UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Érico Marco Dias Alves Pereira

Aprendizado de Representações de Imagens usando Quantização Genética

Belo Horizonte - MG
2021

Érico Marco Dias Alves Pereira

Aprendizado de Representações de Imagens usando Quantização Genética

Versão final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Jefersson Alex dos Santos

Belo Horizonte - MG
2021

Érico Marco Dias Alves Pereira

Image Representation Learning through Genetic Quantization

Final version

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Jefersson Alex dos Santos

Belo Horizonte - MG
2021

Pereira, Érico Marco Dias Alves.

P436i Image representation learning through genetic quantization [manuscrito] / Érico Marco Dias Alves Pereira. - 2021.
xxix, 91 f. il.

Orientador: Jefersson Alex dos Santos.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.75-87.

.

1. Computação – Teses. 2. Aprendizado de representação – Teses. 3. Algoritmos evolucionários – Teses 4. Algoritmos genéticos – Teses. 5.Recuperação de imagems baseada em conteúdo– Teses.. I. Santos, Jefersson Alex dos.II. Universidade Federal de Minas Gerais; Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82.10(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Image Representation Learning through Genetic Quantization

ÉRICO MARCO DIAS ALVES PEREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. JEFERSSON ALEX DOS SANTOS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. RICARDO DA SILVA TORRES
Department of ICT and Natural Sciences - NTNU

PROF. JURANDY ALMEIDA
Instituto de Ciência e Tecnologia - Universidade Federal de São Paulo (UNIFESP)

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 12 de Janeiro de 2021.

To all of those who dream of a brighter future.

Acknowledgments

I would like to register my sincere gratitude to all those that contributed to my education during the Master's and helped me to successfully get through this graduation process.

I would like to thank my advisor, Prof. Jefersson dos Santos, for his guidance, conduct and understanding as well as providing propitious environment for my academic and scientific education. I also thank Prof. Ricardo Torres for his contribution to my research and his elucidating teachings. Furthermore, I thank Prof. Jurandy Almeida e Gisele Pappa for their contribution to this thesis and all the professors from PPGCC-UFMG who transmitted to me their knowledge.

I immensely thank my parents Willianice Pereira and Veríssimo Pereira for all their unconditional support during my whole years of education, allowing me to pursue trajectories of my own will with courage and peace of mind. I also thank the members of my family for their support and guidance. My heartfelt gratitude to my friends from GEIR who have accompanied me for years and shared with me their love, friendship and learning.

I thank my friends from CEFET-MG - Jonatas Cavalcante, Geovane Fonseca, Paula Jácome, Fernanda Duarte and Bruno Maciel - who have brought joy to my life since we met. I also thank my friends from the Light and Escada cells for their love and beneficial impact in my life.

I thank all my colleagues and friends from PATREO and SENSE who contributed to my learning and education. My heartfelt gratitude to Edemir Ferreira, Hugo Oliveira, Matheus Barros, Gabriel Machado, Pedro Gama, João Macedo, Matheus Brito, Caio Cesar, Jéssica Sena, Jesimon Barreto, Carlos Caetano, Victor Hugo, Arthur Jordão, Rafael Prates, Rafael Vareto, Fernando Yamada, Maiko Lie and Antônio Nazaré for all their help and openness.

I am deeply grateful to Jesus for always guiding me with unconditional love and to God for the opportunity to live and complete this fruitful experience.

"Love your life,
perfect your life,
beautify all things in your life."
(Chief Tecumseh)

Resumo

Representações de imagens crucial importância crucial em sistemas de visão computacional pois codificam a informação intrínseca aos pixels e suas relações de uma maneira computacionalmente tratável, permitindo aos algoritmos aprender sobre o conteúdo visual das imagens e tomar decisões a partir disso. O aprendizado de representação de imagens visa fornecer um processo automatizado para a composição das representações otimizadas à uma dada tarefa de visão computacional. O estado-da-arte dessa área de pesquisa - que são as técnicas baseadas em *Deep Learning* - alcançou, nos últimos anos, grandes avanços na solução de problemas estudados há várias décadas pela comunidade de Inteligência Artificial e bateu recordes em diversas tarefas de reconhecimento de padrões. No entanto, essas técnicas geralmente apresentam alta complexidade computacional e demandam uma grande quantidade de recursos como memória de armazenamento, memória de trabalho, capacidade computacional e energia. Além disso, elas comumente requerem grandes conjuntos de dados rotulados a fim de produzir modelos eficazes. Motivados por essas desvantagens, combinamos três pilares para produzir representações com consumo eficiente de recursos: aprendizagem incremental, que otimiza representações sem construí-las do zero, evitando alta complexidade e grande consumo de recursos; algoritmos evolutivos, que fornecem uma otimização escalável, uma cobertura eficiente do espaço de busca e fácil adequação a problemas de otimização combinatória; e otimização de quantização, que é capaz de promover compactação sem reduzir o número de parâmetros. Nós abordamos duas classes essenciais do aprendizado de representações de imagens: representações *shallow* e *deep*. No estudo da primeira classe, propomos a otimização de representações *shallow* e introduzimos uma abordagem baseada em Algoritmo Genético que otimiza a quantização de cores de representações desenhadas manualmente para maior compactação e eficácia na tarefa executada. Avaliamos esta metodologia em tarefas de recuperação de imagens baseadas em conteúdo e obtivemos representações de tamanho menor com precisão significativamente melhor além de superar metodologias baseadas em *Deep Learning*. No estudo da segunda classe, estudamos a otimização de representações *deep* através de uma tarefa de compressão de redes neurais artificiais e propomos um método de quantização de precisão mista pós-treinamento para otimizar os pesos e ativações de modelos convolucionais usando uma busca baseada em Algoritmo Genético multi-objetivo. Avaliamos esta metodologia na tarefa de classificação de imagens usando o *dataset* Imagenet e obtivemos compressão com baixa perda de precisão através da quantização pós-treinamento. Os resultados sugerem que a otimização usando Algoritmo Genético é uma abordagem

promissora para futuras metodologias apresentando um aprendizado de representações altamente eficaz e com consumo eficiente de recursos.

Palavras-chave: Aprendizado de representação, extração de características, algoritmos evolucionários, algoritmo genético, recuperação de imagems baseada em conteúdo, classificação de imagem, quantização de rede neural, quantização de cor, quantização pós-treinamento.

Abstract

Image representations have crucial importance in computer vision systems as they encode the pixels inner and relational information in a computationally tractable form, allowing algorithms to reason about the visual content and take decisions about it. Image representation learning aims to provide an automatized process for composing the most appropriate representations for a given computer vision task. The state-of-the-art of this research area - Deep Learning-based techniques - has achieved, in recent years, major advances in solving problems studied for decades by the Artificial Intelligence community and beat records in several pattern-recognition tasks. However, they usually present high computational complexity and demand a huge amount of resources such as storage memory, working memory, computational power, and energy consumption. Furthermore, they typically require large sets of labeled data to produce effective models. Motivated by these disadvantages, we combine three factors in order to produce resource-efficient representations: incremental learning, that optimizes representations without constructing them from scratch avoiding complexity and high resource consumption; evolutionary algorithms, which provides scalable optimization, efficient search-space cover, and natural suitability for combinatorial problems; and quantization optimization, which often provides compaction without reducing the number of parameters. We address two important branches of image representation learning: shallow and deep representations. Regarding the former, we propose the optimization of shallow representations and introduce a Genetic-Algorithm based approach that optimizes the color-quantization of feature-engineered representations for improved effectiveness and compactness. We evaluated this methodology in content-based image retrieval tasks and obtained representations with significantly improved precision and reduced size besides surpassing deep-learning-based baselines. Regarding the latter, we study the optimization of deep representations through model compression and propose a post-training mixed-precision quantization method to optimize the weights and activations of convolutional neural models using a multi-objective Genetic-Algorithm search. We evaluated this methodology in image classification using Imagenet dataset and obtained compression in post-training quantization with small accuracy drops. Results confirm Genetic Algorithm optimization as a promising approach for highly effective and resource-efficient learning in future methodologies.

Keywords: Representation Learning. Feature Extraction. Evolutionary Algorithm. Genetic Algorithm. Content-Based Image Retrieval. Image Classification. Model Quantization. Color Quantization. Mixed-Precision. Post-training Quantization..

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

Data representations have crucial importance in intelligent systems as they encode data information in a tractable form. They allow pattern recognition algorithms to reason, learn, interpret and understand this information and, consequently, to take improved decisions about the tasks to perform. For instance, in computer vision, image representations condense the visual content into computable image features that encode information about color, texture, shape, semantic artifacts, etc., helping to grasp the inner and relational aspects of huge amounts of pixels. For that reason, much of the computational resources spent in deploying intelligent algorithms are employed in data processing pipelines whose main goal is to generate representations that can support an effective accomplishment of their tasks.

For many years, image representations were mostly a product of Feature Engineering - the process of using hand-designed feature extractors from raw data using expert knowledge about the task. However, it is labor-intensive and time-consuming for the user. Moreover, the domain knowledge required for performing this process hampers the effective applicability of machine learning techniques. Consequently, emerged the need for data representation pipelines that were less human-dependent.

In the last decade, Representation Learning, also referred to as Feature Learning, which is the process of using algorithms to learn features for a given task, has been employed to automatize the feature extraction step and ease the appliance of machine learning. Furthermore, it is a way to make progress towards Artificial Intelligence (AI). According to Bengio et al. [7], an AI must be capable of autonomously understand the world around us, and this will only be achieved if it can learn to identify and disentangle the underlying explanatory factors hidden in low-level sensory data.

The state-of-the-art of Representation Learning are methods based on Deep Learning [57], such as Deep Autoencoders [42, 4] and Convolution Neural Networks [89]. Deep-learning methods construct multiple levels of representation, obtained by composing simple non-linear modules called perceptrons and gouping them into layers. Each perceptron transforms the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. In recent years, they made major

advances in solutions to well-studied AI problems and beat records in several computer vision tasks.

However, Deep Learning methods present high computational complexity and demand a huge amount of resources such as storage memory, working memory, computational power, and energy consumption. Furthermore, they usually require large sets of labeled data to produce effective models and need specific expertise or training for properly design, optimize, and evaluate promising solutions. Consequently, with low availability of resources or when time is scarce, it is desirable to have at disposal alternative approaches that consume less and are easier to use.

Looking at previous feature-engineered methods (e.g. those based on color, texture or object shapes [83, 1]), they usually rely on simpler algorithms and do not depend on large datasets or computationally-demanding learning steps. On the other hand, their feature extractors are application-specific, being less generalizable; considerably dependent on human-labor; and typically present significantly worse results than their Deep Learning counterparts. In this work, we investigate the possibility of automatically optimizing feature-engineered representations to improve their generability, compactness, effectiveness, and human-dependency instead of learning new ones from scratch, which leads to the usage of more complex and costly methodologies.

In a similar direction, recent works have addressed the resource-consumption disadvantages of Deep-learning techniques through approaches of model compression [25, 78, 16], which include tensor decomposition, network pruning, efficient neural architecture design, knowledge distillation, network quantization and more. In the second part of this work, we focus on network quantization [54, 38] – an approach in which the deep model is compressed by reducing the bit-widths of weights and activations. The challenge of quantization is to reduce the resource requirements of the model without compromising its capabilities in performing the task.

Recently, Evolutionary Algorithms (e.g. Genetic Algorithm, Evolution Strategy, Genetic Programming) have shown signs of threatening the hegemony of deep neural networks. Novel evolutionary-based techniques have been approximating the performance of Deep-Learning-based counterparts and even surpassing them in some pattern recognition tasks, such as Reinforcement Learning [95]. Furthermore, they have been largely used alongside deep-learning performing meta-learning tasks [21] such as hyper-parameter search [43], architecture design [64, 102, 29] and model training [104, 47]. As for a fact, evolutionary-based approaches come as promising methodologies for less-costly learning pipelines.

In this work, we address two important branches of representation learning: shallow and deep representations. In the first part (Section 3), we propose the optimization of shallow representations and introduce a Genetic-Algorithm based approach that optimizes the color-quantization of feature-engineered representations for improved effectiveness and

compactness. We evaluated this methodology in content-based image retrieval tasks and obtained representations with significantly improved precision and reduced size and superior results against deep-learning-based baselines. In the second part (Section 4), we study the optimization of deep representations through model compression and propose a post-training mixed-precision quantization method to optimize the weights and activations of convolutional neural models using a multi-objective Genetic-Algorithm search. We evaluated this methodology in image classification using Imagenet dataset and obtained state-of-the-art compression in post-training quantization and small accuracy drops.

1.1 Research Questions

Considering the studies employed in the both parts, we aim to answer the following research questions:

I Can we produce improved representations by optimizing those that already exist (e.g. hand-designed representations and pre-trained Deep Learning models) instead of learning them from scratch?

II How well GA-based Representation Learning performs against the current state-of-the-art, i.e., Deep Learning methods?

III Is it possible to optimize already existing representations (e.g. hand-designed representations and pre-trained Deep Learning models) to be simultaneously more compact and more effective in their task?

1.2 Contributions

In summary, the main contributions of this work are:

General:

1. We investigate the use of incremental representation-learning approaches instead of learning representations from scratch.
2. We study the use of evolutionary algorithms, more precisely genetic algorithms, in image representation learning tasks.
3. We propose methodologies that address disadvantages of Deep Learning models (e.g. high resource consumption) (A) by proposing alternative techniques and (B) by improving them.
4. We introduce approaches for improving shallow and deep representations through quantization optimization.

Shallow Representations:

1. We show that different color quantizations impact the effectiveness performance of feature extractors;
2. We model the search for suitable color quantization using a soft computing apparatus based on the genetic algorithm;
3. We introduce two approaches for supervised representation learning capable of providing compact and more effective representations through color quantization optimization.

Deep Representations:

1. We design a population-based multi-objective solution for post-training mixed-precision quantization.
2. We introduce a per-channel quantization approach capable of providing state-of-the-art compactness for post-training quantization and the best trade-off between accuracy and compactness among post-training mixed-precision methods.

1.3 Publications

Some of the achieved results [85] have been published in the journal *Multimedia Tools and Applications (Springer)*. It is also worth to mention that preliminary results [135] of this work were presented at the *Workshop of Undergraduate Works (WUW)* within the *30th SIBGRAPI - Conference on Graphics, Patterns and Images*, where the work was awarded *Honorable Mention*.

1.4 Outline

This work is organized as follows. Section 2 describes some important background concepts regarding image representations, Genetic Algorithms and quantization. Section 3 presents the study about the optimization of shallow image representations and Section 4 about the deep ones. Section 5 presents the final conclusions and answers the research questions.

Chapter 2

Background Concepts

This section presents background concepts on feature extraction algorithms (Sections 2.1 and 2.2) and genetic algorithms (Section 2.3). The feature extraction algorithms described here refer to methods that are combined with the quantization scheme defined by GA in the experiments of Chapter 3.

2.1 Color Quantization-based Feature Extraction Algorithms

Border/Interior Classification [103] (BIC), is a simple and fast approach for feature extraction which presented prominent results in web image retrieval [83] and remote sensing image classification [28, 80]. This approach relies on an RGB color-space uniformly quantized in $4 \times 4 \times 4 = 64$ colors. After the quantization, the authors propose to apply a segmentation procedure, which classifies the image pixels according to a neighborhood criterion: a pixel is classified as interior if its 4-neighbours (right, left, top, and bottom) have the same quantized color; otherwise, it is classified as border. Then, two color histograms, one for border pixels and other for interior pixels, are computed and concatenated composing a 128-bin representation. In the end, the histograms undergo two normalizations: division by the maximum value, for image dimension invariance, and a transformation according to a discrete logarithmic function ($dLog$), aiming to smooth major discrepancies. When comparing BIC via L1 distance, it was observed that the dLog function is able to increase substantially the effectiveness of histogram-based CBIR approaches and also reduces by 50% the space required to represent a histogram.

Global Color Histogram [106] (GCH) is a widely used feature extractor that presents one of the simplest forms of encoding image information in a representation, a color histogram, which is basically the computation of the pixel frequencies of each color. It relies on the same uniformly quantized RGB color-space such as BIC and, consequently,

produces a feature vector of 64 bins. After the histogram computation, it undergoes a normalization by the max value in order to avoid scaling bias. Additionally, for the same reasons as for BIC, dLog normalization is also applied to the final histogram.

2.2 Autoencoders

An autoencoder [42] is a framework that employs representation learning by optimizing an encoding that reconstructs as well as possible the entry data. It is specified by a explicitly defined feature-extraction function f_θ, called encoder, which allows the computation of a representation $z = f_\theta(x)$ from a given input x, and a parametrised function g_θ, that maps the representation from feature space back to input space producing a reconstruction $r = g_\theta(x)$. The set of parameters θ of the encoder and decoder are learned simultaneously by reconstructing the original input x with the lowest possible discrepancy $L(x, r)$ between x and r, employing a optimization process that minimizes:

$$\Gamma_{AE}(\theta) = \sum_t L(x^{(t)}, g_\theta(f_\theta(x^{(t)}))) \tag{2.1}$$

where $x^{(t)}$ is a training sample.

It is crucial that an autoencoder presents good generalization, i.e., that the produced representations yield low reconstruction error for both train and test samples. For this purpose, it is important that the training criterion or the parametrisation prevents the auto-encoder from learning the identity function to the training samples, which presents zero reconstruction error. This is achieved by imposing different forms of regularisation in different versions of autoencoders. Regularized Autoencoders limit the representational capacity of z provoking a bottleneck effect that does not allow the autoencoder to reconstruct the whole input and forces it to learn more meaningful features. As a consequence, it is trained to reconstruct well the training samples and also present small reconstruction error on test samples, implying generalization.

The most common types of regularised autoencoders include: Sparse autoencoders [88, 79, 69], which limit capacity by imposing a sparsity constraint on the learnt representation of the data; Denoising autoencoders [113, 114], which has the objective of removing noise of an artificially corrupted input, i.e. learning to reconstruct the clean version from a corrupted data; Contractive Autoencoders [91], which penalize the sensitivity of learned features to input variations producing more robust features; and Variational Autoencoders [52], which learn probabilistic latent spaces in order to generate artificial samples.

2.2.1 Winner-Take-All Autoencoders

Winner-takes-all Autoencoders (WTA-AE) [70] are sparse autoencoders that employ two types of sparsity constraints:

- A spatial sparsity constraint, which, rather than reconstructing the input from all of the representational hidden units, selects the single largest value within each feature map, and set the rest to zero. This results in a sparse representation, whose sparsity level is the number of feature maps, and in a reconstruction, which uses only the active hidden units in the feature maps;
- A *winner-take-all* lifetime sparsity constraint, which maintains only the $k\%$ largest values of each feature map, and set others to zero, considering the values selected spatial sparsity within an entire mini-batch.

We choose WTA as baseline because it is one of the most robust and efficient Sparse Enconders – the most effective class of (non-generative) methods based on deep learning that are dedicated for feature extraction/representation learning. WTA autoencoders were capable of aiming at any target sparsity rate, training very fast compared to other sparse autoencoders, and efficiently training all hidden units even under very aggressive sparsity rates (e.g., 1%). Furthermore, the usage of its sparsity properties allows the train of non-symmetrical architectures (different sizes for encoder and decoder) reducing computation and data resource consumption.

