter 3

Precision Optimization

3.1 Introduction

As mentioned in the previous chapter, the number of bits used to represent a data will influence the implementation cost and the quality of the application result. The precision optimization process involves reducing the cost C of an application by modifying the data precision. Let $\mathbf{w}$, be the vector grouping the wordlength w of each data of the considered application. This cost can represent multiple metrics, such as memory cost or energy consumption. The cost reduction is achieved while maintaining the quality at the application output, λ above a threshold, λ_{min}. This precision refinement process can be formulated with the following optimization problem

$$\min_{\mathbf{w}}(C(\mathbf{w})) \quad \text{subject to} \quad \lambda(\mathbf{w}) \geq \lambda_{min} \tag{3.1}$$

In this optimization process, the implementation cost C and the quality λ have to be evaluated according to the data wordlength. To solve this optimization problem, an iterative algorithm will be considered and the cost and the quality will be evaluated multiple times. Thus, this is important to have efficient evaluations (fast and accurate) to limit the optimization time. In the first part of this chapter, existing methods to estimate the effect of quantization error on the application output quality are described. In the second part of this chapter, existing methods to explore the wordlength design space are presented.

3.2 Quantization error estimation

In mathematics and signal processing, quantization is the process of representing a value as an approximate version. In computer applications, any representation can be considered as a quantization of a real value. However, the accuracy of the largest types

of representation, as presented in Section 2.3.1 of Chapter 2, can make them appear close to the real value. Since these types are computationally expensive, quantization is usually used to reduce their impact by modifying the data representation or reducing the wordlength [Bon+22].

In order to refine precision to reduce implementation cost, it is important to estimate the quantization error of a given configuration of data wordlength. This quantization error represents the loss in quality due to quantization. Quality estimation is the major part for the decisions made by the optimization process.

In this section, we present the two approaches for quality estimation: Analytical estimation and simulation-based estimation [BMS22]. We also discuss of the specificities of the quantization process in the context of machine learning applications.

3.2.1 Analytical Estimation

Quantization error modeling

The mathematical model for quantization error is defined according to the format used to represent the data and the application. This model is linked to the absolute and relative errors. Analytical estimation has primarily been used for fixed-point because the absolute error is constant.

Bounds of the quantization error The quantization error depends on the data representation. This error can be represented using the absolute error [BMS22]. Let e_x, be the difference between the desired value, x, and its quantized version, x_Q. The maximum absolute error for fixed-point representation $\Delta^m_{fix} = \max(|x|)$ is constant and independent of the value x. For a given fractional size, W_f, the maximum absolute error Δ^m_{fix} is equal to

$$\Delta^m_{fix} = 2^{-W_f} \tag{3.2}$$

The absolute error for a floating-point representation Δ^m_{fl} is not constant. This error depends on the mantissa size, W_M a and the value x

$$\Delta^m_{fl} = 2^{\lfloor \log_2(x) \rfloor - (W_M+1)} \tag{3.3}$$

These boundaries are relevant for values that can be represented by the data types [BMS22].

For values that are not representable, the absolute error increases linearly. However, data types are typically adapted to represent all possible values in the interval $[\underline{x}, \overline{x}]$.. Using these boundaries allows them to represent the worst-case scenario and ensure the accuracy of the quantization.

Statistical moments of the quantization error The characteristic of the quantization error for fixed-point arithmetic has been extensively studied and especially by Widrow [Wid61] who has proposed a specific model. The quantization of a signal x leads to quantized signal $x_Q(n)$ and a quantization error $e_x(n)$ as illustrated in Figure 3.1. This error is additive *i.e.* $x(n) = x_Q(n) + e_x(n)$ and can be assimilated as a random variable. This random variable is uniformly distributed. The error $e_x(n)$ is uncorrelated with the quantized signal $x(n)$ and is a white noise *i.e.* the sample $e_x(n)$ is not correlated with its previous or future versions : $e_x(n - \tau)$ (with $\tau \neq 0$).

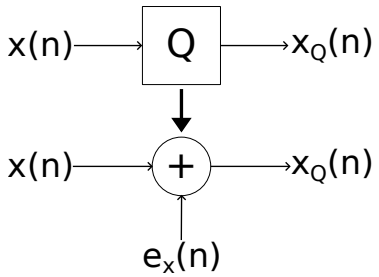

Figure 3.1 – Widrow model for fixed-point quantization noise(extract from [BMS22])

This property enables to estimate the impact of an error $e_x(n)$ at the application output by propagating this error throughout the entire application [HL13; BMS22]. As presented in [CCL99; Mén+10], this model can determine the mean μ_e and standard deviation σ_e of the quantization model with the following expressions

$$\mu_e = \frac{2^{-W_f}}{2}(1 - 2^{-k}) \tag{3.4}$$

$$\sigma_e^2 = \frac{2^{-2W_f}}{12}(1 - 2^{-2k}) \tag{3.5}$$

with f denoting the number of bits required to encode the fractional part after quantization, and k denoting the number of bits removed by quantization.

This approach can also be extended for floating-point arithmetic. In [CM16], the authors describe floating-point error as moving fixed-point error. Moreover, in [Cat+],

the authors describe a custom metric that allows representing errors for fixed-point and floating-point data types. This enables the treatment of the floating-point and fixed-point quantification error as a noise.

Quantization error propagation

The estimation of the quality for the entire application can be determined using multiple methods and depends of the metric used to evaluate the accuracy.

Bounds of the quantization error To propagate the bounds of the quantization error, interval-based arithmetic can be used. This can take the form of interval arithmetic [BMP06; HLR11] or affine arithmetic [FS04; HLR12]. In interval arithmetic, an interval $[\underline{x}, \overline{x}]$ is defined for each variable. These intervals are then propagated through the application according to computational rules. The rules define the error bounds of the operation output according to the error bounds of the inputs in the worst case.

In Affine Arithmetic, the variable is defined by an affine function instead of an interval. This approach allow keeping track of correlations. However, it has a higher computational cost.

Statistical moments of the quantization error To propagate the statistical moments of the quantization error, perturbation theory can be used with the assumption that the quantization error e_Q is small compared to the signal value x. Different methods [SB04; Caf+10] use a first-order Taylor approximation to linearize the propagation of the quantization error through operation. This leads to a time-varying linear expression of the output noise. In [MS02; MRS08], the expression of the second order moment of the output noise are provided. Second-order or affine arithmetic can also be used for noise propagation. These approaches require that the operation is differentiable. Thus, they are limited only to applications containing only smooth operations.

Analytical estimations can not actually be applied to unsmooth operations (non differentiable operations). Over the years, the number of applications with unsmooth operations, such as *min* or *max*, has exploded. This is especially true with the explosion of machine learning applications. This is why estimation by simulation is being explored more and more in the literature.

3.2.2 Estimation by Simulation

The straightforward approach to evaluate the effect of quantization error is based on simulation. The application is executed on an input dataset for two versions corresponding to the reference model and the tested model. The application results between the reference model and the tested model are compared to analyze the quantization error effect on the results as illustrated in figure 3.2. The reference model is a high-precision version of the application using double or single precision format and the tested model correspond to the precision to be tested. Different error metrics can be computed from the application results.

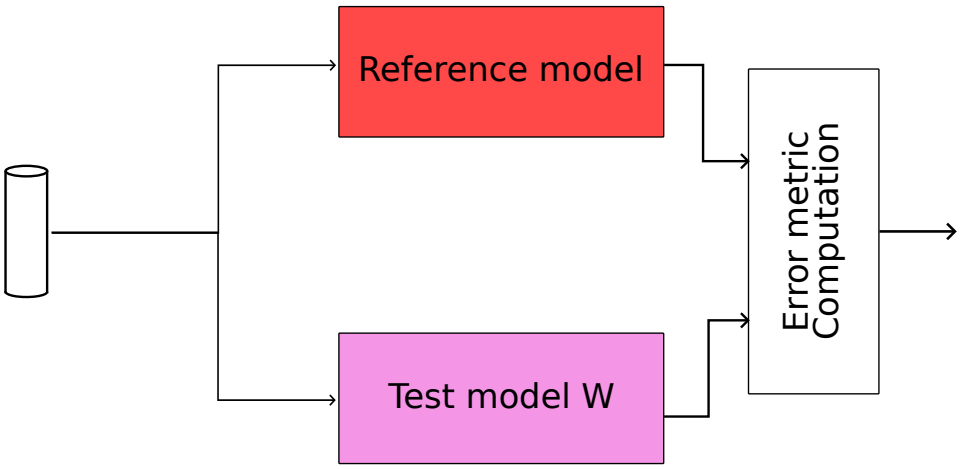

Figure 3.2 – Illustration of the simulation based techniques to determine the error metric (extract from [BMS22])

For the reference model, single or double floating-point are native data-types for computer platforms, this is not the case for the tested fixed-point or custom floating-point data-types. These data types must be emulated on the platform used for simulation. Different `C++` classes have been proposed to emulate the fixed-point mechanisms: `sc_fixed` (*SystemC*) [ACC04], or `ac_fixed` (*Algorithm C Data Types*) [SIE22]. `C++` class attributes define the parameters associated with fixed-point data. Bit-true operations are performed by overloading the different arithmetic operations. The major drawback of the simulation-based techniques is the extremely long simulation times. The simulations are run on floating-point machines, and the extra code used to emulate fixed-point or custom floating-point mechanisms increases execution time by one or two orders of magnitude compared

to traditional simulations with native floating-point data types.

Different techniques have been proposed to reduce this overhead in the context of fixed-point data types. One way is to use more efficient fixed-point data types to reduce the execution overhead. An alternative is to exploit hardware acceleration with FPGA platforms to implement bit-true operations.

To reduce the execution time of quantization error analysis, the number of inputs used for the evaluation can be questioned. Instead of running an exhaustive simulation, Monte Carlo simulation (MCS) are considered. A statistical characterization of the error can be achieved by randomly selecting a set of inputs.

In [BDM19], the authors present a method for estimating noise power at an application's output using MCS. They characterize the noise using multiple values for the inputs and determine the mean and standard deviation of their estimation using estimators. Using the central limit theorem, they determine a confidence interval in their results. By selecting a confidence interval, they determine the minimum number of samples required for their estimation, thereby limiting the execution time. This method allows for simulation while maintaining confidence in the results and reducing the number of required simulations. However, the execution time can be long.

A way to counteract the drawback of the long execution time for quantization error analysis is to approximate the metric instead of evaluate it. In [BMD20], the authors present a method based on a statistical method called Kriging to approximate the noise power due to quantization error. This method estimates the potential value of a fixed-point configuration in a space using nearby configurations evaluated with a defined confidence in the estimation. This enables increasing the number of points estimated without increasing the number of performed simulations.

3.3 Design space exploration for quantization

As presented before, the precision refinement process can be formulated with the following optimization problem

$$\min_{\mathbf{w}}(C(\mathbf{w})) \quad \text{subject to} \quad \lambda(\mathbf{w}) \geq \lambda_{min} \tag{3.6}$$

where C, is the implementation cost, λ, the quality at the application output, λ_{min}, the quality constraint and $\mathbf{w}$, a vector grouping the wordlength w of the N_v variables considered in the optimization process.

In this optimization process, multiple configurations are tested in order to determine an optimized balance between quality degradation and cost reduction (energy consumption or speed). One common method is to replace the original type with the multiple normalized types presented in Section 2.3.1 of Chapter 2. As the space to explore is small, it is possible to explore exhaustively. This method is particularly adapted for software implementation. Indeed, general processors such as CPUs or GPUs are designed and optimized for these types; it is hard to justify the use of a more custom format. [Mic+22]. Custom format are more adapted for hardware implementation such as ASIC of FPGA [DSC17].

To explore this trade-off, it is important to consider configurations of custom wordlengths for fixed-point and floating-point numbers. Exploring custom wordlengths could reveal significant design possibilities. For example, reducing N variables originally represented with 32-bit IEEE 754 floating-point numbers composed of an exponent and a mantissa of sizes 7 and 23, respectively, to a custom floating-point number leads to the exploration of a space composed of $7^N \times 23^N$ configurations. However, this number increases quickly, making it impossible to explore every configuration. For this reason, this section presents methods for exploring the design space. First, we present two heuristic optimizations methods for quantization that explores part of the space to find a sufficient minimum. Then, we present specific methods that use the particularities of machine learning to find an adequate quantization.

3.3.1 Heuristic optimization for quantization

Heuristic optimization is an optimization method used to explore a set of points (wordlength configurations). They are used when the space cannot be explored linearly and when the size or estimation time of the points is too important to explore exhaustively. In this section, we present two heuristics used and adapted for wordlength optimization.

MIN+1

The MIN+1 algorithm [GCa22] is an heurisitic for generic optimization process. This algorithm is composed of two steps. First, a minimal point is determined to start the greedy algorithm deployed in the second step. All the variables of vector $\mathbf{w}$ are set to their maximal value and one variable $\mathbf{w}_i$ is decreased until the quality constraint λ_{min} is no more fulfilled. The minimal value fulfilling the constraint following only its dimension is recorded in the vector $\mathbf{w}_{\min}$. A dichotomy approach can be used to accelerate the

determination of the minimal value. This process is repeated for the N_v variables of vector $\mathbf{w}$. In the second step, a mildest-ascent greedy algorithm is applied to increment by one bit, one of the variables of vector $\mathbf{w}$ at each iteration. This second step starts with the initial solution $\mathbf{w}_{\min}$, for which the quality constraint is not fulfilled. At each iteration, a gradient is computed for each variable to find the best direction $s^*(\mathbf{w})$ *i.e.* the variable for which a one-bit word-length increment leads to the best quality improvement. The optimization algorithm stops when the quality constraint is fulfilled. The algorithm is described in Algorithm 1. This algorithm has the advantage to find a solution quickly. However, this speed lead to a non-optimal solution. For this reason, complementary algorithms such as TABU have been developed.

Algorithm 1 MIN+1 algorithm

Input :$\mathbf{w} \in W^{N_v}$
$\mathbf{w}^* \leftarrow \mathbf{w}$
▷ Gradient descent
for $i \in [1; N_v]$ **do**
  $\mathbf{w}_{current} \leftarrow x$
  while $\lambda(\mathbf{w}_{current}) > \lambda_{min}$ **do**
    $\mathbf{w}^i_{current} \leftarrow \mathbf{w}^i_{current} - 1$
  end while
  $\mathbf{w}^{*i} \leftarrow \mathbf{w}^i_{current} + 1$
end for
▷ Mildest-ascent greedy algorithm
while $\lambda(x^*) > \lambda_{min}$ **do**
  $\mathbf{w}^* \leftarrow s^*(\mathbf{w}^*)$
end while
  return $\mathbf{w}^*$

TABU

TABU search is a meta-heuristic optimization method used as a complement to other heuristic optimization methods [HS21]. This method explores more points around the solution previously found, trying to leave a potential local minimum.

TABU search is based on the neighborhood ($S(\mathbf{w})$) exploration of a point $\mathbf{w}$ with knowledge of previous explored points. This knowledge is used to fill a TABU list T which contains already explored trajectories. At a step k, the best trajectory $s^*(\mathbf{w}_k)$ that is not in T is applied to the current point $\mathbf{w}_k$. The next point $\mathbf{w}_{k+1}$ is then defined through

this trajectory, and T is updated with the explored points [Glo89a; Glo89b]. The global algorithm is described in Algorithm 2.

Algorithm 2 TABU search (general algorithm)

Input :$x \in X$
$k \leftarrow 0$
$T \leftarrow \emptyset$
while $S(x) \backslash T \neq \emptyset$ and $k < k_{max}$ **do**
  Select $s^*(x)$ in $S(x) \backslash T$
  $x \leftarrow s^*(x)$
  $T \leftarrow T \cup x$
  if $C(x) < C(x^*)$ **then**
    $x* \leftarrow x$
  end if
  $k \leftarrow k + 1$
end while
  return x^*

A version specifically designed for wordlength optimization has been proposed in [NMS11]. In this version, the process oscillates between two states: sinking and climbing. The sinking state begins with the current quality $\lambda_{current}$ above the limit λ_{min}. It will reduce each variable by one according to dimension that are not in T and select the trajectory with the smaller trade-off of quality and cost. The point with the best trajectory will be selected as the new current one. This step continues while the quality is above the minimum value. Then, the process switches to the second state, climbing. This step will increase each variable by one according to dimensions that are not in T and select the trajectory with the best trade-off quality/cost. If during this process, a dimension reaches the space boundary, it is added to the TABU list T. This step continues while the quality is over the minimal. Then, the process will add the last dimension climbed to T. If T does not contain all the dimensions, the process switches back to the sinking state. At each selected point, if the quality is over the minimal criterion and the cost smaller than the best one found so far, the point is saved as the new best. The algorithm is described in Algorithm 3.

This method solves two issues for wordlength optimization: the huge optimization time and the trade-off between quality and cost. The huge optimization time is handled by the TABU list T. Each time the optimal trajectory $s^*(\mathbf{w})$ is selected to exceed λ_{min}, the dimension of the trajectory is added to T and locked. Furthermore, if a dimension

Algorithm 3 TABU search for wordlength

Input :$x_{current} \in X^d$
$x^* \leftarrow x_{current}$ ▷ Definition of best as current
$k \leftarrow 0$
$T \leftarrow \emptyset$
state $\leftarrow$ sinking
while $|T| < d$ and $k < k_{max}$ **do**
  for $i \notin T$ **do**
    if state=sinking and $x^i_{current} > X_{min} + 1$ **then**
      Estimate $\hat{\delta}_i$ by decreasing the ith variable.
    else if state=climbing and $x^i_{current} < X_{max} - 1$ **then**
      Estimate $\hat{\delta}_i$ by increasing the ith variable.
    else
      $T \leftarrow T \cup i$ ▷ Add to the TABU list a dimension that reach a limit
    end if
  end for
  if state=sinking **then**
    $i \leftarrow$ position(min($\hat{\delta}$))
    $x^i_{current} \leftarrow x^i_{current} - 1$ ▷ Follow the best gradient
    if $\lambda(x_{current}) < \lambda_{min}$ **then**
      state $\leftarrow$ climbing
    end if
  else
    $i \leftarrow$ position(max($\hat{\delta}$))
    $x^i_{current} \leftarrow x^i_{current} + 1$ ▷ Follow the best gradient
    if $\lambda(x_{current}) > \lambda_{min}$ **then**
      state $\leftarrow$ sinking
      $T \leftarrow T \cup i$
    end if
  end if
  if $\lambda(x_{current}) > \lambda_{min}$ and $C(x_{current}) < C(x^*)$ **then**
    $x^* \leftarrow x$ ▷ Update best point
  end if
  $k \leftarrow k + 1$
end while
  return x^*

should increase outside the defined boundary of the search space, the dimension is also added to the TABU list. The number of dimensions to explore is then reduced across the optimization process, minimizing the number of configurations to test and, therefore, the time required to obtain a good solution. For the trade-off between accuracy and cost, the version uses a gradient that take into account both the cost and quality. This gradient $\delta(\mathbf{w}_{tested}, \mathbf{w}_{current})$ is described with the following equation 3.7.

$$\delta(\mathbf{w}_{tested}, \mathbf{w}_{current}) = \frac{\lambda(\mathbf{w}_{tested}) - \lambda(\mathbf{w}_{current})}{C(\mathbf{w}_{tested}) - C(\mathbf{w}_{current})} \tag{3.7}$$

The MIN+1 and TABU algorithms are examples of generic algorithms that allow one to obtain solutions that balance cost and quality. However, their generic nature and low configuration requirements come at the cost of the quality of the final results. These algorithms mostly find local minima instead of global minima. To find results closer to global minima, more specific strategies have been developed. Machine learning is a great example of this kind of development.

3.3.2 Machine learning quantization optimization strategy

The field of quantization optimization has recently gained popularity alongside the evolution of machine learning applications. These applications are important in terms of size and energy consumption. The reductions brought about by modifying the data representation are even more significant.

Furthermore, their nonlinear structure presents new challenges for precision refinement. For this reason, specific optimization strategies have been developed: Quantization-Aware Training and Post-Training Quantization.

Quantization-Aware Training

Quantization-Aware Training is a quantization method specific to machine learning applications. This method is performed during the training of a model.

The first approach is to adapt the weight representation. Quantizing the weight of the application before realizing the gradient descent takes into account the quantization noise introduced by this step. Some adaptations are required to apply backpropagation, which is used during training, to a non-quantization operator. The mainstream approach is the Straight Through Estimator (STE) [BLC13], which approximates the rounding operation by an identity function and ignores it. This prevents the gradient from being approximated

to zero everywhere due to quantization. Some non-STE approaches have also been developed in the literature, such as those in [Len+17; Ali+20]. These methods allow reduction of the weight precision while maintaining similar quality in machine learning applications. Since the weights represent most of the memory consumption and the variables used for computation, the speed and cost gains are significant [Gho+21].

Another approach is to consider quantization parameters during training. One way to do this is to learn the clipping range of activation functions with quantization, as described in PACT [Cho+18].

Quantization-Aware Training is an efficient method of quantization for machine learning applications that obtains good performances [Kul+22]. However, it is very time-consuming. This method is particularly useful for applications that will be deployed on a single architecture type in large quantities. The benefits of quantization offset the additional training costs. For more modular quantization, post-training quantization is used.

Post-Training Quantization

Post-training quantization is another quantization method for machine learning applications. This method is implemented after the training process on machine learning. It makes quantization more flexible for multiple architectures, cost, and quality optimization. However, it is less accurate and robust than Quantization-Aware Training.

This challenge has been addressed in the literature through multiple approaches. These approaches can focus on quantizing the weight, the activation function, or both.

An example of focusing on weight quantization can be found in [FAG19]. In this article, the authors analyze inherent bias in the mean and variance of weight values and then correct for it thanks to quantization bias.

An example of quantization focusing on the activation function can be found in [Nag+20]. In this article, the authors present another method for rounding quantized data to more easily handle activation functions.

Finally, an example of strategies that focus on quantizing both the activation function and the weights can be found in [CKK19]. In this article, the authors attempt to minimize the L2 distance between the referenced model tensor in floating-point and the tested model tensor in reduced precision by testing multiple configurations.

The goal of this PhD is to develop generic optimization tools for all types of applications, including machine learning. These tools can be classified as a type of post-training quantization on weights and intrinsic data. The goal in developing these tools is to opti-

mize quantization in a short amount of time.

3.4 Conclusion

The precision optimization aims at finding the best compromise between cost and quality. This optimization process involve two main tasks : quantization error estimation and design space exploration.

Quantization error estimation is the concept of determining the error introduced by a precision configuration. There are two main approaches for quantization error estimation: analytical estimation and simulation-based estimation. Analytical estimation represents the quantization-induced degradation as a mathematical model. The error is then propagated through the application using signal theory. This method is fast and reliable, but it cannot be used with unsmooth operations, which are increasingly present in applications. On the other hand, simulation-based estimation is a more generic but leads to long evaluation time. It involves comparing a quantified model to a golden reference using multiple values for the inputs. Due to its generic nature, this method will be developed in this thesis.

Design space exploration methods allow selecting solutions leading to good cost and quality trade-off. These methods allow exploring the wordlength design space with a reduced number of configurations and thus a reasonable optimization time. These methods can be generic, such as the MIN+1 and TABU algorithms, or specific to applications, such as quantization-aware training for machine learning applications. For the sake of generality, the first type of method will be considered in this thesis.

In the following section, we will present our contributions. These contributions aim to reduce the time required for a precision optimization process. For that reason, we propose contributions for design space exploration for custom floating-point data representation in chapter 4 and contributions for reducing the required time for using simulations in chapters 5 and 6.

Part II

Contributions

Chapter 4

Precision refinment for custom floating-point

4.1 Introduction

Data processing applications are mostly designed using simple or double floating-point precision. These data types are used to represent a high precision for most applications, and they are easily accessible on the processor. However, the level of accuracy brought by these types can be oversized for most of the applications. This oversizing can lead to longer execution times, higher circuit area and energy costs than necessary. These overrides hinder to integrate those applications on smaller chips.

The most common technique used to lower these costs is to use data types leading to lower cost like fixed-point arithmetic. With a limited wordlength, it reduces costs and execution time. However, the use of small floating-point representation are more and more explored. They have the advantage to represent a large range of data with a small number of bits.

With the increase of energy consumption due to memory transfer compared to computation, this number of bits reduction could lead to a global consumption that is smaller than with fixed-point arithmetic. This explains the development of normalized, smaller floating-point representations, such as half-precision floating point (FP16) and 8-bit floating-point representation (FP8). [Mic+22; Aa22]. However, this type can still be not adapted. The use of non-normative data type with floating-point representation allows to obtain the benefit from this smaller data-type while reducing even more the cost. To avoid oversizing while maintaining good accuracy, we need a flow that explores the wordlength configuration to find the optimized cost reduction trade-off.

This chapter presents a general optimization flow for custom floating-point arithmetic. This flow is inspired of flow for fixed-point arithmetic. First, section 4.2 describes the custom floating-point representation, existing frameworks for optimization, the constraints

for emulation and the correlation between the exponent and the mantissa. Section 4.3 describes the flowchart and proposed strategies to optimize the variables. Finally, section 4.4 presents the results obtained from different applications.

This contribution has been published first at DASIP 2024 and then extended and submitted in Journal of System Architecture.

4.2 Custom floating-point

4.2.1 Custom floating-point description

Custom floating-point is a representation method based on the IEEE 754 normalized floating-point representation [08]. It aims to reduce the number of bits used to represent a number while preserving the benefit of floating-point representations. This benefit represents an important range of numbers that are close to and far from 0.

Similar to normalized floating-point, custom-floating point numbers are composed of two main parts, an exponent and a mantissa. An implicit part is also present which is the bias on the exponent. Each of these parts can be modified to fit the range to represent and the accuracy of the representation. The exponent size, W_E, and the mantissa size, W_M, are the two parts that can be reduced in comparison to the IEEE754 norm. The application cost and speed can be reduced by these number of bits reductions. The bias can also be modified to fit with the defined exponent size. The modified exponent, $\tilde{E}$, mantissa, $\tilde{M}$, and bias, $\tilde{B}$, allow representing a number, X, according to a formula similar to the IEEE754 norm:

$$X = (-1)^S(1 + \tilde{M}) \times 2^{\tilde{E}-\tilde{B}} \tag{4.1}$$

Modifying the bias and the exponent size allows modifying the range R as follows:

$$R = [2^{-(2^{w_E}-B)}; 2^{2^{w_E}-B}] \tag{4.2}$$

This method of representation actually leads to a lot of constraints that will be explored in the next sections.

4.2.2 Existing frameworks for wordlength optimization

Different frameworks have been proposed in the literature to optimize custom floating-point according to a maximal degradation. In this section, we present three frameworks: Bitsize, Minibit+, and TrueFloat.

Bitsize [GML04]

The Bitsize framework aims at optimizing the wordlength for fixed-point and floating-point data types. This framework uses an analytical approach for quantization error evaluation. The error model is efficient in terms of error evaluation time but limited to smooth operations. An operation is considered as smooth if the output is a continuous and differentiable function of the inputs. Furthermore, this method unifies the wordlength for all data. This unification leads to a less optimized solution for cost and accuracy.