2.3 Genetic Algorithms

This section describes two optimization algorithms that belong to the class of Genetic Algorithms (GAs): the first (Section 2.3.1) corresponds to the simplest version of the canonical Genetic Algorithm and aims to solve problems that have a single objective, and the second (Section 2.3.2) describes a version of GA that was designed to solve multi-objective problems and produces a population of solutions.

2.3.1 Simple Genetic Algorithm

GA is a bio-inspired optimization heuristic that mimics natural genetic evolution to search the optimal in a solution space [36]. It models potential solutions for the problem as individuals of a population and subjects them to an iterative process of combinations and transformations towards an improved population, i.e., a population with better solutions for the target problem.

At each step, GA probabilistically selects individuals from the current population, called parents, in an operation called tournament, in which individuals are grouped and only the best ones are selected. From this selection, GA exchanges genetic material of the individuals in order to produce new individuals of the next generation. This operation is known as cross-over. Some individuals are also selected to undergo a mutation operation, which consists in randomly changing small pieces of the individual representation. This new individual is also integrated into the new generation [22]. Typically a few of the best individuals of the population also compose the new one, a practice known as elitism. When a new generation is formed, its individuals are evaluated by means of a fitness function, which assesses the individual (solution) performance on the target problem. According to this function score, the algorithm selects the parent individuals that will generate the next population, simulating a natural selection process. At the end of the process, when the stopping condition is satisfied, the expected result is the best-performing individual, i.e., the one that best solves the target problem.

2.3.2 Multi-objective Genetic Algorithm

In multi-objective optimization problems, often there is no solution that is optimal in all objectives simultaneously. Consequently, the preferred solving approach is to find all non-dominating optimal solutions, i.e. the solutions which compose the optimal Pareto frontier. Among the possible Pareto-optimal points, the decision maker may want to select one point over the other depending on the situation; before taking any decision, he or she may want to know the other possible Pareto-optimal solutions.

A convenient method would be one that can find multiple Pareto-optimal solutions simultaneously so that decision makers may be able to choose the most appropriate solution for the current situation. Since genetic algorithms deal with a population of points instead of one point, multiple Pareto-optimal solutions can be captured in the population, in a single run. In this sense, Deb et al. [24] proposed Non-dominated Sorting Genetic

Algorithm (NSGA-II): a multi-objective GA that groups the population into Pareto frontiers and selects the superior groups to compose the next population, pushing the best frontier further and returning its solutions as final answer.

NSGA-II differs from a simple genetic algorithm only in the way the selection operator works. The crossover and mutation operators remain as usual. NSGA-II has a scheme for selecting new population based on two main operators: the Fast Non-dominated Sorting and a parameter-free metric for promotion of population diversity, called crowding distance (CD).

The Fast Non-dominated Sorting (detailed in Algorithm 1) consists in grouping a population P in r subgroups F_1, F_2, ..., F_r – which correspond to Pareto frontiers – such that $F_1 = \{$individuals of P that are not dominated by any other of $P\}$ (frontier of rank 1), $F_2 = \{$individuals of $P \setminus F_1$ that are not dominated by any other of $P \setminus F_1\}$ (frontier of rank 2), ..., $F_r = \{$individuals of $P \setminus F_1 \cup F_2 \cup \ldots \cup F_{r-1}$ that are not dominated by any other of $P \setminus F_1 \cup F_2 \cup \ldots \cup F_{r-1}$ $\}$ (frontier of rank r), where $P = F_1 \cup F_2 \cup \ldots \cup F_r$.

Algorithm 1 Fast Non-dominated Sorting

```
1   Let P be a population of individuals
2   Let S_p be the set of individuals dominated by p
3   Let H be a set of individuals
4   Let n_p and n_q be counters for the number of individuals that dominates p and q
5   For each individual p ∈ P do
6       For each individual q ∈ P do
7           If(p ≺ q) then                          if p dominates q then
8               S_p ← S_p ∪ q                       include q in S_q
9           Else If(q ≺ p) then                     if p is dominated by q then
10              n_p+ = 1                            increment n_p
11      End For
12      If(n_p = 0) then                            if no solution dominates p than
13          F_1 ← F_1 ∪ p                           p is a member of the first front
14  End For
15  i = 1
16  While(F_i ≠ ∅) do
17      H ← ∅
18      For each p ∈ F_i do                         for each member p in F_i
19          For each q ∈ S_p do                     modify each member from S_p
20              n_q− = 1                            decrement n_q
21              If(n_q = 0) then                    if n_q is zero
22                  H ← H ∪ q                       q goes to list H
23          End For
24      End For
25      i+ = 1
26      F_i ← H                                     i-th front is formed with current members of H
27  End While
```

The value of the crowding distance (CD) metric for each solution (computed using Algorithm 2) is based on its distance from the closest neighbour solutions for each objective function within the same rank. The higher the CD value, the more distant is the solution from its neighbours in the search space, making it more preferable in the selection for the next generation of the population. To maintain candidate solutions at the extremities of a frontier, its CDs are given an infinite value. In a problem with two objective functions, CD is the semi-perimeter of a rectangle whose vertices are the solutions nearest neighbours.

Algorithm 2 Crowding Distance computation

```
1   Let I be a set of individuals
2   Let M be the list of objectives
3   l = |I|
4   For each i ∈ 0..l − 1 do
5       I[i] = 0                                          initialize distances
6   End For
7   For each objective m ∈ M do
8       I ← sort(I, m)                                    sort by each objective value
9       I[0]_distance = I[l − 1]_distance = ∞              guarantee selection of extremities
10      For each i ∈ 1..l − 2 do
11          I[i]_distance += (I[i + 1].m − I[i − 1].m)     compute distances
12      End For
13  End For
```

The selection procedure for the new generation of the population (detailed in Algorithm 3) is carried out as follows: the current population is merged with its offspring and grouped according to the Fast Non-dominated Sorting procedure. Individuals with lower ranks will receive preference to entry into the next generation. To select an individual between 2 or more within the same rank, the individual with the highest CD value will have the preference.

Algorithm 3 NSGA-II Selection

```
1   Let P be the set of parent individuals
2   Let Q be the set of offspring individuals
3   Let S be the set of selected individuals
4   F ← fast_nondominated_sorting(P ∪ Q)
5   i = 1
6   While(|S| < |P|) do
7       crowding_distance_computation(F_i)
8       S ← S ∪ F_i
9       i += 1
10  End While
11  S ← S ∪ F_i[0 : |P| − |S|]
```

2.4 Quantization

Quantization is the process of constraining a continuous and large set of values, such as the real numbers, to a discrete set, such as the integers. In this work, we deal with quantization processes over the 24-bit RGB encoding in color-based image representations (Section 3.3) and from 32-bit floating point to equal or less than 8-bit integer in deep neural networks (Section 4.3.1).

Bellow we address some pertinent characteristics of quantizations: the form in which the low-precision levels are distributed over the high-precision range (Section 2.4.1), the symmetry of the high-precision range (Section 2.4.2) and the form how the high-precision are projected over the quantization levels (Section 2.4.3).

2.4.1 Distribuition of Quantization Levels

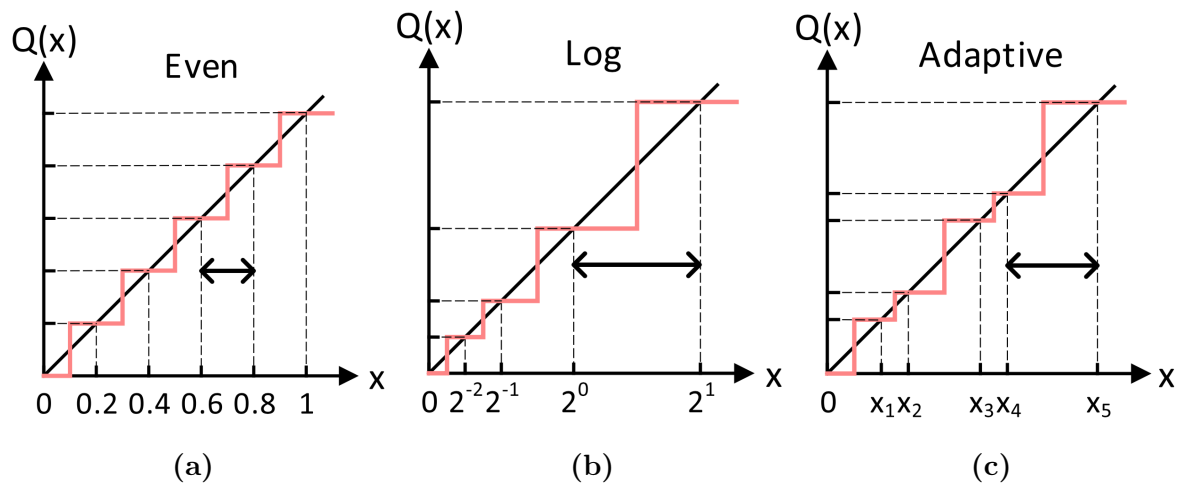


Figure 2.1. Distribution of discrete levels [25]. (a) Uniform distribution with even step length. Non-uniform distribution with (b) logarithmic or (c) adaptive step length.

Quantization schemes differ in the form that its levels are distributed across the signal interval to be quantized. According to this aspect, there are basically two classifications of quantizations: Uniform quantizations and Non-uniform quantizations. The quantization whose quantization levels are uniformly spaced is termed as a Uniform quantization. The type of quantization whose the quantization levels are unequal and following non-uniform distributions, – such as exponential families distributions – is termed as a Non-uniform quantization. Some non-uniform quantizations distributions are opti-

mized for the signal distribution, they are usually prominent of adaptive processes such as statistics-based processes and clustering. The step length and dynamic range can be fixedly predefined or dynamically determined by data. Non-uniform distribution usually has variable step length that selects important data regions or provides a wider dynamic range. Figure 2.1 exemplifies some quantization schemes.

2.4.2 Symmetry

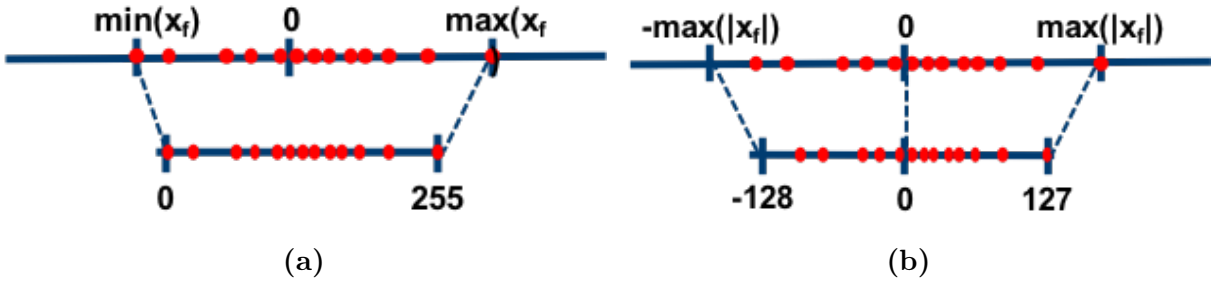


Figure 2.2. Quantization Symmetry [134]. (a) Asymmetric with respect to the zero-point. (b) Symmetric with respect to the zero-point.

This property says about the symmetry of the quantization interval with respect to the zero-point. In asymmetric quantization (Fig .2.2a), we use actual min/max values of the high-precision axis to determine the quantization range ($[min(x_f), max(x_f)]$). In symmetric quantization (Fig .2.2b), the quantization range is defined using the absolute maximum value of the high-precision axis ($[-max(|x_f|), max(|x_f|)]$), allowing a broader range.

2.4.3 Level Projection

A important aspect in the quantization process is how to project the original high-precision data to a discrete space. There are only two approaches for level projection (Fig. 2.3): (a) the deterministic approach and (b) the stochastic approach. The former projects the high-precision data to the nearest discrete level. The latter has the possibility of projection to one of the two nearby adjacent levels (to the left or to the right). The probability p of projecting to each one is determined by the distance from the original

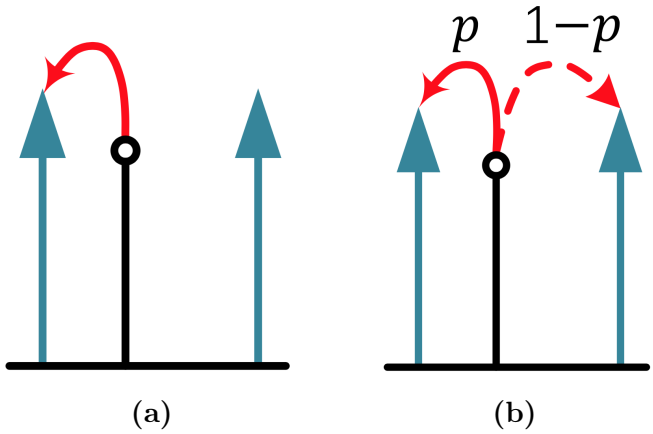


Figure 2.3. Level projection [25]. (a) Deterministic—transition to the nearest level. (b) Stochastic—transition to nearby adjacent levels controlled by probability.

data to the nearby discrete levels.

Chapter 3

Optimizing Shallow Representations

3.1 Introduction

It is known that the form in which multimedia data, especially images, are represented can highly impact the performance of machine learning methods typically used in visual pattern recognition tasks, such as content-based image retrieval (CBIR) [101], object detection [123], remote sensing image analysis [23], and image classification [66]. In the last years, representation learning [7], which consists of the process of using pattern recognition algorithms to find representations optimized for a given data domain and/or task at focus, has become a tendency. In fact, the current state-of-the-art methods for representation learning, which are based on deep learning [57] techniques, in many cases present considerable gains in terms of the image content description quality.

However, the use of these methods present serious drawbacks, such as the broad range of hyper-parameters and possible architectures, the huge computational workload spent to train existing models, the big amount of labeled data required to produce effective models, and the need for specific expertise or training for properly designing, optimizing, and evaluating promising solutions.

Representation learning methods usually employ one of two main approaches: those that learn representations from a feature set provided by a hand-crafted extractor and those that completely compose new ones without any prior feature extraction (from scratch). The latter approach often leads to the usage of more complex and consequently costly methodologies, such as deep learning. Such complexity, however, should be avoided in the generation of representative features. A few years ago, before the arising of deep neural networks, hand-crafted feature extractors were used to encode image visual properties (e.g., color, texture, or shape) into effective representations [83, 28, 84]. In general, those solutions rely on less costly algorithms and do not depend on previously annotated datasets or time-consuming learning steps. On the other hand, these feature extractors are application-dependent, being less generalizable.

In this chapter, we propose a hybrid scheme, focused on color quantization, which

aims to take advantage of both research venues. We propose data-driven color quantization schemes, which improve the effectiveness of hand-crafted feature extractors, as it allows for the identification of discriminative visual features. Our representation learning scheme exploits a particular characteristic of the current image context, its color distribution, a simple but yet suitable visual cue in several applications [60, 74, 50]. We hypothesize that data-driven quantization optimizations are able to positively impact the quality of image content description approaches, leading to effective and efficient representations. In this chapter, we investigate how these optimizations can be performed effectively and efficiently and to what extent.

Our color quantization optimization relies on a soft computing framework, implemented using genetic algorithms (GA). GA is an evolutionary algorithm widely used to solve optimization problems. According to its formulation, a population of individuals, representing possible solutions to a problem, evolves over generations, subjected to genetic operations. The goal is to find the best individuals, i.e., the best solutions for the problem. In our color quantization problem, a GA individual encodes how color channels should be divided in order to improve the effectiveness of feature extractors. To the best of our knowledge, this is the first work to use GA to model the representation learning problem.

In summary, the main contributions of this chapter are:

1. We show that different color quantizations impact the effectiveness performance of feature extractors;

2. We model the search of suitable color quantization using a soft computing apparatus based on the genetic algorithm;

3. We introduce two approaches for supervised representation learning capable of providing compact and more effective representations through color quantization optimization.

In summary, the main novelty of our work relies on the presentation of an integrative framework for the implementation of effective image search systems that combines several concepts, approaches and techniques, such as, Genetic Algorithm optimization, Color Quantization, Representation Learning, Feature Extraction, and Content-Based Image Retrieval.

We conducted a series of experiments in order to evaluate the robustness of the proposed approaches in content-based image retrieval tasks, considering eight well-known datasets containing images with different visual properties. Experimental results indicate that the approach focused on the representation effectiveness outperformed the baselines in all tested scenarios. The other approach, which focuses not only on the effectiveness

but also on the size of the generated feature vectors, was able to produce competitive results by keeping or even reducing the final feature vector dimensionality up to 25%.

The remainder of this chapter is organized as follows. Section 3.2 presents related work. Section 3.3 describes the proposed color quantization schemes and their use in CBIR tasks. Section 3.4 details the experiments performed to assess the effectiveness and efficiency of the proposed methods. Section 3.5, in turn, presents and discusses achieved results. Finally, Section 3.6 presents our main conclusions and outlines possible future research directions.

3.2 Related Work

Image representation learning (a.k.a. feature learning) consists in automatically discovering the representations needed for object detection or classification from raw images. It is a set of approaches that aim at making it easier to extract useful information when building classifiers or other predictors [7]. In other words, feature learning allows to find the most suitable or discriminative representation from the raw data according to some constraint imposed by the target application. Thus, it is also commonly known as data-driven features because of its contraposition to engineered or hand-crafted features.

Although feature learning has been an active research area for a long time, the development of effective techniques (mainly based on deep learning) has been boosted in the last decade mainly due to the spread use of powerful computational resources, which were motivated by the development of graphical processing units (GPUs). Many successful recent feature representation approaches are based on deep belief nets [41], denoising auto-encoders [113], deep Boltzmann machines [94], K-Means-based feature learning [17], hierarchical matching pursuit [13], and sparse coding [125]. Regarding image representation learning, the most successful approaches are based on the Convolutional Neural Networks (CNNs) [56].

Although, by definition, a large number of techniques perform feature learning, the term is most commonly employed by the community that develops methods based on deep learning or probabilistic graphical models. These methods are the basis for most of the state-of-the-art approaches for pattern recognition and computer vision. Despite the recent great success of these approaches, they still have several limitations, such as a large number of parameters for optimization and the difficulty in designing network architectures.

Evolutionary algorithms are meta-heuristic optimization techniques that use mechanisms inspired by biological evolution (e.g., reproduction, mutation, recombination, and

selection). They have been widely employed in a myriad of frameworks developed for image analysis and retrieval usually for feature fusion [20] or selection [46, 76]. In the last few years, evolutionary algorithms have also been successfully employed for neural networks architecture search [105, 121]. Nonetheless, we did not find other works that directly model feature learning as an evolutionary algorithm-based problem from the raw data.

In this work, we propose to learn image features from images via genetic algorithms by color quantization optimization. Some works developed quantization learning using evolutive heuristics for image segmentation [67]. Scheunders [97] handles the quantization problem as global image segmentation and proposes an optimal mean squared quantizer and a hybrid technique combining optimal quantization with a Genetic Algorithm modelling [36]. Further, the same author [97] presents a genetic c-means clustering algorithm (GCMA), which is a hybrid technique combining the c-means clustering algorithm (CMA) with Genetic Algorithm. Lastly, Omran et al. [82] developed colour image quantization algorithm based on a combination of Particle Swarm Optimization (PSO) and K-means clustering.

Regarding the effects of colour quantization on image representations, Ponti et al. [87] approached the colour quantization procedure as a pre-processing step of feature extraction. They applied four fixed quantization methods – Gleam, Intensity, Luminance, and a concatenation of the Most Significant Bits (MSB) – over the images of three datasets and then used four feature extractors – ACC, BIC, CCV, and Haralick-6 – to compute representations intended to solve the tasks of Image Classification and Image Retrieval. Their conclusions show that it is possible to obtain compact and effective feature vectors by extracting features from images with a reduced pixel depth and how the feature extraction and dimensionality reduction are affected by different quantization methods.

New approaches based on deep learning developed in the last ten years have revolutionized the learning of representations from data. Regarding the learning of representations for images, convolutional networks have established themselves as the most effective solution. However, its use still has some limitations, such as: (1) they require a large amount of data for training from scratch; (2) traditional networks have a large number of parameters. Therefore, some works have been proposed in order to mitigate these limitations and produce more compact networks [70, 130]. In this context, approaches based on nature-inspired/evolutionary algorithms have emerged as an alternative to optimize network architectures in various ways [93, 10]. Although color quantization approaches are less used nowadays than in the past for image representation, they are still an alternative to obtain compact and effective representation for some applications, such as color-based image retrieval [127, 86, 100, 11].

To the best of our knowledge, our work is the unique that provides an application-driven way to learn compact representation from color quantization.

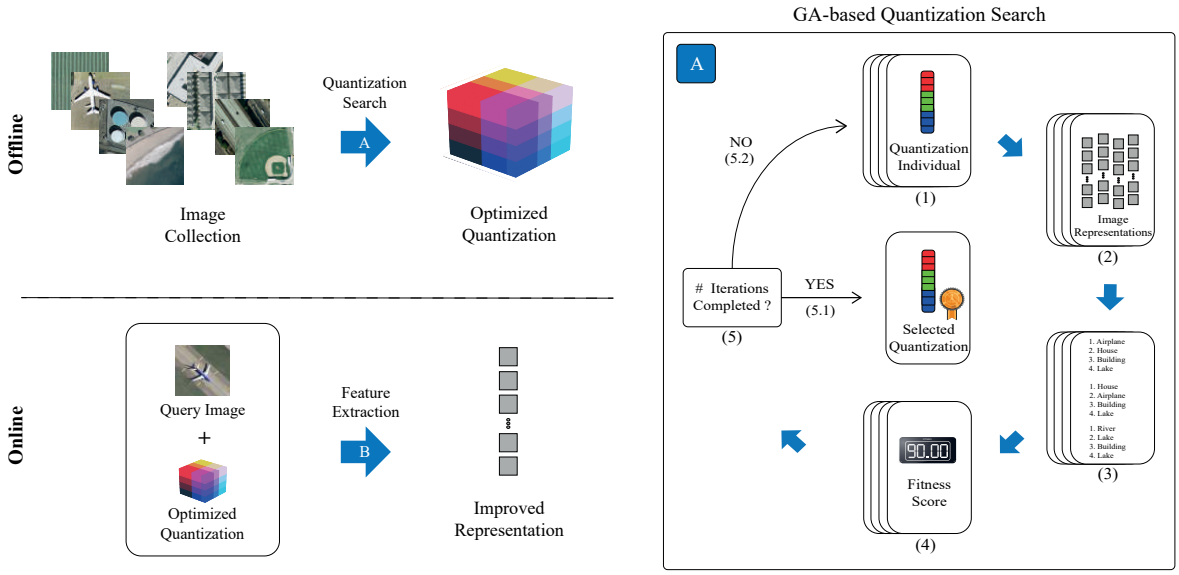


Figure 3.1. Overview of the proposed approach. First, (A) we use Genetic Algorithm to search for an optimized color quantization. Later, (B) the resulting quantization is incorporated into the feature extractor to generate improved image representations. The GA-based quantization search proceeds as follows: first, (1) a population of encoded color quantizations is randomly produced; second, (2) sets of image representations of the whole collection are produced being each one according to one quantization color space; third, (3) similarity rankings for all to all images are computed within each representation set; and fourth, (4) a fitness score is computed to measure each retrieval effectiveness. Finally, if the stopping condition is met or the total number of iterations is achieved, (5.1) the quantization of the highest fitness of the last population is selected as the optimized colour space, otherwise, (5.2) a new population is created, via crossover and mutation operations over the current population, initiating the next iteration.

3.3 GA-based Color Quantization

In this chapter, we introduce the use of Genetic Algorithm to learn an optimized color quantization for a given image domain. Figure 3.1 provides an overview of the entire process, which is composed of two main steps: (A) quantization search, and (B) feature extraction. These steps are described next.

3.3.1 Quantization Search

We propose the use of Genetic Algorithm [36] to learn the best color quantization for a given collection. GA has been a widely used approach for finding near-optimal solutions for optimization problems. One remarkable property of this optimization apparatus

relies on its ability on performing parallel searches starting from multiple random initial search points and considering several candidate solutions simultaneously. Consequently, it represents a fair alternative to an exhaustive search strategy, which would be unfeasible given the number of possible solutions.

According to this optimization algorithm, an individual corresponds to a representation of a potential solution to the problem that is being analyzed. In our modeling, each individual represents a possible color quantization, as detailed in Section 3.3.1.1. During the evolution process, described in Section 3.3.1.2, these individuals are gradually evolved. At the end of the evolutionary process, the best-performing individual, which encodes a quantization that leads to an improved representation, is selected.

3.3.1.1 Quantization Encoding

In our modeling, a quantization is represented in a GA individual as follows: Let $\mathbb{M}$ be a color model composed of three channels. Without loss of generality, we will assume the RGB color model from now on. Assume that each channel is divided into 256 discrete levels, i.e., eight bits can be used to define the number of colors in each channel. In the case of the traditional 24-bit RGB model, there are almost 17 million ($256 \times 256 \times 256$) different colors.

In our formulation, a 24-bit long GA individual encodes the number of partitions of the different channels. Figure 3.2 (top) presents the typical 24-bit RGB channel partitioning. Figure 3.2 (middle) illustrates a possible GA individual encoding how each channel should be divided. Figure 3.2 (bottom), in turn, illustrates the resulting color quantization after using the GA individual encoding.

Figure 3.3 presents the RGB color space before (a-b) and after (c-d) using the GA-based encoding defined in Figure 3.2(middle). Figures 3.3(b) and 3.3(d) present different views of the same color space presented in Figures 3.3(a) and 3.3(c), respectively.

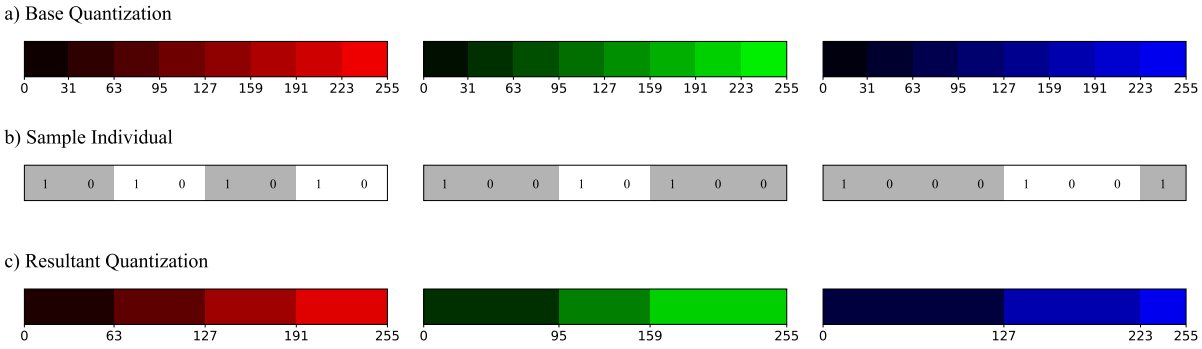


Figure 3.2. Our modeling takes reference from a base quantization (a) representing each interval of color tonalities as a bit in individuals implemented as binary arrays (b). These bits dictate the union of intervals producing a new quantization (c): if a bit is set, its respective interval has its own position, otherwise, it is aggregated to the immediate previous interval. The first bit of each color axis is forced to always be set.

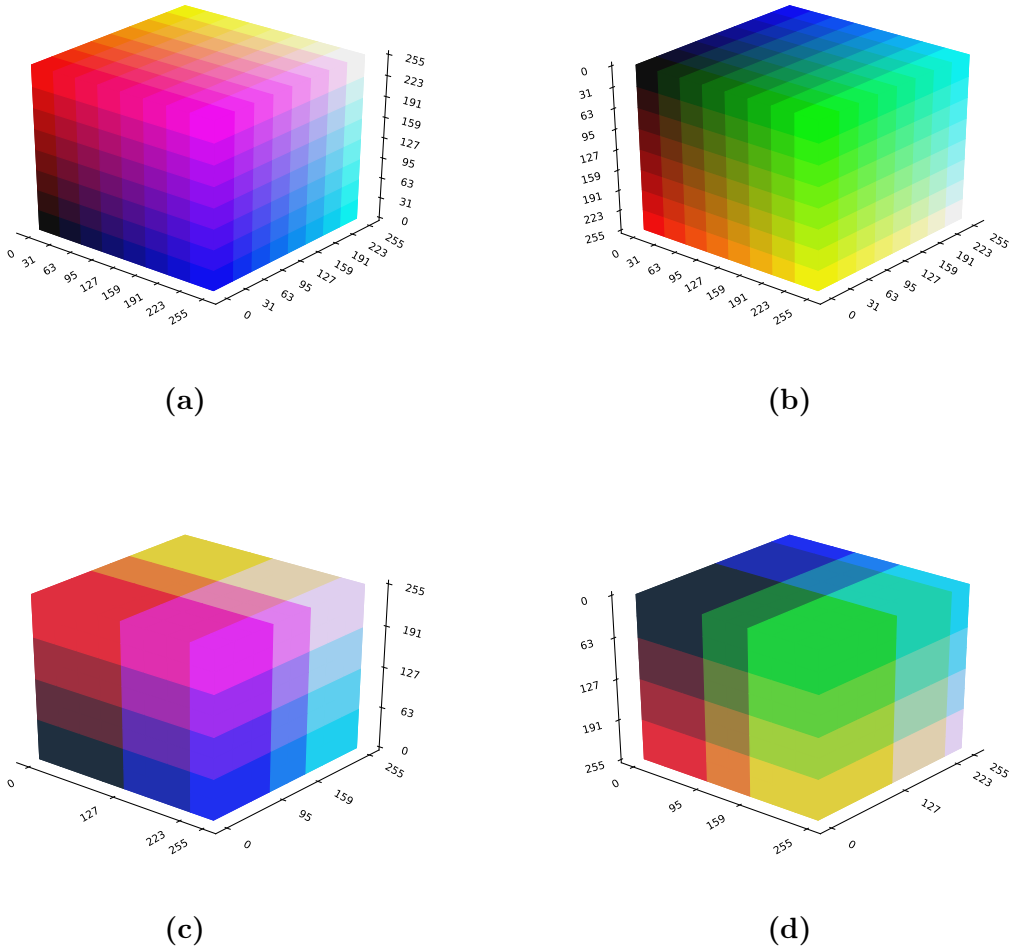


Figure 3.3. (a) RGB color space using the traditional 8-bit quantization per channel. (b) The same color space presented in (a), but rotated in 180º over the Z axis. (c) Color space after applying the GA individual illustrated in Figure 3.2 (middle). (d) The same color space presented in (c), but rotated in 180º over the Z axis.

Algorithm 4 GA-based quantization search

1 Let T be a training set
2 Let P, S_e e S_t be sets of pairs $(q, fitness_q)$, where q and $fitness_q$ are an individual and its fitness, respectively
3 $P \leftarrow$ Initial random population of individuals
4 **For each** generation g of N_g generations **do**
5 **For each** individual $q \in P$ **do**
6 $fitness_q \leftarrow fitness(q, T)$
7 **End For**
8 $S_e \leftarrow elitism(k, P)$
9 $S_t \leftarrow tournament(n_t, P)$
10 $P \leftarrow S_e \cup mutation(S_t) \cup crossover(S_t)$
11 **If** stopping condition is met
12 Break outer loop
13 **End If**
14 **End For**
15 Select the best individual $q^* = \arg\max_{q \in P}(fitness_q)$

3.3.1.2 GA-based Quantization Search

Algorithm 4 illustrates the proposed GA-based quantization. The population starts with individuals created randomly (line 3). The population evolves generation by generation through genetic operations (line 4). A function (described in Section 3.3.3) is used to assign the fitness value for each individual (lines 5-7), i.e., to assess how well an individual solves the target problem. According to the elitism operation, the k best individuals of the current generation are recorded (line 8). Then, individuals from P are selected according to a tournament operation of n_t-sized groupings (line 9). After that, the next generation is formed from the union of the resulting individuals from the operations of mutation and cross-over over the tournament selection and those selected in elitism (line 10). If the stopping condition (discussed on Section 3.4.3) were met, the iterations stop (lines 11-13). The last step is concerned with the selection of the best individual q^* of all generations (line 15). The individual q^* is used later to define the quantization used in the feature representation process. For details regarding genetic operators (cross-over, mutation, tournament and elitism), refer to Section 3.4.3.

3.3.2 Feature Extraction

In the second phase, the best individual, i.e., the one which leads to the best quantization q^* is used with the feature extractor algorithm to produce a color image representation. In order to do that, it was necessary to implement a slightly modified version of the feature extractor, that incorporates the capacity of generating representations according to a specified color quantization. Equations 1, 2, and 3, where M_c is the maximum color axis size and q^* is the quantization individual, define how to calculate the new R, G, and B (referred to as R_{new}, G_{new}, and B_{new}, respectively) values for each pixel. In this work, according to empirical observations, M_c was chosen as 8.

$$R_{new} = \left(\sum_{i=0}^{r} q^*[i] \right) \times \frac{|R_{axis}|}{256}, \tag{3.1}$$

$$where \quad r = R \times \frac{M_c}{256}; \quad |R_{axis}| = \sum_{l=0}^{M_c} q^*[l]$$

$$G_{new} = \left(\sum_{j=N}^{g+M_c} q^*[i] \right) \times \frac{|G_{axis}|}{256}, \tag{3.2}$$

$$where \quad g = G \times \frac{M_c}{256}; \quad |G_{axis}| = \sum_{m=M_c}^{2M_c} q^*[m]$$

$$B_{new} = \left(\sum_{k=2M_c}^{b+2M_c} q^*[i] \right) \times \frac{|B_{axis}|}{256}, \tag{3.3}$$

$$where \quad b = B \times \frac{M_c}{256}; \quad |B_{axis}| = \sum_{n=2M_c}^{3M_c} q^*[n]$$

Figure 3.4 shows the visual effect of different color quantizations on different sample images. The first column shows the RGB color spaces defined according to the specified quantizations: the original space in which the image is captured, a widely-used handcrafted quantization scheme using 64 colors, and an example of optimized quantization defined by our method. The remaining columns show sample images after using each quantization scheme. Original images are shown in the top line. Above each quantized image, we present the color spectrum and its respective histogram.

[1]We recommend colourful printing for adequate visualization.

Figure 3.4. This figure[1] shows the visual effect of different color quantizations on different sample images. The first column shows the RGB color spaces defined according to the specified quantizations: the original space in which the image is captured, a widely-used hand-crafted quantization scheme using 64 colors, and an example of optimized quantization defined by our method. The remaining columns show sample images after using each quantization scheme.

3.3.3 Individual Fitness Computation

The use of the proposed GA-based quantization leads to discriminative features, which may be useful in different applications, such as Image Classification [28], Image Retrieval [83], and Object Recognition. In this chapter, we opted for evaluating the method in the context of Content-Based Image Retrieval (CBIR) [101] tasks. The goal of this task is to retrieve the most relevant images from a collection, given their similarity to a given query image. The similarity computation relies on the use of a distance (or similarity) function applied to feature vectors, which encode their content (in our case, their color properties).

We first extract feature vectors from all images within a collection, by taking into account feature extractors that benefit from the learned color quantization. Collection images are later ranked according to the distance of their feature vectors to the feature vector of a query using the Manhattan Distance (L1). Two images belonging to the same class are assumed to be relevant to each other. Given a query image, our goal is to produce a ranked list with collection images of the same class of the query on top positions. The more relevant images on top positions, the more effective is the ranked list, i.e., the more effective is the description approach.

More formally, an image *img* is firstly encoded through a feature extraction procedure, which allows quantifying the similarity between images. Let C =

$\{img_1, img_2, \ldots, img_n\}$ be a collection with n images. Let $\mathcal{D}$ be a descriptor, which can be defined as a tuple (ϵ, δ) [19], where:

- ϵ: $img_i \rightarrow \mathbb{R}^d$ is a function, which extracts a feature vector $v_{\vec{i}}$ from an image img_i;
- δ: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$ is a function that computes the distance between two images according to the distance between their corresponding feature vectors.

The distance between two images img_i, img_j is computed as $\delta(\epsilon(o_i), \epsilon(o_j))$. The Euclidean distance is commonly used to compute δ, although the proposed ranking method is independent of distance measures. A similarity measure $\rho(img_i, img_j)$ can be computed based on distance function δ and used for ranking tasks. We will use $\rho(i, j)$ from now on to simplify the notation.

The target task refers to retrieving multimedia objects (e.g., images, videos) from C based on their content. Let img_q be a query image. A ranked list τ_q can be computed in response to img_q based on the similarity function ρ. The ranked list τ_q=(img_1, img_2, ..., img_n) can be defined as a permutation of the collection C. A permutation τ_q is a bijection from the set C onto the set $[N] = \{1, 2, \ldots, n\}$. For a permutation τ_q, we interpret $\tau_q(i)$ as the position (or rank) of the image img_i in the ranked list τ_q. If img_i is ranked before img_j in the ranked list of img_q, i.e., $\tau_q(i) < \tau_q(j)$, then $\rho(q, i) \geq \rho(q, j)$.

Given a training set composed of a set of queries and their respective list of relevant objects, the fitness of an individual is measured as a function of the quality (effectiveness) of ranked lists produced for each query, considering the use of a feature extractor implemented using the GA-based quantization. The more relevant images found at top positions, the better the GA individual is.

3.3.4 Computational Complexity of GA-based Quantization Search

The GA training procedure takes $\mathcal{O}(N_g \times N_i \times F)$, where N_g is the number of generations considered in the evolution process, N_i is the number of individuals in the population, and F is the cost for evaluating the fitness function.

The costs for computing F depends on the number of training samples N_s and the size of pre-computed histograms S_h. The later, in the worst case, is k^3, where k is the number of bins in a color axis. As overlying detailed base color spaces does not improve the results, k is typically small, making S_h also small ($k = 8$ and $S_h = k^3 = 512$

in our experiments). As a consequence, F takes $\mathcal{O}(N_s \times S_h)$ for feature extraction and $\mathcal{O}({N_s}^2 \times logN_s)$ for computing rankings, then $\mathcal{O}(F) = \mathcal{O}({N_s}^2 \times logN_s)$.

Finally, the whole procedure takes $\mathcal{O}(N_g \times N_i \times {N_s}^2 \times logN_s)$ to find the final quantization. Recall that the training process is performed offline.