Minibit+ [Osb+07]

Minibit+ is an increment of the Minibit approach [Lee+05] dedicated to fixed-point optimization. In this extension, the transformation focuses on non-uniform optimization and floating-point optimization. The flow uses as input a C++ code. It starts with a range analysis using affine arithmetic. It then performs a coarse-grained accuracy analysis followed by a fine-grained accuracy analysis to determine the data wordlength. The main limitation is its lack of support for control structures such as conditional structures and loops.

TrueFloat [FCF23]

TrueFloat is a framework for the exploration of custom floating-point data types. It is integrated within the open-source HLS tool Bambu [Fa21] to optimize the data types inside the HLS flow. This framework starts with a transformation of C/C++ variables and operations to a customized counterpart. An optimization is done following a dynamic range analysis of the variable to optimize. The output of the entire process is a fully synthesized VHDL/Verilog representation.

In the Bitsize, Minibit+ and TrueFloat frameworks, the strategies to optimize the wordlength of both the exponent and the mantissa are not detailed. Furthermore, the

user can not use a quality degradation of the output as a metric for the implementation cost optimization.

4.2.3 Custom floating-point emulation

As far as we know, there are not many tools for simulating custom floating-point representations in the literature. Existing tools such as flopoco [DK24],ct_float [Bar17] or ac_float [SIE22] can only represent custom floating-point with fixed bias. To explore methodologies that take into account smaller exponents, mantissas, and custom biases, we developed a C++ class to simulate their impact on data.

This class is defined using an exponent size w_E, a mantissa size w_M, and a custom bias B. Shrinkage due to quantification is applied to data X_0 when it needs to be stored. To determine this shrinking, we first determine if the absolute value of the number X to represent falls within the range that can be represented with the exponent size and custom bias defined. If $|X|$ is not in the range R defined in equation 4.2, the value stored will be either $-\infty$ or $+\infty$, depending on the sign of X_0. If $|X|$ is below the range, the value is stored using the subnormal representation with the mantissa. Finally, if $|X|$ is within the range, a treatment is performed on the mantissa part. To replicate the impact of reducing the mantissa size, we set every bit in positions superior to w_M to 0.

This method aims to provide a representation with custom bias, but it comes with certain drawbacks. Firstly, it requires considerable calculations for each operation, leading to an increase in simulation time for every application. Secondly, it leads to inconsistency between operations, making it challenging to determine the new bias of results when adding numbers with different custom biases. For example, the range of an addition result's isn't always linearly linked to the numbers used for the addition. Finally, the inclusion of a custom bias in the inputs of an optimization process adds an extra dimension to the inputs, potentially resulting in a more approximate quality/cost trade-off at the end of the optimization process while requiring more time for optimization.

For these reasons, the following analysis has been conducted considering only a custom exponent and mantissa. The simulations were performed using ac_float [SIE22].

4.2.4 Exponent and Mantissa correlation

A second constraint specific to custom floating-point is that modifying the exponent and mantissa sizes have a correlated impact. Specifically, changing the exponent size will

affect the range for the maximum and minimum representable values, while altering the mantissa size will impact the absolute error proportionally to the exponent. The absolute error between a number X in its floating-point representation and its custom floating-point representation $\hat{X}$ with a mantissa of size w_M can be determined using the formula :

$$|X - \hat{X}| = 2^{\lfloor \log_2(X) \rfloor} \sum_{k=w_M+1}^{+\infty} a_k 2^{-k} \tag{4.3}$$

With a_k denoting the sequence of numbers indicating whether the $-k$ power of two is necessary for representing the number.

The relative error can be determined using the formula:

$$|\frac{X - \hat{X}}{X}| = \frac{\sum_{k=w_M+1}^{+\infty} a_k 2^{-k}}{\sum_{k=1}^{+\infty} a_k 2^{-k}} \tag{4.4}$$

When differentiating both formulas with respect to the mantissa size w_M, the following results are obtained:

$$\frac{d|X - \hat{X}|}{dw_M} = 2^{\lfloor \log_2(X) \rfloor} \sum_{k=w_M+1}^{+\infty} -k a_k 2^{-(k-1)} \leq 0 \tag{4.5}$$

$$\frac{d\frac{|X-\hat{X}|}{X}}{dw_M} = \frac{\sum_{k=w_M+1}^{+\infty} -k a_k 2^{-(k-1)}}{\sum_{k=1}^{+\infty} a_k 2^{-k}} \leq 0 \tag{4.6}$$

These formulas illustrate that increasing the mantissa size reduces the absolute and relative error as can be seen in Figures 4.2 and 4.4. These figures show how the absolute and relative errors change according to the value being represented and the used format. By extension, decreasing the mantissa size increases the absolute and relative error. This increase will also be proportional to the value being represented for the absolute error (figure 4.1 and 4.2). Increasing the exponent size allows for representing numbers with higher errors linked to the mantissa size (figure 4.1 and 4.3). Therefore, both values are correlated, and optimization for custom floating-point requires taking both of them into account.

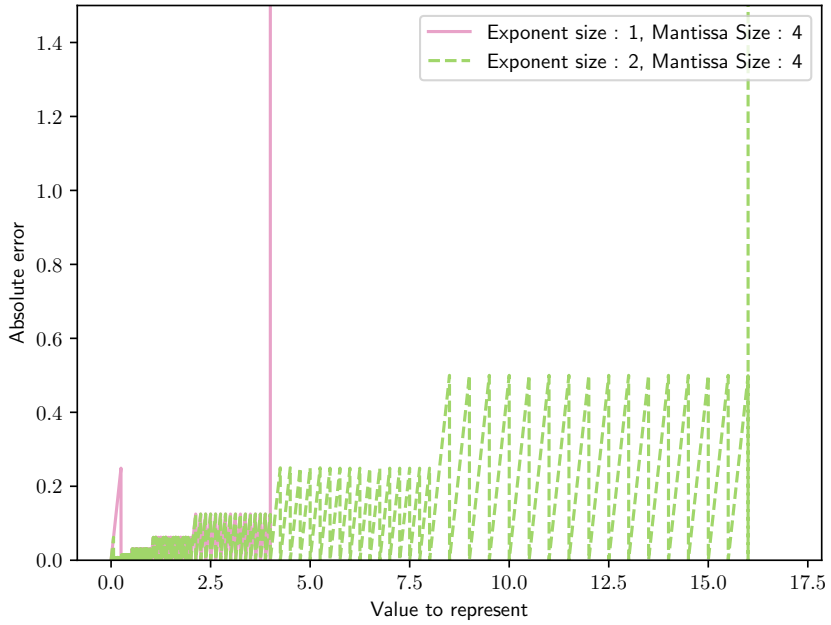

Figure 4.1 – Absolute error of a value according to the value represented and the exponent size using truncation

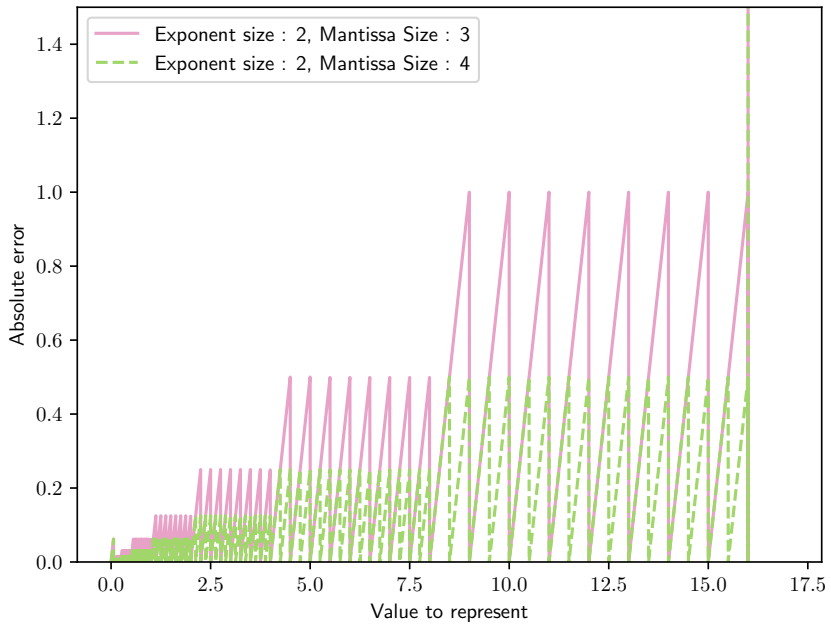

Figure 4.2 – Absolute error of a value according to the value represented and the mantissa size using truncation

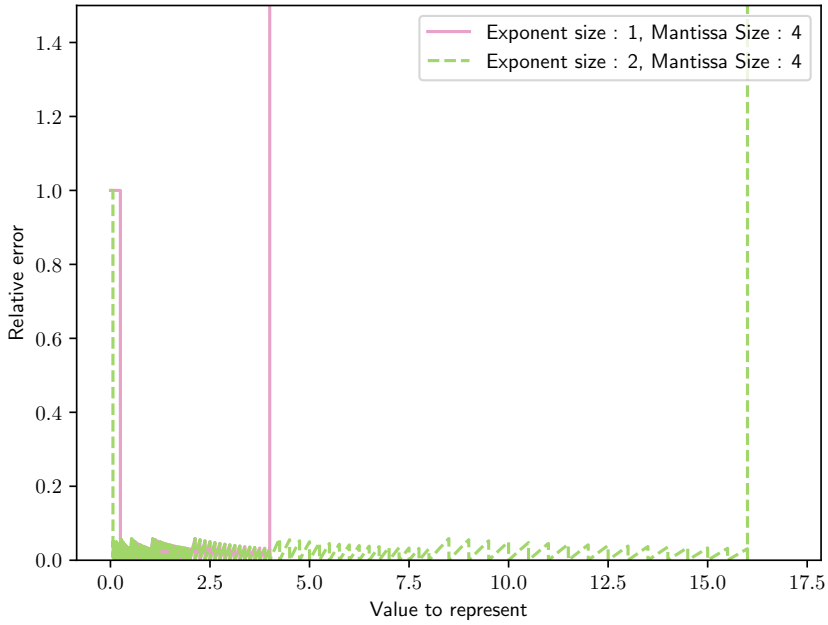

Figure 4.3 – Relative error of a value according to the value represented and the exponent size using truncation

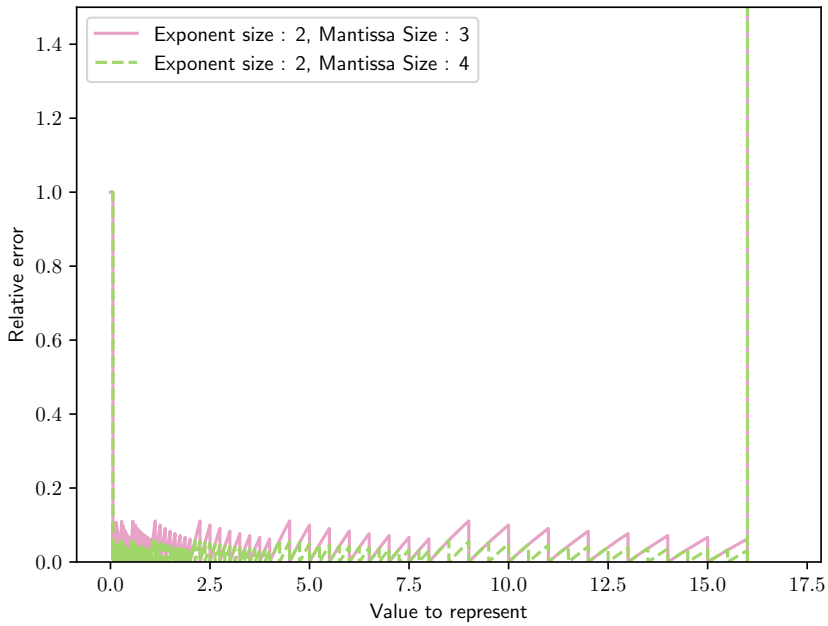

Figure 4.4 – Relative error of a value according to the value represented and the mantissa size using truncation

4.2.5 Range Analysis

To adjust the exponent size of a floating-point format for a variable and to ensure no overflow, the maximal and minimal values of a variable must be known. For that reason, an range analysis of the variable is required. In this thesis, the range R is estimated using a dynamic analysis carried-out by the WeDoLow tools [WeD].

This method is commonly used for fixed-point refinement to determine the size of the integer part. The integer part size W_I is set to represent in binary the maximal value reachable by a variable X. The integer part size can be determined using the following formula

$$W_I = \lfloor \log_2(\max(|X|)) \rfloor + 1 \tag{4.7}$$

The determined integer part is then used to optimize the fractional part. Adapting this method for custom-floating point will allow to reduce the number of relevant wordlength configurations.

In custom floating-point, the exponent needs to represent the range for number under and above 1. However, the range that can be represented depends if the bias can be modified or not. If the bias can be modified, the exponent need to represent the distance between the middle of the $\log_2$ of the range. The corresponding exponent size w_{E_b} can be determined using the formula :

$$w_{E_b} = \left\lfloor \log_2\left(\left| \frac{\log_2(\max(|X^*|)) + \log_2(\min(|X^*|)}{2} \right| \right) \right\rfloor + 1 \tag{4.8}$$

with $X^* := X \backslash 0$, the possible values that can be represented by the variable without 0.

If the bias can not be modified, the exponent need to reach every power of two represented by the variable. The corresponding exponent size w_E can be determined using the formula :

$$w_E = \lfloor \log_2(\max(|\log_2(\max(|X^*|))|, |\log_2(\min(|X^*|))|)) \rfloor + 1 \tag{4.9}$$

The different range represented with w_E and w_{E_b} are illustrated with an example in Figure 4.5. In this example, a variable from a machine learning application is considered, and the distribution of the $\log_2$ of the value is depicted. The top arrow illustrates the range that would be reached using the standard IEEE 754 half type ($w_E = 5$). This range is widely oversized for the values to represent in this use-case. The second arrow illustrates

the range of a custom floating-point that represents all the possible values without a custom bias ($w_E = 3$). This arrow oversize less than the IEEE 754 half type, however, we can observe than an oversize on the exponent above 5. Finally, the bottom arrow illustrates the range of a custom floating-point that represents all the possible values with a custom bias set in the middle of the possible values ($w_{E_b} = 2$). This method allows to fit completely without oversize on the range of values to represent.

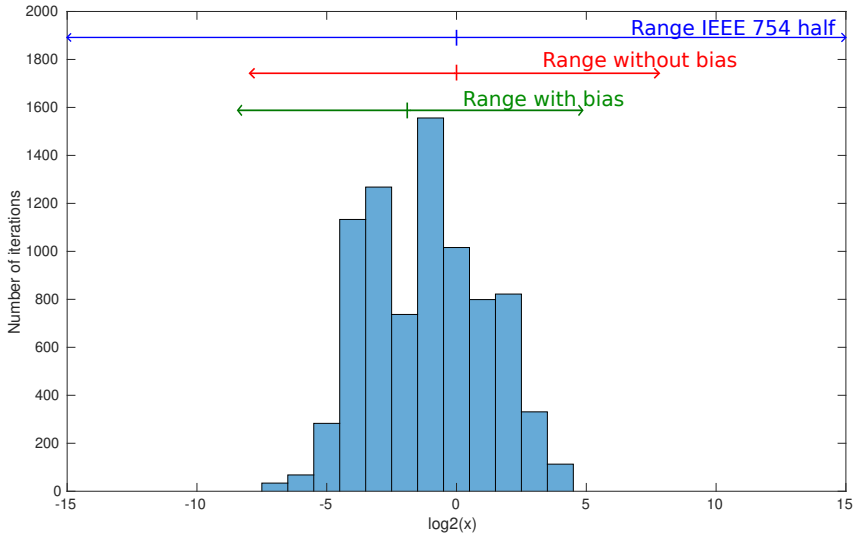

Figure 4.5 – Representation of the range to represent with different exponent for the range analysis of a variable

4.3 Proposed Method

4.3.1 Custom floating-point flowchart

The proposed framework aims at converting automatically a C or C++ code with standard floating-point data types (float or double) to a C++ source code with custom floating-point data types for which the data wordlength has been optimized according to an application quality criterion. This output source code can then be used at the input of high-level synthesis tools such as Vitis or Catapult for hardware implementation.

This framework is based on tools described in section 3.2 of chapter 3 and is described in Figure 4.6. The contributions were done for the exponent wordlength and in the optimization process. The exponent wordlength determination were adapted according to the process described in section 4.2.5. The optimization process aims at determining the

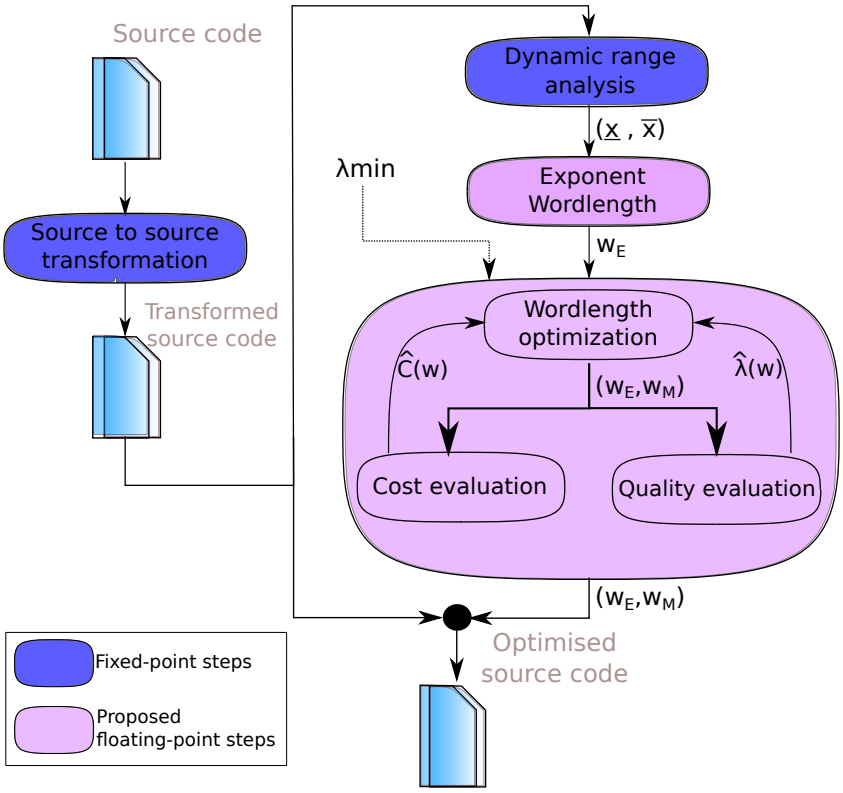

Figure 4.6 – Proposed framework for custom floating refinement.

wordlengths that minimize the cost satisfying a user-defined quality constraint λ_{min}.

$$\min_{\mathbf{w}}(\hat{C}(\mathbf{w})) \quad \text{subject to} \quad \hat{\lambda}(\mathbf{w}) \geq \lambda_{min} \tag{4.10}$$

with $\hat{C}$ the implementation cost estimation for a given vector of wordlength $\mathbf{w}$. This vector of wordlength is the concatenation of the vector of exponent values $\mathbf{w_E}$ and the vector of mantissa values $\mathbf{w_M}$. This entire optimization process is dependent on two metrics: the cost evaluation and the quality evaluation.

Cost Evaluation

Different metrics like architecture area, energy consumption or memory space can be considered for the cost C. For this study, the focus is on energy consumption E. The energy consumption of the complete system is supposed to be additive. Thus, the global energy consumption is a weighted sum of the energy consumed by each operation. The dynamic energy of an operation depends on its type (addition, multiplication, memory

transfer, etc.) and the input and output wordlengths.

The cost $E(\mathbf{w})$ is then determined thanks to the equation 4.11.

$$E(\mathbf{w}) = \sum_{op \in \mathbb{O}} (E_{op}(w^{IN1}, w^{IN2}, w^{OUT}) + E_{mem}(op)) \times N_{op} \tag{4.11}$$

with $\mathbb{O}$ all the operations executed in the complete system and N_{op} the number of times the operation op is executed with w^{IN1} and w^{IN2} the inputs wordlength, w^{OUT} the output wordlength and $E_{mem}(op)$ the operation memory cost.

For a given technology, (ASIC, FPGA, etc.), a characterization process is required in which the operations costs are evaluated for different combinations of exponents and mantissa wordlengths. For this study, a library [al17] for 28nm FDSOI technology is used. This library provides the implementation cost for arithmetic operations, but not for memory transfer. The energy consumption of the complete system is supposed to be additive.

In this thesis, to estimate the memory transfer energy, the HLS tools assumptions are considered: 1) Inputs for the main function are stored in DRAM; 2) Tables used inside the function are stored in SRAM; 3) Other variables are stored in registers. To determine these categories, thanks to dedicated types, the number of allocations of a variable is determined. To estimate the energy transfer with this assumption, a coefficient α representing the relation between an operation and the memory transfer as shown in paper [Jou+21] is used. This coefficient α is then extrapolated to the considered for 28nm FDSOI technology library. The memory cost $E_{mem}(op)$ for an operation op is then determined as proportional to the operation op realizes with the maximal wordlength of all the data used in the operation with the following expression

$$E_{mem}(op) = \alpha E_{op}(w^{+}, w^{+}, w^{+})) \tag{4.12}$$

with $w^{+} = \max(w_e^{IN1} + w_m^{IN1}, w_e^{IN2} + w_m^{IN2}, w_e^{OUT} + w_m^{OUT})$.

The cost $E(\mathbf{w})$ is then determined thanks to the following expression

$$E(\mathbf{w}) = \sum_{op \in \mathbb{O}} (E_{op}(w^{IN1}, w^{IN2}, w^{OUT}) + \alpha E_{op}(w^{+}, w^{+}, w^{+})) \times N_{op} \tag{4.13}$$

This model is used in this thesis to abstract the architecture. The flow can be adapted to any architecture.

Quality Evaluation

As mentioned in Section 3.2.2 of chapter 3, Simulation-based and analytical approaches can be considered to evaluate the quality metric λ at the output of an application according to the data wordlength $\mathbf{w}$. In this thesis, simulation-based techniques are used. The quality metric is statistically evaluated from data collected with a Monte-Carlo simulation carried out on the input data. For the simulation, the source code is modified to integrate the custom floating-point data types. The `ac_type` datatypes [SIE22] have been used.

The combination of the numerous iterations of the optimization process and the high quality evaluation time leads to a huge global optimization time. As described in Chapter 1, the time of optimization depends on three parameters: the number of configurations N_C, the number of input samples N_S and the execution time for one sample t_s. In order to minimize optimization time without compromising the quality of the results, it is important to develop specific strategies for floating-point optimization to limit the number of configurations N_C tested during the optimization process. These strategies should focus on the structure of floating-point arithmetic, as well as the impact of the exponent and the mantissa on the quality of the system.

Optimization implementation

With the cost and quality evaluation processes described above, the optimization process can be implemented. The mantissa and exponent wordlength must be optimized for each variable. Thus, this optimization is done in a subspace of $\mathbb{N}^{2dN_v}$ with N_v the number of variables to optimize. This subspace is limited in its dimensions by the wordlength of the type used in the source code. Since the optimization is done in a highly dimensional space of natural numbers, it requires the use of heuristic methods to obtain a result in a reasonable amount of time. In this thesis, an adapted version of a meta-heuristic search methods employing local search is proposed. Theses methods are MIN+1 and TABU described in section 3.3.1.

These optimization processes raise the question of parallelization. Each part of the optimization can be parrallelized. First of all, the quality evaluation using a simulation can be parallelized. For a same input, the reference model and the tested model can run simultaneously. The MIN+1 algorithm can also be parallelized. The gradient descent used in the first step of the algorithm is done on all the dimensions of the variable. Each of these gradient descents are independent of each other. On a pool of P threads, each thread is in

charge of a full gradient descent process on the variable. As soon as one is over, another gradient descent process is launched. This is represented in figure 4.7.a.

The TABU algoritm and the mildest-ascent greedy algorithm used in the second step of the MIN+1 algorithm requires a different kind of parallelization. Indeed, to select the direction to follow, the impact of each modification on each of the variables need to be known. At each step, the impact of a modification on each dimension is estimated in parallel. All the data are then merged to determine the selected configuration for the next step. The process is summarized in figure 4.7.b.

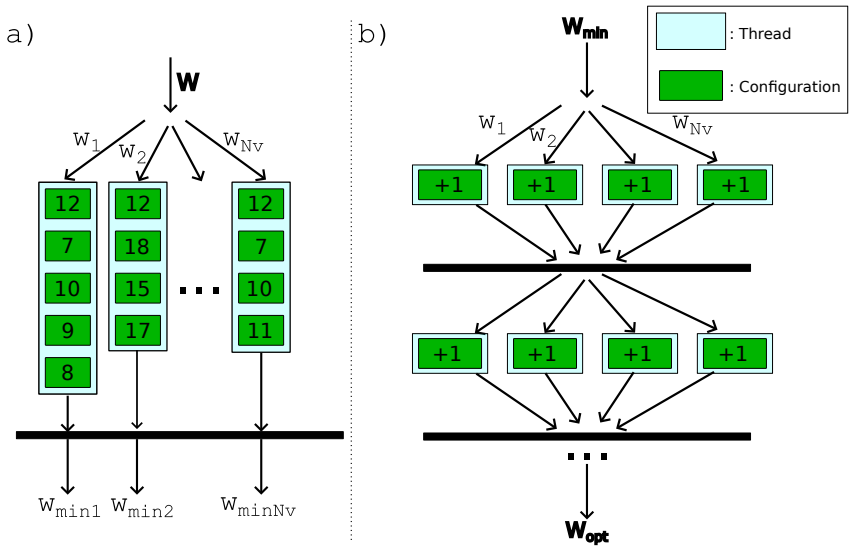

Figure 4.7 – Diagram of the parralelization process. a) For the gradient descent in the MIN+1. b) For the mildest-ascent greedy of the MIN+1 and the TABU.

The execution time for each task can not be estimated beforehand. The order of execution is then selected by the order in the input vector. The speedup with parallel processing is dependent on the number of threads and the input vector size. Figure 4.8 shows the evolution of speedup according to the number of threads for optimizing an IIR application. A complete optimization was done with an input vector size of 7. We can observe that this implementation leads to a maximal speedup of 4 for this use-case. This speedup is linear until reaching its maximal value for a number of threads matching with the input vector size. This is explained by the fact that the number of possible tasks are reached. Any added threads will be empty, which leads to no speedup increase.

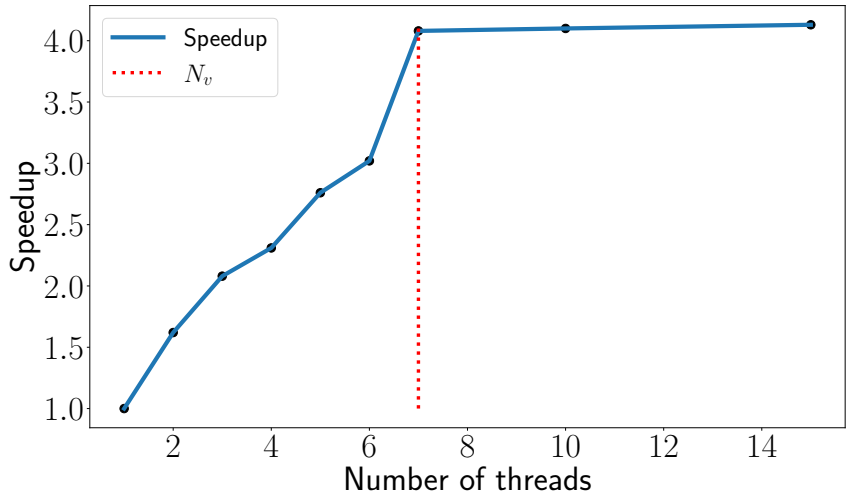

Figure 4.8 – Evolution of the speedup according to the number of threads for a complete optimization process for an application with $N_v = 7$.

4.3.2 Global Wordlength Optimization strategies

In this section, the proposed strategies to optimize the wordlength of both the exponent and mantissa are detailed. As shown in Equation 1.2, the optimization time is proportional to the number of tested configurations. This latter depends on the optimization search space. The search space is linked to three parameters:

— N_v Number of variables to optimize

— $\mathbb{E}_r \doteq \{\mathbf{x} \in \mathbb{N}^{N_v} : 1 \leq x_i \leq w_{E_i}\}$ Reduced search space for the exponent wordlength

— $\mathbb{M} \doteq \{\mathbf{x} \in \mathbb{N}^{N_v} : 1 \leq x_i \leq W_m\}$ Search space for the mantissa wordlength

with W_m being the mantissa wordlength of the reference type (24 for single precision, 53 for double precision) and w_{E_i} the maximal number of bits required for exponent of the ith variable according to the range analysis described in Section 4.2.5. In this contribution, we present four strategies that take advantage of the properties of the search space.

Exponent and mantissa wordlength simultaneous optimization

This first approach optimizes the wordlengths of the exponent and mantissa simultaneously. The maximal exponent wordlength is defined in vector $\mathbf{w}_E^{\max}$ computed with Equation 4.9 with the help of the range analysis step.

The variable for the optimization process is a $2.N_v$-length vector $\mathbf{w}$ composed of the exponent wordlength w_{E_i} and the mantissa wordlength w_{M_i} for each variable v_i of the set

$\mathbb{V}$.

The domain to explore is then $\mathbb{E}_r \times \mathbb{M}$. The number of elements n_p of the search space can be expressed with the following expression

$$n_p = W_M^{N_v} \times \prod_{i=0}^{N_v} w_{E_i}^{\max} \tag{4.14}$$

By considering the exponent wordlength as a variable to optimize this approach provides more degrees of freedom to find an optimized solution.

Indeed, values with high or low exponent can have an important impact on the estimation of the reduced exponent wordlength w_{E_i} without an important statistical impact on the quality. w_{E_i} is then a good starting point for the optimization process, but not the best solution. The exponent part needs to be optimized to determine its adequate value. Furthermore, for small exponent values, the unnormalized representation can help to represent a part of the smallest values. To optimize this unnormalized representation, both the mantissa and exponent need to be taken into account.

The use of the reduced space $\mathbb{E}_r$, instead of a space composed of the maximal exponent wordlength like for the mantissa, allows reducing the optimization time. Indeed, the values higher than $w_{E_i}^{\max}$ will lead to the same quality results for a superior cost and are irrelevant.

Nevertheless, this approach leads to the largest number of possibilities and thus tends to the highest optimization time of the four proposed methods.

Mantissa wordlength only optimization

This second approach aims at optimizing only the mantissa wordlength. The exponent wordlength is set to the value obtained with Equation 4.9 ($\mathbf{w}_e = \mathbf{w}_e^{\max}$). The variable for the optimization process is a N_v-length vector $\mathbf{w}$ composed only of the mantissa wordlength w_{M_i} for each variable v_i of the set $\mathbb{V}$.

This a priori information allows limiting the search space to the set $\mathbb{M}$. Compared to the first approach, the number of variables in the optimization process has been divided by two. Thus, the number of tested configurations during the optimization process and the optimization time are reduced. The number of elements n_p of the search space can be expressed with the following expression

$$n_p = W_M^{N_v} \tag{4.15}$$

Nevertheless, this approach can lead to a solution less optimized compared to the one obtained with the first approach. This approach does not explore the potential benefit of reducing slightly the exponent wordlength in relation to the value obtained with Equation 4.9.

Exponent and mantissa wordlength sequential optimization

This third approach is a middle ground between the two others. The optimization problem is divided into two optimization sequential processes. First the exponent wordlengths are optimized and then the mantissa. The variable for the first optimization process is a N_v-length vector $\mathbf{w}$ composed of the exponent wordlength w_{E_i} for each variable v_i of the set $\mathbb{V}$. The variable for the second optimization process is a N_v-length vector $\mathbf{w}$ composed of the mantissa wordlength w_{M_i} for each variable v_i of the set $\mathbb{V}$.

The optimization is performed on a subset of the search space $\mathbb{E}_r \times \mathbb{M}$. The number of elements n_p of the search space can be expressed with the following expression

$$n_p = W_M^{N_v} + \prod_{i=0}^{N_v} w_{E_i}^{\max} \tag{4.16}$$

Since this approach is a middle ground between the two other methods in terms of exploration space, the number of configurations required tends to be a middle ground. However, optimizing with a sequential optimization algorithm such as MIN+1 can lead to the exploration of more configurations especially for high λ_{min}.

This solution brings a new hyperparameter to the optimization process : $\lambda_{E_{min}}$. The quality degradation due to finite precision must be budgeted between the two independent optimization processes. In this work, the goal $\lambda_{E_{min}}$ for the first optimization process was set to the quality of the original system.

Separate input between MIN+1 and TABU

This last approach aims at reducing the optimization time by considering separate inputs for the MIN+1 and TABU algorithms. In this strategy, the MIN+1 algorithm is executed with the algorithm described in the *Mantissa wordlength only optimization* section considering only the mantissa part of each variable being analyzed. The exponent wordlength is set to the value obtained with Equation 4.9 after the dynamic range analysis ($\mathbf{w}_e = \mathbf{w}_e^{\max}$) . The variable for the optimization process is a N_v-length vector $\mathbf{w}$ composed

only of the mantissa wordlength w_{M_i} for each variable v_i of the set $\mathbb{V}$. This selection then reduces the number of configurations required in this first step.

However, to obtain a better solution than the one obtained with the second approach, the exponent part of the variables is considered for the input of the TABU search. They are conserved as the value obtained with Equation 4.9($\mathbf{w}_e = \mathbf{w}_e^{\max}$). The variable for the optimization process is then a $2.N_v$-length vector $\mathbf{w}$ composed of the exponent wordlength w_{E_i} and the mantissa wordlength $\hat{w}_{M_i}$ obtained after MIN+1 algorithm for each variable v_i of the set $\mathbb{V}$.

The optimization is first performed on a subset of the search space $\mathbb{M}$ then on a subset of the search space $\mathbb{E}_r \times \mathbb{M}$. The number of elements n_p of the search space can be expressed with the following expression

$$n_p^{MIN+1} = M^{N_v}, \quad n_p^{TABU} = M^{N_v} \times \prod_{i=0}^{N_v} w_{E_i}^{\max} \tag{4.17}$$

This method is a middle ground between the first two strategies. The exponent part of any variable having an impact on the range, any small modification could lead to an important quality degradation. This important degradation for a small modification makes them a priori exploration candidate for the TABU search. Furthermore, they can be quickly added to the TABU list reducing the number of configurations required.

However, the result obtained at the end of the MIN+1 part with only the mantissa could lead to a starting point located in an inadequate part of the exploration space. Combined with the reduced configurations explored with the TABU search, this could lead to a worse configuration than strategies 1 and 3.

A summary of the strategies and the part of the variables (exponent, mantissa) involved in the optimization process is described in the Table 4.1.

Table 4.1 – Part of the variables (exponent, mantissa) involved in the optimization process according to the strategy. A ✓ symbol specify that this part is involved in the optimization process. A $\checkmark^n$ show an order in the optimization

Strategy :	1		2		3		4	
Optimization part:	Min+1	Tabu	Min+1	Tabu	Min+1	Tabu	Min+1	Tabu
Exponent	✓	✓			$\checkmark^1$	$\checkmark^1$		✓
Mantissa	✓	✓	✓	✓	$\checkmark^2$	$\checkmark^2$	✓	✓

4.4 Experimental results

To determine the efficiency of the given optimization and the different strategies, the full optimisation process was carried out on three different applications.

4.4.1 IIR filter

The first application is a Linear Time-Invariant (LTI) system corresponding to an Infinite Impulse Response (IIR) filter. This band-pass filter is composed of four cascaded biquad cells. To estimate the quality metric λ, a chirp is used as input signal for both the reference model with single-precision floating-point data types (FP32) and the tested model. The quality metric λ is defined as the Signal to Quantization Noise Ratio between the output signals of these two models for this chirp. As an LTI application including only linear operations, wordlength modification has a linear impact on the quality metric. This application leads to a relatively small number of variables to optimize : $N_v = 7$.

4.4.2 Kalman filter

The second application is a Kalman filter. This Kalman filter has 3 inputs to define a measured position and estimate the gravity constant. The quality metric λ is defined as the inverse of the error. The error corresponds to the mean squared error between the real and estimated values of the Kalman filter output. To estimate this quality metric λ, 50 trios of inputs are considered. These 50 trios of inputs aim to represent both common and extreme cases. This application leads to an important number of variables to optimize : $N_v = 21$.

4.4.3 Squeezenet CNN

The third application is a Squeezenet convolutional neural network (CNN). The quality metric λ is defined as

$$\lambda = 1 - p \tag{4.18}$$

where p is the mean error rate on the top-1 class. For each input image, the most probable class (top-1) given as an output for the FP32 model and for the tested model are compared. To estimate its quality λ, a Monte-Carlo simulation is performed on a dataset of 4520 images. This application is the most intricate application of the three

considered. Its structure is composed of multiple unsmooth operators such as maxpool. These unsmooth operators can lead to an important variation on the result for a small wordlength modification. A common wordlength is considered for each layer of the CNN, which leads to a number of variables to optimize $N_v = 15$. In terms of number of variables to optimize, this is in a middle ground between the IIR filter and the Kalman filter

4.4.4 Results

To evaluate the effectiveness of our proposed approaches, we explored multiple optimizations for various applications and strategies. These optimizations lead to different quality-cost tradeoffs at the end of the optimization process.

The efficiency of our approaches is presented in Figures 4.9, 4.10 and 4.11 by providing the couple Quality - Cost associated with the final solution, corresponding to the configuration obtained at the end of the complete optimization process. In this figure, we present the results for different values of λ_{min}, illustrated with dashed lines. The x-axis represents the normalized power consumption according to the golden model. The y-axis represents the quality metric. The Pareto front including all the strategies is drawn with a dotted line. We also provides a reference point as a FP-16 configuration. The FP-16 configuration is a configuration where all the variables wordlength are set according to the IEE754 half precision floating-point data type.

These results illustrate a reduction in terms of cost for the all applications and methods compares to the FP32 model. Furthermore, the results obtained for our application are better than those of the FP16 model. This demonstrates the advantage of custom floating-point types over generic data types.

The Pareto front highlights two dominant strategies out of the four proposed. The strategy 2 and strategy 4 (detailed in section 4.3) composes most of the Pareto front for all the tested applications. These differences in results can be explained by the curse of high dimensionality. These strategies have a smaller number of variables to optimize and so a smaller optimization search space to explore to find the minimal solutions compared to the two other strategies. Since the MIN+1 and TABU optimization processes are heuristic, they cannot explore all the configurations. The quality of the solution degrades as the design space increases in dimension.The higher dimensional spaces to explore in strategy 1 and 3 makes it harder to find optimized configurations. The heuristic algorithms are then less efficient with too many configurations to test.

In simple cases to optimize, such as the IIR and the Kalman filter, the strategy 2 and

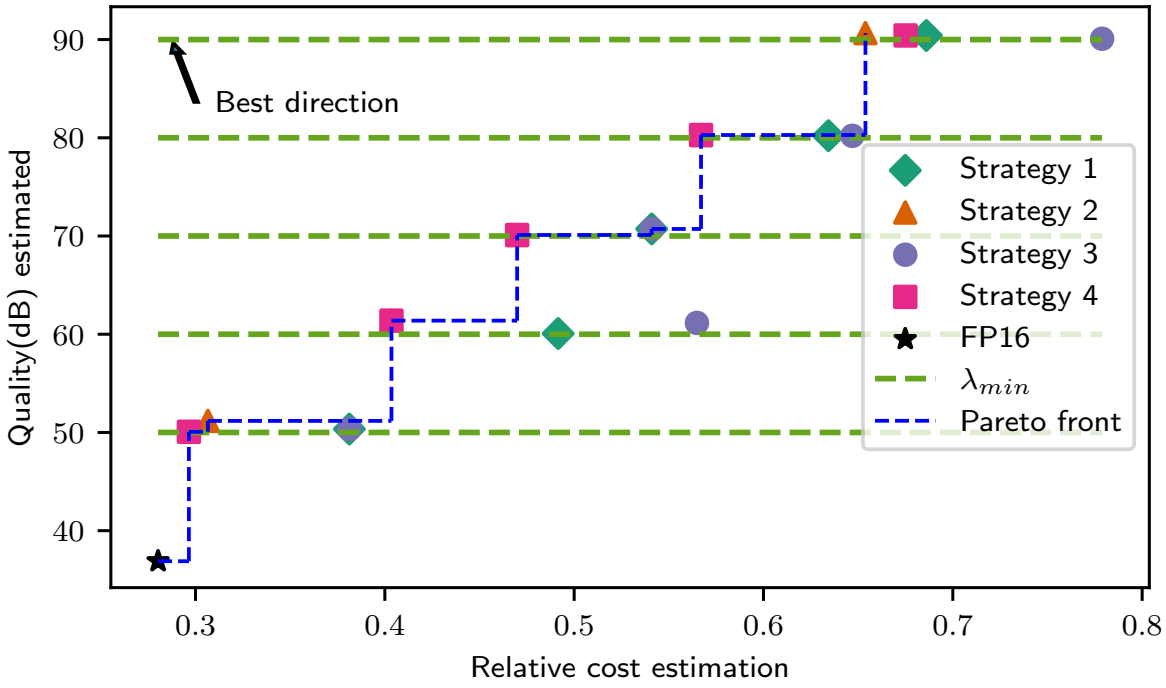

Figure 4.9 – Pareto chart of normalized energy consumption according to the golden model and final accuracy obtained according to the strategy and the minimal quality λ_{min} for IIR filter application.

4 give similar final trade-offs. However, for a complex application such as Squeezenet, a difference between strategies 2 and 4 is highlighted. For a similar quality constraint λ_{min}, strategy 4 usually achieves a better trade-off than strategy 2.

An important element between the four strategies is the number of configurations tested for the complete optimization process. The Figures 4.12, 4.13 and 4.14 present the number of configurations tested during the optimization process according to quality constraint λ_{min} and the strategy. These figures illustrate an important difference between strategies 2 and 4 in terms of the number of tested configurations compared to strategies 1 and 3. Strategy 2 required fewer configurations to find the final solution than strategy 4. This difference is explained by the number of dimensions added during the TABU optimization process. Under certain conditions, Strategy 4 may require fewer configurations than Strategy 2. This difference appears if, early in the TABU optimization process, a variable of the exponent part is modified to lead to an excellent trade-off. This new configuration will lead the exploration process to a space that cannot be reached by strategy

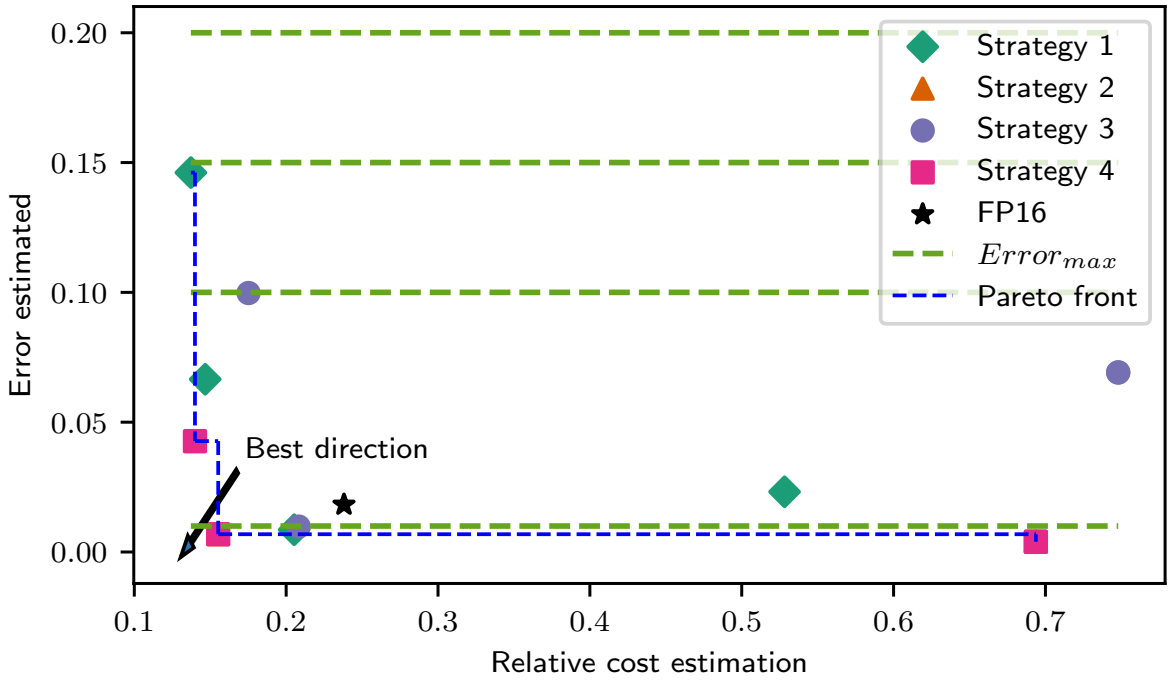

Figure 4.10 – Pareto chart of normalized energy consumption according to the golden model and final accuracy obtained according to the strategy and the minimal quality λ_{min} for Kalman filter application.

2. However, these cases are not predictable and Strategy 2 is generally faster than the other proposed strategies.

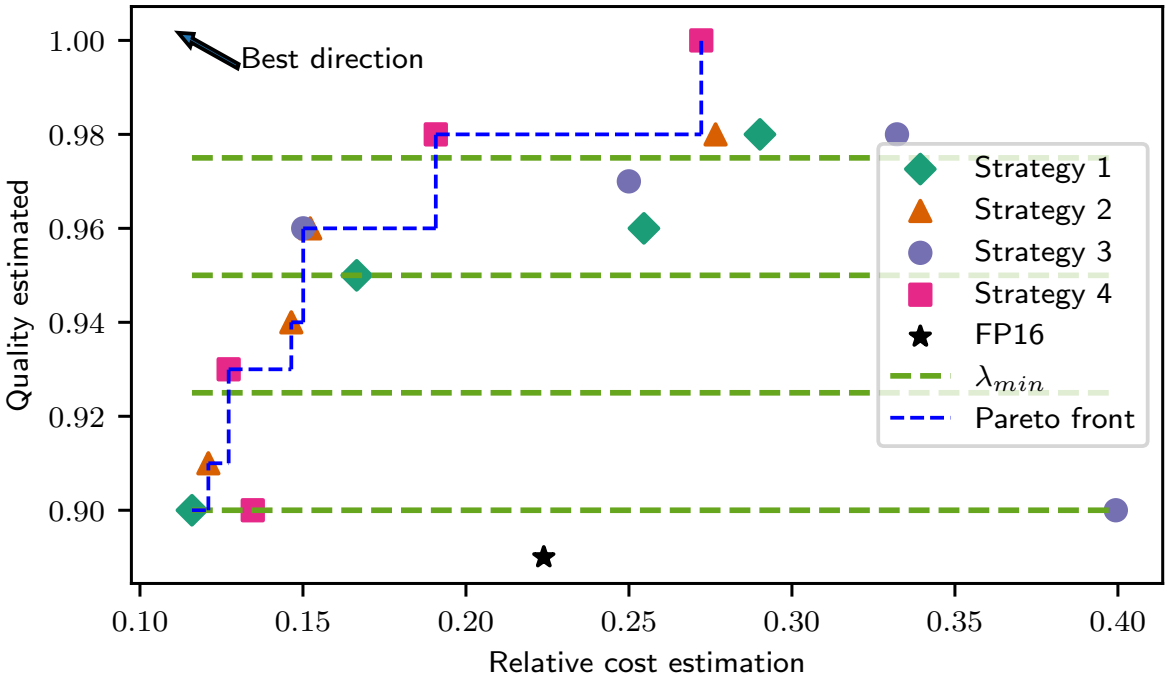

Figure 4.11 – Pareto chart of normalized energy consumption according to the golden model and final accuracy obtained according to the strategy and the minimal quality λ_{min} for Squeezenet application.

4.5 Conclusion

This chapter proposes a new method for custom floating-point refinement for data-oriented applications described with a C or C++ source code. This method takes as inputs a C or C++ source code to output a refined version of this source code with optimized data types. In this method, a dynamic analysis is performed to determine information for the optimization process. This optimization process is then achieved using heuristic algorithms: MIN+1 and TABU. These heuristic algorithms determine the optimized implementation cost for a given minimum quality level for a custom floating-point word length configuration. This allows users to explore multiple cost and quality compromises.

To optimize the exponent and mantissa word length according to a user-defined quality constraint, four strategies were established. Theses strategies are based on data dynamic range analysis to adapt the exponent wordlength.

The results show an improvement in terms of power consumption. Two strategies (strategie 2 and strategie 4) from all the proposed in this chapter obtains better results

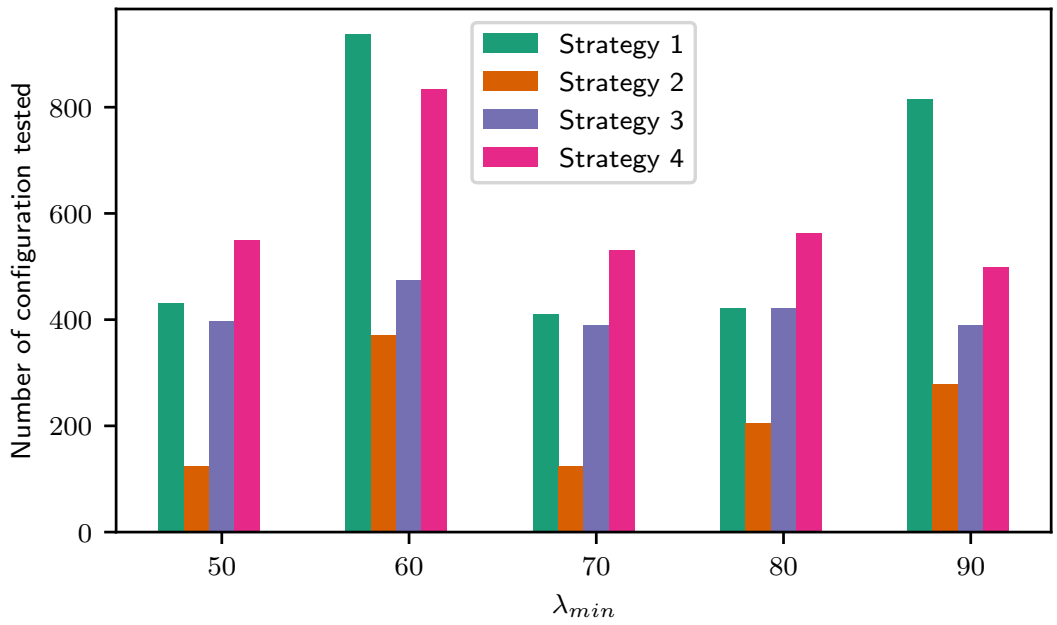

Figure 4.12 – Number of configurations used for the full optimization process according to the strategy and the minimal quality λ_{min} for IIR filter application.

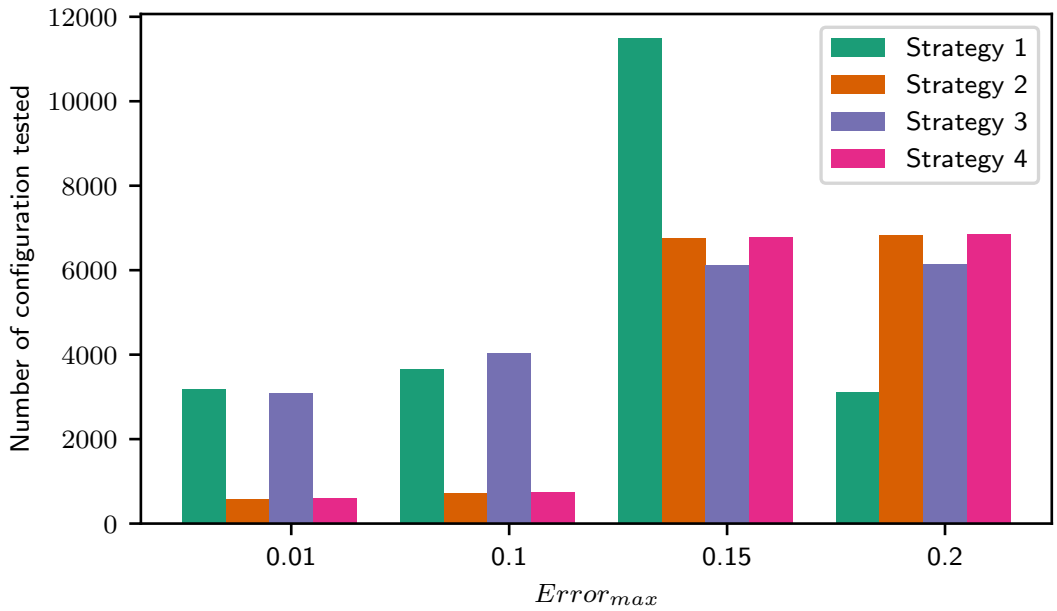

Figure 4.13 – Number of configurations used for the full optimization process according to the strategy and the minimal quality λ_{min} for Kalman filter application.