3.3.5 Quantization Approaches

In this chapter, we propose two formulations of the GA-based quantization method. The first, named Unconstrained Approach (UA), is intended to provide a quantization focused on generating representations that have the best possible effectiveness performance. The second, named Size-Constrained Approach (SCA), focuses not only on effectiveness aspects, but also on the size of the representation. The goal is to find the best-performing individual, which leads to feature vectors with a pre-defined size, i.e., the target feature vector size is defined a priori. From the implementation point of view, the GA-based quantization approach assigns a negative fitness score for the individuals that present dimensions higher than the pre-defined feature vector size. As a consequence, this latter formulation tends to produce more compact representations.

3.4 Experimental Setup

In this section, we present the adopted experimental setup, which concerns the image datasets considered (Section 3.4.1), the configuration of parameters of the method (Section 3.4.3), the baselines used for comparative analysis (Section 3.4.2), the metrics used to evaluate the effectiveness and compactness of the produced feature vectors (Section 3.4.4), and the employed experimental protocol (Section 3.4.5).

3.4.1 Datasets

In order to assess the effectiveness of the employed quantization approach, we conducted experiments using eight different image datasets, which are described next.

Table 3.1. Image datasets and statistics

Dataset	# of samples	# of classes	Images content
Coil-100 [77]	7,200	100	objects
Corel-1566 [115]	1,566	43	mixed (objects, landscapes etc)
Corel-3906 [115]	3,906	85	mixed (objects, landscapes etc)
ETH-80 [58]	3,280	80	objects
MSRCORID [18]	4,320	20	mixed (scenes and objects)
Groundtruth [62, 61]	1,285	21	landscapes
Supermarket Produce [92]	2,633	15	fruits
UC Merced Land-use [122]	2,100	21	aerial scenes

For convenience, Table 3.1 summarizes some important information about them.

- ***Coil-100***: This dataset [77] comprises images of 100 everyday objects, being each one used to define a different class. Pictures of each object were taken in 72 different poses composing a total set of 7,200 images. Some samples of this dataset are shown in Figure 3.6.

- ***Corel-1566 and Corel-3906***: These datasets [115] correspond to two sets from a collection with 200,000 images from the Corel Gallery Magic–Stock Photo Library 2. The first (Fig. 3.7) contains 1,566 samples distributed among 43 classes, while the second (Fig. 3.8) contains 3,906 samples among 85 classes. Besides the image quantity, the main difference between them is that the latter presents more intra-class variability.

- ***ETH-80***: This dataset [58] was originally tailored to the task of object categorization. It includes images of 80 objects from 8 basic-level categories. Each object is represented by 41 views over the upper viewing hemisphere, performing a total of 2,384 images. Some samples of this dataset are shown in Figure 3.9.

- ***Groundtruth***: This dataset [62, 61] contains a variety of 1,285 scenes and objects grouped among 21 high-level concepts, such as: Arbor Greens, Australia, Barcelona, Cambridge, Campus In Fall, Cannon Beach, Cherries, Columbia George, Football, Geneva, Green Lake, Greenland, Indonesia, Iran, Italy, Japan, Leafless Trees, San Juans, Spring Flowers, Swiss Mountains, Yellow Stone. Figure 3.12 depicts some of its classes.

- ***Microsoft Research Cambridge Object Recognition Image Database (MSRCORID)***: This collection [18] contains a set of 4,320 images of scenes, objects and landscapes. Its images are grouped into 20 categories: Aeroplanes, Cows, Sheep, Benches and Chairs, Bicycles, Birds, Buildings, Cars, Chimneys, Clouds, Doors, Flowers, Kitchen Utensils, Leaves, Scenes Countryside, Scenes Office, Scenes

Urban, Signs, Trees, Windows. Some samples of this dataset are shown in Figure 3.10.

- ***Supermarket Produce***: This dataset [92] contains images of fruits and vegetables collected from a local distribution center. It comprises 2,633 images distributed into 15 different categories: Plum, Agata Potato, Asterix Potato, Cashew, Onion, Orange, Tahiti Lime, Kiwi, Fuji Apple, Granny-Smith Apple, Watermelon, Honeydew Melon, Nectarine, Williams Pear, and Diamond Peach. Figure 3.11 depicts some samples of its categories.

- ***UC Merced Land-use***: This dataset [122] is composed of 2,100 aerial scene images divided into 21 classes selected from the United States Geological Survey (USGS) National Map. Its 21 categories are Agricultural, Airplane, Baseball Diamond, Beach, Buildings, Chaparral, Dense Residential, Forest, Freeway, Golf Course, Harbor, Intersection, Medium Density Residential, Mobile Home Park, Overpass, Parking Lot, River, Runway, Sparse Residential, Storage Tanks, and Tennis Courts. Some samples of this dataset are shown in Figure 3.5.

(a) Forest **(b)** Beach **(c)** Tennis Court

(d) Dense Residential **(e)** Medium Residential **(f)** Sparse Residential

Figure 3.5. Examples of the *UC Merced Land-use* dataset.

(a) Object 13 **(b)** Object 14

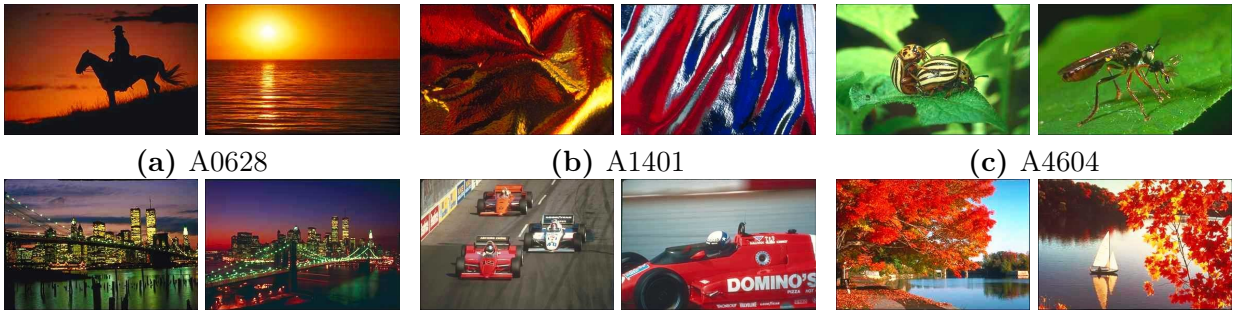
(c) Object 87 **(d)** Object 81

Figure 3.6. Examples of the *COIL-100* dataset.

(a) A6140 **(b)** A14935 **(c)** A2231

(d) A0908 **(e)** A7840 **(f)** A0004 **(g)** A12147

Figure 3.7. Examples of the *COREL-1566* dataset.



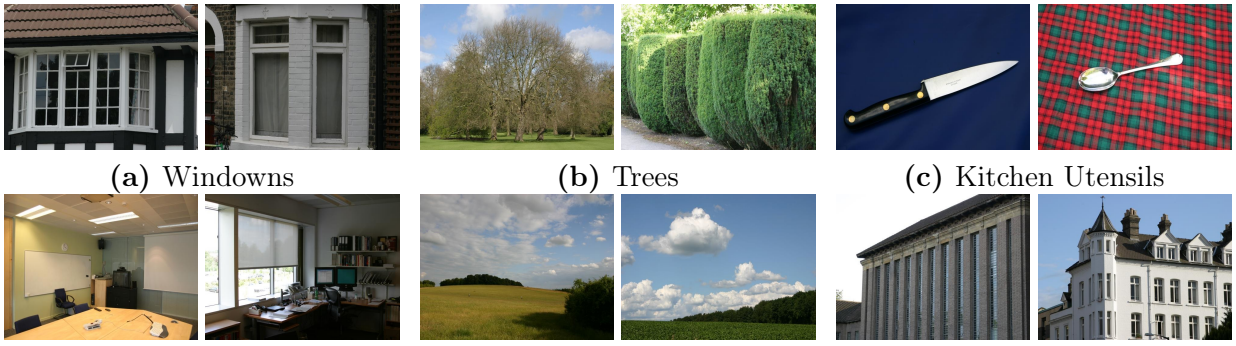
(a) A0628 **(b)** A1401 **(c)** A4604

(d) A4932 **(e)** A7601 **(f)** A14208

Figure 3.8. Examples of the *COREL-3906* dataset.

. (a) Car (b) Cow (c) Cup

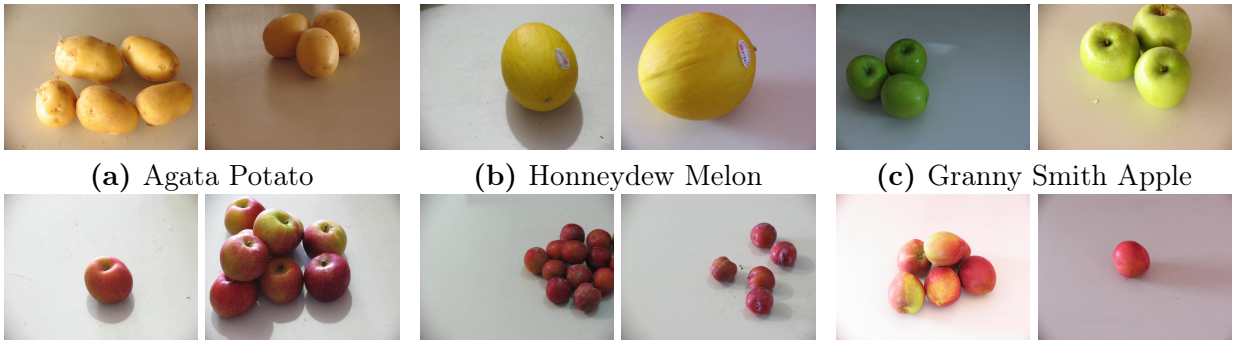
Figure 3.9. Examples of the *ETH-80* dataset.



(a) Windowns **(b)** Trees **(c)** Kitchen Utensils

(d) Scenes Office **(e)** Scenes Countryside **(f)** Buildings

Figure 3.10. Examples of the *MSRCORID* dataset.



(a) Agata Potato **(b)** Honneydew Melon **(c)** Granny Smith Apple

(d) Fuji Apple **(e)** Plum **(f)** Nectarine

Figure 3.11. Examples of the *Supermarket Produces* dataset.

(a) Cambridge **(b)** Australia **(c)** Columbia Gorge

(d) Barcelona **(e)** Indonesia **(f)** Geneva **(g)** Japan

Figure 3.12. Examples of the *Groundtruth* dataset.

3.4.2 Baselines

3.4.2.1 Feature Extraction Algorithms

In order to demonstrate the impact of using the learned quantizations in the generation of more effective image representations, we compare GA-based feature extractors with similar formulations *without* any quantization procedure. We use the BIC and the GCH original formulations (see Section 2.1) as baselines.

3.4.2.2 *Winner-Take-All* Autoencoder

We also perform comparisons with autoencoders (see Section 2.2), a class of methods based on Deep Learning – the state-of-the-art framework for computer vision – dedicated to performing representation learning and, consequently. They are, therefore, suitable recent approaches for comparison purposes.

According to Makhzani et al. [69], Sparse Autoencoders (SAE) yield the best performance than other types, such as Denoising Autoencoders, for feature extraction in tasks such as image classification. Among SAEs, we selected Winner-Take-All Autoencoders WTA-AE (see Section 2.2) which hold some advantages in comparison with other SAEs including the capability of aiming any sparsity rate, efficient training and resource consumption besides allowing the use of reduced architectures.

Among the WTA-AE proposed configurations, we selected the CONV-WTA autoencoder, which is a non-symmetric architecture where the encoder consists of a stack of three 256-units ReLU convolutional layers (5×5 filters) and the decoder is a 256-units linear deconvolutional layer of larger size (11×11 filters): 256conv3-256conv3-256conv3-256deconv7. It also maintains N_u hidden representation units between the encoder e decoder. This is the same architecture used by Makhzani et al. [70] in experiments for the CIFAR-10 dataset [55], an image dataset of a domain similar to the ones used in our experiments.

Following the instructions of Makhzani et al., with the purpose of composing representations adequate to being used on an image classification/retrieval setting, we employed, after training, max-pooling on the last N_u feature maps of the encoder, over 6×6 regions at strides of 4 pixels to obtain the final representation of $N_u \times 8 \times 8 = N_u \times 64$

Table 3.2. Genetic algorithm parameters. The indicated variables refer to Algorithm 4.

Two-point Cross-over Probability	60%
One-point Mutation Probability	40%
Number of Generations (N_g)	200
Population Size	200
Tournament (n_t)	5
Elitism (k)	1%

total size. In order to allow a fair performance comparison between the different-sized representations of WTA-AE and SCA, we employed Principal Components Analysis [118] – a well-known data projection algorithm – on the WTA-AE representation as dimensionality reduction procedure where the number of dimensions corresponded to the imposed representation size limits.

3.4.3 Parameters

Table 3.2 presents the values adopted for the GA-based quantization learning process. The values chosen for population size, cross-over, mutation, elitism, and tournament parameters were defined empirically, but all of them represent typical values employed in GA-based optimization solutions. Initially, it was applied a parameter search according to a 2^k Fractional Factorial Design (please refer to item 16.3.3 of [48]) over a portion of the dataset. For the parameters which presented major sensitivities, a binary search was employed for the exploration of different values.

The total number of generations was defined aiming to ensure convergence of the evolutionary algorithm. However, we empirically observed that typically the best fitness value is not significantly improved after remaining unchanged for more than 50 iterations. Thus, one might impose a stopping condition regarding the fitness value as an option to avoid unnecessary iterations.

We assess the quality of ranked lists defined by image representations obtained by means of a GA individual through the FFP4 function [31]. This score is defined for a given query image q as:

$$FFP4_q = \sum_{i=1}^{|D|} r_q(d_i) \times k_8 \times k_9^i \tag{3.4}$$

where D is the image dataset; $r_q(d) \in [0, 1]$ is the relevance score for the image d_i associated to the query, it being 1 if relevant and 0 otherwise; and k_8 and k_9 are two scaling

factors adjusted to 7 e 0.982 respectively. The final fitness score is computed as the mean $FFP4$ for all images $q \in D$.

As Fan et al. [31] explain, FFP4 is a utility function based on the idea that the utility of a relevant document decreases with its ranking order. More formally, we need a utility function $U(x)$ which satisfies the condition $U(x_1) > U(x_2)$ for two ranks x_1 and x_2 which $x_1 < x_2$. Although there are many possible functions $U(x)$, we decided to use FFP4 as it presents good results in previous works [20] applying this measure on similar evolutionary approaches that address rank-based tasks. According to Fan et al., this function and its associated parameters were chosen after exploratory data analysis.

For the baseline WTA-AE, we set the parameters as: number of hidden representation units $N_u = 1024$ and *winner-take-all* lifetime sparsity $k = 40\%$ empirically selecting them within the ranges {64, 128, 256, 512, 1024} and {5%, 10%, 20%, 30%, 40%, 50%, 80%}, respectively.

3.4.4 Evaluation Metrics

3.4.4.1 Precision-Recall Curves

The most traditional measures to evaluate retrieval effectiveness over a set of queries are Precision and Recall [3]. Precision measures the proportion of relevant images regarding the answer set, while Recall measures the proportion of relevant images retrieved in the answer set regarding all relevant images existing in the database.

A perfect system would provide a Precision equal to 1 (all the retrieved images are relevant) and a Recall also equal to 1 (all the relevant images were retrieved). In practice, there is an inverse relationship between them: the more items the system returns, the higher the likelihood that relevant documents will be retrieved (increasing recall). However, this comes at the cost of also retrieving many irrelevant documents (decreasing precision). Therefore, in general, it is necessary to define a compromise between them.

In our case, we chose a measurement that considers Precision and Recall as functions of each other, generating interpolated Precision-Recall curves (11 points) whose the precision points P given by

$$P(r_i) = \max_{\forall j | r_i \leq r_j} P(r_j) \tag{3.5}$$

where $i, j \in 0, 1, ..., 10$ represent recall levels.

In order to evaluate the retrieval effectiveness over a set of query images Q, an averaged Precision-Recall curve is computed according to

$$\overline{P}(r_i) = \sum_{q=1}^{|Q|} \frac{1}{|Q|} P_q(r_i) \tag{3.6}$$

where P_q corresponds to the precision of the q-th query image.

3.4.4.2 MAP: Mean Average Precision

In some cases, the Precision-Recall curves appear occluded or inter-crossed, restraining a proper visual comparison. Because of the compromise between Precision and Recall, it is possible to employ a combination of the two measures as a single metric. This is the case of Mean Average Precision [3] which provides a convenient measure to quantitatively compare Precision-Recall curves and is defined as

$$MAP = \frac{1}{|Q|} \sum_{q=1}^{|Q|} AP_q \tag{3.7}$$

$$AP_q = \frac{1}{|R_q|} \sum_{k=1}^{|R_q|} P(R_q[k]) \tag{3.8}$$

where R_q is the set of relevant images in the dataset Q for each image q.

3.4.4.3 P@10

As observed on real-world applications of CBIR, the user gives prior attention to a small group of the top answers, corresponding to the first page of results, usually preferring to reformulate the query instead of checking the next pages. The *Precision-Recall* curves and *MAP* do not provide an adequate measurement for the effectiveness of these top results as they generally consider longer portions of the ranking. To address this issue, we also measured the precision at the top-10 results (P@10) [3].

Due to the proximity of some measures and aiming to provide accurate comparisons

between the methods and its baselines, we used the Student's Paired t-Test [51] (p-value < 0.05) to statistically verify the results of Precision-Recall, MAP, and P@10.

3.4.4.4 Representation Size

To evaluate the descriptions dimensionality and possibly detect occurrence compactness regarding the previous methods, we measured the representation size, defined as the total number of bins that compose the histogram representations.

3.4.5 Experimental Protocol

In order to evaluate the proposed method, we conducted a k-fold cross-validation. According to this protocol, the dataset is randomly split into k mutually exclusive samples subset (folds) of about the same size. Then, the $k-1$ subsets are chosen as training set, and the remaining one as test set. The execution is repeated k times, and for each time, a different subset (without replacement) is chosen as the current test set and the remaining compose the training set.

We carried out all experiments considering $k = 5$ folds. As a consequence, for each experiment, the method was executed 5 times using 80% of the dataset as training set and 20% as test set.

3.5 Results and Discussion

This section compares the results of the proposed methods and baselines according to the evaluation measures.

First, we present the results of the UA methods with regard to Precision-Recall (Figs. 3.13 and 3.15), P@10 (Figs. 3.14a and 3.16a), MAP (Figs. 3.14b and 3.16b), and representation size (Fig. 3.17) for all datasets and feature extractors. Next, we present charts comparing the SCA results for Precision-Recall (Figs. 3.19-3.22), P@10 (Figs. 3.23 and 3.24), MAP (Figs. 3.25 and 3.26), and representation size (Fig. 3.18).

In the figures, the symbols above each pair of measures indicate whether the proposed method yields statistically better ⊕, worse ⊖, or similar ⊜ results to those observed for the baselines (the minimum between BIC/GCH and WTA-AE), considering rejection of the null hypothesis when p-value < 0.05.

The following sections present and discuss the experimental results and provides comparisons between these two proposed approaches and baselines.