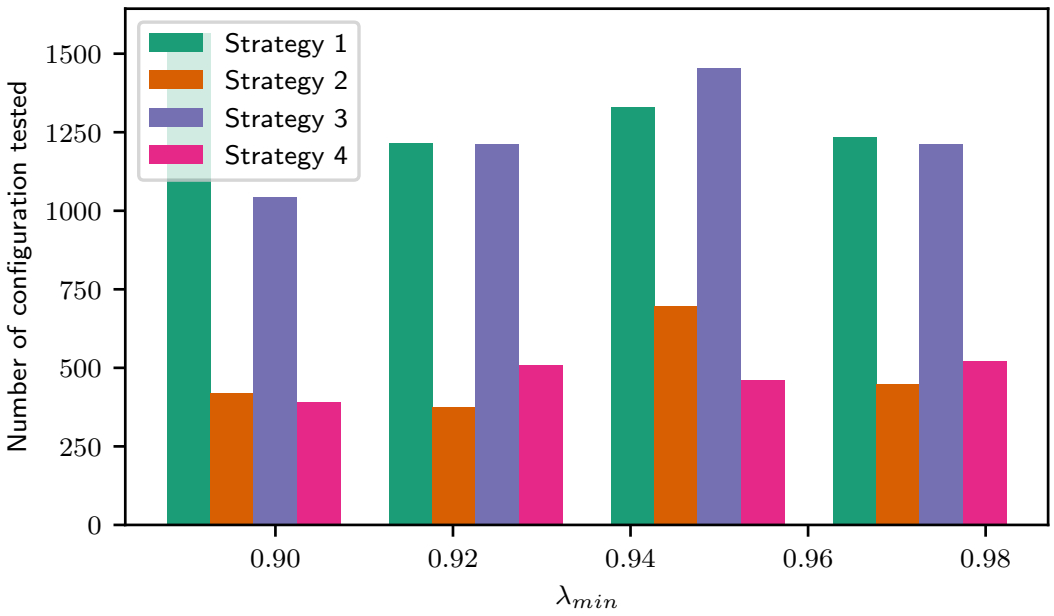

Figure 4.14 – Number of configurations used for the full optimization process according to the strategy and the minimal quality λ_{min} for Squeezenet application.

than the others. The first one sets the exponent wordlength using the dynamic range and explores different mantissa wordlengths with the MIN+1 and TABU algorithms. This strategy obtains correct results with the smaller number of configurations tested. The second one sets the exponent wordlength using the dynamic range and explores different mantissa wordlength with the MIN+1 algorithm. It then explores different mantissa and exponent wordlength with the TABU algorithm. This strategy obtains the best results but require to test a higher number of configurations.

These strategies reduce the number of configurations and the time required for optimizing the quantification of a generic application. In the next chapter, we will explore another method to reduce the optimization time by minimizing the number of input samples used for the test.

Chapter 5

Stratified sampling for precision optimization

5.1 Introduction

As stated in section 1, various parameters influence the execution time of the precision optimization process. To reduce this optimization time, a promising way is to reduce the number of input samples tested during the quality estimation process. The execution time of the quality estimation process is directly proportional to this number of samples.

Two methods exist for setting the number of input samples tested during the quality estimation process. The first one is to define a dataset that is representative of most of the use cases and to test all the inputs. Although this method is consistent, it may require the consideration of a lot of unnecessary data to ensure that as many cases as possible are covered. This method was used for experiments in chapter 4 and is commonly used for applications based on machine learning. The second method involves treating quality as a random variable and using Monte Carlo simulations to evaluate its statistical parameters. These simulations are performed by sampling the input dataset.

This method is based on estimation theory and can be implemented with the algorithm described in Section 3.2.2 of chapter 3. This theory allows defining the minimal number of input samples required to a have a given confidence in the estimation. The minimal number of samples depends on three parameters: the user-defined absolute error δ and the confidence interval ε_{max} and the variance of the outputs σ. Since the parameters δ and ε_{max} are user-defined, to reduce the number of samples required for a confident estimation, σ needs to be reduced.

In this chapter we propose to exploit the Stratified sampling concept to reduce the number of input samples for the quality evaluation process. This sampling method involves grouping inputs with similar behavior to estimate the mean in each group. If done correctly, this grouping reduces the variance and, consequently, the number of required in-

puts. This contribution has been published in Design & Diagnostics of Electronic Circuits & Systems (DDECS) 2025.

First, in Section 5.2, the confidence interval in the context of Monte-Carlo simulations is presented. Secondly, in Section 5.3 the proposed method based on Stratified Sampling is described. Thirdly, in Section 5.4, different metrics are proposed to group inputs into different subspaces required for stratified sampling. Finally, Section 5.5 presents the experimental results.

5.2 Confidence interval with Monte-Carlo simulations

In this chapter, we consider data processing oriented applications for which the quality criteria λ for finite precision refinement is based on the comparison between y_T, the output obtained with the tested model (*i.e. finite precision*), and y_R, the golden reference output as described in section 3.2.2 of chapter 3. As illustrated in Figure 5.1, the error metric computation is set to an equality between y_R and y_T.

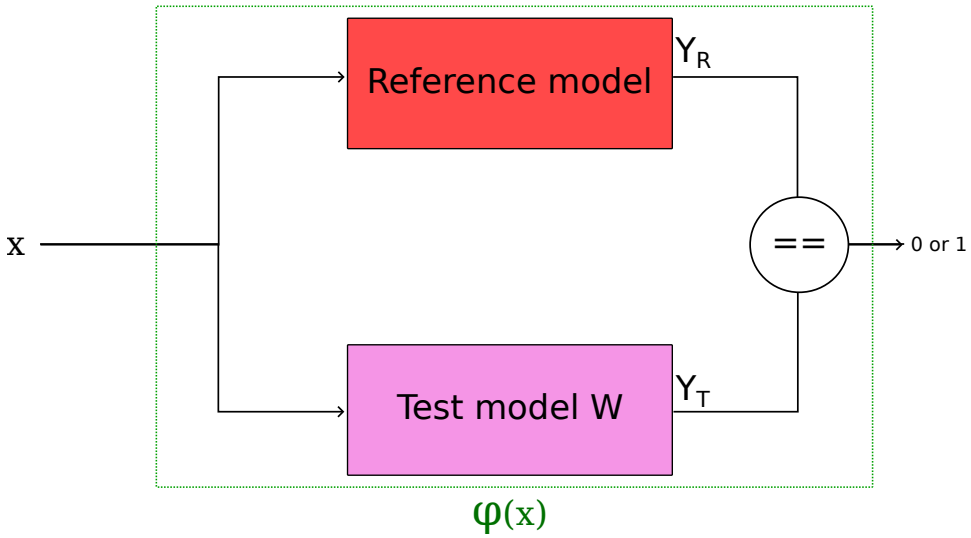

Figure 5.1 – Simulation of an input sample x for a configuration

The result of the comparison of y_T and y_R is a boolean φ with value 0 when y_T and y_R are not identical (failure) or 1 otherwise (success). In the context of DNN used for solving classification tasks, $\varphi = 1$ if the Top-M classes (*i.e* the M output classes with the highest probabilities) obtained with the tested model are identical to the Top-M classes obtained with the golden reference.

Since φ takes only the values 0 and 1 with a known proportion of $\lambda\%$, φ can be approximated as a random variable. This random variable can be represented with a Bernoulli distribution with parameter λ. The quality of a configuration corresponding to the mean of this Bernoulli distribution is the parameter λ [Bru10]. We want to estimate this parameter through an estimator, $\hat{\lambda}$. In this representation, realizing a MCS (equation 5.1) to obtain the value $\hat{\lambda}$ corresponds to using the minimal mean squared error (MMSE) to estimate the mean λ of a random variable φ.

$$\hat{\lambda} = \frac{1}{N} \sum_{k=1}^{N} \varphi(x_k) \tag{5.1}$$

As an estimator, the obtained value $\hat{\lambda}$ depends on the N tested inputs x. The histogram of possible values for $\hat{\lambda}$ can be modeled through a Gaussian distribution using the law of large numbers [Wal13]. The mean of the Gaussian is equal to the real value of λ and its variance depends on the number of inputs used for the estimation [Wal13]. To define a confidence in the estimator, two values are required: the absolute error ε and the confidence interval δ. ε represents the quantile for δ of the distance between the real value λ and the estimated one $\hat{\lambda}$. To find the minimal number of samples required for the MCS, the user can define ε_{max} which sets the maximal ε that the user tolerate.

The term δ represents the probability that the estimation $\hat{\lambda}$ is at a maximal distance ε of the real value λ [GT13]. The expression of the confidence interval δ is given in Equation 5.2 and illustrated in Fig. 5.2.

$$\mathbb{P}(\sqrt{N}|\hat{\lambda} - \lambda| \leq \varepsilon) = \delta \tag{5.2}$$

The minimum number of input samples N_s to obtain a given confidence can be derived using the inverse cumulative distribution function for the normal distribution z with the following expression

$$N_s \geq z(\overline{q})^2 \frac{\sigma^2}{\varepsilon_{max}^2} \tag{5.3}$$

with $\overline{q} = \frac{1+\delta}{2}$ the upper quantile as illustrated in Fig. 5.2.

For a set (δ, ε_{max}), the only parameter remaining is the variance of the simulation result σ. Since $\varphi(x)$ follows a Bernoulli distribution, σ can be estimated through the unbiased estimator $\hat{\sigma}$ presented in Equation 5.4 [Wal13].

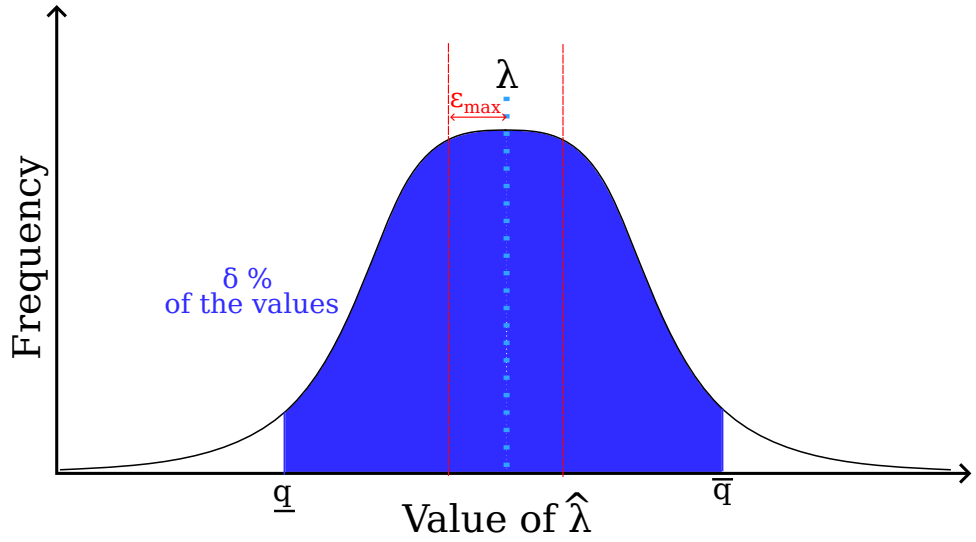

Figure 5.2 – Histogram for the estimator $\hat{\lambda}$.

$$\hat{\sigma} = \hat{\lambda}(1-\hat{\lambda})\frac{N}{N-1} \tag{5.4}$$

The simulation-based estimation requires a large number of simulations to achieve a confident estimate, which can be time-consuming compared to an analytic estimation. However, it does not have constraints on the application or refinement method. Any system that can be represented as shown in Figure 5.1 can be estimated for quality using this method. Developing methods to reduce the parameters δ, ε_{max} and σ will allow for a fast and generic estimation method for precision optimization with a reduced number of samples.

Since the parameters δ and ε_{max} are user-defined, reducing the number of samples required for a confident estimation necessitates to reduce σ. The next section describes a variance reduction method, stratified sampling, to accomplish this.

5.3 Proposed method based on Stratified Sampling

5.3.1 Stratified Sampling concept

Stratified sampling is a variance reduction sampling method. It is based on conditional probability and requires that the input space, Ω, to be stratified into S independent subspaces, Ω_j. For these subspaces, the probability of being selected is known, as described

in [Bru10; SM96]:

$$p_j = \mathbb{P}[\Omega_j] \tag{5.5}$$

Every subdivision meeting this requirement can be used for stratified sampling. However not all subdivision can lead to a variance reduction. Stratified sampling relies on the idea that the inputs belonging to a subdivision share common features and this can lead to a smaller variance.

However, using subdivisions leads to an adjustment of the mean estimator, $\hat{\mu}_{strat}$. This mean estimator is defined by drawing, for each subspace, a number n_j of inputs x_k^j. This new unbiased estimator $\hat{\mu}_{strat}$ can then be described as a weighted sum of the different mean estimators $\hat{\mu}_{n_j}$ of the subspaces, as expressed in equation 5.6 [Bru10].

$$\hat{\mu}_{strat} = \sum_{j=1}^{S} p_j \frac{1}{n_j} \sum_{k=0}^{n_j} \varphi(x_k^j) = \sum_{j=1}^{S} p_j \hat{\mu}_{n_j} \tag{5.6}$$

Using this estimator requires to adapt the variance estimator as follows [Bru10]

$$\hat{\sigma}_{strat} = \sum_{j=1}^{S} \frac{p_j^2}{n_j} \hat{\sigma}_j^2 \tag{5.7}$$

with $\hat{\sigma}_j^2$ the variance as defined in equation 5.4 for inputs x_k^j.

This variance is, in the worst case, equal to the variance of classical Monte-Carlo simulations, otherwise it is smaller. The variance reduction is dependent of n_j the number of samples drawn for a subdivision. An optimal n_j^* can be formulated with the following equation:

$$n_j^* = \frac{p_j \sigma_j}{\sum_{i=1}^{S} p_i \sigma_i} \tag{5.8}$$

However, this optimal n_j^* depends on σ_j which has to be estimated. A good alternative is to draw for each subspace Ω_j, a number of inputs n_j proportional to N, the total number of inputs that we want to draw

$$n_j = p_j.N \tag{5.9}$$

5.3.2 Subspace selection

To adapt the stratified sampling presented in Section 5.3.1 for quality estimation process we propose Algorithm 4 to reduce the minimum number of samples required for estimation. As seen in equation 5.3, the minimal number of samples N_s depends on the variance σ of the drawn inputs and this variance can not be determined in advance. Consequently, an iterative process is used to refine progressively the minimal number of samples.

This algorithm takes as input each subspace d in Ω and its probability of being drawn $p(d)$. It also requires the maximal absolute error ε_{max} and the confidence interval δ defined by the user. Finally, a batch size N_b is defined and the number of tested images will be a multiple of N_b. A small value for N_b will lead to a number of tested images close to N_s. However, N_b needs to be large enough so that each iteration of the process follows the law of large numbers. For that reason, the batch size N_b is set to 50.

This iterative process continues while the estimated absolute error $\hat{\varepsilon}$ exceeds the user-defined maximal absolute error ε_{max} or all inputs have been tested (line 4). For all subspace d, the number of inputs n_d is computed according to equation 5.9 (line 7). From the remaining inputs in a subspace d, n_d inputs are drawn (line 8). These inputs are added to the list of inputs that have been drawn for this subdivision (line 9). If all inputs in a subspace d have been drawn, the list of inputs remains the same for each subsequent iteration. This list of inputs is then used to determine the quality $\hat{\lambda}_d$ and the variance $\hat{\sigma}_d$ for this subdivision according to Equation 5.1 and Equation 5.4 (lines 10,11). Once $\hat{\lambda}_d$ and $\hat{\sigma}_d$ have been estimated for all subspace d, the quality $\tilde{\lambda}$ and the absolute error $\hat{\varepsilon}$ for the configuration is estimated (lines 13,14,15). If the estimated absolute error is lower than the maximal absolute error, the algorithm returns the last quality estimate. Otherwise, it performs another iteration with N_b new inputs.

In figure 5.3 we illustrate this iterative process with and without stratified sampling. The graph depicts the boundaries $\overline{q}$ and $\underline{q}$ of the estimated quality according to the number of samples used for the estimation. These two versions have been tested on a wordlength configuration leading to a quality $\lambda = 0.77$ for a Squeeznet applications. For both of them, $\overline{q}$ and $\underline{q}$ has been reported to represent the confidence interval $\delta = 95\%$. The red curve represents the boundaries ($\overline{q}$ and $\underline{q}$) for MCS without stratified sampling. The purple curve represents the boundaries for MCS with stratified sampling as described in algorithm 3.

We can observe that the boundaries tend to the real value λ over the iteration. Furthermore, the boundaries for an MCS without Stratified cross ε_{max} set to 0.01 after the

Algorithm 4 Sampling Monte Carlo estimation for wordlength optimization

Input :$z(\overline{q}), \varepsilon_{max}, N_b, D$ and p
$Xs_d = \emptyset \ \forall d \in \Omega$
$k \leftarrow 0$
while $\hat{\varepsilon} \geq \varepsilon_{max}$ and $kN_b < card(\Omega)$ **do**
    $k \leftarrow k + 1$
    for $d \in \Omega$ **do**
        $n_d \leftarrow round(N_b p(d))$
        $X_d \leftarrow$Draw n_d images without replacement
        $Xs_d \leftarrow Xs_d \cup X_d$
        Estimate $\hat{\lambda}_d$ on Xs_d
        Estimate $\hat{\sigma}_d$ on Xs_d
    end for
    $\tilde{\lambda} \leftarrow \sum_{d\in\Omega} p(d)\hat{\lambda}_d$
    $\hat{\sigma} \leftarrow \sum_{d\in\Omega} \frac{p(d)^2}{kn_d}\sigma_d^2$
    $\hat{\varepsilon} \leftarrow z(\overline{q})\sqrt{\frac{\tilde{\lambda}(1-\tilde{\lambda})}{kN-1}}$
end while
    return $\tilde{\lambda}$

test of 2600 inputs. On the other hand, with stratified sampling, the boundaries cross ε_{max} after the test of only one batch of 50 inputs.

5.4 Metric for grouping inputs

The stratification sampling algorithm presented in section 5.3.2 requires to group the inputs into different subspaces d. These subspaces must be selected such that the input belonging to a subspace d leads to a similar behavior in terms of quantization process. Satisfying this condition will allow reducing the variance of the output quality metric. By extension, the minimum number of samples required to obtain a given confidence interval will be reduced. In this section, different metrics have been explored and their efficiency has been evaluated. These explorations are illustrated using SqueezeNet applications and datasets.

5.4.1 Specific metrics for classification problems

The proposed method can be adapted to any application that involves comparing the outputs of a golden reference to those of a tested model, as illustrated in Figure 5.1. This

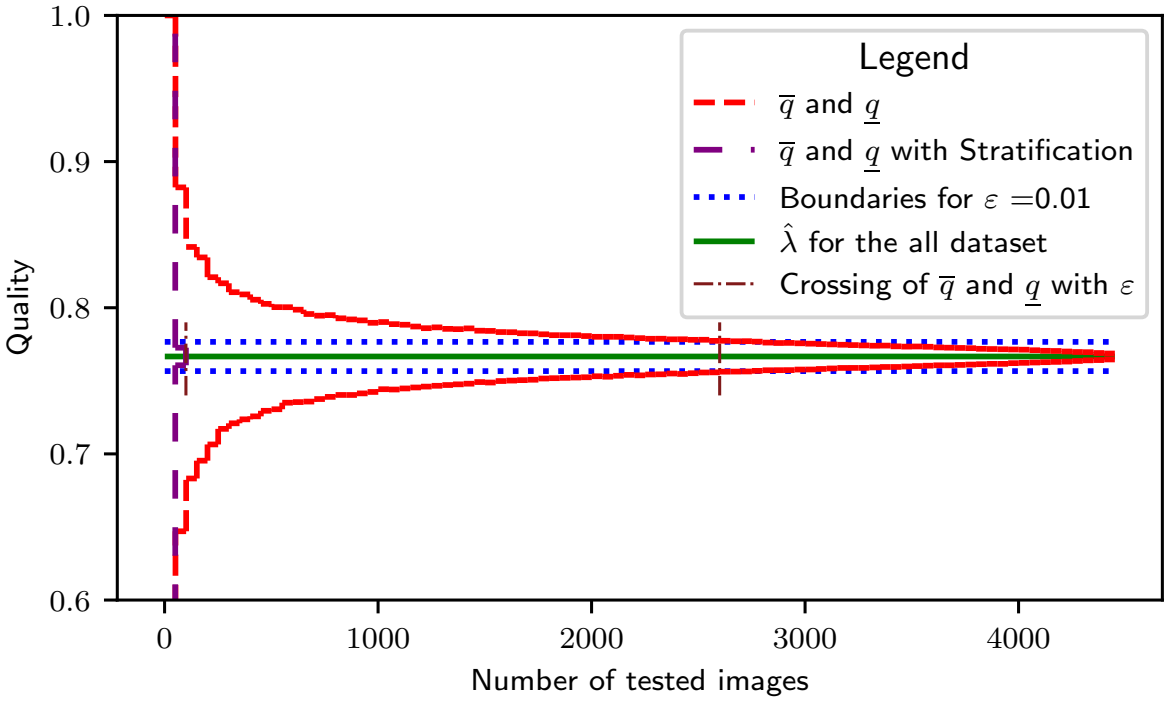

Figure 5.3 – Evolution of the confidence with and without Stratified using a batch size of $N_b = 50$ on a Squeezenet application.

model is particularly well-suited for classification applications that require equal outputs. For this reason, we have proposed metrics that utilize the specific characteristics of these applications.

Top 1 value

The first proposed metric is based on the probability associated to the most probable output (Top 1) for a given input.

In this chapter, the generic method to estimate quality degradation is to compare the outputs between the golden reference and the tested (quantized) version. In the context of classification application, an error occurs when the class predicted by the reference model is different from the class predicted by the tested model. It means that the quantization error has modified the predicted class. The assumption for this metric is that the input will be more resilient to degradation due to quantization when its Top 1 probability is high. In other words, the confidence in the class is high when the Top 1 probability is high. Grouping inputs with similar Top 1 probabilities will lead to similar resilience to

quantization process. This could result in small variance among the outputs of each group.

Distribution among the different group for Squeezenet dataset is shown in figure 5.4. The group Ω_j of value p_j is defined for an input i with a top-1 probability of p_{i1} using the following equation:

$$i \in \Omega_j \text{ if } p_j = \frac{\lfloor p_{i1} \times 10 \rfloor}{10} \tag{5.10}$$

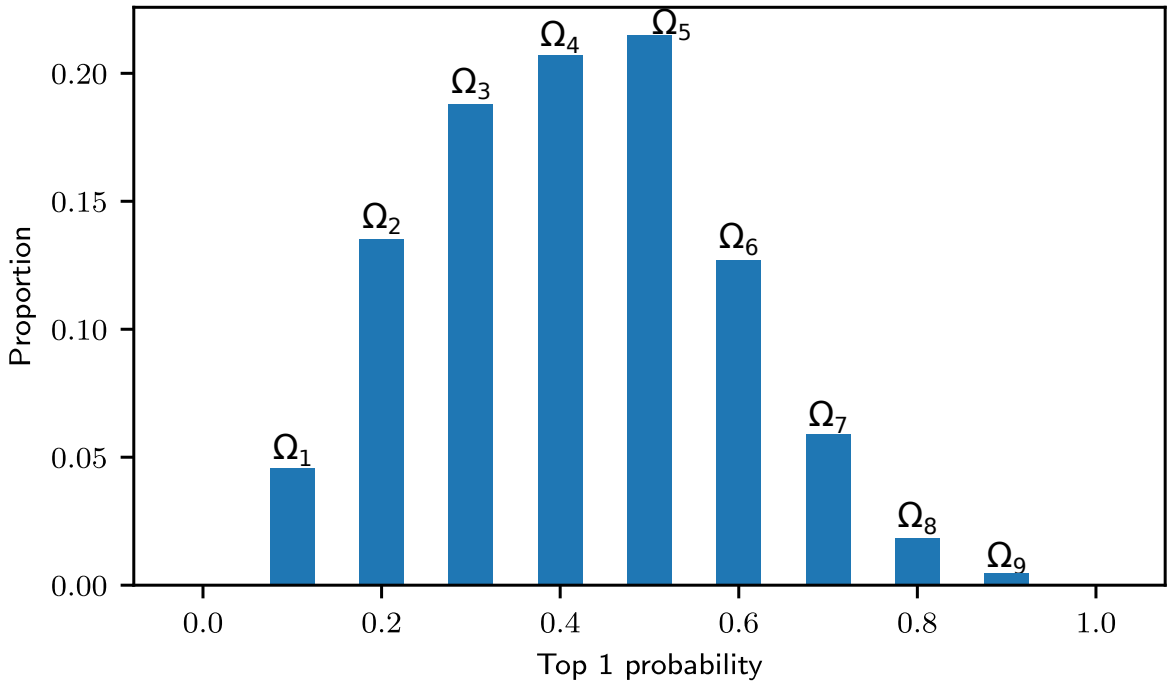

Figure 5.4 – Proportion of inputs according to the Top 1 probability for the 4520 images in the dataset.

Top 2 value

The second proposed metric is based on the probability associated to the second most probable output (Top 2) for a given input. This metric is based on the same assumptions as the Top 1 metric. It assumes that the second most probable output will replace the most probable ones due to quantzation process. The assumption for this metric is that the input image will be more resilient to degradation due to quantization when its Top 2 probability is low. A high Top 2 probability would then imply greater sensitivity to quantization degradation. Grouping inputs with similar Top 2 probabilities will lead to

similar resilience to quantization process. This could result in small variance among the outputs of each group.

Distribution among the different group for Squeezenet dataset is shown in figure 5.5. The group Ω_j of value p_j is defined for an input i with a top-2 probability of p_{i2} using the following equation:

$$i \in \Omega_j \text{ if } p_j = \frac{\lfloor p_{i2} \times 10 \rfloor}{10} \tag{5.11}$$

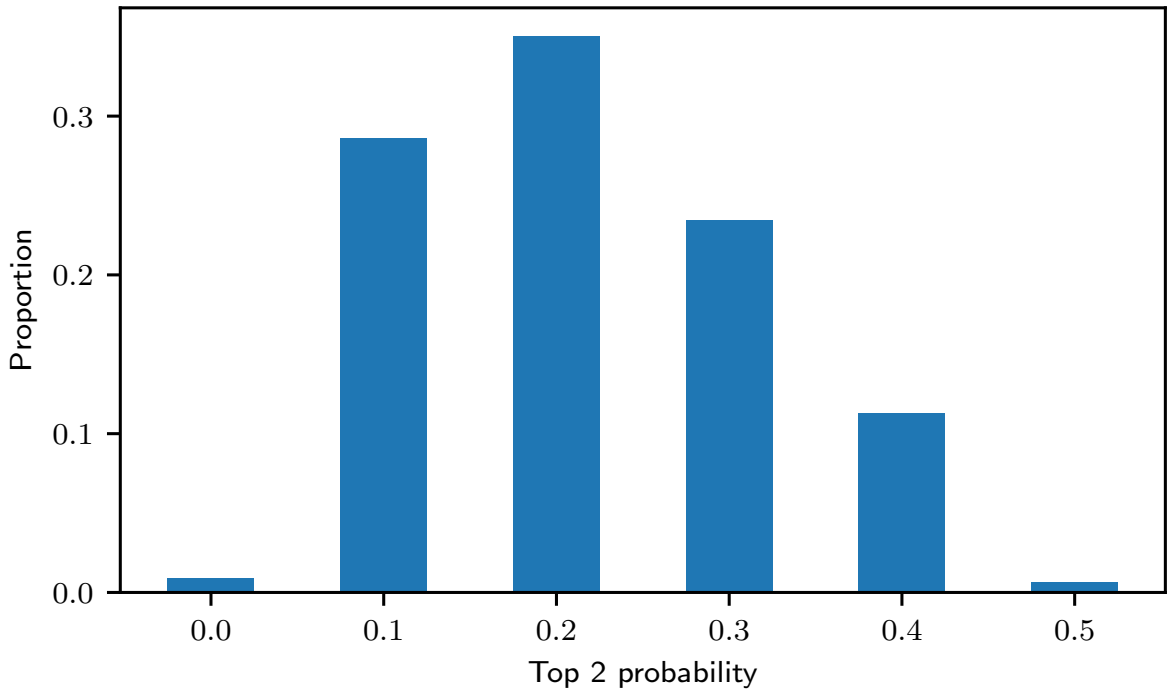

Figure 5.5 – Proportion of inputs according to the Top 2 probability for the 4520 images in the dataset.