3.5.1 Unconstrained Approach

Observing the Precision-Recall curves for the BIC feature extractor (Fig. 3.13), the UA outperforms its baselines for all datasets. According to the P@10 measurements (Fig. 3.14a), the method also presents, on average, more relevant results in the first positions of the ranking for all datasets. The superior MAP results (Fig. 3.14b) confirm the superiority of UA, as this measure takes into account the performance of the evaluated methods for the whole Precision-Recall curve.

Similar results were observed when the GCH feature extractor is considered. Figs. 3.15, 3.16a, and 3.16b provide the effectiveness results in terms of Precision-Recall, MAP, and P@10, respectively. For all datasets, but for the *Supermarket Produce*, the proposed UA approach yielded better results than those of the baseline. For the *Supermarket Produce* dataset, no statistical difference was observed.

With regard to the representation sizes (Fig. 3.17), the differences between the proposed methods and the baselines are very high. Comparing to the feature extractors, the representations produced by our method approach were, on average, around 521% larger for BIC (Fig. 3.17a) and around 328% larger for GCH (Fig. 3.17b). A possible reason relies on the fact that the fitness function used for evaluating the genetic algorithm individuals prioritizes the representation effectiveness performance on the retrieval task, i.e., the optimization process is not guided to guarantee compact representations. In this scenario, the proposed method quantized more regions in the color space, leading to representation with higher dimensions. The Size-Constrained Approach (SCA), whose results are discussed next, addresses this issue.

3.5.2 Size-Constrained Approach

In the evaluation of the SCA approach, we varied the number of bins in the ranges $\{16, 32, 64, 96, 128, 256, 384\}$ and $\{8, 16, 32, 48, 64, 128, 192\}$ for BIC and GCH approaches, respectively. These ranges were defined based on a logarithmic sequence of proportions (12,5%, 25%, 50%, 100%, 200%) of the baselines vector sizes and some additional points among them (75% and 300%) to provide a clearer view of the performances behaviour. Figs. 3.19 and 3.20 present the Precision-Recall curves for the BIC-based approaches and WTA-AE for all datasets, considering these different feature vector sizes. We can observe that the proposed method yielded comparable or better results than those observed for the baselines for feature vectors whose size is higher than 96 for the majority of the datasets. In fact, the smaller the feature vector size, the worse the results of SCA when compared to the baselines. Similar results were observed for the GCH-based approaches at Figs. 3.21 and 3.22.

Figs. 3.23 and 3.24 provide the P@10 results for the SCA method when compared with baselines for both BIC and GCH description approaches, respectively. Figs. 3.25 and 3.26, in turn, provide the MAP results for both BIC and GCH description approaches, respectively. Results related to MAP and P@10 demonstrated that, regardless the feature extraction method considered, the use of the SCA approach is able to create quite effective description approaches, without a high cost in terms of storage requirements, i.e., in terms of the feature vector size.

Figure 3.18 shows the sizes of the produced representations given the respective size upper-bounds. For the BIC approach (Fig. 3.18a) representations whose size reached or were very close to the imposed upper-bound were produced, showing a tendency for generating quantizations with strong tonality detailing. In contrast, the results for the GCH approach (Fig. 3.18b) are quite different. For example, for the upper limits 128 and 192, the produced representations were considerably smaller than the maximum size. In these cases, the more effective representations are not necessarily the ones with the highest possible dimensionality. This finding means that increasing the number of tonalities does not necessarily lead to performance improvements. In other words, the proposed methods are able to generate representations that are significantly smaller than the predefined upper-bound but with high effectiveness.

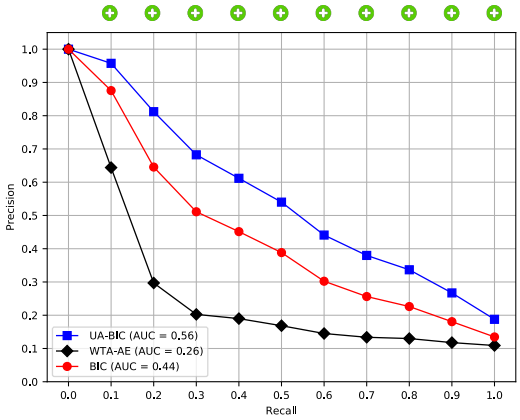
3.6 Conclusions

We proposed two approaches of a representation learning method, which intends to provide more effective and compact image representations by optimizing the color quantization for the image domain. We performed experiments on eight different image datasets comparing the results with a pre-defined quantization approach and a Sparse Autoencoder in terms of performance on content-based image retrieval tasks. Methods are also evaluated in terms of the representation dimensionality.

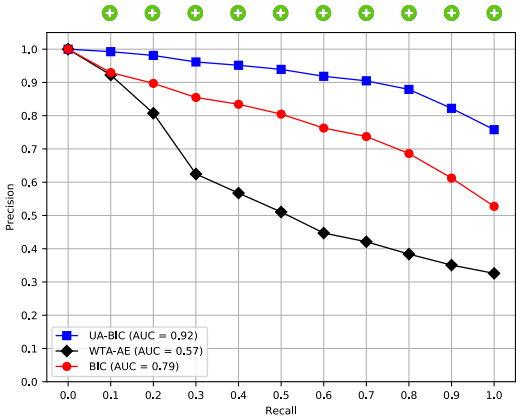
The first approach, named Unconstrained approach, produced representations that outperformed the hand-crafted baselines in terms of effectiveness but presented feature vectors with several times higher dimensionality. It also outperformed the effectiveness of the autoencoder representation but presenting intensively lower sizes. The second approach, which imposes a limitation on the representation dimension (Size-Constrained approach), presented, in general, better effectiveness results for the same dimensionality (e.g., 128 bins). In other situations, this approach reduced the representation size up to 50%, maintaining statistically comparable performance to the hand-crafted baselines. Finally, the SCA approach also produced results that imposed a reduction of more than 75% of the storage requirements, but presented poor effectiveness performance, showing the existence of a trade-off between compactness and effectiveness.

Since the representations are based on color histograms, the over- and the sub-sampling of specific color space regions allows for the identification of more effective representations and, consequently, improvements in the search performance. Furthermore, a domain-oriented quantization allows for discarding the less contributing tonalities resulting in a possible reduction of the representation size.

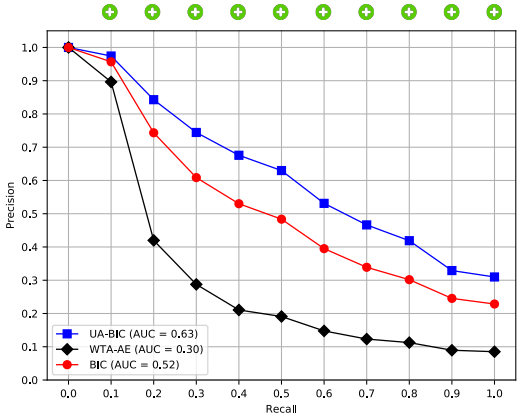
In the end, the results confirm our hypothesis, for the tested scenarios, that it was possible to produce more effective and compact fitness by exploring a color quantization optimized for the image domain. Moreover, our method is capable of improving already existent feature extraction methods by providing descriptions more effective in terms of representation quality and more compact according to a parametric upper bound. This research, therefore, opens novel opportunities for future investigation. We plan to assess the effects of the proposed quantization approaches to other image processing applications such as image classification [59], image segmentation [34] and image dehazing [129]. We also plan to investigate the impact of the resulting quantization when combined with deep-learning-based feature extractors.



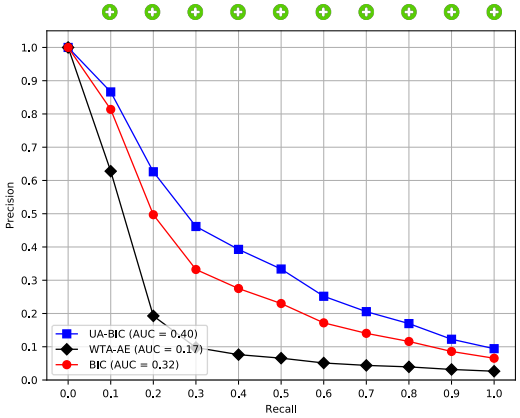
(a) Groundtruth



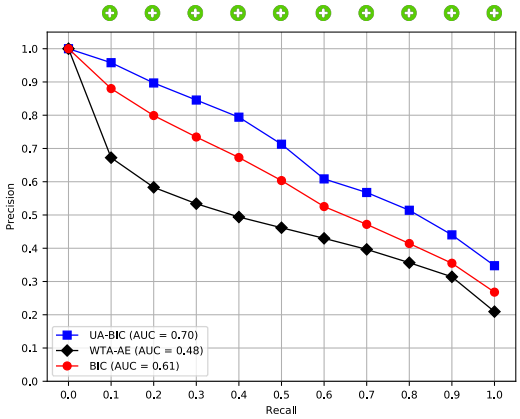
(b) Coil-100



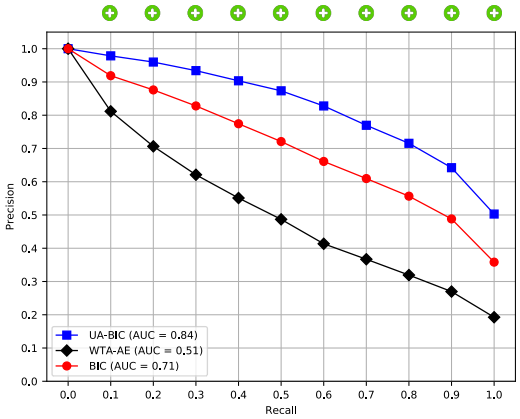
(c) Corel-1566



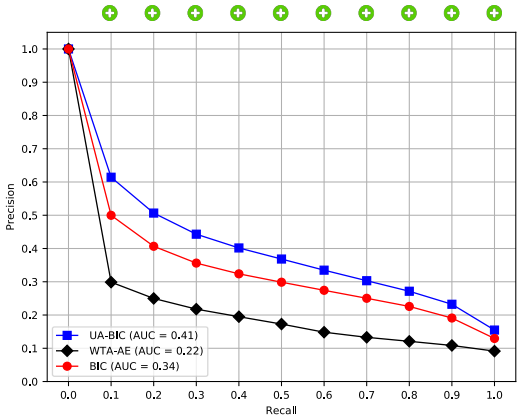
(d) Corel-3906



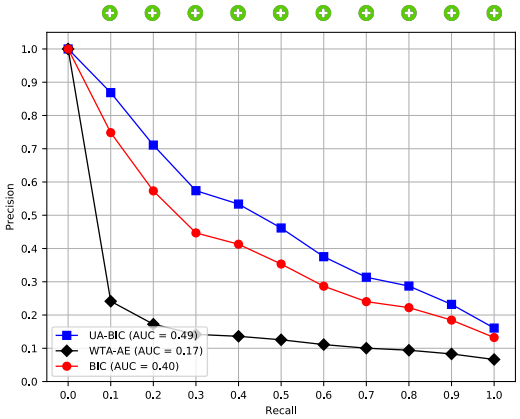
(e) ETH-80



(f) Supermarket P.

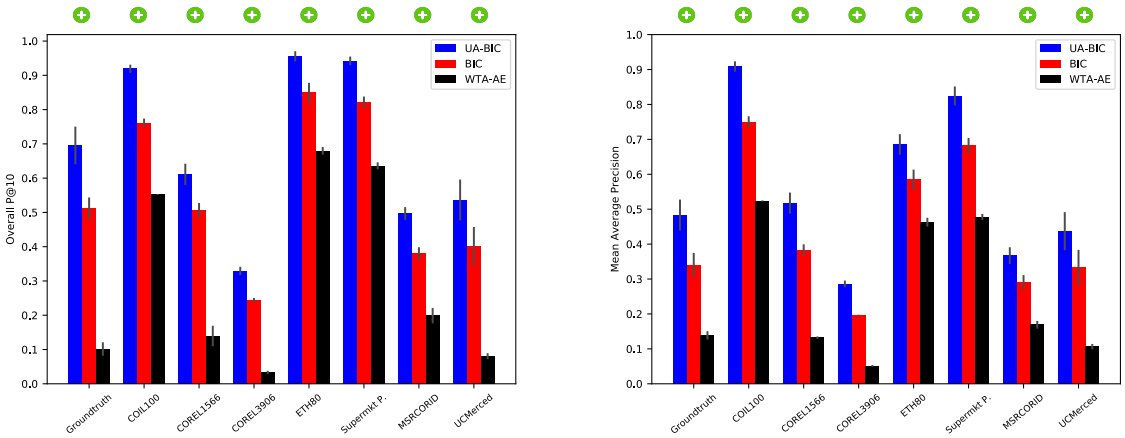


(g) MSRCORID



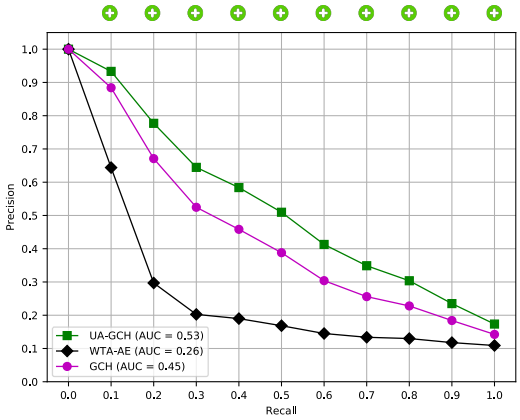
(h) UCMerced Land-use

Figure 3.13. Comparison between the Precision-Recall Curves of the UA method, WTA Autoencoder and the BIC feature extractor.

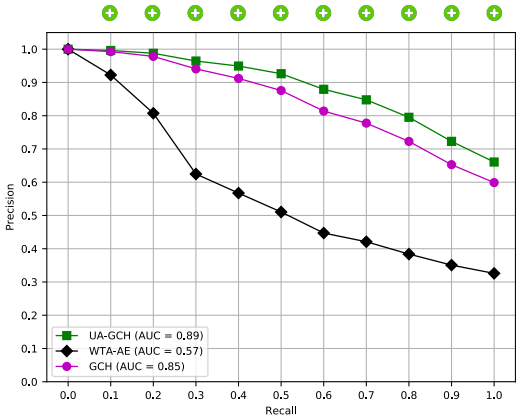


(a) P@10 **(b)** MAP

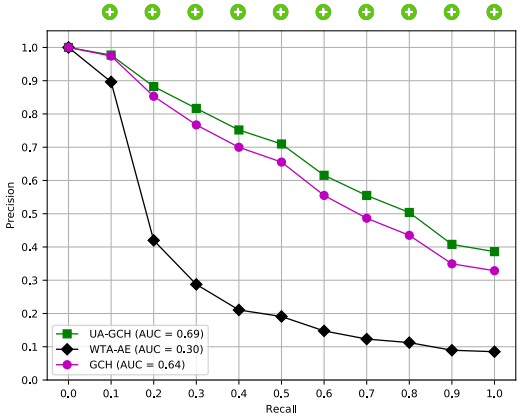
Figure 3.14. Comparison between the (a) P@10 and (b) MAP results of UA, WTA Autoencoder and the **BIC** feature extractor.



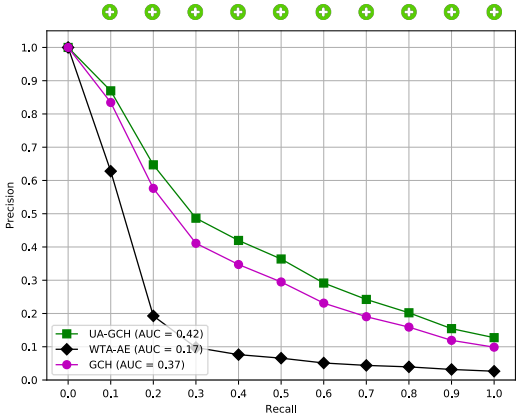
(a) Groundtruth



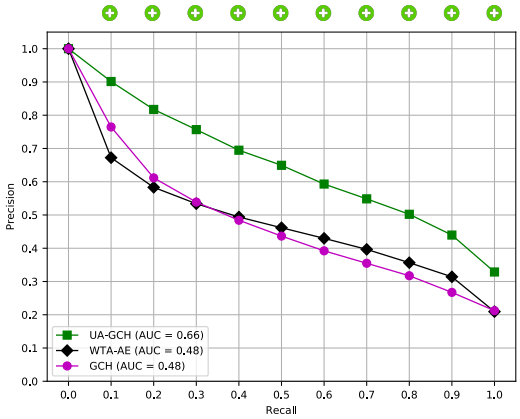
(b) Coil-100



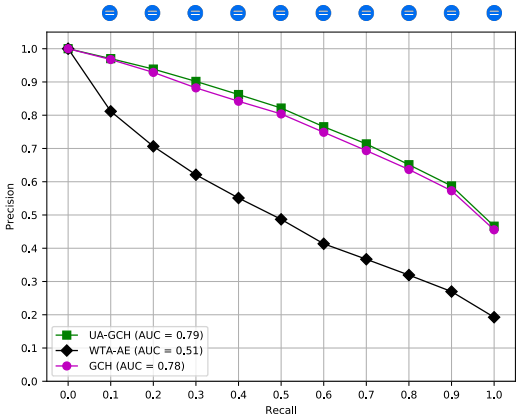
(c) Corel-1566



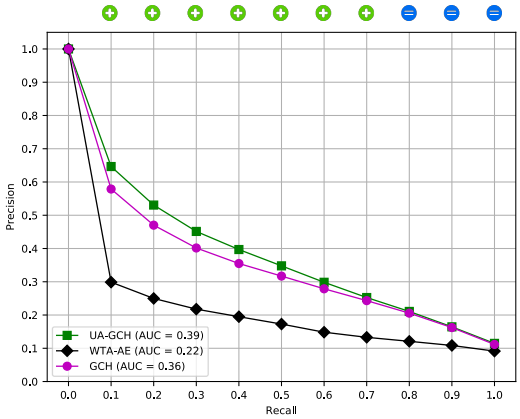
(d) Corel-3906



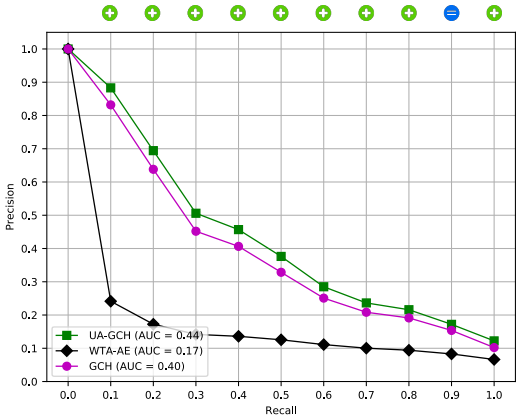
(e) ETH-80



(f) Supermarket P.