Delta Top 1 Top 2 value

The third proposed metric is based on distance between the Top 1 and the Top 2 probabilities This metric is based on the same assumptions as the Top 2 metric. However, instead of assuming that a high Top 2 probability will facilitate the second most probable outputs to replace the most probable, we assume that a small distance between those two probability will facilitate it. A small distance between the Top 1 and the Top 2 would then imply greater sensitivity to quantization degradation. Distribution among the

different group for Squeezenet dataset is shown in figure 5.6. The group Ω_j of value p_j is defined for an input i with a top-2 probability of p_{i2} and a top-1 probability of p_{i1} using the following equation:

$$i \in \Omega_j \text{ if } p_j = \frac{\lfloor (p_{i1} - p_{i2}) \times 10 \rfloor}{10} \tag{5.12}$$

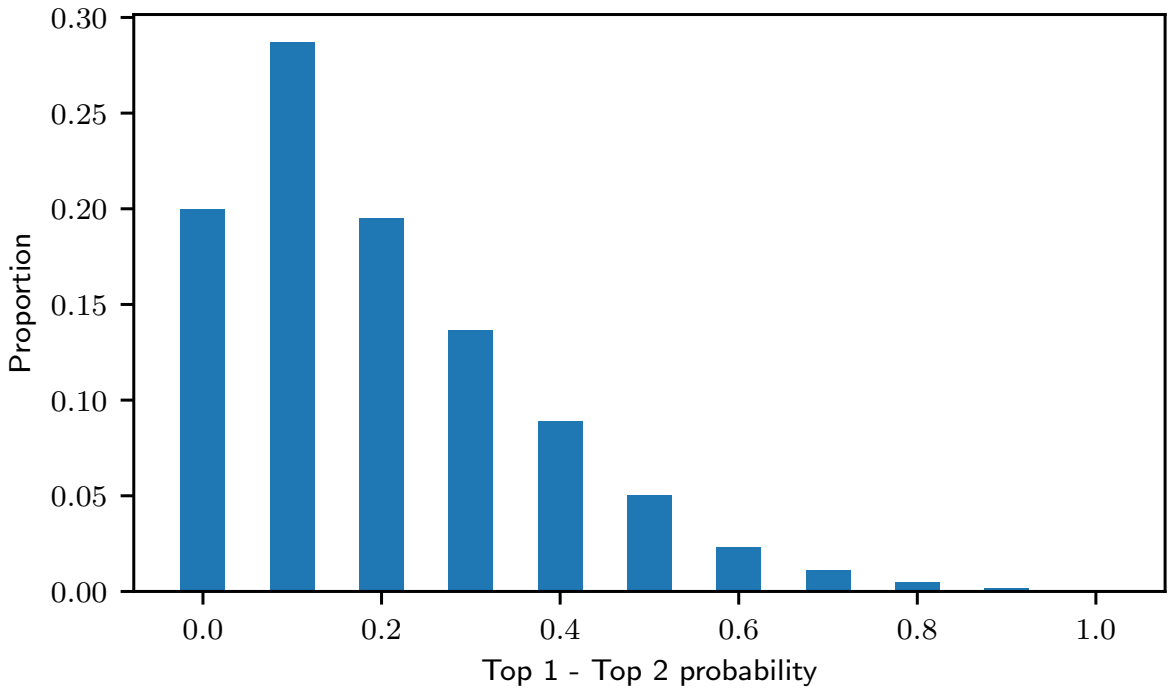

Figure 5.6 – Proportion of inputs according to the difference between the top 1 and the top 2 probabilities for the 4520 images in the dataset.

5.4.2 Difficulty

We propose a metric called *difficulty*, available for any generic application targeted in this chapter. This metric aims at evaluating, for a given input, the difficulty to quantify the application for this input. This metric is defined using a uniform wordlength for all the data inside the whole application. This corresponds to reducing the design space of the precision optimization to a single variable. From this exploration, we extrapolated similar behavior on the full space. The value b_x for an input x is the minimal value for which $\varphi_b(x) = 1$. For a DNN, this value corresponds to the lowest wordlength for which the Top-M matches with the golden reference. The value b_x can be found for each input

using a dichotomy approach.

All the inputs with the same value b_x are grouped together in the subspace S_b. Distribution among the different group for Squeezenet dataset is shown in figure 5.7

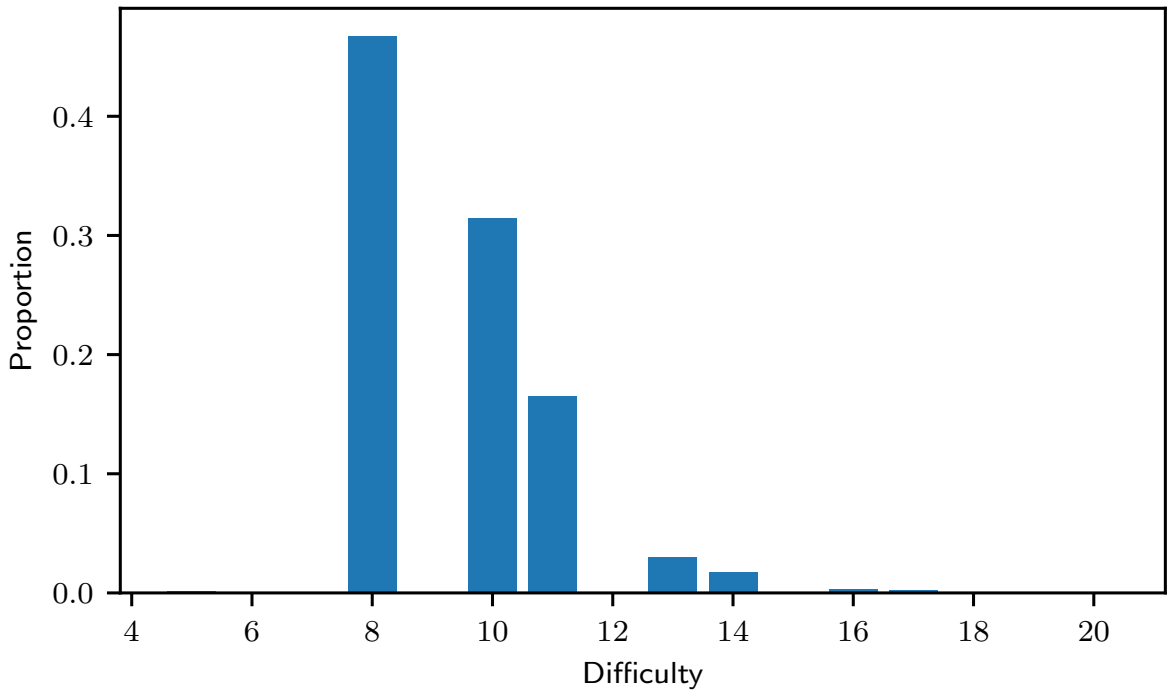

Figure 5.7 – Proportion of inputs according to their difficulty estimated for the 4520 images in the dataset.

5.5 Experimental results

To evaluate the efficiency of the proposed approach, the experiments have been conducted on a wordlength optimization algorithm for custom floating-point arithmetic and the quality is evaluated by Monte-Carlo Simulations.

5.5.1 Comparison of the metric for input grouping

To compare the metric to group the inputs, described in Section 5.4, we conduct experiments on the Squeezenet version described in Section 4.4 of chapter 4. These experiments were conducted using 126 quantization configurations. For each of these configurations,

the quality is estimated on the complete dataset and is assumed to be the real value λ. The configurations were selected to represent a quality in the range $[0; 1]$.

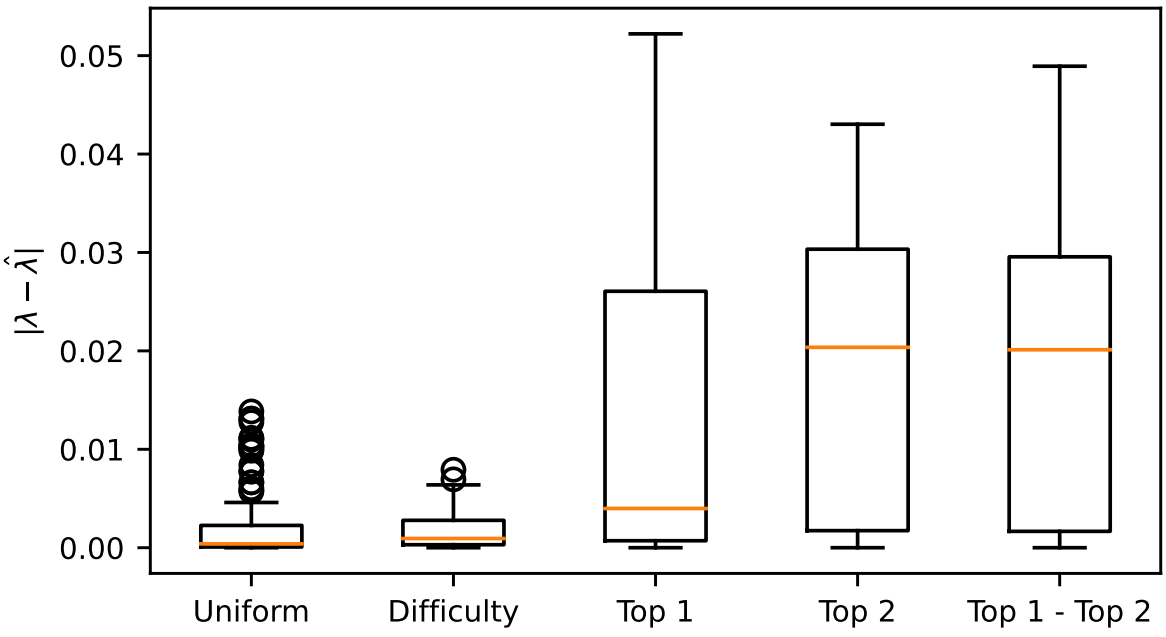

Figure 5.8 – Distance between the estimated value and the real value according to the metric used for stratification sampling on Squeezenet configurations.

A comparison of theses metrics can be seen in Figures 5.8, 5.9 and 5.10. In Figure 5.8, we represent the variation of the distance between the real quality λ estimated on the complete dataset, and the quality estimated $\hat{\lambda}$ using Algorithm 4 according to the metric used. For all the configurations, the mean is represented by an orange line. The boxplot represents 95% of the values, and the minimum and maximum values are represented by black lines. Figure 5.8 shows that, for all metrics except the difficulty metric, MCS using uniform sampling results in closer and more consistent results than stratified sampling. For example, the Top-1 metric estimation error varies greatly, ranging from 0.00015 to 0.024, whereas the uniform sampling varies from 0.0001 to 0.0020.

However, achieving uniform sampling required a large number of images, as illustrated in Figure 5.9. In Figure 5.9 and 5.10, the mean number of samples used to estimate the quality $\hat{\lambda}$ and its variation according to the quality λ are presented. The quality λ is used as abscissa because the confidence in the estimation $\hat{\varepsilon}$ is based on the following formula [BDM19]:

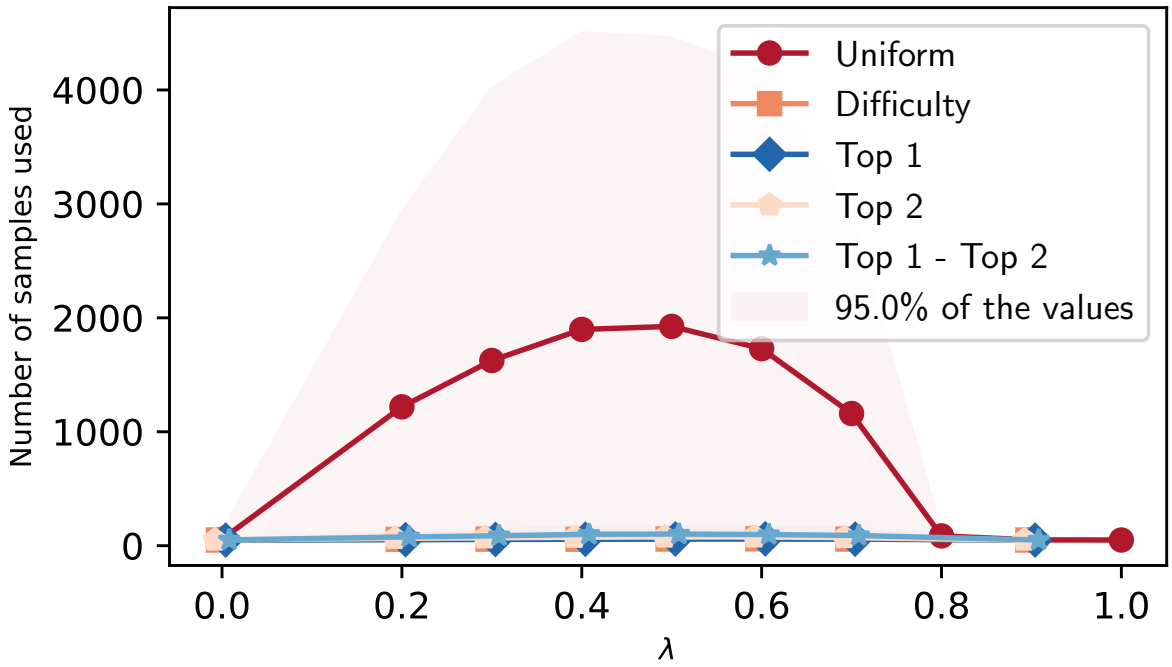

Figure 5.9 – Mean number of samples used to estimate the quality according to the quality and the sampling method.

$$\hat{\varepsilon} = z(\overline{q})\sqrt{\frac{\hat{\lambda}(1-\hat{\lambda})}{N-1}} \tag{5.13}$$

This leads to mean number of samples depending on the estimated quality, $\hat{\lambda}$, with a maximum reach for $\hat{\lambda} = 0.5$ and a minimum reach for $\hat{\lambda} = 0$ or 1. This phenomenon can be observed in our experimental results. The difficulty metric obtains a result similar to that of uniform sampling. As Figure 5.8 shows, the absolute error is close to that of uniform sampling. As can be seen in Figure 5.10, this metric also required fewer images to obtain these results. These results can be explained by the a priori analysis used to create this metric. The difficulty metric will then be used in further experiments.

5.5.2 Comparison on a complete optimization

To experiment the impact of our proposed Stratified sampling approach, a complete wordlength optimization is achieved on the Squeezenet Convolutional Neural Network (CNN) described in Section 4.4 of chapter 4. Two versions of a Squeezenet CNN are con-

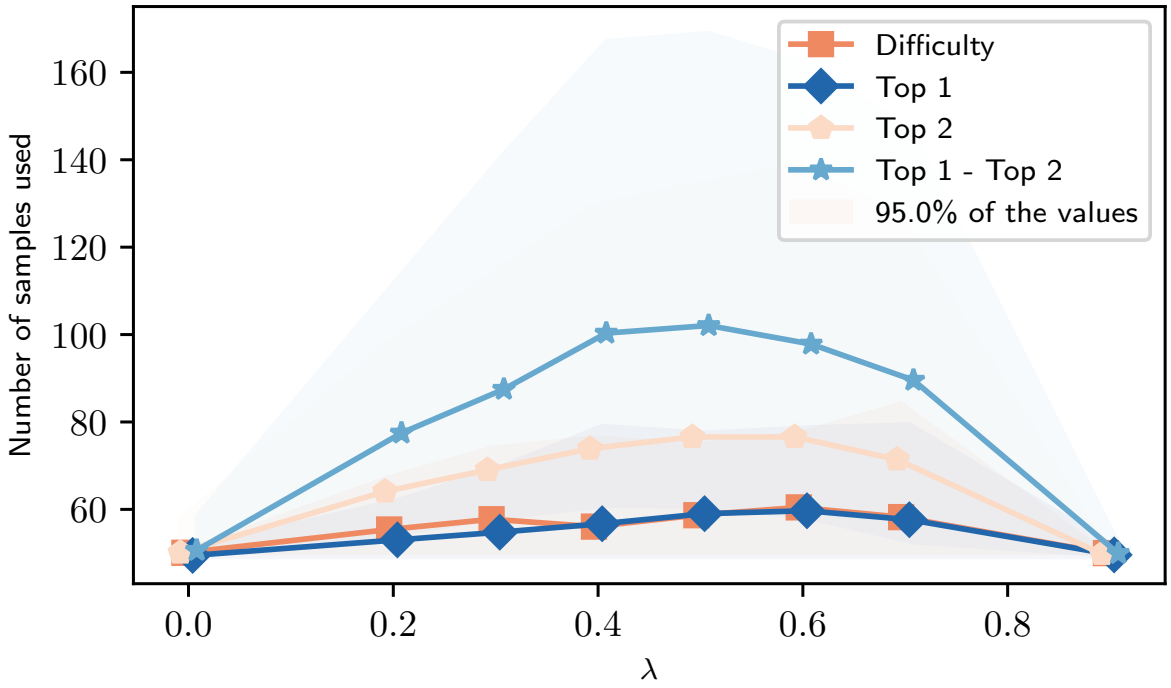

Figure 5.10 – Mean number of samples used to estimate the quality according to the quality and the sampling method.

sidered. Squeezenet is used for image classification and is composed of 15 layers. The two squeezenet versions differ in the number of variables to optimize, which affects the complexity of the optimization algorithm. For the smaller one (SQUEEZENET-8v), 8 variables are involved in the optimization process and for the bigger one (SQUEEZENET-15v) 15 variables are involved in the optimization process. The 15 variables were determined using the methods described in Chapter 4. The 8 Variables version is a version that has been simplified by removing the variables that are not frequently utilized. In this experiment, the φ function returns 0 if the Top-1 is different between the golden reference (floating-point version - FP32) and the tested version. The φ function returns 1 otherwise. The tested version is based on the *ac_float* library [SIE22].

To evaluate the impact of Stratified sampling, we compare our proposed approach described in algorithm 3 with the uniform MCS. The input database is made-up of 4520 images. As mentioned in the previous section, the metric used to group input into subdivision is the *Difficulty*, and is done beforehand as described in Section 5.4.2. For each difficulty value, the proportion of images is shown in Figure 5.7.

The optimization process determines a relative cost gain compared to the initial version

Table 5.1 – Results of the optimization process for SQUEEZENET-8v and SQUEEZENET-15v with and without Stratified sampling.

		SQUEEZENET-15v		SQUEEZENET-8v	
	Stratified	Without	With	Without	With
Relative Cost	Mean	0.179	0.188	0.149	0.175
	Variance	1E-4	1E-4	6E-5	5E-4
λ	Mean	0.984	0.957	0.974	0.967
	Variance	2E-4	1E-6	3E-4	1E-4

based on FP32. The estimation being a random process, it can impact the optimization process and consequently the final result. We perform 5 optimizations with the same parameters ($\lambda_{min} = 0.95, \delta = 99\%, \varepsilon_{max} = 0.01$) to compensate for possible variations in the results. The mean and the variance of the relative cost and the quality obtained at the end the optimization process are presented in Table 5.1. These results show that the solution of the optimization process is not significantly modified by Stratified sampling. However, for a same confident interval the results with stratified requires less images as shown in Figure 5.11.

The figure 5.11 illustrates the mean number of images required to estimate the quality $\hat{\lambda}$ of a configuration $\mathbf{w}$. The number of images used by the approach based on uniform MCS is on average equal to 523 for the SQUEEZENET-15v and 748 for the SQUEEZENET-8v. The number of images used by our approach based on stratified sampling is on average equal to 66 for the SQUEEZENET-15v and 69 for the SQUEEZENET-8v. The number of images used by the approach based on uniform MCS is higher of an order of magnitude compared to the proposed approach based on stratified sampling. For stratified sampling, the number of images used is then, on average, close to the minimal number, which is the batch size $N_b = 50$.

This reduction of number of images is even more important when scaled by the number of configurations tested to find a solution in the MIN+1 algorithm. This is illustrated in Figure 5.12. In our proposed approach a pre-processing phase is required to compute the difficulty associated to each image. This phase is done only once and then can be used for the different wordlength optimization processes. For the 4520 input images, 18062 simulations have been carried-out to compute the difficulty of all input images. Even if this phase is taken into account, our approach is still faster than the approach based on uniform MCS.

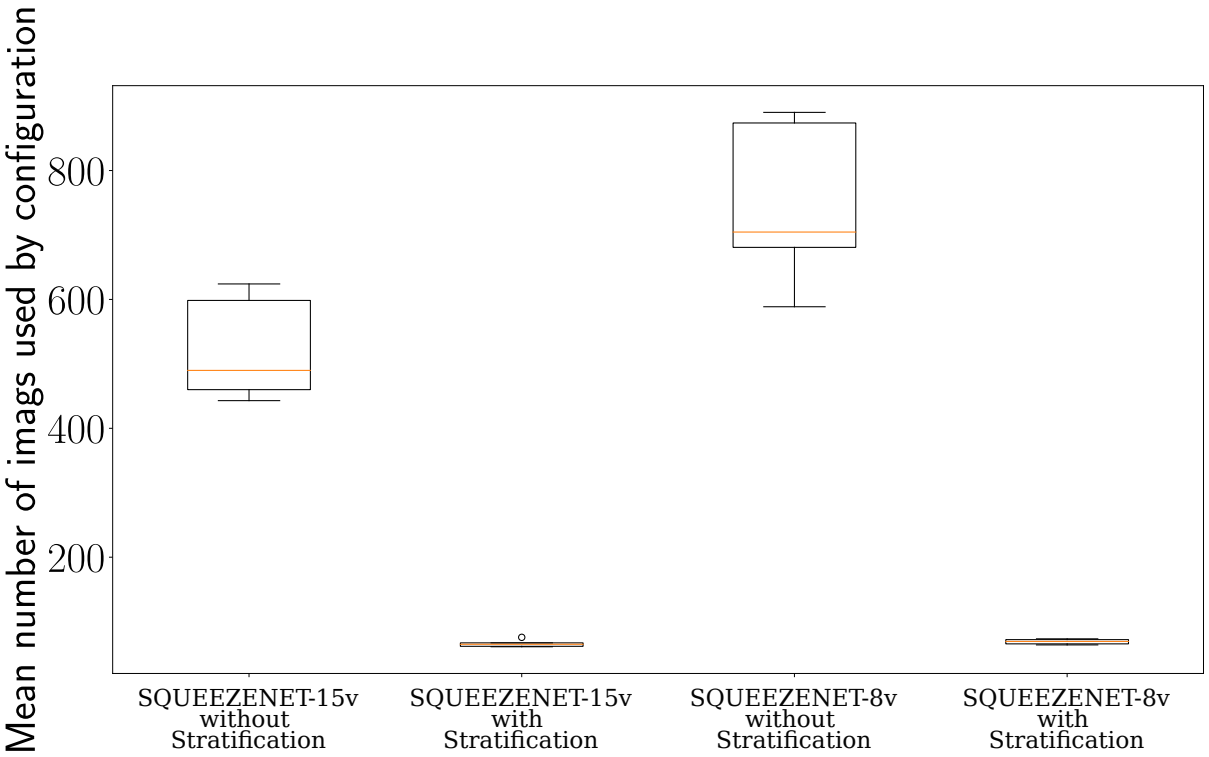

Figure 5.11 – Mean number of images required for one quality estimation for SQUEEZENET-15v and SQUEEZENET-8v with Uniform and Stratified sampling.

5.6 Conclusion

This chapter proposes a method to reduce the number of inputs to obtain a confident estimation of quality degradation due to quantization. This method relies on a variance reduction method of Monte-Carlo simulation : Stratified sampling. The use of Stratified sampling requires subdividing the input dataset into subspaces. This subdivision will define the variance reduction.

The proposed approach uses simulations of the targeted application to identify similar features in the inputs. A difficulty metric is computed for each input image and inputs having the same difficulty are grouped together in the same subdivision.

With these difficulties, we propose an iterative algorithm to determine the quality of a system. This iterative algorithm computes a quality estimation using successive batches of inputs to tend to the theoretical minimal required number of inputs. For each batch, the quality and their respective variances are estimated through simulation for each difficulty value. The number of inputs used for each of these estimation being proportional to the size of the difficulty subdivision. A weighted sum of these estimation gives the global

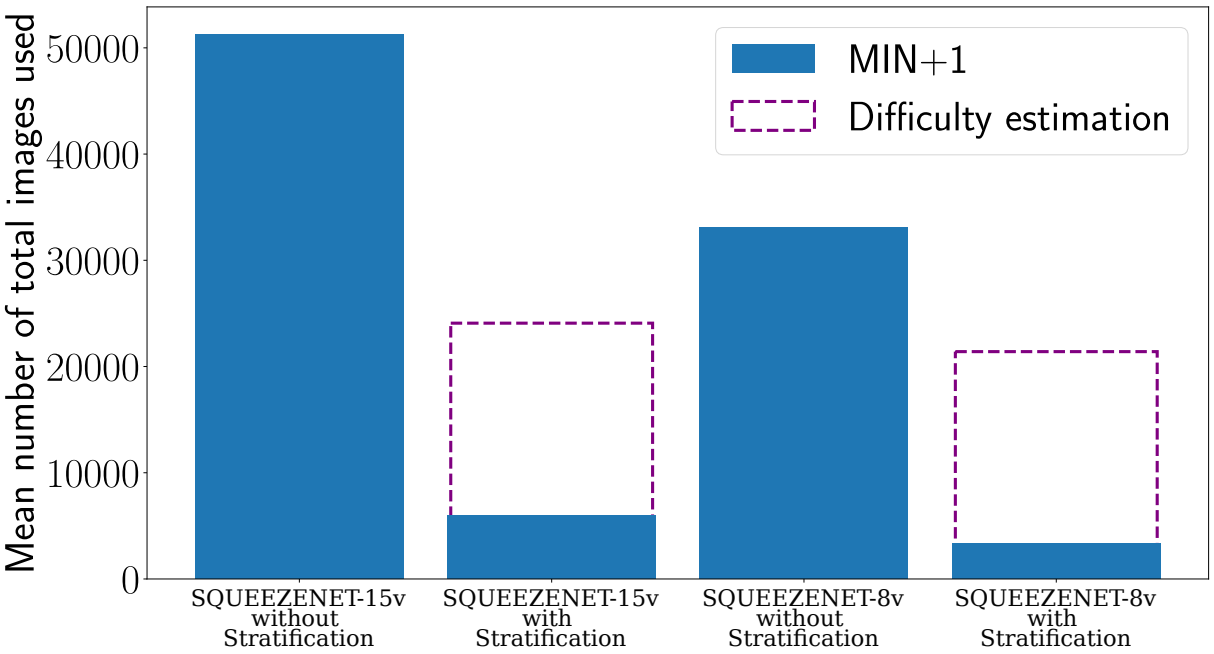

Figure 5.12 – Mean number of images required for a complete optimization process for SQUEEZENET-15v and SQUEEZENET-8v with and without stratified sampling.

quality estimation.

We experiment this method on a complete wordlength optimization for two versions of a Squeezenet CNN. We compare the results and the number of inputs required with and without the stratified method. The results show a similarity in the output of the optimization process but a reduction of one order of magnitude of the number of inputs tested with the Stratified approach.

Chapter 6

Importance Sampling for precision optimization

6.1 Introduction

In Chapter 5, we attempted to minimize the number of samples used for quality estimation by employing a variance reduction method : Stratified Sampling. This method enables a reduction in the number of samples required while maintaining similar confidence in the results. However, utilizing this method and achieving comparable results necessitates the use of a specific metric that requires some preprocessing beforehand to be applied in a complete optimization process. The reduction achieved compensates for these preprocessing requirements. Employing a simpler metric could potentially eliminate the need for this preprocessing requirement and reduce the overall time required for an optimization process.

To employ a simpler metric, we investigated another variance reduction method called Importance Sampling, which is presented in Section 6.2. This method uses a custom density function to weight the samples. The method can be adapted for the density function of diverse metrics in an attempt to implement them. In Section 6.3, we present simulations of importance sampling utilizing specific Probability Density Functions (PDF). We explored multiple PDFs to find one that requires the least number of samples for confident quality estimation. In Section 6.4, we present the use of an adaptive algorithm for importance sampling. This algorithm attempts to mimic the hypothetical minimum variance PDF using an iterative process. Finally, in Section 6.5, we present experiments conducted on the Squeezenet application utilizing adaptive importance sampling.

6.2 Importance Sampling

The Importance Sampling (IS) is a methodology that was primary used for rare event inference in statistical physics. It was then used for variance reduction, since it reduce the computational cost for rare event to append. [Bug+17]

IS work by modifying the function used to draw inputs. In Monte Carlo simulation has presented in section 3.2.2 of chapter 5, the distribution $\pi(X)$ uses to draw an input gives to every possible input an equal chance to be drawn. So, inputs with a lot of information, such as inputs that have a huge chance to fail for a quality to estimate near to 1, have a low probability to be drawn. Defining a distribution $q(X)$ that increase the probability for this data to be drawn will then increase the confidence in the estimation. However, if the function $q(X)$ is not correctly selected, it could reduce the confidence in the estimation and then could increase the number of required inputs [ZSB23].

Modify the probability distribution require modifying the estimator to avoid bias in the result. For that, every result of the function φ need to be weighted as in equation 6.1.

$$\tilde{\lambda}(\mathbf{w}) = \frac{1}{\sum_{k=0}^{K} w(X^k)} \sum_{k=0}^{K} w(X^k)\varphi(x^k) \tag{6.1}$$

The weight $w(x)$ for each value x being described as the relation between the original distribution and the new one (equation 6.2) [Bug+17; ZSB23].

$$w(X) = \frac{\pi(X)}{q(X)} \tag{6.2}$$

Figure 6.1 shows the density functions $\pi(x)$ and $q(x)$ and the deduced weight $w(x)$. This illustrates that the weight reduces the impact of an input, x, for which the probability of being drawn was increased. Otherwise, the weight increases its impact. This compensates for the bias by balancing the increased probability.

6.3 Probability Density Function selection

To apply importance sampling for quantization, it's necessary to define a new probability density function $q(x)$ that weights the more important inputs for MCS. By employing predefined types of PDF, only their parameters will need to be determined beforehand for use with Importance Sampling. This could result in a reduction in processing required before the quantization optimization process.

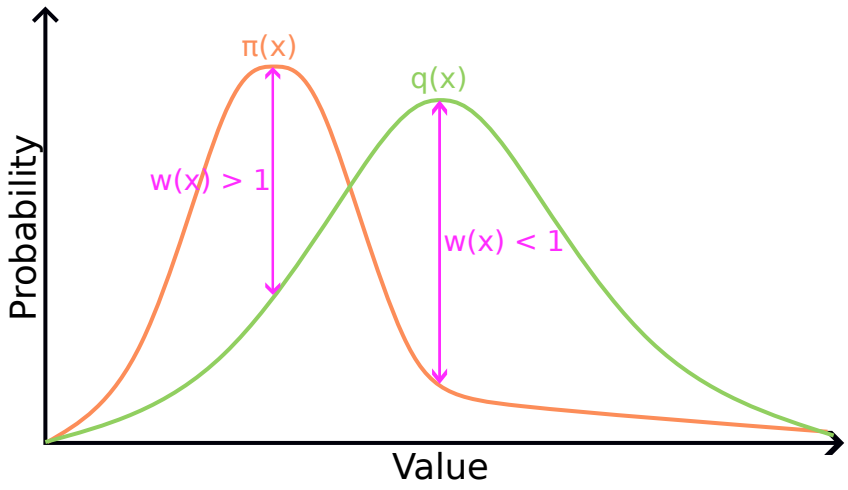

Figure 6.1 – Representation of the weights according to the original and the new probability distributive function

In this section, we realized simulations on two different types of PDF that match with the histograms of the different metric present in section 5.4 of chapter 5 The goal of theses simulations is to explore if using a predefined PDF can lead to a decrease in the number of inputs compare to an uniform sampling method.

6.3.1 Simulation using Gaussian distribution

The first type of PDF are Gaussian based PDF. This distribution types has been chosen because it matches with the histogram of three of our metrics : Top-1, Top-2 and Top-1 - Top-2. The Gaussian PDF can be defined using two parameters, the mean μ and the variance θ with the equation 6.3.

$$p(x) = \frac{1}{\sqrt{2\pi\theta^2}} e^{-\frac{(x-\mu)^2}{2\theta^2}} \tag{6.3}$$

To try to explored the impact of quality estimation using MCS, multiple values have been explored for the terms μ and θ. The mean μ ranges were defined between the max and minimal of the metrics. For the difficulty metrics, the range were between $[8; 12]$. For the others metrics, the range were between $[0; 1]$. The variance θ were defined between $[0.5; 3]$.

For each parameter tested values, the quality was estimated for multiple quantization configurations. The error between the estimated quality $\hat{\lambda}$ obtained from Importance Sampling and Uniform Sampling have been compared with the quality estimated on the

complete dataset, which is considered as the real quality value λ. For each configuration, μ and θ, the estimation has been repeated 50 times to avoid that rare events affect the results.

The results are shown in Figure 6.2, which illustrates the mean number of samples used by the metric according to the quality λ to estimate. The 95% quantiles are also shown to demonstrate variation.

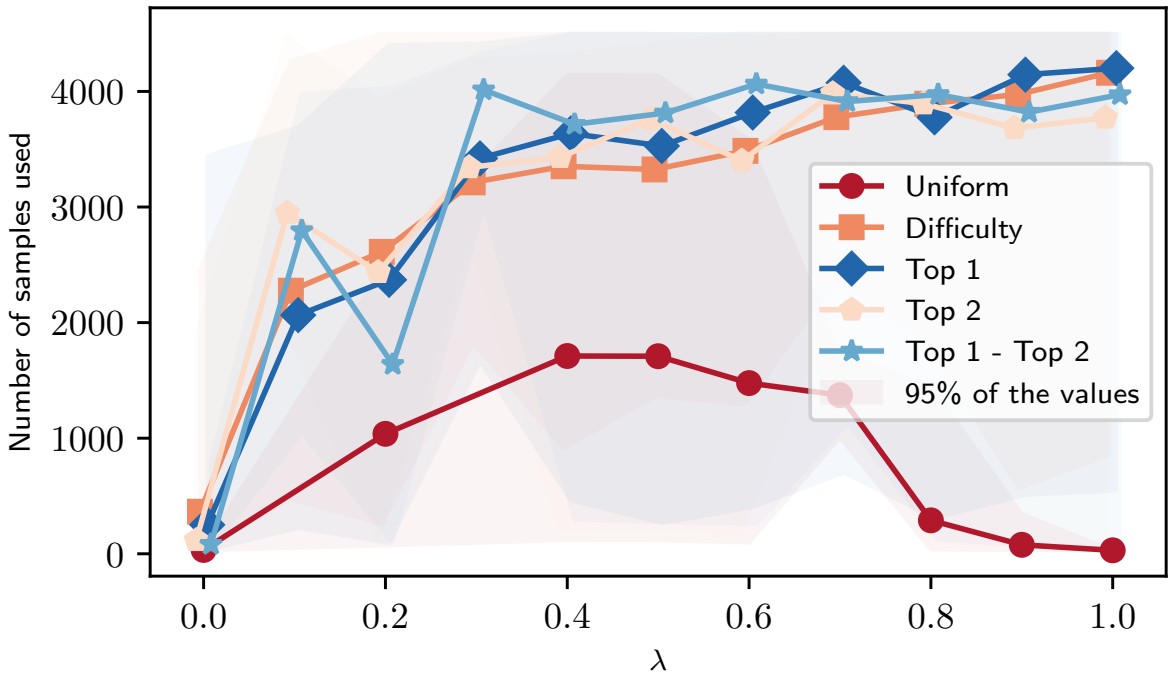

Figure 6.2 – Distance between the esimtaed value and the real value according to the metric used for Importance Sampling on Squeezenet configurations.

As shown in Figure 6.2, with these configurations, the minimum number of samples required to estimate quality is nearly as close as the number of samples for the case of Uniform Sampling. In the worst-case scenario, the complete dataset may be tested to estimate the quality with a given confidence. In most cases, the number of samples is more significant than Uniform sampling. This PDF does not accelerate estimation compared to Uniform Sampling. Consequently, another type of PDF corresponding to the Zipf distribution has been tested.

6.3.2 Simulation using Zipf distribution

The Zipf PDF has been chosen because this distribution matches with the histogram of the difficulty metric. The Zipf PDF can be defined using two parameters, the power a and the location l with the equation 6.4.

$$p(x) = \frac{1}{\zeta(a)(x-l)^a} \tag{6.4}$$

with ζ the Riemann zeta function.

To explore the impact of quality estimation using MCS with this PDF, multiple values have been explored for the parameters a and l. The location l ranges were defined between $[6; 7]$ for the difficulty metric and set to 0 for the other metrics in order to represent all inputs. The power a was defined between $[1; 5]$ for all metrics.

For each parameter tested value, the quality was tested for multiple quantization configurations. The error between the estimated quality $\hat{\lambda}$ obtained from Importance Sampling and Uniform Sampling was compared with the quality estimated on the complete dataset, which is considered as the real quality value λ. For each configuration, a and l, the estimation has been repeated 50 times to avoid that rare events affect the results.

The results can be seen in figure 6.3. As shown in Figure 6.3, with these configurations, the results are similar to the one obtain with Gaussian density function. This two tests illustrates that the methods to select a defined density function does not lead to relevant results to reduce the number of input samples.

6.3.3 Curse of High dimensionality

The Importance Sampling method, unlike Stratification Sampling, does not have certainty in its variance reduction. In some cases, the variance can even be increased, leading to a higher number of inputs required for confidence estimation. However, it has been demonstrated that a minimal local probability density function q_{opt} always exists for a given case (equation 6.5).

$$q_{opt}(x) = \frac{\phi(x)f(x)}{\lambda} \tag{6.5}$$

This minimal function is difficult to find in practice because it depends on the desired quality λ as a parameter. The experiments conducted in Sections 6.3.1 and 6.3.2 demonstrate that finding a PDF that closely approximates q_{opt} can be challenging in our case.

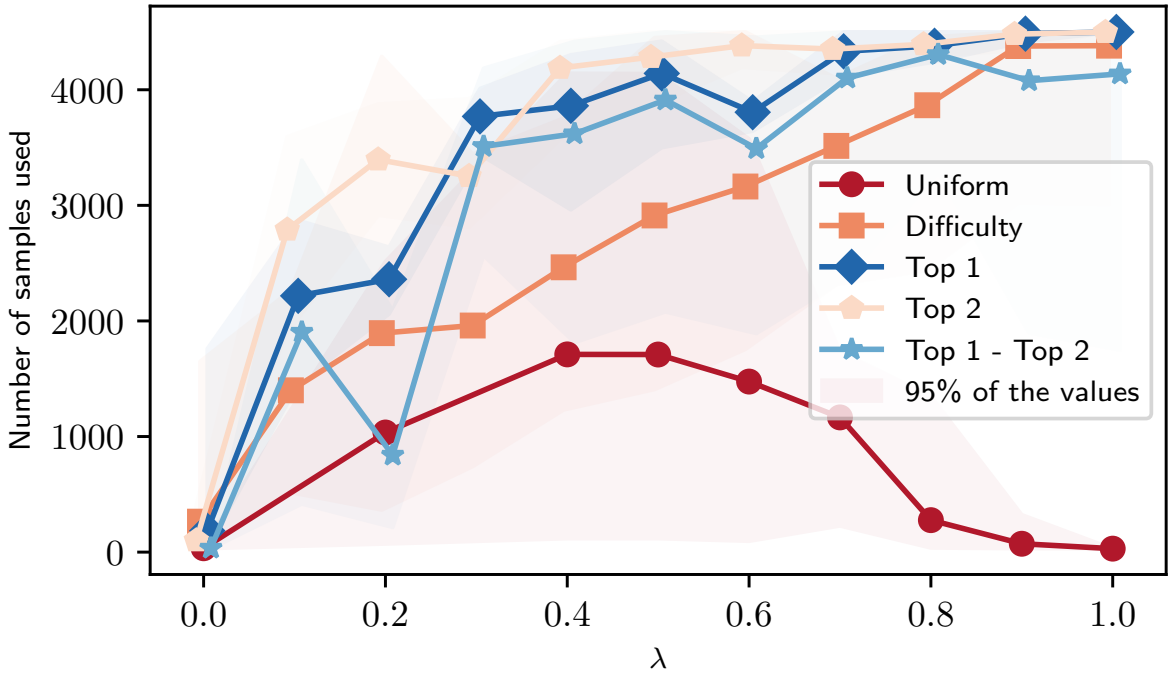

Figure 6.3 – Distance between the estimated value and the real value according to the metric used for Importance Sampling on Squeezenet configurations.

This difficulty can be attributed to the simplifications made when creating the metrics, which attempt to replicate a more complex multi-dimensional space that is our dataset. The loss of information resulting from this projection may explain why it is difficult to find a simple PDF for Importance Sampling.

For this reason, we explored iterative algorithms designed to converge towards the optimal probability density function to reduce variance and thus the number of samples.

6.4 Adaptive Importance Sampling

On way to exploit importance sampling is to approximate the hypothesized minimum variance function $q_{opt}(x)$ defined in equation 6.5 for variance reduction. The optimal PDF cannot be used in practice since it is defined using the desired quality. However, by employing an iterative algorithm, we can approach this optimal configuration. In Algorithm 5, an algorithm based on the one described in [BMM13] has been proposed.

The proposed algorithm begins with a first iteration on N_b images, using the histogram of the metric as the definition of q_0. Initializing with the metric facilitates an easier

tendency towards a more optimized function. However, for subsequent quality estimations during the optimization process, q_0 is set the most recent final distribution q_k. For each subsequent iteration, a new batch of N_b images is drawn from the dataset using the PDF q_k. With this new data, the quality parameter $\hat{\lambda}_k$ is estimated across all batches. The weight for each batch is determined using the corresponding PDF h_k employed to draw the images. Once $\hat{\lambda}_k$ has been estimated, the PDF used to draw future images q_k is determined based on the quality parameter $\hat{\lambda}_k$, the previous PDF q_k, the standard d-dimensional Gaussian function K_d with zero mean and covariance identity matrix, and the bandwidth B_k optimized according to the Asymptotic Mean Integrated Square Error criterion [BMM13; Sil86]. The variance $\hat{\sigma}$ is then determined to estimate the confidence interval, $\hat{\varepsilon}$. The process continues until $\hat{\varepsilon}$ is less than the user-defined maximal confidence interval ε_{max}.

Algorithm 5 Adaptive Importance Sampling Algorithm to estimate the quality $\hat{\lambda}$

Input :$d(X), z(\overline{q}), \varepsilon_{max}$ and N_b
$X = \emptyset$
$k \leftarrow 0$
$q_0 = f$
while $\hat{\varepsilon} \geq \varepsilon_{max}$ **do**
    $k \leftarrow k + 1$
    $I \leftarrow$Draw N_b images
    $\hat{\lambda_k} \leftarrow \frac{1}{kN} \sum_{j=0}^{k-1} \sum_{i=1}^{N} \phi(\mathbf{y}_i) \frac{f(\mathbf{y}_i)}{h_j(\mathbf{y}_i)}$
    $q_k(\mathbf{y}) \leftarrow \frac{1}{kN\hat{\lambda_k}B_k} \sum_{j=0}^{k-1} \sum_{i=1}^{N} \frac{f(\mathbf{y}_i^j)}{h_j(\mathbf{y}_i^j)} K_d(B_k^{-1}(\mathbf{y} - \mathbf{y}_j))$
    $\hat{\sigma} \leftarrow \hat{\lambda}(1 - \hat{\lambda}) \frac{kN_b}{kN_b - 1}$
    $\hat{\varepsilon} \leftarrow z(\overline{q}) \frac{\hat{\sigma}}{\sqrt{kN_b}}$
    $k = k + 1$
    $P = P_{k-1}$
end while
    return $\hat{\lambda}$

6.5 Experimental results

To evaluate the proposed algorithm 5, this algorithm was implemented on a MIN+1 optimization procedure described in Section 3.3.1 of Chapter 3. The experiments were conducted for the optimization of a fixed-point version of a 15-variable Squeezenet application, as presented in Section 4.4.3. This algorithm has been tested using the difficulty

Table 6.1 – Mean number of calls of the Squeezenet by execution (N_S) and time of execution of the algorithm according to λ_{min}, the ε and the method of Sampling for a $\sigma = 0.05$

		Uniform		Importance		Gain relative (%)	
λ_{min}	ε	N_S	Time	N_S	Time	N_S	Time
0.90	0.1	6100	24035	6000	23573,2	1,64	1,92
0.95	0.05	12450	49886	11262,5	44411,5	9,54	10,97
0.99	0.1	5875	22964,25	5600	21911,25	4,68	4,59
	0.01	77850	298938,6	40170	159358,8	48,40	46,69

metric presented in section 5.4.2 of chapter 5 due to its stability. Various pairs $(\lambda_{min}, \varepsilon)$ were explored to assess the potential benefits of the adaptive algorithm. The results are presented in Table 6.1.

The results demonstrate that the adaptive importance sampling method can achieve a reduction in the mean number of tested samples and, consequently, a time savings for the complete optimization process cab be obtained. Gains vary between 1.64% and 48.4%, depending on the confidence interval ε. A larger ε results in a higher number of samples required, allowing more iterations for the iterative algorithm to converge towards an optimal density. This more optimal density reduces the number of samples needed to estimate quality.

The gains obtained were promising and indicate a direction for this method. However, in its current form, the gain is not as significant as that achieved by the stratification sampling method presented in Chapter 5. The gains achieved using stratification were around 90%, even when accounting for the preliminary simulations required. Even considering these preliminary simulations, the gain was still 50%, which exceeds the 48.4% obtained in the best-case scenario by the adaptive importance sampling method. Furthermore, the presented methods still rely on preprocessing simulations to estimate the initial PDF based on the difficulty metric described in Chapter 5.

6.6 Conclusion

This chapter proposes a new method to reduce the number of inputs required for obtaining a confident estimation of quality degradation due to quantization. The method utilizes a variance reduction technique in Monte-Carlo simulation called Importance Sampling, which requires the use of a new probability density function that weights the inputs

with more information.

Initially, we attempted to implement common probability density functions such as Gaussian and Zipf on the metrics developed in Chapter 5. However, this approach did not lead to a reduction in the number of tested samples and, in most cases, resulted in an increase in the number of tested samples. This increase can be attributed to the sensitivity of Importance Sampling and the complexity of the data being modeled into a single metric.

Subsequently, we proposed an adaptation of an iterative process that tends towards the theoretical optimal density function for Importance Sampling. This iterative algorithm computes a quality estimation using successive batches of inputs to tend towards the theoretical minimal number of inputs. For each batch, the quality and their respective variances are estimated through simulation, and a new PDF to draw inputs is modified at each batch. The new PDF is determined by assuming the quality estimates as the real quality for estimation purposes. This assumption allows for creating a supposed optimal density function to draw a new batch of images.

This method has been experimented on a complete precision optimization process for the Squeezenet CNN. The number of tested inputs has been compared with and without the Adaptive Importance Sampling method. The results showed a reduction in the number of tested samples. Consequently, the optimization time has been reduced up to 50% in the best-case scenario. While this reduction is promising for the use of Importance Sampling, it is still inferior to the one obtained using Stratification Sampling.

Chapter 7

Conclusion and future works

7.1 Thesis contributions

This thesis took place in the context of rethinking our entire production process to answer the evolving climate conditions. To address this issue, the different aspects of a product life cycle must be considered. This includes all the process from design to the end of life. The usage impact can affect a product in multiple ways, but a prominent way is through energy consumption. In this thesis, the approximate computing paradigm is exploited to reduce the implementation cost and more specifically the energy consumption. Approximate computing aims at reducing the implementation cost by potentially reducing the quality of an application result. This can be done at multiple levels: computation, hardware, or data. The data-level has been considered in this thesis.

Reducing the energy consumption of a data-oriented application can be achieved by selecting an efficient arithmetic representation and by optimizing the precision, *i.e.* the number of bits used to represent the data. The two commonly used arithmetic corresponding to floating-point and fixed point are considered in this thesis. The precision optimization process involves reducing the cost of an application by modifying the data precision while maintaining a quality level at the application output. The confidence in the quality estimation is also a parameters to take into account as it impact the results and the optimization time. This thesis focuses on designing methods to reduce the optimization time for precision refinement optimization. This optimization time depends on the number of configurations evaluated by design space exploration, the number of samples used to estimate quality, and the simulation time for a sample. In this thesis, we proposed contributions to reduce both the number of configurations, N_c, and the number of samples, N_s.

Precision refinement for custom floating-point. To reduce the number of configurations tested in the wordlength optimization process, we proposed an approach that

optimizes data wordlength for applications using custom floating-point datatype. This method takes as inputs a C or C++ source code to output a refined version of this source code with optimized data types.

In this method, a dynamic range analysis is performed to determine information for the optimization process. The optimization process is then achieved using heuristic algorithms corresponding to the MIN+1 and TABU. To optimize the exponent and mantissa wordlength according to a user-defined quality constraint, four strategies were established. Theses strategies are based on data dynamic range analysis to set the maximal exponent wordlength and simulations to adapt the complete wordlength. The results show an improvement in terms of energy consumption. Two strategies obtains better results than the others. The first one sets the exponent wordlength using the dynamic range and explores different mantissa wordlengths with the MIN+1 and TABU algorithms. This strategy obtains good results with the smaller number of tested configurations. The second one sets the exponent wordlength using the dynamic range and explores different mantissa wordlength with the MIN+1 algorithm. It then explores different mantissa and exponent wordlengths with the TABU algorithm. This strategy obtains the best results but require to test a higher number of configurations.

Stratified Sampling to accelerate quality degradation evaluation. The second contribution exploits stratified Sampling method. This method aims at reducing the global variance of the output quality estimation. The use of stratified Sampling requires subdividing the input dataset into subspaces. The quality of the subdivision will define the variance reduction efficiency. The proposed approach uses simulations of the targeted application to identify similar quantification resiliency in the inputs. We defined a difficulty metric from these simulations. The metric value associated to an input image corresponds to the minimal uniform wordlength allowing to satisfy the quality constraint. Input images having the same difficulty are grouped together in the same subdivision.

An iterative algorithm has been proposed to determine the quality of a system. This iterative algorithm computes a quality estimation using successive batches of inputs to tend to the theoretical minimal number of inputs. For each batch, the quality and their respective variances are estimated through simulation for each difficulty value. The number of inputs used for each of these estimation being proportional to the size of the difficulty subdivision. A weighted sum of these estimations gives the global quality estimation.

This method has been experimented on a complete wordlength optimization process

for two versions of a Squeezenet CNN. The quality of the estimation has been evaluated and the numbers of tested inputs with and without the stratified method have been compared. The results show similar trade-offs and confidence at the optimization process' end but a reduction of one order of magnitude for the number of tested inputs with the proposed approach.

This method enables a reduction in the number of tested samples while maintaining similar confidence in the results. However, utilizing this method and achieving comparable results necessitates the use of a specific metric that requires some preprocessing beforehand to be applied in a complete optimization process. The reduction achieved compensates for these preprocessing requirements. Employing a simpler metric could potentially eliminate the need for this preprocessing requirement and reduce the overall time required for an optimization process.

Importance Sampling to accelerate quality degradation evaluation. The third contribution exploits importance Sampling method. Importance Sampling uses a custom density function to weight the samples. The method can be adapted for the density function of diverse metrics in an attempt to implement them. Initially, we attempted to implement common PDFs such as Gaussian and Zipf on the metrics developed in Chapter 5. However, this approach did not lead to a reduction in the number of tested samples and, in most cases, resulted in an increase in the number of tested samples. This increase can be attributed to the sensitivity of Importance Sampling and the complexity of the data being modeled into a single metric.

Subsequently, we proposed an adaptation of an iterative process that tends towards the theoretical optimal PDF for Importance Sampling. This iterative algorithm computes a quality estimation using successive batches of inputs to tend towards the theoretical minimal number of inputs. For each batch, the quality and their respective variances are estimated through simulation, and a new PDF to draw inputs is modified at each batch. The new PDF is determined by assuming the quality estimates as the real quality for estimation purposes. This assumption allows to create a supposed optimal density function to draw a new batch of images.

This method has been evaluated on a complete precision optimization process for the Squeezenet CNN. The number of tested inputs has been compared with and without the Adaptive Importance Sampling method. The results showed a reduction in the number of tested samples. Consequently, the optimization time has been reduced up to 50% in the

best-case scenario. While this reduction is promising for the use of Importance Sampling, it is still inferior to the one obtained using Stratification Sampling.

7.2 Future works

7.2.1 Short term perspectives

Some short term perspectives can be envisaged to improve the efficiency of the proposed methods. These modifications would require testing and confirmation, but they are a promising way to obtain better performances.

Fine-grained batch size

One straightforward improvement can be achieved on method proposed in Chapter 5 to estimate the quality using simulation. To reach a theoretical minimum number of samples, we used an iterative algorithm. This algorithm works in batches of N_b inputs, which are added at each iteration to increase the total number of tested inputs and the confidence in the estimation. For instance, if $N_b = 50$, the first iteration will test 50 inputs. The second iteration will test a second batch of 50 inputs, bringing the total number of inputs tested to 100. These batches are drawn until the desired level of confidence is reached.