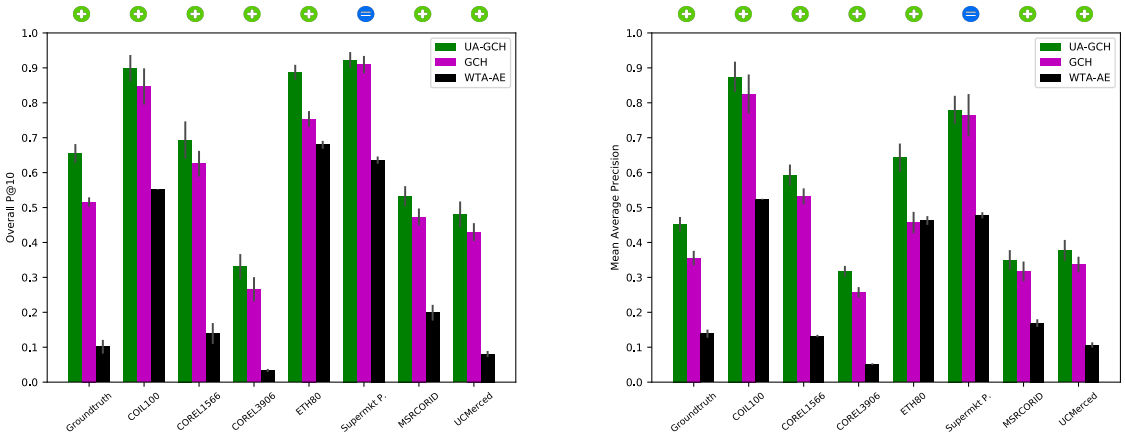


(g) MSRCORID



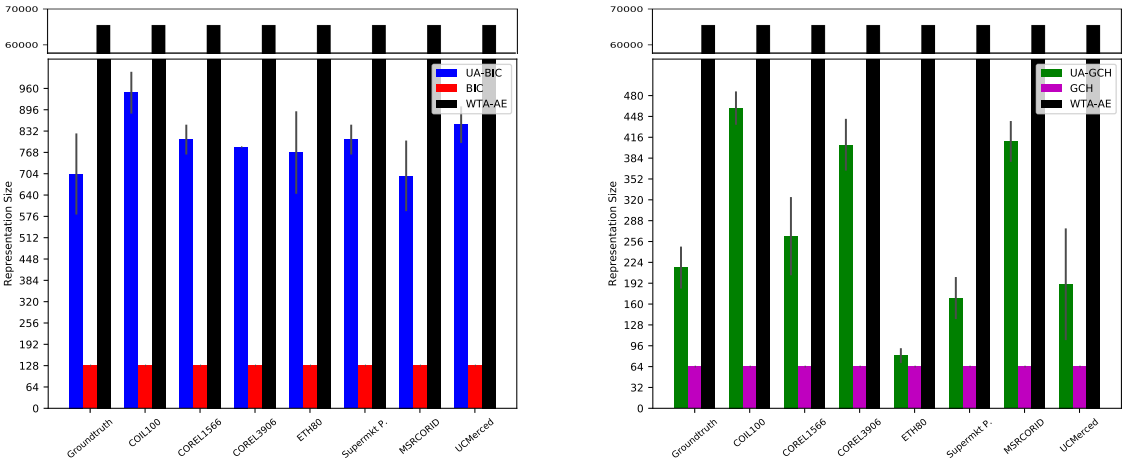
(h) UCMerced Land-use

Figure 3.15. Comparison between the Precision-Recall curves of UA, WTA Autoencoder and GCH feature extractor.



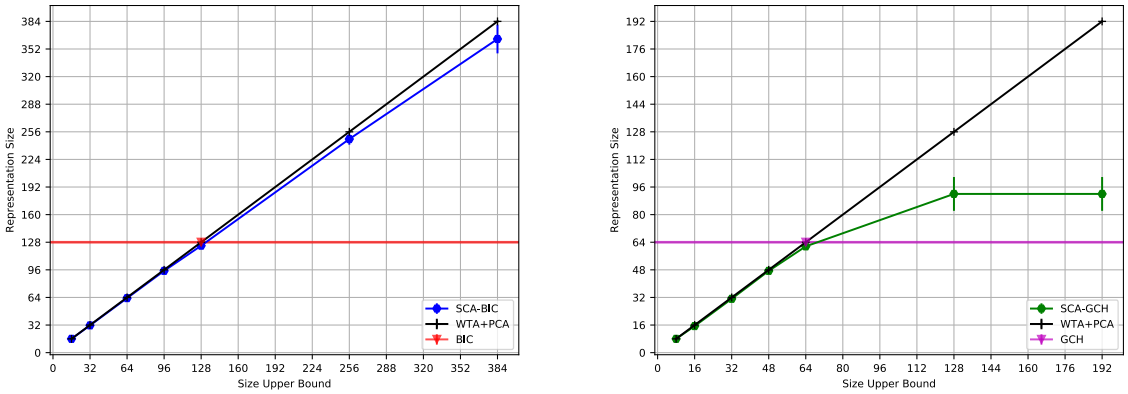
(a) P@10 **(b)** MAP

Figure 3.16. Comparison between the (a) P@10 and (b) MAP results of UA, WTA Autoencoder and the **GCH** feature extractor.



(a) BIC **(b)** GCH

Figure 3.17. Comparison between the representation size results of UA, WTA Autoencoder and the feature extractors: (a) BIC and (b) GCH. The upper windows show a cut of the highest columns while the lower windows show a view of the bottom.



(a) BIC **(b)** GCH

Figure 3.18. Comparison between the representation size results of SCA, WTA Autoencoder and the feature extractors: (a) BIC and (b) GCH, for the ETH-80 dataset. The appendix A of this article contains the same comparison for the remaining datasets.

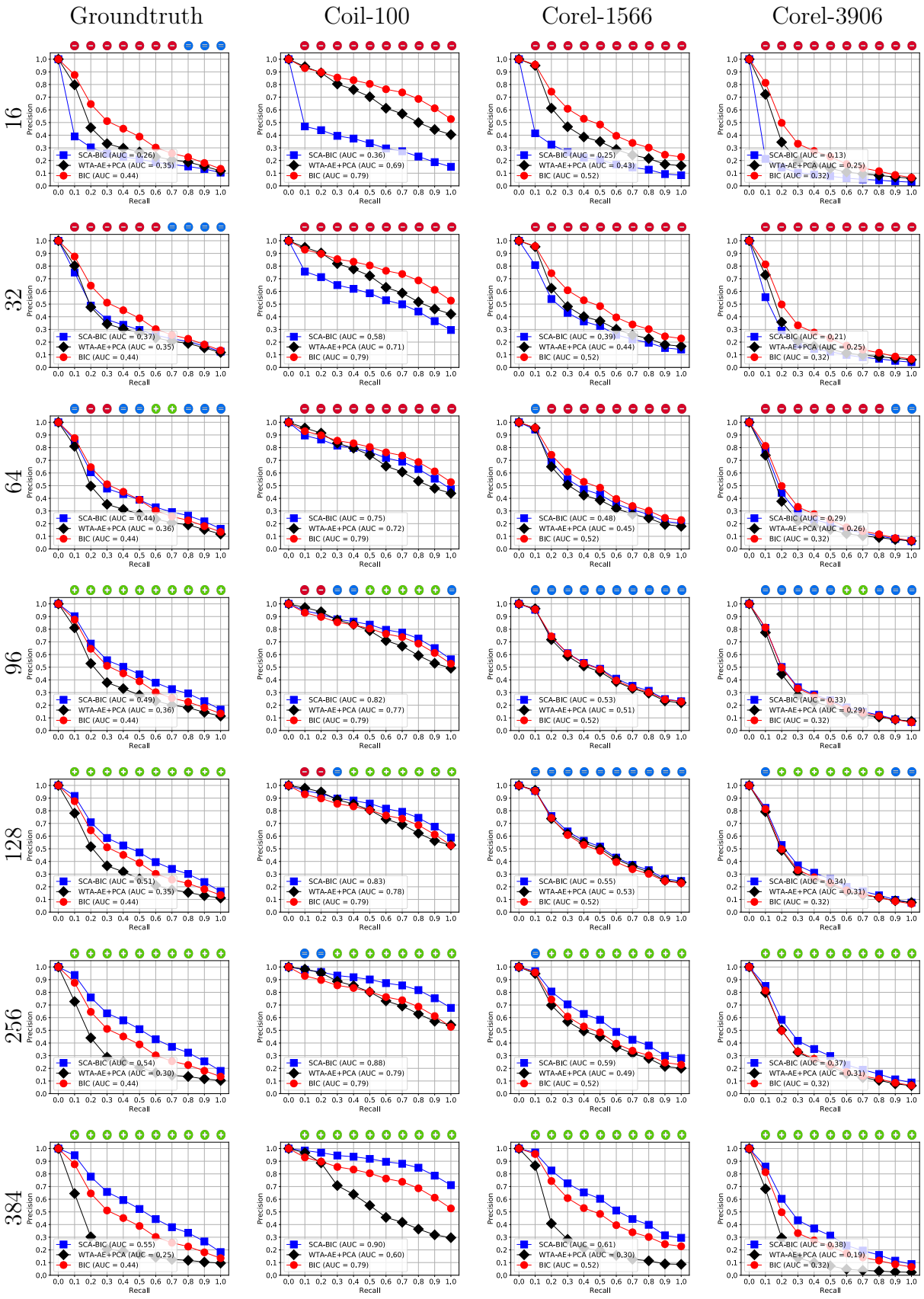


Figure 3.19. Comparison between the Precision-Recall curves of SCA, WTA Autoencoder and BIC feature extractor considering all representation size limits for the datasets *Groundtruth*, *Coil-100*, *Corel-1566*, and *Corel-3906*. We recommend colourful printing for adequate visualization.

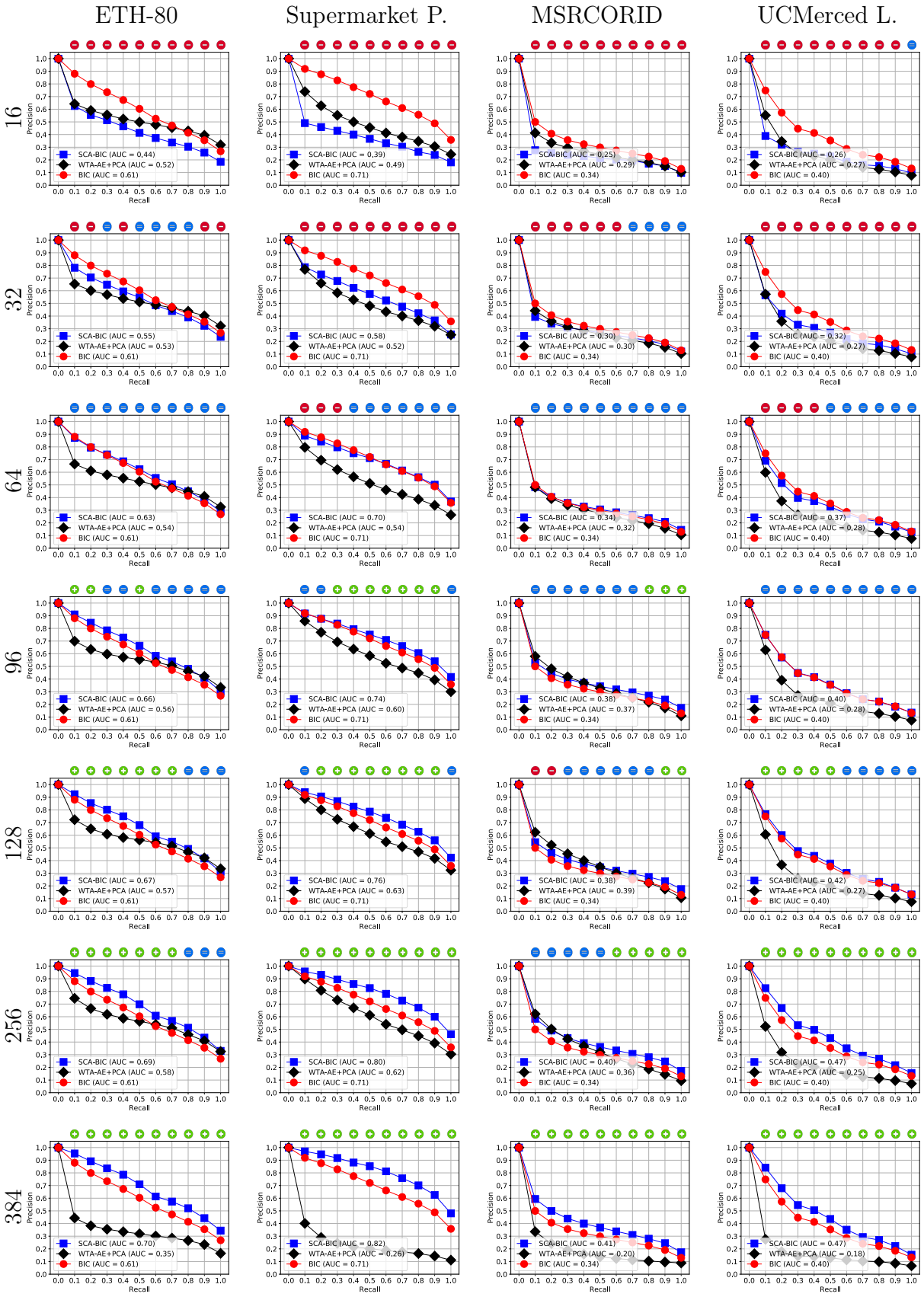


Figure 3.20. Comparison between the Precision-Recall curves of SCA, WTA Autoencoder and BIC feature extractor considering all representation size limits for the datasets *ETH-80*, *Supermarket Produce*, *MSRCORID*, and *UCMerced Landuse*. We recommend colourful printing for adequate visualization.

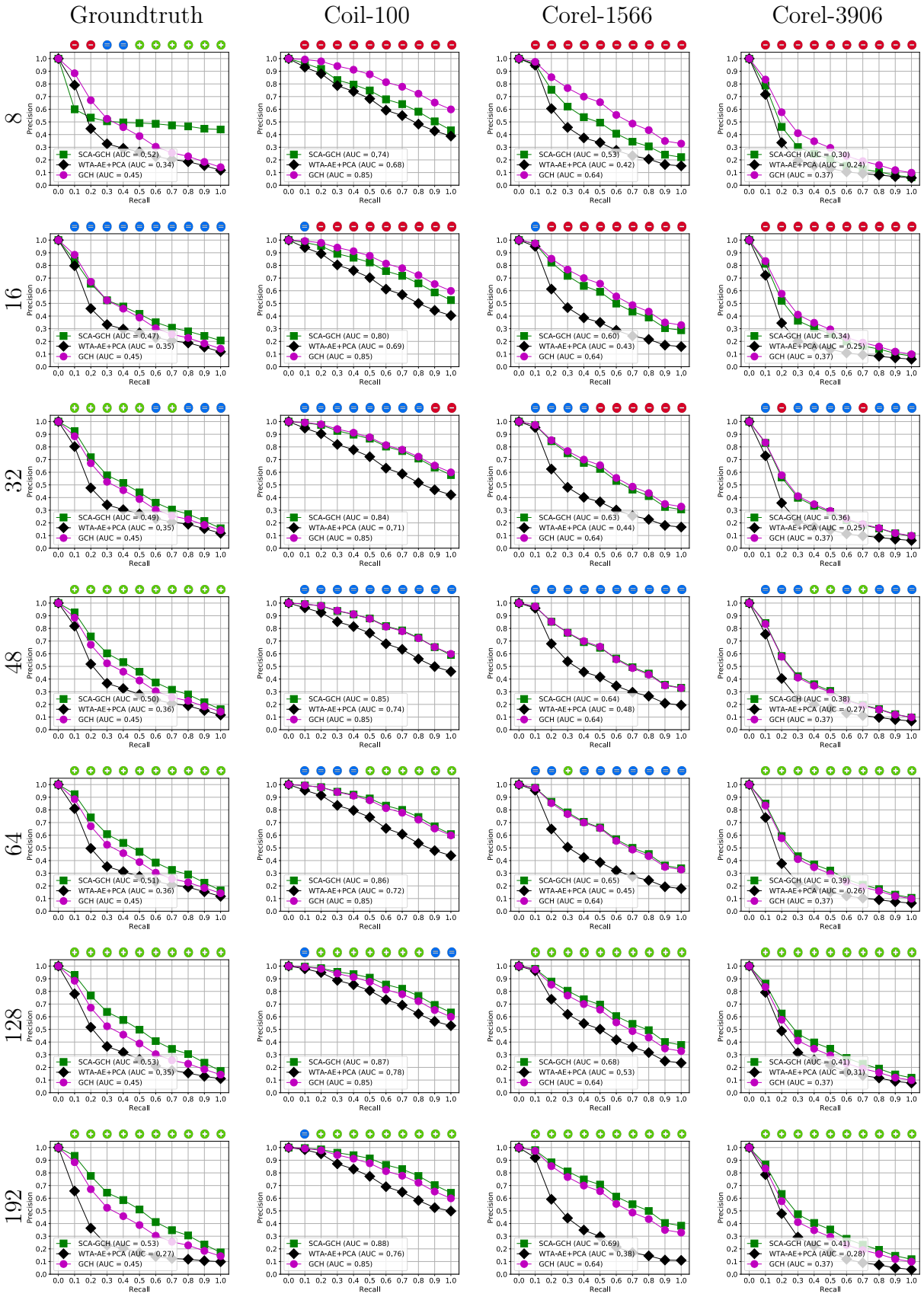


Figure 3.21. Comparison between the Precision-Recall curves of SCA, WTA Autoencoder and GCH feature extractor considering all representation size limits for the datasets *Groundtruth*, *Coil-100*, *Corel-1566*, and *Corel-3906*. We recommend colourful printing for adequate visualization.

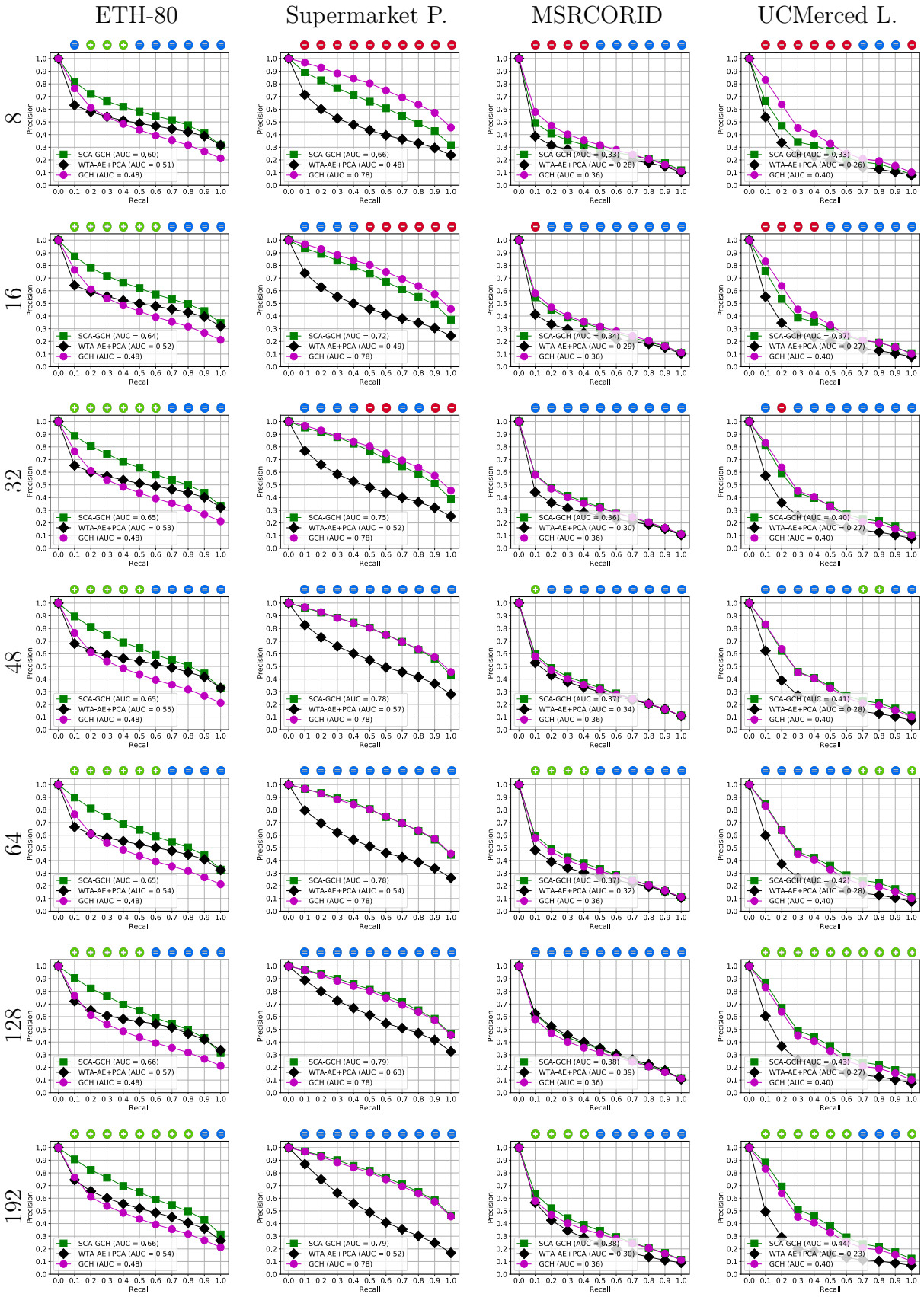
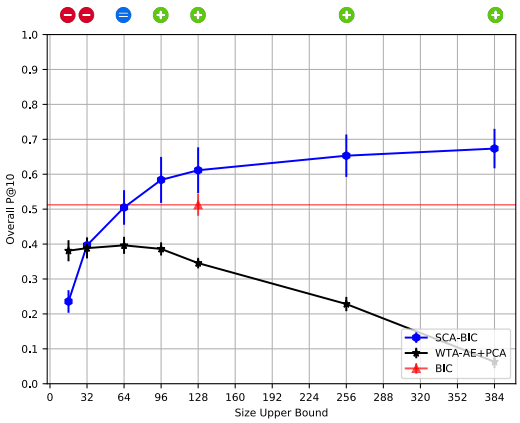
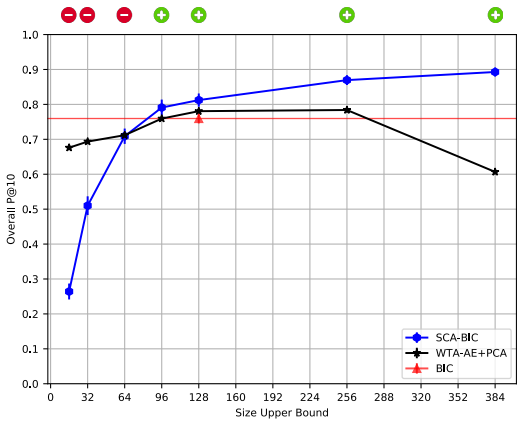


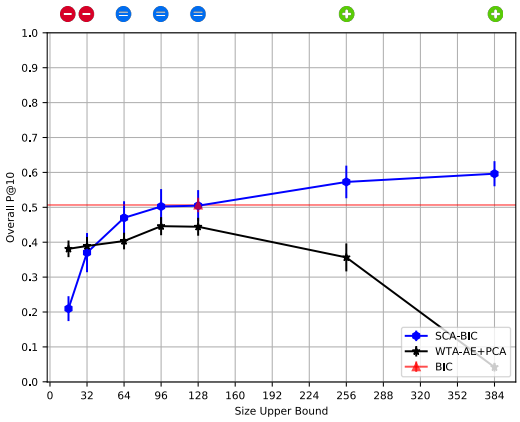
Figure 3.22. Comparison between the Precision-Recall curves of SCA, WTA Autoencoder and GCH feature extractor considering all representation size limits for the datasets *ETH-80*, *Supermarket Produce*, *MSRCORID*, and *UCMerced Landuse*. We recommend colourful printing for adequate visualization.



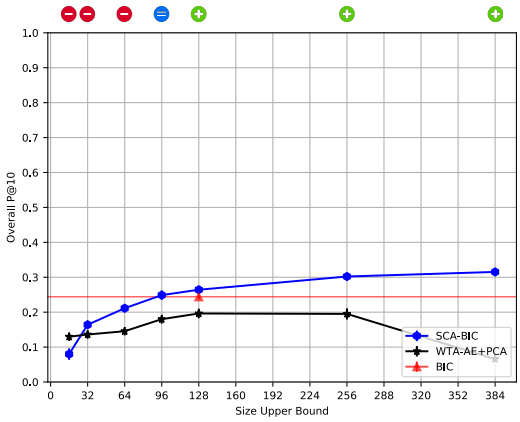
(a) Groundtruth



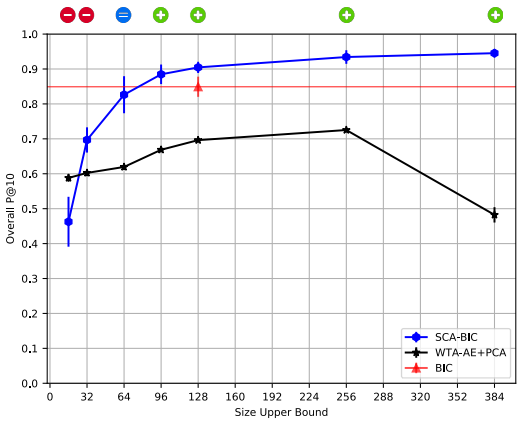
(b) Coil-100



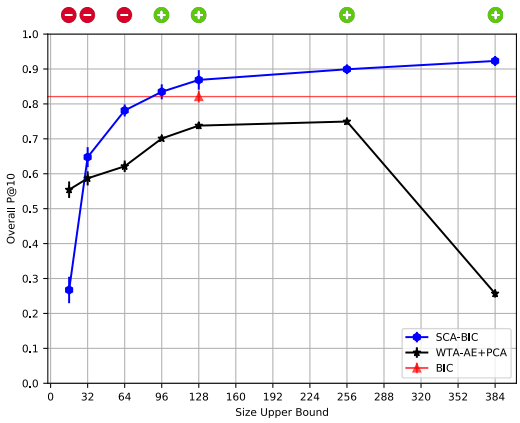
(c) Corel-1566



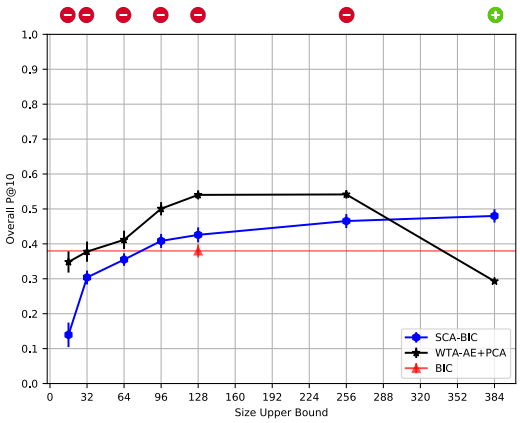
(d) Corel-3906



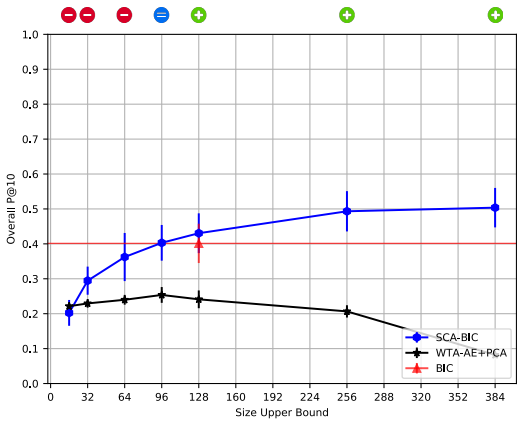
(e) ETH-80



(f) Supermarket Produce

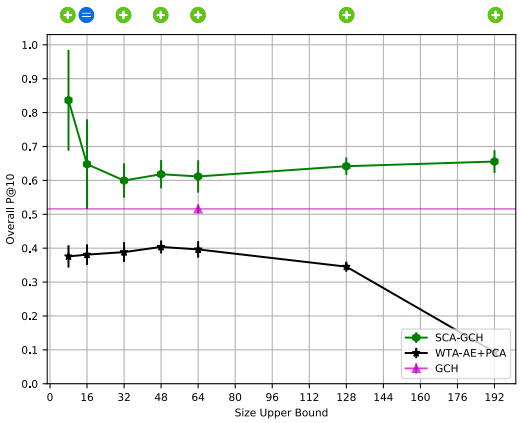


(g) MSRCORID

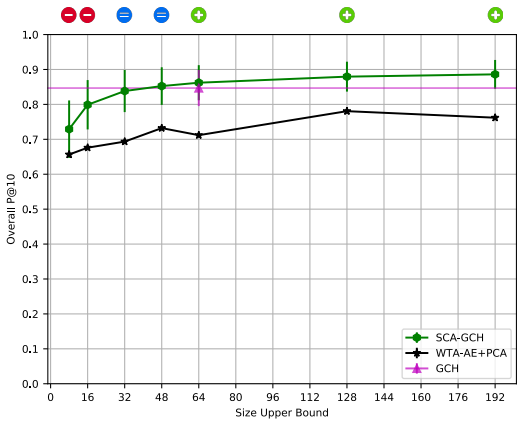


(h) UCMerced Land-use

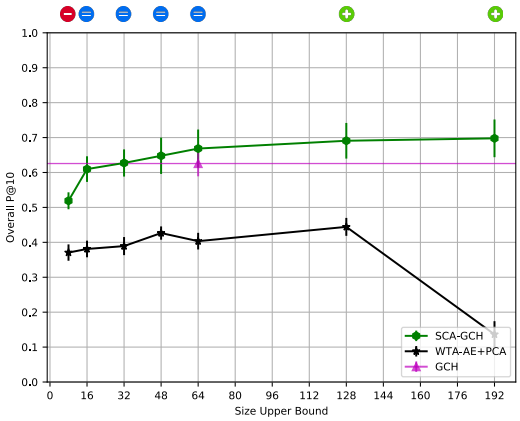
Figure 3.23. Comparison between the P@10 results of SCA, WTA Autoencoder and BIC feature extractor



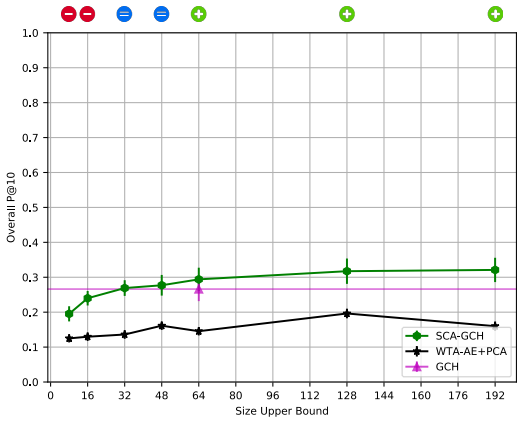
(a) Groundtruth



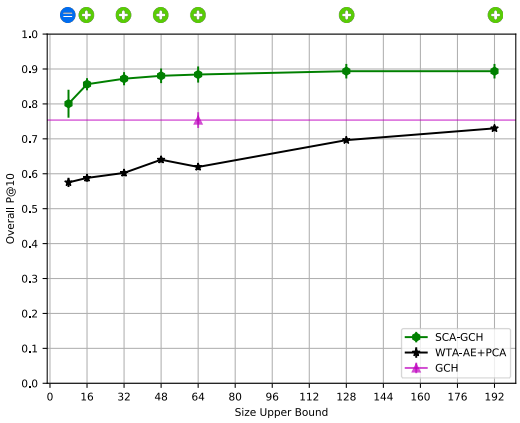
(b) Coil-100



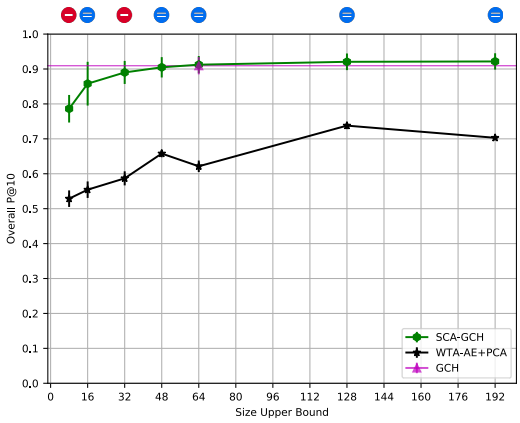
(c) Corel-1566



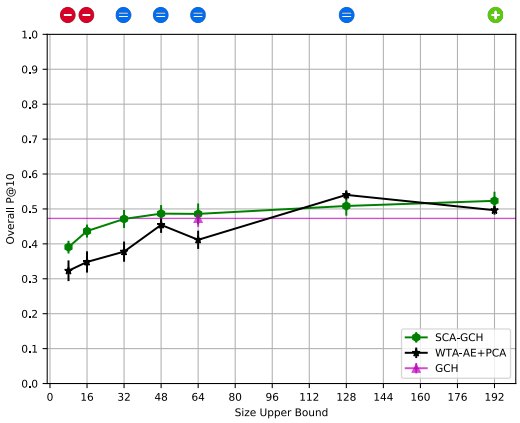
(d) Corel-3906



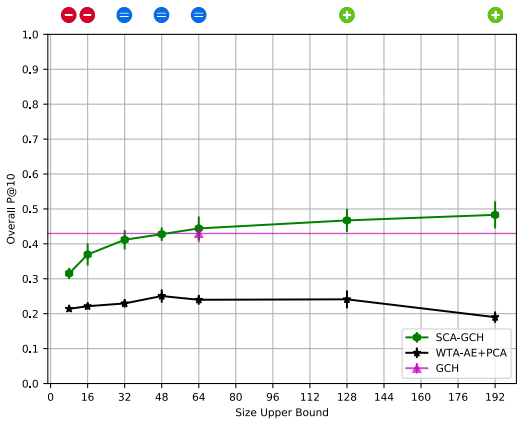
(e) ETH-80



(f) Supermarket Produce

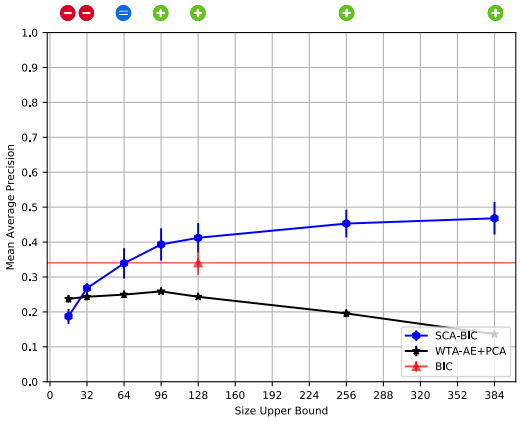


(g) MSRCORID

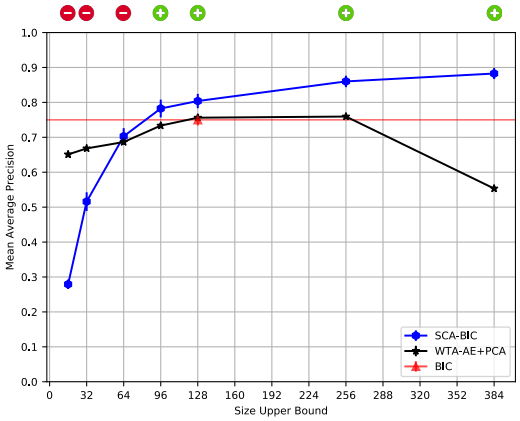


(h) UCMerced Land-use

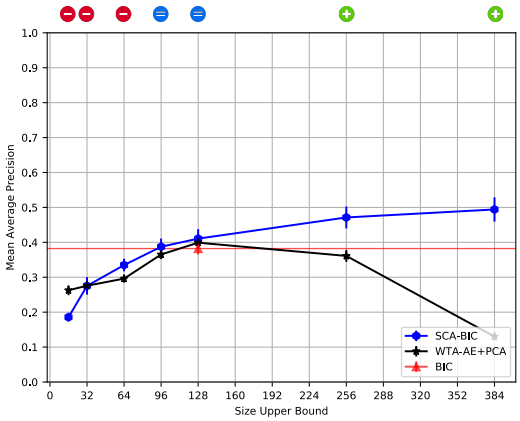
Figure 3.24. Comparison between the P@10 results of SCA, WTA Autoencoder and GCH feature extractor