To ensure correct estimations, it is important to have a large enough sample size to follow the law of large numbers. This is why we assume a constant batch size of a significant amount. However, the law of large numbers only applies to the total number of inputs used for estimation. Therefore, it is possible to adapt the algorithm with a variable batch size. The first batch still needs to follow the law of large numbers, but subsequent batches can be reduced to more accurately tend toward a minimum. An adaptive size proportional to the current iteration's confidence interval can be used. If the estimated confidence interval is far from the desired confidence interval, an important batch size can be used. However, as the confidence interval tends to the targeted value, the batch size can be reduced. For example, the initial size, $N_b = 50$, could be kept. However, a second iteration with $N_b = 20$ could be performed, followed by an iteration with $N_b = 10$ if the desired confidence isn't reached. This would bring the total number of tested inputs to 80, compared to the 100 that our method would have used. This methodology still needs to be tested to ensure its compatibility with Stratified Sampling and refined in terms of successive sizing.

Variance reduction method and Kriging

Another way to reduce the time required for precision refinement optimization is to reduce the number of output application quality evaluations. In [BMD20], the authors propose a method called Kriging for this purpose. Kriging is a stochastic spatial interpolation technique from geostatistics. It predicts a value in a random field based on the known values of that field. The authors adapted this method for precision refinement optimization. They incorporated it into the MIN+1 algorithm. When new configurations need to be evaluated, the method determines whether the previously evaluated configurations allow for an estimation using Kriging with enough confidence. If so, the value is retained and used in the MIN+1 process. No simulations are then performed. Otherwise, simulations are performed, and the configuration is stored for subsequent configuration evaluation. The authors present impressive results: 90% of configurations are evaluated through Kriging instead of simulations, reducing optimization time by a factor of 10.

We did not explore this method for our proposed Stratified and Importance Sampling models to avoid confusion regarding error propagation. Indeed, since the Kriging evaluation is based on previously evaluated configurations, potential errors introduced by Stratified or Importance Sampling could propagate to the Kriging evaluation of configurations. This could have caused us to evaluate the combination of Stratified/Importance Sampling and Kriging instead of the methods individually. Now that we have observed an improvement with stratified Sampling, exploring that combination is a potential method of improvement.

7.2.2 Long term perspectives

Some long term perspectives can be envisaged by modifying the paradigm explored, looking into security or LCA aspects.

Attack friendly precision refinment optimization

Another approach could be modifying the paradigm for precision refinement. Security is especially important for embedded systems since the data is contained inside them. This raises the question of resilience to attacks. Each method presented in this thesis is based on quality estimation in a safe space. The results are computed without additional external noise or attacks.

As stated in [Lom+25], the impact of quantization on different types of attacks has

not been widely studied in the literature. Nevertheless, for attacks on hardware architecture, the attack depends on the data representation used by the applications [MBC21]. In [Lom+25], the results illustrate differences in resilience between 4-bit and 6-bit quantizations. The 4-bit version of the application obtains less information than the 6-bit version. Therefore, the impact of quantization on resilience to attacks is a factor that could be applied to our proposed algorithm. Furthermore, the increase in complexity brought about by mixed-precision architecture could lead to greater resilience when using our proposed methods. To evaluate this, the attack sequence could be incorporated into the estimation by simulation, as illustrated in Figure 7.1. Submitting systems to an attack and comparing the information the attacked obtain can determine a new quality metric.

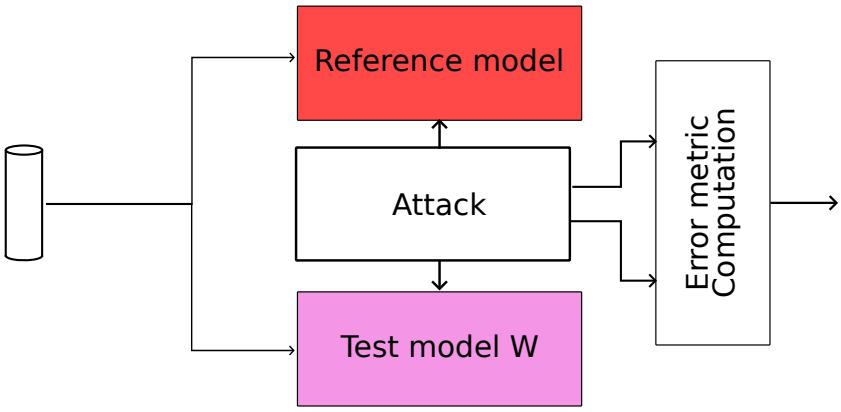

Figure 7.1 – Proposed modifications to measure the attack impact on quantization during optimization.

However, quantization could also penalize the defender's tactics. For example, software jammed-based countermeasures [KMH25] are a defensive method that adds random noise to each of the RGB values of each pixel. This noise is mostly applied to the lower bits. Since our quantization method mostly affects the lower bits, it is possible that our reduction will lead to less noise resilience, thereby reducing the flexibility of the Software-Jamming-based countermeasure. One possibility to explore is adding these defense methods before the output comparison (Figure 7.2).

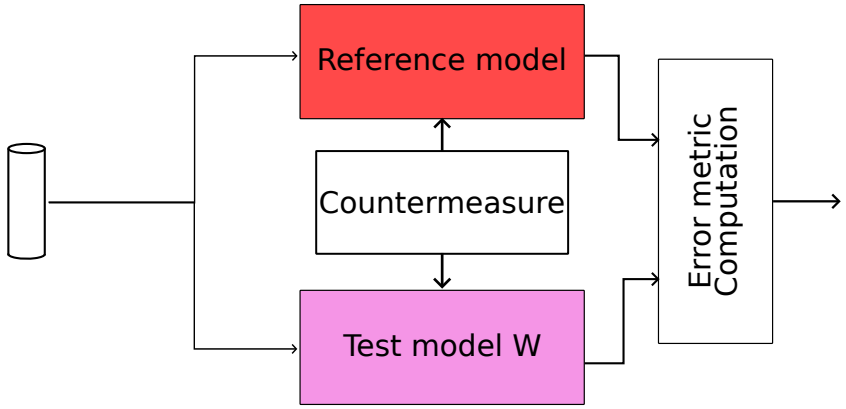

Figure 7.2 – Proposed modifications to measure the countermeasure efficiency on quantization during optimization.

Both of these methods could lead to modify the definition of the metric λ or the creation of a second metric μ different from the quality without attack. These two metrics evolving differently, the optimization method would need to be updated to use Multi-Objective optimization methods [Hua+21].

Confidence and statistics in electronics life cycle assessment

The expertise developed in this thesis can be exploited in a broader context. This thesis aims to reduce the optimization time for precision refinement using statistical tools. This work takes place in an effort to reduce the energy consumption of ICT applications and, consequently, the global impact of ICT on carbon emissions. However, the full life cycle need to be optimized to reduce this impact. This optimization requires to estimate the carbon emissions for each life stage of a product. Life Cycle Assessment (LCA) [HS02] is the right tool to evaluate the environmental impact.

In LCA, matrix computations are considered to represent the impact. These methods make it possible to determine the environmental impact of a product. However, most of the data used in the LCA databases are based on fixed values, usually the mean. Upper boundaries are also used to determine a worst-case scenario. However, as we presented in this thesis, using statistical tools allows for more flexibility and better representation of the information. In [Hei24], the authors called for the use of statistical tools in LCA such as uncertainty and sensitivity analysis. These methods are based on MCS and the relationship between multiple MCS estimations. They define the results and confidence

of the computation between two variables estimated using MCS.

Another possibility could be using the PDF of the values used in matrix computation methods instead of the means. This could allow for more cases to be represented. Furthermore, these methods could allow one to represent the probability of the worst-case scenario and determine whether it is negligible. Finally, these methods could pave the way for time-dependent estimation of usage according to usage (i.e., an application being used differently at different times of day).

Appendix A

French summary

A.1 Contexte Général

A.1.1 Électronique et développement durable

Les conditions climatiques en constante évolution à travers le monde nous obligent à repenser l'ensemble de notre processus de production. Certaines lois et normes ont été établies pour répondre à ces exigences. Par exemple, au niveau national, le Code de l'énergie français de 2015 (article L100-4) [Fra23] prévoit une réduction de 40% des émissions de gaz à effet de serre (GES) d'ici 2030 et, au niveau européen, l'Union européenne s'est fixé pour objectif d'atteindre la neutralité carbone d'ici 2050 [dév25].

Pour répondre à ces critères, tous les secteurs de production doivent être pris en compte. Selon [Com25], l'impact du secteur des technologies de l'information et de la communication (TIC), de la production à la fin de vie, représente 4% des émissions totales de GES. Bien que ce chiffre semble faible, chaque secteur doit réduire son impact afin d'atteindre la réduction souhaitée. C'est pourquoi la recherche sur l'analyse du cycle de vie (ACV) [HS02] a été développée au fil des ans pour les appareils électroniques [Boy11]. Cette analyse prend en compte les cinq étapes de la vie d'un produit : 1) Extraction des ressources ; 2) Fabrication ; 3) Transport ; 4) Utilisation ; 5) Fin de vie.

Cette méthode permet de déterminer l'impact de chaque étape du cycle de vie et de cibler les efforts à fournir pour une optimisation globale. Par exemple, dans [Lou+23], les auteurs fournissent une ACV d'un ordinateur Dell utilisé pendant cinq ans. La figure A.1 présente les émissions de GES exprimées en kilogrammes d'équivalent CO2 (kgCO2eq) pour chaque étape de l'ACV de l'ordinateur. La plupart des émissions sont dues aux composants de l'ordinateur Dell, avec 248,17 kgCO2eq. La deuxième source d'émissions la plus importante est la consommation d'électricité pendant sa durée de vie, avec 29,79 kgCO2eq.

Pour chaque étape du cycle de vie, plusieurs approches peuvent être envisagées [Pel+24]

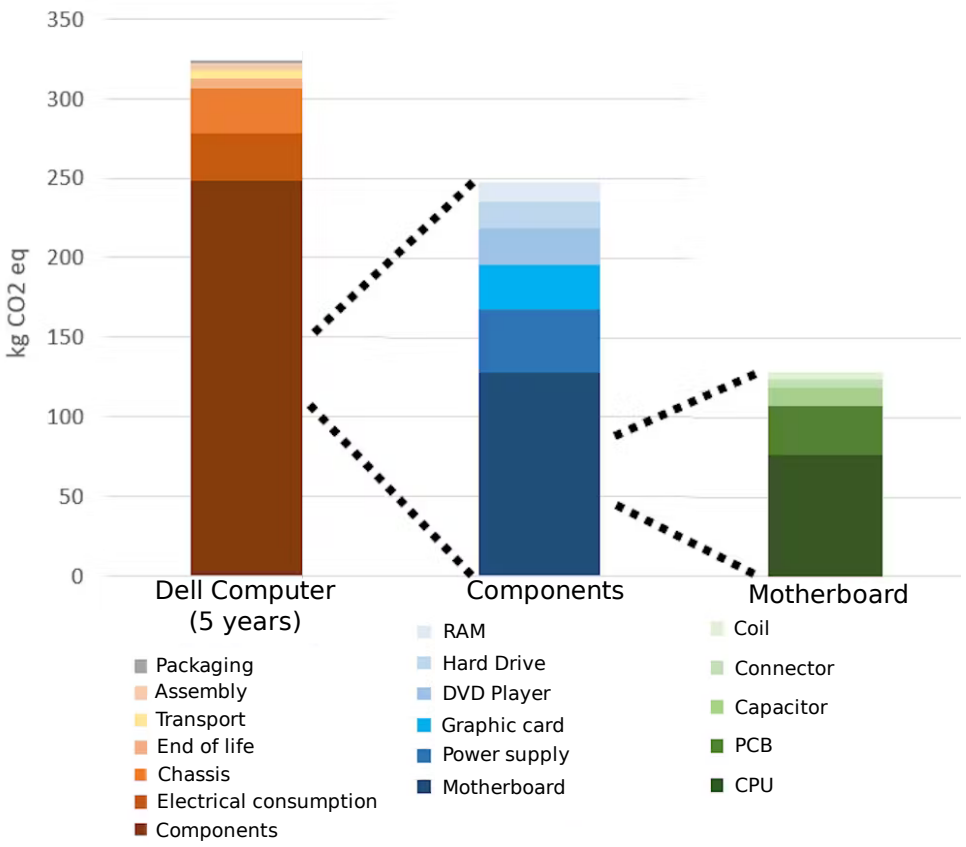

Figure A.1 – Émissions en équivalents CO2 pour un ordinateur Dell sur 5 ans [Lou+23](extrait de [Pel+24]).

afin de réduire l'impact environnemental. Par exemple, augmenter la recyclabilité d'un produit réduit son impact en fin de vie. Augmenter la durée de vie du produit réduit son impact pendant son utilisation et globalement. Enfin, réduire la consommation énergétique d'un produit réduit son impact pendant son utilisation. Néanmoins, ces recommandations vont à l'encontre de la tendance actuelle des applications informatiques et électroniques, comme présenté dans la section suivante.

A.1.2 Calculs à plus grande échelle

Au fil des ans, la complexité de nombreuses applications a considérablement augmenté. Cette augmentation de la complexité s'explique par les performances accrues requises pour ces applications et les tâches de plus en plus complexes qu'elles exécutent. Un bon exemple de cette évolution est la croissance du domaine de l'apprentissage automatique. La

figure A.2, extraite de [Bia+18], représente la précision des réseaux neuronaux convolutifs (CNN) en fonction de leur nombre de paramètres et d'opérations en virgule flottante (FLOP). On observe une tendance à l'augmentation de la taille pour gagner en précision.

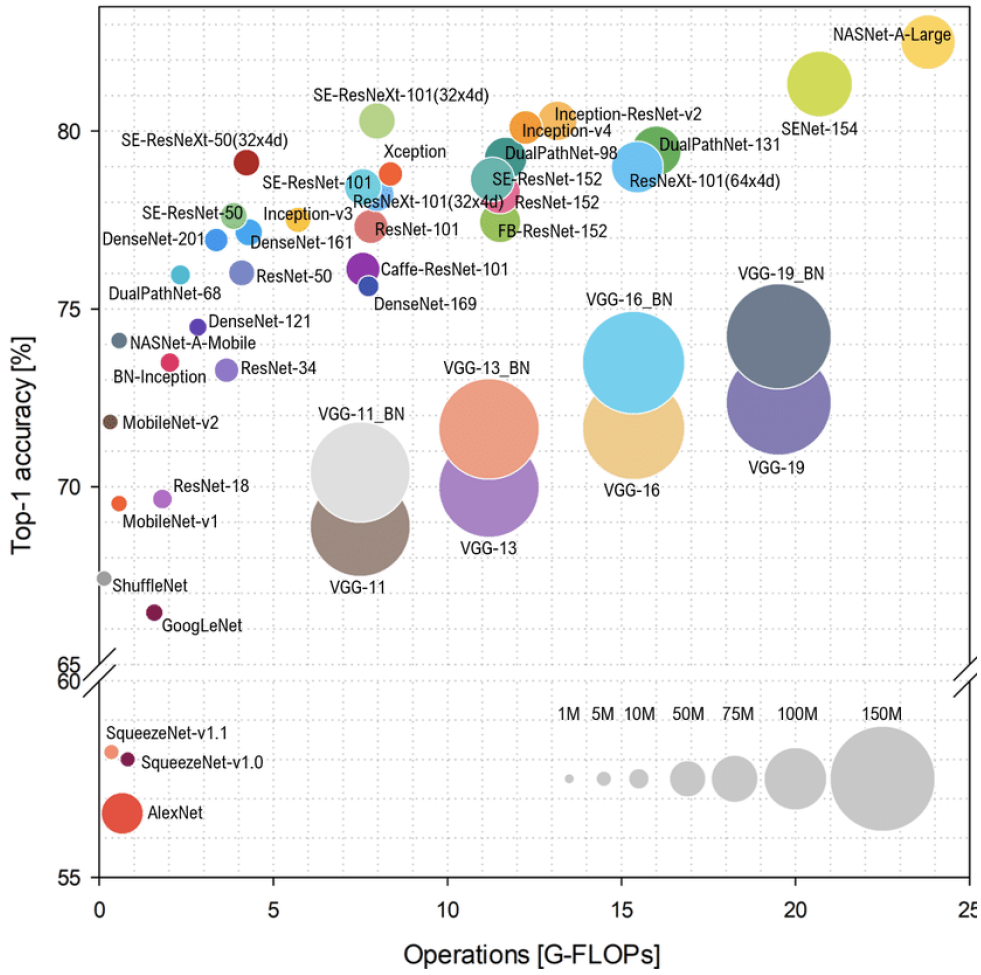

Figure A.2 – Diagramme comparatif de différents CNN en fonction du nombre d'opérations et de la précision [Bia+18].

La radioastronomie en est un autre bon exemple. Dans le cadre de leurs études, l'augmentation de la résolution des radiotélescopes entraîne une augmentation du volume de données produites. Cela incite le domaine à rechercher une optimisation économe en énergie [Cor+22].

Cette tendance à l'augmentation va à l'encontre des recommandations climatiques définies dans la section A.1.1. Elle entraîne également des coûts importants pour l'entreprise. Pour ces raisons, des solutions ont été étudiées afin de réduire les calculs des applications, comme présenté dans la section suivante.

A.1.3 Solutions pour la réduction des calculs

Dans [Eec23], les auteurs présentent une formule permettant de synthétiser les émissions mondiales provenant de sources humaines liées à l'utilisation d'appareils électroniques, F_{op}, en fonction du nombre de puces produites, C ; de l'efficacité énergétique (consommation d'énergie par puce pendant leur durée de vie), $\frac{E}{C}$; et de l'intensité carbone, $\frac{F}{E}$. Cela donne l'équation suivante :

$$F_{op} = C \times \frac{E}{C} \times \frac{F}{E} \tag{A.1}$$

Cette équation offre plusieurs moyens de réduire l'impact d'une application. Le premier consiste à réduire le nombre de puces ou produites, C. Ce paramètre relève principalement d'une décision politique et dépend des deux autres paramètres. En effet, une réduction de la consommation d'énergie entraîne souvent ce que l'on appelle l'effet rebond. L'effet rebond est une augmentation de la production liée à une réduction des coûts. Cependant, il est toujours important de réduire les autres variables afin de minimiser l'impact global si le nombre de puces reste le même. L'intensité carbone $\frac{F}{E}$ dépend du pays et du mode de production de l'électricité. L'efficacité énergétique, $\frac{E}{C}$, est davantage liée au domaine de l'ingénierie électronique. Il existe plusieurs méthodes pour améliorer l'efficacité énergétique, en optimisant les logiciels et le matériel.

Plus précisément, dans le cadre de ces travaux de recherche, le domaine du calcul approximatif a été exploré. Le calcul approximatif consiste à ajuster la tolérance de qualité d'une application afin de réduire la consommation d'énergie et d'accélérer le traitement. Nous avons exploré cette méthode en raison de son potentiel à être généralisée à de nombreuses applications grâce à des outils automatisés. Le concept de calcul approximatif a été largement étudié au cours des deux dernières décennies et s'est concrétisé par des outils commerciaux de conception assistée par ordinateur (CAO).

Par exemple, WeDoLow, une entreprise française fondée par Justine Bonnot, qui exploite plus de 20 ans de recherche dans des institutions telles que l'IETR et l'Inria, vise à fournir un cadre d'optimisation pour améliorer l'efficacité et réduire le coût de mise en œuvre des systèmes embarqués C/C++ [WeD]. Elle se concentre sur plusieurs axes, tels que la mise en œuvre de l'optimisation SIMD, la réalisation d'approximations polynomiales, l'optimisation de la taille des vecteurs pour réduire la mémoire et la quantification des variables. Son outil effectue des analyses statiques et dynamiques du code afin de déterminer les optimisations potentielles et la plage des variables.

En tant que partenaire de cette thèse, les outils analytiques de WeDoLow ont été utilisés sur l'application pour effectuer l'analyse et la transformation du code dans le cadre du processus d'optimisation. L'outil analytique utilisé est un compilateur source-à-source. Leurs outils visent à être intégrés dans les flux de travail d'autres entreprises qui développent des applications, dans le but d'augmenter les performances des applications et de réduire leur consommation énergétique. Ces outils ont été utilisés pour l'optimisation en virgule fixe et seront utilisés pour nos tests de performance et nos contributions. Un facteur crucial pour ce type d'outil de CAO est le temps d'optimisation. Ce temps d'optimisation doit être raisonnable afin que l'outil soit pleinement adopté par les développeurs. Cette thèse explorera les méthodes permettant de réduire ce temps d'optimisation.

A.2 Objectifs et Contributions

Cette thèse se concentre sur le développement d'outils visant à réduire le coût de mise en œuvre des applications génériques, avec une attention particulière portée à la consommation d'énergie. Le développement de ces outils repose sur l'idée de réduire la consommation d'énergie des puces tout au long de leur cycle de vie. Cela fait référence à l'efficacité énergétique $\frac{E}{C}$ dans l'équation A.1. Proposer des outils génériques complets facilite la mise en œuvre de la réduction de la consommation d'énergie dans l'ensemble du flux de travail d'un produit. Les intégrer dans des outils largement utilisés, tels que l'intégration continue, pourrait conduire à leur adoption généralisée dans l'industrie. Cependant, les outils génériques limitent la méthodologie utilisée. Toutes les méthodes de calcul approximatif ne peuvent pas être transformées en méthodes génériques. Par exemple, la méthode consistant à réduire la fréquence ou la tension de fonctionnement pour réduire les coûts énergétiques, telle que décrite dans [MYD20], dépend de l'architecture. Dans ce contexte, cette thèse se concentre sur la quantification et le raffinement de la précision.

Le processus de raffinement de précision vise à optimiser la longueur des mots de données. La réduction de la longueur des mots de données ou la modification du type de représentation peut réduire la consommation d'énergie [Jou+21]. Cependant, cela présente l'inconvénient d'introduire une erreur entre la valeur réelle et la valeur représentée, ce qui peut entraîner une dégradation de la qualité à la sortie de l'application. L'impact de toute modification sur cette dégradation de la qualité a été étudié. Certains outils [Bon+22] ont été développés à l'aide d'une représentation mathématique afin de déterminer la propagation des erreurs dues à la quantification dans un algorithme et son impact sur l'ensemble

de l'application. Ces méthodes sont très efficaces pour estimer la dégradation de la qualité.

Cependant, ces méthodes sont difficiles à étendre aux opérations non lisses (opérations non différentiables). Ce type d'opérations est de plus en plus répandu, en particulier avec l'essor des applications d'apprentissage automatique. Cette thèse se concentre sur le développement d'outils génériques permettant d'évaluer l'impact de la quantification à l'aide de simulations Monte-Carlo.

Les approches basées sur la simulation consistent à comparer un modèle de référence idéal avec sa version quantifiée [Bon+22]. En appliquant la même entrée aux deux modèles, il est possible de comparer les deux versions de la sortie du modèle. En effectuant cette comparaison avec plusieurs valeurs pour les entrées, il est possible d'identifier une tendance représentant la dégradation due à la quantification. Bien que cette méthode soit plus généralisable, obtenir une estimation fiable nécessite de nombreuses valeurs pour les entrées et entraîne un temps de simulation long. Cette longue durée de simulation est un problème, en particulier lorsqu'elle est mise à l'échelle d'un processus d'optimisation complet qui explore l'espace de conception de la longueur des mots.

Pour optimiser une application à l'aide de la simulation, il est important d'explorer un grand nombre de configurations, N_C. Chaque configuration nécessite de tester N_S entrées, et chaque simulation prend un certain temps, t_S. Le temps d'optimisation total, t_{opt}, peut alors être exprimé comme suit :

$$t_{opt} = N_C \times N_S \times t_s \tag{A.2}$$

La réduction de chacun de ces paramètres permettra de réduire le temps d'optimisation. Le paramètre N_C dépend de la méthode utilisée pour explorer l'espace de conception. Cela dépend de l'algorithme sélectionné pour l'optimisation et du nombre de variables à optimiser. Le paramètre N_S dépend de la manière dont la qualité est estimée et du niveau de confiance souhaité dans l'estimation. Le paramètre t_s dépend de la méthode utilisée pour simuler l'impact d'une précision limitée. L'objectif de cette thèse est d'accélérer le processus d'optimisation pour le raffinement de la précision. Le paramètre t_s a été exploré dans la littérature [SIE22; DK24], mais son champ d'application est davantage lié aux outils de compilation et à la synthèse. Cette thèse se concentrera sur l'exploration de N_S, le nombre d'échantillons d'entrée, et de N_C, le nombre de configurations.

Dans cette thèse, afin d'accélérer le processus d'optimisation pour le raffinement de précision, les contributions suivantes ont été proposées.

Amélioration de la précision pour les nombres à virgule flottante personnalisés. Pour la première contribution, nous proposons une approche qui optimise la longueur des mots de données pour les applications utilisant un type de données à virgule flottante personnalisé. Cette approche détermine la longueur des mots de l'exposant et de la mantisse pour chaque donnée. Elle réduit le nombre de configurations à explorer, N_C, en analysant la plage dynamique des données. À partir de cette analyse, une limite est déterminée pour la longueur de mot de l'exposant. Ensuite, le problème d'optimisation est décomposé en quatre stratégies d'optimisation. Ces stratégies répartissent l'optimisation entre la réduction des longueurs de mot de l'exposant et de la mantisse. Ces stratégies ont été testées sur trois applications différentes afin d'estimer le compromis entre le coût de mise en œuvre et la qualité.

Échantillonnage stratifié pour accélérer l'évaluation de la dégradation de la qualité. Les deux autres contributions explorent des outils statistiques permettant de réduire le nombre d'échantillons N_S utilisés pour évaluer la dégradation de la qualité due à la quantification. La deuxième contribution exploite la méthode d'échantillonnage stratifié. Cette méthode regroupe les échantillons présentant un comportement similaire afin de réduire la variance globale de la qualité estimée pour la configuration. Grâce à une analyse préalable, cette méthode permet de réduire d'un ordre de grandeur le nombre d'échantillons nécessaires pour obtenir un intervalle de confiance similaire.

Échantillonnage préférentiel pour accélérer l'évaluation de la dégradation de la qualité. Comme la contribution précédente, cette troisième contribution vise à réduire le nombre d'échantillons N_S utilisés pour évaluer la dégradation de la qualité due à la quantification. Cette contribution est basée sur l'échantillonnage préférentiel, qui modifie la loi utilisée pour prélever des échantillons afin d'obtenir des échantillons plus informatifs.

A.3 Plan

Cette thèse est divisée en deux parties. La première partie fournit les informations de base nécessaires. Le chapitre 2 présente en détail les concepts d'informatique approximative et de précision. Nous présentons les méthodes issues de la littérature pour les trois niveaux du calcul approximatif : le niveau matériel, le niveau logiciel et le niveau des données. Nous nous concentrons ensuite davantage sur le niveau des données

en décrivant les méthodes de représentation des données. Le chapitre 3 présente divers outils d'optimisation de la précision. Dans ce chapitre, nous présentons deux méthodes d'estimation de l'erreur de quantification : l'estimation analytique et l'estimation basée sur la simulation.

La deuxième partie de la thèse présente les contributions proposées au cours de ces travaux de recherche. Le chapitre 4 présente la première contribution. L'approche proposée, qui optimise la longueur des mots de données pour les applications utilisant un type de données à précision flottante personnalisée, y est décrite. Après la présentation des caractéristiques de la précision flottante personnalisée, la méthode proposée est expliquée et les quatre stratégies d'optimisation permettant de déterminer la longueur des mots de l'exposant et de la mantisse sont détaillées. Enfin, les résultats expérimentaux sont fournis et les performances des différentes stratégies sont comparées. Cette contribution a d'abord été publiée dans DASIP 2024, puis étendue et soumise au Journal of System Architecture.

Le chapitre 5 présente la deuxième contribution. La méthode proposée, basée sur l'échantillonnage stratifié, réduit la variance globale de l'estimation de la qualité de sortie en regroupant les échantillons présentant un comportement similaire. Après la présentation de l'intervalle de confiance sur l'estimation de la qualité dans le contexte des simulations Monte-Carlo, la méthode proposée, basée sur l'échantillonnage stratifié, est décrite. Différentes mesures pour regrouper les données d'entrée sont évaluées. Enfin, les résultats expérimentaux sont présentés et l'efficacité de l'approche proposée est évaluée pour les différentes mesures de regroupement des entrées. Cette contribution a été publiée dans Design & Diagnostics of Electronic Circuits & Systems (DDECS) 2025.

Le chapitre 6 présente la troisième contribution. La méthode proposée, basée sur l'échantillonnage préférentiel, modifie la loi d'échantillonnage afin d'obtenir des échantillons plus informatifs. Après la présentation de la méthode proposée, les différentes fonctions de densité de probabilité sélectionnées pour modifier la loi d'échantillonnage sont détaillées. Ensuite, un algorithme d'échantillonnage préférentiel adaptatif est proposé pour estimer la qualité. Enfin, les résultats expérimentaux sont présentés et les performances des approches proposées sont évaluées.

Enfin, dans le chapitre 7, nous concluons cette thèse et proposons des perspectives de recherche.

BIBLIOGRAPHY

[08] *IEEE Standard for Floating-Point Arithmetic*, Aug. 2008.

[Aa22] A.Kuzmin and et al, *FP8 Quantization: The Power of the Exponent*, tech. rep., Aug. 2022.

[ACC04] ACCELERA Systems Initiative, *systemc.org*, 2004, URL: https://systemc.org/.

[ACN14] Roberto Airoldi, Fabio Campi, and Jari Nurmi, « Approximate computing for complexity reduction in timing synchronization », *in*: *EURASIP Journal on Advances in Signal Processing* 2014 (2014).

[Ahm+23] Amjed Abbas Ahmed et al., « Design of Convolutional Neural Networks Architecture for Non-Profiled Side-Channel Attack Detection », *in*: *Elektronika ir Elektrotechnika* (2023).

[al17] B.Barrois et al, « Customizing fixed-point and floating-point arithmetic — A case study in K-means clustering », *in*: *International Workshop on Signal Processing Systems (SiPS)*, Oct. 2017, pp. 1–6.

[Ali+20] Milad Alizadeh et al., « Gradient Regularization for Quantization Robustness », *in*: *arXiv: Learning* (2020).

[Bar17] Benjamin Barrois, « Methods to Evaluate Accuracy-Energy Trade-Off in Operator-Level Approximate Computing », *in*: 2017.

[BDM19] Justine Bonnot, Karol Desnos, and Daniel Menard, « Accuracy Evaluation Based on Simulation for Finite Precision Systems Using Inferential Statistics », *in*: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, ISSN: 2379-190X, May 2019, pp. 1508–1512, DOI: 10.1109/ICASSP.2019.8682631.

[Bia+18] Simone Bianco et al., « Benchmark Analysis of Representative Deep Neural Network Architectures », *in*: *IEEE Access* 6 (2018), pp. 64270–64277, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2018.2877890.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville, « Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation », *in*: *ArXiv* abs/1308.3432 (2013).

[BMD20] Justine Bonnot, Daniel Menard, and Karol Desnos, « Fast Kriging-based Error Evaluation for Approximate Computing Systems », *in*: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ISSN: 1558-1101, Mar. 2020, pp. 1384–1389, DOI: 10.23919/DATE48585.2020.9116320, URL: http://ieeexplore.ieee.org/abstract/document/9116320.

[BMM13] Mathieu Balesdent, Jérôme Morio, and Julien Marzat, « Kriging-based adaptive Importance Sampling algorithms for rare event estimation », *in*: *elsevier* (2013).

[BMP06] Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion, « The design of the Boost interval arithmetic library », *in*: *Theor. Comput. Sci.* 351 (2006), pp. 111–118.

[BMS22] Alberto Bosio, Daniel Ménard, and Olivier Sentieys, eds., *Approximate Computing Techniques: From Component- to Application-Level*, en, Cham: Springer International Publishing, 2022, ISBN: 978-3-030-94704-0 978-3-030-94705-7, DOI: 10.1007/978-3-030-94705-7.

[BNM16] Justine Bonnot, Erwan Nogues, and Daniel Menard, « New non-uniform segmentation technique for software function evaluation », *in*: *2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2016, pp. 131–138, DOI: 10.1109/ASAP.2016.7760782.

[Bon+18] Justine Bonnot et al., « Cassis: Characterization with Adaptive Sample- Size Inferential Statistics Applied to Inexact Circuits », *in*: *2018 26th European Signal Processing Conference (EUSIPCO)* (2018), pp. 677–681.

[Bon+19] Justine Bonnot et al., « Adaptive simulation-based framework for error characterization of inexact circuits », *in*: *Microelectronics Reliability* (2019).

[Bon+22] Justine Bonnot et al., « Approximate Computing at the Algorithmic Level », en, *in*: *Approximate Computing Techniques*, Springer, Cham, 2022, pp. 109–142, ISBN: 978-3-030-94705-7, DOI: 10.1007/978-3-030-94705-7_5, URL: https://link.springer.com/chapter/10.1007/978-3-030-94705-7_5.

[Boy11] Sarah Boyd, « Life-Cycle Assessment of Semiconductors », *in*: 2011.

[Bru10] Bruno Tuffin, *La Simulation de Monte-Carlo*, Lavoisier, Méthodes Stochastiques Appliquées, 2010, ISBN: 978-2-7462-2551-3.

[Bug+17] Monica F. Bugallo et al., « Adaptive Importance Sampling: The past, the present, and the future », en, *in*: *IEEE Signal Processing Magazine* 34.*4* (July 2017), pp. 60–79, ISSN: 1053-5888, 1558-0792, DOI: 10.1109/MSP.2017.2699226, URL: https://ieeexplore.ieee.org/document/7974876/.

[Caf+10] Gabriel Caffarena et al., « SQNR Estimation of Fixed-Point DSP Algorithms », *in*: *EURASIP Journal on Advances in Signal Processing* 2010 (2010), pp. 1–12.

[Cam+18] Vincent Camus et al., « Design of Approximate Circuits by Fabrication of False Timing Paths: The Carry Cut-Back Adder », *in*: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8 (2018), pp. 746–757.

[Cat+] Daniele Cattaneo et al., « Architecture-aware Precision Tuning with Multiple Number Representation Systems », en, *in*: *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA: IEEE, pp. 673–678, ISBN: 978-1-6654-3274-0, DOI: 10.1109/DAC18074.2021.9586303, URL: https://ieeexplore.ieee.org/document/9586303/.

[CCL99] George A. Constantinides, Peter Y. K. Cheung, and Wayne W. C. Luk, « Truncation noise in fixed-point SFGs [digital filters] », *in*: *Electronics Letters* 35 (1999), pp. 2012–2014.

[Cho+18] Jungwook Choi et al., « PACT: Parameterized Clipping Activation for Quantized Neural Networks », *in*: *ArXiv* abs/1805.06085 (2018).

[CKK19] Yoni Choukroun, Eli Kravchik, and Pavel Kisilev, « Low-bit Quantization of Neural Networks for Efficient Inference », *in*: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019), pp. 3009–3018.

[CM16] Gabriel Caffarena and Daniel Menard, « Quantization Noise Power Estimation for Floating-Point DSP Circuits », *in*: *IEEE Transactions on Circuits and Systems II: Express Briefs* 63.*6* (June 2016), Conference Name: IEEE Transactions on Circuits and Systems II: Express Briefs, pp. 593–597, ISSN: 1558-3791, DOI: 10.1109/TCSII.2016.2530802.

[Com25] European Commision, *Rolling Plan 2025 | Interoperable Europe Portal*, 2025, URL: https://interoperable-europe.ec.europa.eu/collection/rolling-plan-ict-standardisation/rolling-plan-2025.

[Cor+22] Stefano Corda et al., « Reduced-Precision Acceleration of Radio-Astronomical Imaging on Reconfigurable Hardware », *in*: *IEEE Access* 10 (2022), pp. 22819–22843.

[CP13] Manik Chugh and Behrooz Parhami, « Logarithmic arithmetic as an alternative to floating-point: A review », *in*: *2013 Asilomar Conference on Signals, Systems and Computers*, ISSN: 1058-6393, Nov. 2013, pp. 1139–1143, DOI: 10.1109/ACSSC.2013.6810472.

[De +19] Florent De Dinechin et al., « Posits: the good, the bad and the ugly », en, *in*: *Proceedings of the Conference for Next Generation Arithmetic 2019*, Singapore Singapore: ACM, Mar. 2019, pp. 1–10, ISBN: 978-1-4503-7139-1, DOI: 10.1145/3316279.3316285.

[dév25] Commissariat général au développement, *Engagements de l Union europeenne*, 2025, URL: https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/2025/chiffres-cles-du-climat/fr/20-engagements-de-l-union-europeenne.php.

[DK24] Florent de Dinechin and Martin Kumm, *Application-Specific Arithmetic*, Springer, 2024.

[DSC17] Roberto Dicecco, Lin Sun, and Paul Chow, « FPGA-based training of convolutional neural networks with a reduced precision floating-point library », *in*: *2017 International Conference on Field Programmable Technology (ICFPT)* (2017), pp. 239–242.

[DT05] F. de Dinechin and A. Tisserand, « Multipartite table methods », *in*: *IEEE Transactions on Computers* 54.*3* (2005), pp. 319–330, DOI: 10.1109/TC.2005.54.

[Eec23] Lieven Eeckhout, « Kaya for Computer Architects: Toward Sustainable Computer Systems », *in*: *IEEE Micro* 43 (2023), pp. 9–18.

[Fa21] F.Ferrandi and et al, « Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications », *in*: *2021 58th ACM/IEEE Design Automation Conference (DAC)* (2021), pp. 1327–1330.

[FAG19] Alexander Finkelstein, Uri Almog, and Mark Grobman, « Fighting Quantization Bias With Bias », *in*: *ArXiv* abs/1906.03193 (2019).

[FCF23] Michele Fiorito, Serena Curzel, and Fabrizio Ferrandi, « TrueFloat: A Templatized Arithmetic Library for HLS Floating-Point Operators », en, *in*: *Embedded Computer Systems: Architectures, Modeling, and Simulation*, ed. by Cristina Silvano, Christian Pilato, and Marc Reichenbach, Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2023, pp. 486–493, ISBN: 978-3-031-46077-7, DOI: 10.1007/978-3-031-46077-7_35.

[Fer+21] Fabrizio Ferrandi et al., « Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications », en, *in*: *2021 58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA: IEEE, Dec. 2021, pp. 1327–1330, ISBN: 978-1-6654-3274-0, DOI: 10.1109/DAC18074.2021.9586110, URL: https://ieeexplore.ieee.org/document/9586110/.

[Fra23] Parlement Français, *LES OBJECTIFS DE LA POLITIQUE ÉNERGÉTIQUE (Articles L100-1 A à L100-5) - Légifrance*, 2023, URL: https://www.legifrance.gouv.fr/codes/section_lc/LEGITEXT000023983208/LEGISCTA000023985174?init=true&page=1&query=L.+100-4&searchField=ALL&tab_selection=all&anchor=LEGIARTI000047717642#LEGIARTI000047717642.

[Fru+15] Fabio Frustaci et al., « SRAM for Error-Tolerant Applications With Dynamic Energy-Quality Management in 28 nm CMOS », *in*: *IEEE Journal of Solid-State Circuits* 50 (2015), pp. 1310–1323.

[FS04] Luiz Henrique de Figueiredo and Jorge Stolfi, « Affine Arithmetic: Concepts and Applications », *in*: *Numerical Algorithms* 37 (2004), pp. 147–158.

[GCa22] G.Caffarena, « Wordlength Optimization of Fixed-Point Algorithms », *in*: *Approximate Computing Techniques: From Component-to Application-Level*, Springer, 2022, pp. 261–284.

[Gho+21] Amir Gholami et al., *A Survey of Quantization Methods for Efficient Neural Network Inference*, en, June 2021, URL: http://arxiv.org/abs/2103.13630.

[Glo89a] F. W. Glover, « Tabu Search - Part I », *in*: *INFORMS J. Comput.* 1 (1989), pp. 190–206.

[Glo89b] F. W. Glover, « Tabu Search - Part II », *in*: *INFORMS J. Comput.* 2 (1989), pp. 4–32.

[GML04] A.A. Gaffar, O. Mencer, and W. Luk, « Unifying bit-width optimisation for fixed-point and floating-point designs », *in*: *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2004, pp. 79–88, DOI: 10.1109/FCCM.2004.59.

[GT13] Carl Graham and Denis Talay, *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*, en, vol. 68, Stochastic Modelling and Applied Probability, Berlin, Heidelberg: Springer, 2013, ISBN: 978-3-642-39362-4 978-3-642-39363-1, DOI: 10.1007/978-3-642-39363-1, URL: https://link.springer.com/10.1007/978-3-642-39363-1.

[Gup+11] Vaibhav Gupta et al., « IMPACT: IMPrecise adders for low-power approximate computing », *in*: *IEEE/ACM International Symposium on Low Power Electronics and Design* (2011), pp. 409–414.

[GY17] John L Gustafson and Isaac Yonemoto, « Beating Floating Point at its Own Game: Posit Arithmetic », en, *in*: (2017), DOI: 10.14529/jsfi170206.

[Hei24] Reinout Heijungs, *Probability, Statistics and Life Cycle Assessment: Guidance for Dealing with Uncertainty and Sensitivity*, Cham: Springer International Publishing, 2024, ISBN: 978-3-031-49316-4 978-3-031-49317-1, DOI: 10.1007/978-3-031-49317-1, URL: https://link.springer.com/10.1007/978-3-031-49317-1.

[HL13] Thibault Hilaire and Benoit Lopez, « Reliable implementation of linear filters with fixed-point arithmetic », *in*: *SiPS 2013 Proceedings* (2013), pp. 401–406.

[HLR11] Jiawei Huang, John Lach, and Gabriel Robins, « Analytic error modeling for imprecise arithmetic circuits », *in*: *Proc. SELSE* 51 (2011), pp. 64–69.

[HLR12] Jiawei Huang, John C. Lach, and Gabriel Robins, « A methodology for energy-quality tradeoff using imprecise hardware », *in*: *DAC Design Automation Conference 2012* (2012), pp. 504–509.

[HS02] Reinout Heijungs and Sangwon Sun, « The computational structure of life cycle assessment », *in*: *The International Journal of Life Cycle Assessment* 7 (2002), p. 314.

[HS21] Van-Phu Ha and Olivier Sentieys, « Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning », *in*: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France: IEEE, Feb. 1, 2021, pp. 1542–1547, ISBN: 978-3-9819263-5-4, DOI: 10.23919/DATE51398.2021.9474209, URL: https://ieeexplore.ieee.org/document/9474209/.

[Hua+21] Yicun Hua et al., « A Survey of Evolutionary Algorithms for Multi-Objective Optimization Problems With Irregular Pareto Fronts », *in*: *IEEE/CAA Journal of Automatica Sinica* 8 (2021), pp. 303–318.

[HYS20] Van-Phu Ha, Tomofumi Yuki, and Olivier Sentieys, « Towards Generic and Scalable Word-Length Optimization », en, *in*: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France: IEEE, Mar. 2020, pp. 1668–1673, ISBN: 978-3-9819263-4-7, DOI: 10.23919/DATE48585.2020.9116564, URL: https://ieeexplore.ieee.org/document/9116564/.

[Jou+21] Norman P. Jouppi et al., « Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product », en, *in*: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Valencia, Spain: IEEE, June 2021, pp. 1–14, ISBN: 978-1-6654-3333-4, DOI: 10.1109/ISCA52012.2021.00010, URL: https://ieeexplore.ieee.org/document/9499913/.

[KAK18] S. Karen Khatamifard, Ismail Akturk, and Ulya R. Karpuzcu, « On Approximate Speculative Lock Elision », *in*: *IEEE Transactions on Multi-Scale Computing Systems* 4 (2018), pp. 141–151.

[KGE11] Parag Kulkarni, Puneet Gupta, and Milos D. Ercegovac, « Trading Accuracy for Power with an Underdesigned Multiplier Architecture », *in*: *2011 24th Internatioal Conference on VLSI Design* (2011), pp. 346–351.

[KMH25] Taiki Kitazawa, Shohei Matsumoto, and Yuichi Hayashi, « Robustness Evaluation of Software-Jamming Countermeasure against Multivariate and Nonlinear Analysis », *in*: *2025 International Symposium on Electromagnetic Compatibility – EMC Europe* (2025), pp. 881–886.

[Kul+22] Uday Kulkarni et al., « A Survey on Quantization Methods for Optimization of Deep Neural Networks », *in*: *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Dec. 2022, pp. 827–834, DOI: 10.1109/ICACRS55517.2022.10028742.

[Lee+05] Dong-U Lee et al., « MiniBit: bit-width optimization via affine arithmetic », en, *in*: *Proceedings of the 42nd annual conference on Design automation - DAC '05*, San Diego, California, USA: ACM Press, 2005, p. 837, ISBN: 978-1-59593-058-3, DOI: 10.1145/1065579.1065799, URL: http://portal.acm.org/citation.cfm?doid=1065579.1065799.

[Len+17] Cong Leng et al., « Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM », *in*: *AAAI Conference on Artificial Intelligence*, 2017.

[Lom+25] Guillaume Lomet et al., « Side-Channel Extraction of Dataflow AI Accelerator Hardware Parameters », *in*: *2025 IEEE 31st International Symposium on On-Line Testing and Robust System Design (IOLTS)* (2025), pp. 1–7.

[Lou+23] Philippe Loubet et al., « Life cycle assessment of ICT in higher education: a comparison between desktop and single-board computers », *in*: *The International Journal of Life Cycle Assessment* 28 (2023), pp. 255–273.

[Maz+17] Sana Mazahir et al., « Probabilistic Error Modeling for Approximate Adders », *in*: *IEEE Transactions on Computers* 66 (2017), pp. 515–530.

[MBC21] Saurav Maji, Utsav Banerjee, and Anantha P. Chandrakasan, « Leaky Nets: Recovering Embedded Neural Network Models and Inputs Through Simple Power and Timing Side-Channels—Attacks and Defenses », *in*: *IEEE Internet of Things Journal* 8 (2021), pp. 12079–12092.

[Mén+10] Daniel Ménard et al., « Quantization mode opportunities in fixed-point system design », *in*: *2010 18th European Signal Processing Conference* (2010), pp. 542–546.

[Mic+22] Paulius Micikevicius et al., *FP8 Formats for Deep Learning*, en, Sept. 2022.

[Mig17] Szymon Migacz, « 8-bit Inference with TensorRT », en, *in*: (2017), URL: https://www.cse.iitd.ernet.in/~rijurekha/course/tensorrt.pdf.

[MRS08] Daniel Ménard, Romuald Rocher, and Olivier Sentieys, « Analytical Fixed-Point Accuracy Evaluation in Linear Time-Invariant Systems », *in*: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55 (2008), pp. 3197–3208.

[MS02] Daniel Ménard and Olivier Sentieys, « Automatic evaluation of the accuracy of fixed-point algorithms », *in*: *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition* (2002), pp. 529–535.

[MYD20] Thibaut Marty, Tomofumi Yuki, and Steven Derrien, « Safe Overclocking for CNN Accelerators Through Algorithm-Level Error Detection », *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39 (2020), pp. 4777–4790.

[Nag+20] Markus Nagel et al., « Up or Down? Adaptive Rounding for Post-Training Quantization », *in*: *ArXiv* abs/2004.10568 (2020).

[Nev+21] Yarib Nevarez et al., « Accelerating Spike-by-Spike Neural Networks on FPGA With Hybrid Custom Floating-Point and Logarithmic Dot-Product Approximation », *in*: *IEEE Access* 9 (2021), Conference Name: IEEE Access, pp. 80603–80620, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2021.3085216.

[NMS11] H.-N. Nguyen, D. Menard, and O. Sentieys, « Novel algorithms for word-length optimization », *in*: *2011 19th European Signal Processing Conference*, 2011, pp. 1944–1948.

[Osb+07] W.G. Osborne et al., « Automatic Accuracy-Guaranteed Bit-Width Optimization for Fixed and Floating-Point Systems », *in*: *2007 International Conference on Field Programmable Logic and Applications*, ISSN: 1946-1488, Aug. 2007, pp. 617–620, DOI: 10.1109/FPL.2007.4380730.

[Pel+24] Maxime Pelcat et al., *Comment rendre l'électronique plus soutenable ?*, The Conversation, Mar. 24, 2024, DOI: 10.64628/AAK.9w9urup45, URL: http://theconversation.com/comment-rendre-lelectronique-plus-soutenable-221145.

[Rah+14] Amir Rahmati et al., « Refreshing Thoughts on DRAM : Power Saving vs . Data Integrity », *in*: 2014.

[Ren+12] Lakshminarayanan Renganarayana et al., « Programming with relaxed synchronization », *in*: *RACES '12*, 2012.

[SB04] Changchun Shi and Robert W. Brodersen, « A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs », *in*: *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)* 3 (2004), pp. III–373.

[Sid+11] Stelios Sidiroglou-Douskos et al., « Managing performance vs. accuracy trade-offs with loop perforation », en, *in*: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, Szeged Hungary: ACM, Sept. 2011, pp. 124–134, DOI: 10.1145/2025113.2025133.

[SIE22] SIEMENS EDA, *Algorithmic C Datatypes reference manual*, Aug. 2022.

[Sil86] Bernard W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.

[SM95] Debjit Das Sarma and David W. Matula, « Faithful bipartite ROM reciprocal tables », *in*: *Proceedings of the 12th Symposium on Computer Arithmetic* (1995), pp. 17–28.

[SM96] Ravindra Singh and Naurang Singh Mangat, « Stratified Sampling », *in*: *Elements of Survey Sampling*, Dordrecht: Springer Netherlands, 1996, pp. 102–144, ISBN: 978-94-017-1404-4, DOI: 10.1007/978-94-017-1404-4_5, URL: https://doi.org/10.1007/978-94-017-1404-4_5.

[Sur+15] Arjun Suresh et al., « Intercepting Functions for Memoization: A Case Study Using Transcendental Functions », *in*: *ACM Trans. Archit. Code Optim.* 12.*2* (June 2015), ISSN: 1544-3566, DOI: 10.1145/2751559, URL: https://doi.org/10.1145/2751559.

[Wal13] Sean Wallis, « Binomial Confidence Intervals and Contingency Tests: Mathematical Fundamentals and the Evaluation of Alternative Methods », en, *in*: *Journal of Quantitative Linguistics* 20.*3* (Aug. 2013), pp. 178–208, ISSN: 0929-6174, 1744-5035, DOI: 10.1080/09296174.2013.799918, URL: http://www.tandfonline.com/doi/abs/10.1080/09296174.2013.799918.

[WeD] WeDoLow, *WedoLow - Software engineering - Optimisation logiciel*, fr-FR, URL: https://www.wedolow.com/.

[Wid61] Bernard Widrow, « Statistical analysis of amplitude-quantized sampled-data systems », *in*: *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry* 79 (1961), pp. 555–568.

[Yao+21] Zhewei Yao et al., *HAWQV3: Dyadic Neural Network Quantization*, en, June 2021, URL: http://arxiv.org/abs/2011.10680.

[Zha+14] Qian Zhang et al., « ApproxIt: An approximate computing framework for iterative methods », *in*: *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, ISSN: 0738-100X, June 2014, pp. 1–6, DOI: 10.1109/DAC.2014.6881424, URL: https://ieeexplore.ieee.org/document/6881424.

[ZSB23] Hong Zhang, Lukai Song, and Guangchen Bai, « Active Kriging-Based Adaptive Importance Sampling for Reliability and sensitivity Analyses of Stator Blade Regulator », en, *in*: *Computer Modeling in Engineering & Sciences* 134.*3* (2023), pp. 1871–1897, ISSN: 1526-1506, DOI: 10.32604/cmes.2022.021880, URL: https://www.techscience.com/CMES/v134n3/49742.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Titre : Accéleration du temps d'optimisation de taille de données pour une contrainte de qualité à l'aide de méthodes statistiques

Mot clés : Calcul approximé – Quantification – Exploration d'espace de conception – Échantillonage préférentiel – Échantillonage stratifié

Résumé : L'explosion de la complexité algorithmique et de la taille des applications au cours des dernières années a apporté de nouveaux défis. Notamment avec l'évolution des applications de machine learning. Cela a causé une augmentation des temps de calculs et de l'espace mémoire nécessaire, et par conséquent de la consommation d'énergie. Une méthode pour réduire l'ensemble de ces paramètres est de réduire la taille des variables utilisées pour stocker les données. Cette méthode a pour désavantage d'impliquer une potentielle réduction de la qualité d'une application. Afin de pouvoir propager l'utilisation de cette méthode, il est important de définir des outils d'optimisation permettant de définir un coût minimal sous une contrainte de qualité. Dans cette thèse, nous proposons de développer des méthodes de quantification généraliste afin de réduire leur temps d'optimisation. Tout d'abord, nous proposons une méthode d'optimisation généraliste pour des nombres à virgule flottante de taille personnalisée. Ensuite, nous proposons une méthode généraliste d'estimation de dégradation de qualité. Cette méthode utilise les outils statistiques d'échantillonnage préférentiel ou d'échantillonnage stratifié pour obtenir des résultats de confiance en un temps réduit.

Title: Precision refinement acceleration using statistical methods for quality estimation

Keywords: Custom floating-point - Quantification - Wordlength Optimization - Importance sampling - Stratified sampling

Abstract: The explosion in algorithmic complexity and data volumes in recent years has brought new challenges. This is particularly true with the evolution of machine learning applications. This caused an increase in computation times and memory space requirements, and consequently in energy consumption. One method of reducing all of these parameters is to reduce the size of the variables used to store data. The disadvantage of this method is that it can potentially reduce the quality of an application. In order to promote the use of this method, it is important to define optimization tools that allow a minimum cost under a defined quality constraint. In this thesis, we propose to develop general-purpose quantization methods in order to reduce their optimization time. First, we propose a general optimization method for custom-sized floating-point numbers. Next, we propose a general method for estimating quality degradation. This method uses the statistical tools of importance sampling or stratified sampling to obtain reliable results in a shorter time.