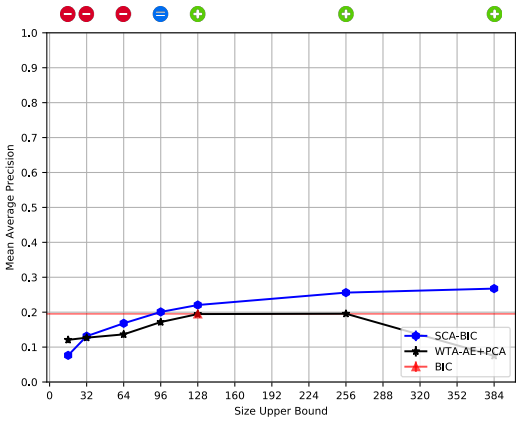
(a) Groundtruth



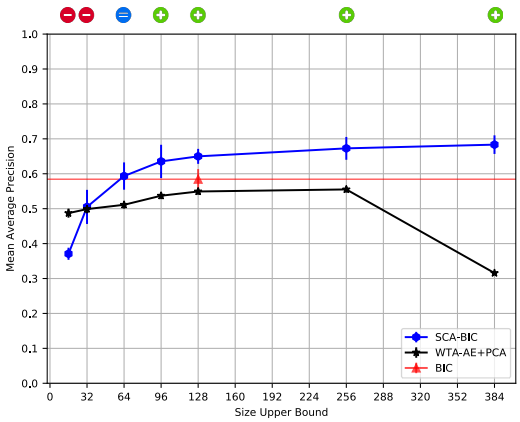
(b) Coil-100



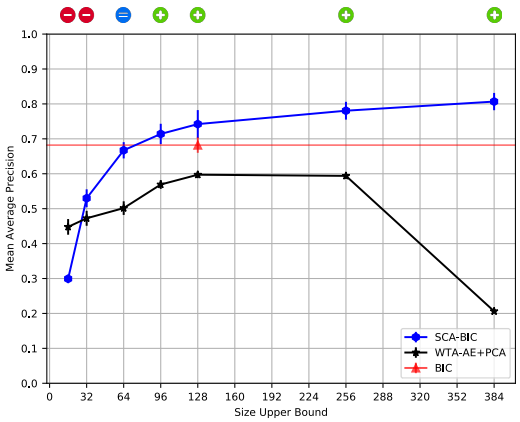
(c) Corel-1566



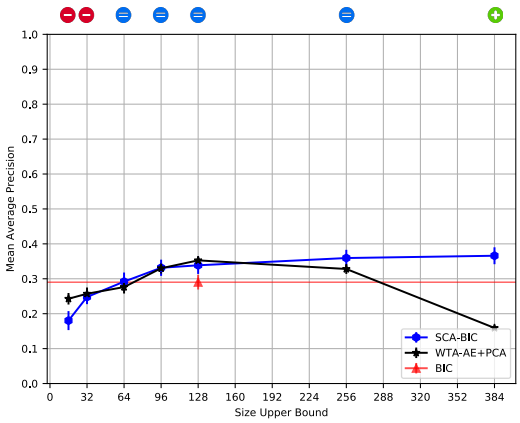
(d) Corel-3906



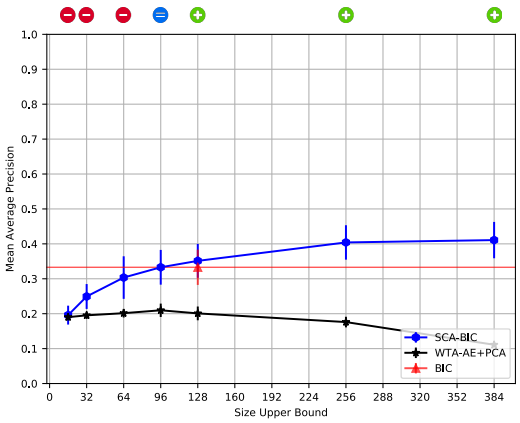
(e) ETH-80



(f) Supermarket Produce

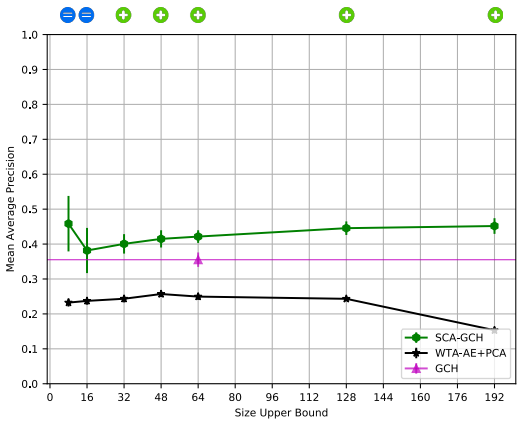


(g) MSRCORID

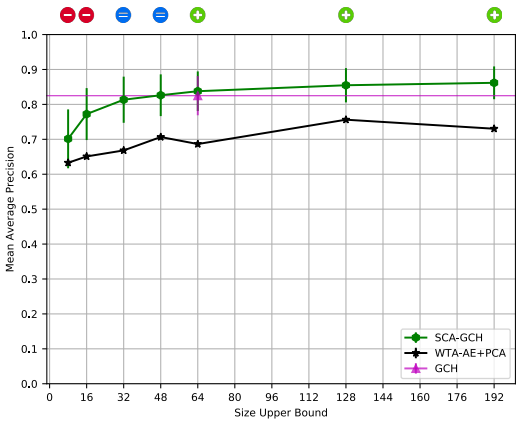


(h) UCMerced Land-use

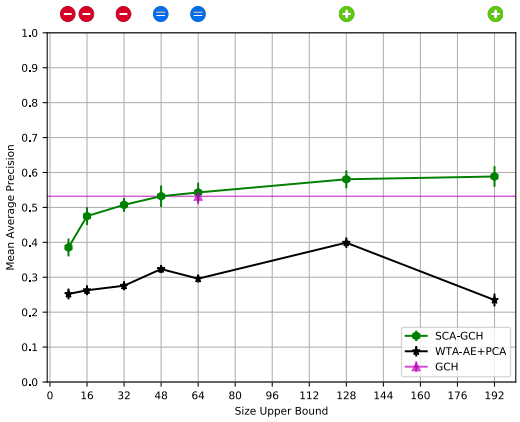
Figure 3.25. Comparison between the MAP results of SCA, WTA Autoencoder and BIC feature extractor



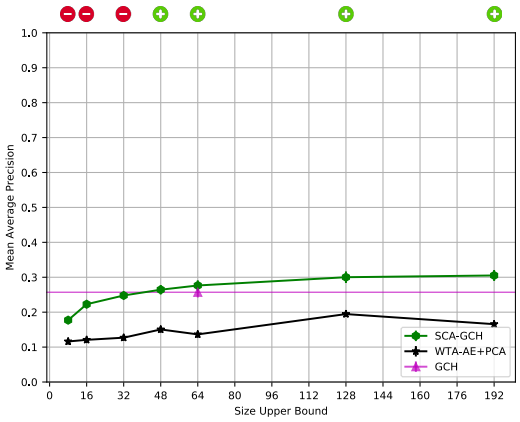
(a) Groundtruth



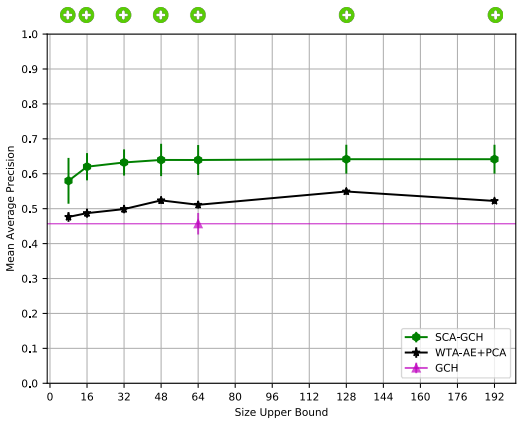
(b) Coil-100



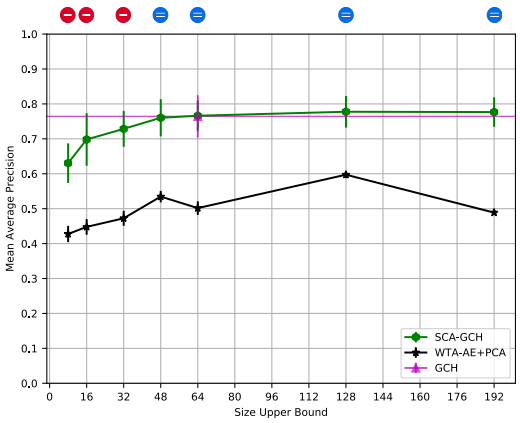
(c) Corel-1566



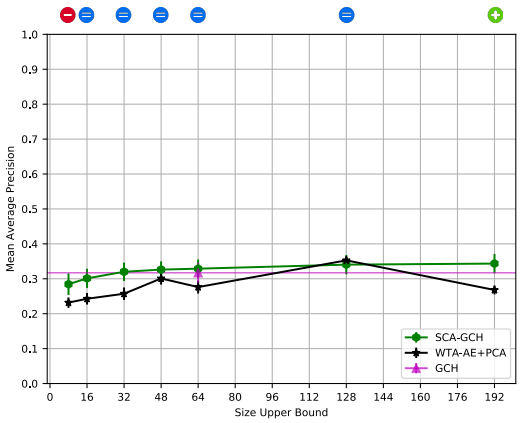
(d) Corel-3906



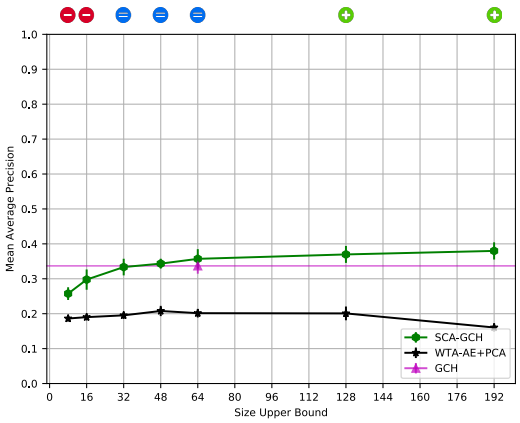
(e) ETH-80



(f) Supermarket Produce



(g) MSRCORID



(h) UCMerced Land-use

Figure 3.26. Comparison between the MAP results of SCA, WTA Autoencoder and GCH feature extractor

Chapter 4

Optimizing Deep Representations

4.1 Introduction

In recent years, deep neural networks (DNNs) have composed the state-of-the-art methodologies for several computer vision tasks including image classification [89], object detection [131], semantic segmentation [107], etc. However, they present high computational complexity and demand a huge amount of resources such as storage memory, working memory, computational power and energy consumption. Consequently, deploying these models can be a challenging task, specially in environments of limited resources such as embedded systems and mobile devices. Many works address this issue proposing different approaches of model compression [25, 78, 16], which include weight sharing, tensor decomposition, network pruning [12], neural architecture search [29, 117, 64, 90], efficient neural architecture design [108, 44, 14, 119, 35, 68], knowledge distillation [37], and quantization [54, 38].

In this chapter, we focus on quantization – an approach in which the model is compressed by reducing the bit-widths of weights and activations. Usually quantization represents the model with low precision format, transforming floating-point values and operations to fixed-point. Besides reduction in memory requirements, quantization usually results in the reduction of computing time and energy consumption. The challenge of quantization is to reduce the model requirements without compromising its capabilities in performing the task.

Mixed-Precision Quantization

Recent works on quantization produced state-of-the-art results using mixed precision quantization — a type of quantization in which the bit-widths are not constant across

the model (weights and activations). As methods that assign uniform bit-width across the whole model are not optimal, mixed precision quantization can optimize bit-width allocation and improve the representational ability of DNNs without additional computation costs. The portions of the network that have most impact on the performance are assigned more bits, while less influential portions are assigned fewer bits.

Mixed-precision results have motivated significant adoption in industry and academia. Emergent DNN hardware accelerators recently began to support mixed precision, specially at low bit-width (1-8 bits). For instance, in 2018, Apple started to support mixed precision for the neural network inference in its line of processors with the A12 Bionic chip, and NVIDIA introduced the Turing GPU architecture [81, 72] that supports 1-bit, 4-bit, 8-bit and 16-bit arithmetic operations. In the same year, NTNU and Xilinx Research Labs proposed bit-level flexible hardware designs such as BISMO [111, 112], which has a bit-serial multiplier that supports operations of 1 to 8 bits, and Georgia Tech and UC San Diego proposed BitFusion [99], which supports multiplications of 2, 4, 8 and 16 bits in a spatial manner.

Post-Training Quantization

The majority of literature on network quantization requires either training the model from scratch or employing a fine-tunning step in order to execute the quantization process. This approach, known as training-aware quantization, usually perform, at training time, calculations necessary to determine the final bit-widths. Consequently, these methods typically require access to the full training dataset, which may be unavailable, in many real-world scenarios, for reasons such as logistical hurdles, privacy and security concernments, etc. Good examples are medical data, bio-metric data, or user information used in recommendation systems and authentication procedures. Furthermore, the training/fine-tuning process is often time-consuming and computationally expensive. For instance, in online applications, where a model needs to be constantly updated on new data and deployed every few hours, there may not be enough time for recurrent training or fine-tuning. Therefore, it is desirable to apply quantization without inner training and access to the complete dataset.

A viable alternative can be found in post-training quantization: a methodological approach which executes quantization over already trained models and only needs a small set of samples to accomplish the whole process. Furthermore, by being incremental to training, this approach allows the direct usage of pre-trained models made available by the scientific community, providing reproducibility and instant-ready applicability.

Per-Channel Quantization

One of the most important properties of a quantization scheme is its granularity. A quantization approach can define one quantizer for the whole model, one for each layer (per-layer quantization) or one for each channel (per-channel quantization). Although per-channel settings tend to produce more complex quantizations, they provide a higher specialization to the weights and implicate in more effective representability.

Probably motivated by simplicity of implementation, works of the literature comprising post-training quantization mostly employ per-layer quantizers. However, experiments of the literature [54] show that quantizing model weights with a per-channel quantizer usually provides significantly superior results and produced accuracies close to floating point across a wide range of networks.

Multiobjective Optimization

When applying a quantization process in a neural network, one might be interested in improving several pertinent aspects of the model such as performance in the task, compactness, latency of decision making and energy consumption. So, in order to obtain better results, it is desirable that all aspects could be explicitly optimizable.

The majority of works of the literature about neural network quantization present optimization processes based on only one objective. Those that simultaneously optimize more than one metric employ a unification approach that scalirizes a vector of objectives into only one objective by averaging the objectives with a weight vector. This is a simple solution to use, but the obtained solution largely depends on the weight vector used in the scalarization process.

Determining the appropriate weight vector can be a difficult task as it must consider metrics normalization, shape of the search space as well as encoding the priority order of objectives. Moreover, it is not guaranteed that the optimal solutions to the unified objective correspond to the pareto-optimal solutions of the multiobjective search space. Typically, when used for finding multiple solutions, the unified method has to be applied many times to hopefully find a different solution at each simulation run.

Over the past two decades, evolutionary algorithms (EAs), such as NSGA-II [24] (refer to Section 2.3.2), have been used as alternative for finding multiobjective solutions. Since EAs work with a population of solutions, with an emphasis for moving toward the Pareto-optimal region, an EA can naturally find multiple Pareto-optimal solutions in one

single simulation run.

4.1.1 Contributions

In this chapter, we study the optimization of deep neural models through network quantization. We take into account the above considerations combining powerful approaches of mixed-precision, post-training per-channel quantization and multi-objective optimization to propose a data-cheap multi-solution methodology capable of provide fast, compact and accurate DNN models. We conduct experiments comparing all mixed-precision post-training methodologies of the available literature and found that our approach produced state-of-the-art compactness for post-training quantization and close to full-precision accuracy. It also maintained the best trade-off between accuracy and compactness among all baselines.

We therefore provide the following contributions:

1. We design a population-based multi-objective solution for post-training mixed-precision quantization.
2. We introduce a per-channel quantization approach capable of providing state-of-the-art compactness for post-training quantization and the best trade-off between accuracy and compactness among post-training mixed-precision methods.

4.2 Related Work

4.2.1 Quantization

Quantization methods can be roughly divided into two main families: quantization-aware training techniques and post-training quantization techniques.

Quantization-Aware Training

In quantization-aware training techniques, a model undergoes a (gradient-based) optimization process during which the quantization is computed. As the gradients of quantization functions are usually zero, most quantization-aware training techniques use the Straight-Through Estimator (STE) [45, 8] for estimating gradients for these functions and make backward-propagation feasible.

During training, the network weights are usually optimized in full-precision, its quantized versions are used in feed-forward to compute the loss and full-precision weights are then updated via back-propagation. After training, the full-precision values can be discarded. Works of the literature [109, 9, 2, 124] discuss training aspects and demonstrate theoretical analysis.

Quantization-aware training techniques mostly differ in their choice of quantizers: Uniform [30, 49, 110] or Non-uniform [6, 73, 128, 132] (clustering-based, logarithmic and others); the parametrization of the quantizers: choices of thresholds, bit-widths, step size, etc.; and details in their training procedure.

Post-Training Quantization

In post-training techniques, a trained full-precision model is quantized without finetuning or retraining. These methods usually need only a small set of examples or any data at all [54].

Recent efforts have been made to enhance the performance of post-training quantization. He et al. [40] used unlabeled data to re-estimate the statistical parameters of the batch normalization layer improving the accuracy of quantized models. Banner et al. [5] combined per-channel bit-allocation, bias correction, and analytical clipping for integer quantization. Choukroun et al. [53] employed linear quantization as a Minimum Mean Squared Error (MMSE) problem for weights and activations, allowing 4-bit precision inference in hardware of limited resources. Nagel et al. [75] proposed a data-free quantization method that equalizes the weights by making use of a scale-equivariance property of activation functions and corrects error biases introduced during quantization.

A key challenge in this area of research is to compress the model without significant performance loss. Post-training techniques usually suffer higher accuracy degradation in comparison to quantization-aware training, especially at high compression rates.

4.2.2 Mixed-Precision Quantization

Among the methods mentioned above, all the model weights or activations are treated equally and assigned the same bit-width. As the parameters of different portions of the network contribute differently to the overall results, the bit-widths should be determined for each portion. In mixed-precision quantization, the bit-widths are not constant across the model (weights and activations) and the precision is adjusted to each part according to its interference in quantization error or task performance. Mixed-Precision methods can also be grouped in quantization-aware training and post-training quantization.

Quantization-Aware Training

Mixed-precision quantization-aware works employed a variety of strategies to find the appropriate bit-width allocation. Wang et al. [116] employed reinforcement learning and hardware simulator feedback signals (latency and energy) to determine layers bit-widths. Dong et al. [27, 26] used second-order gradient information to determine the sensitivity of each layer and accordingly assign the bit-precision setting. Uhlich et al. [110] determined bit-widths by learnable parameters whose gradients are estimated using STE. Wu et al. [120] converted the bit-widths search into a network architecture search. In this solution, the search space of all possible quantization schemes becomes a search for a sub-graph in a super-net. Zhou et al. [133] theoretically analyzed the correlation between the quantization error and the model accuracy and then optimized the bit-width allocation for different layers. Fromm et al. [33] also determined bit-widths according to the quantization residual of a pre-trained network. Lin et al. [63] used the signal-to-quantization-noise ratio (SQNR) to measure the effect of quantization error and determined the bit-width for each layer.

Post-training Quantization

Recently, mixed-precision methods have been proposed for post-training quantization. Their most pertinent aspects are summarized in Table 4.1.

Table 4.1. Post-training Mixed-Precision Quantization methods. The cells in bold correspond to the best design choices for a quantization method according to the literature.

Method	Symmetry	Granularity on weights	Granularity on activations	Estimator Approach	Objective Function Principles	Objective Function Nature
GAQ [63]	Symmetric	Per-layer	**Per-layer**	Genetic Algorithm	Accuracy and Compression	Unified approach
EvoQ [123]	**Asymmetric**	Per-layer	Uniform (8 bits)	Evolutionary Search	Logits Distance	Mono-objective
ZeroQ [16]	**Asymmetric**	Per-layer	Uniform (6 bits)	Dynamic Programing	Accuracy and Compression	Unified approach
MGQ (ours)	**Asymmetric**	**Per-channel**	**Per-layer**	Genetic Algorithm	Accuracy and Compression	**Multi-objective**

Long et al. proposed GAQ [65], a layer-wise quantization method based on Genetic Algorithm that is retraining-free and requires a small evaluation dataset. However, GAQ employs symmetric quantization, which usually provides lower accuracy than the asymmetric choice [54]. Furthermore, although its optimization process considers both accuracy and model compression, these metrics are unified into one objective through a scalarization process, which is complex to set up correctly and not as effective as a truly multi-objective function.

Yuan et al. proposed EvoQ [126] that employs evolutionary search to achieve mixed-precision quantization with limited data. To improve the search efficiency, it uses a mutation operation guided by the sensitivity of model layers to quantization. EvoQ has an objective function that measures the output difference between the quantized model and the full-precision model. The authors argue that this objective allows model optimization with fewer samples, as model logits contain more information than output labels alone. However, minimizing logits distance does not necessarily translate in accuracy maintenance. This function leads to the reproduction of a larger portion of the model output than the fraction that provides the label decision. Consequently, the quantized model with the closest outputs may not be the one that provides more correct labels.

Cai et al. [15] proposed a zero-shot quantization method named ZeroQ. It optimizes the model using synthetic data engineered to match the statistics of batch normalization across different layers of the network. It employs the same bit-width across all activations, which provides worse levels of accuracy and compression than (per-layer or per-channel) customized quantizers, and, as well as GAQ, presents a unified objective, which is not ideal for multi-metric scenarios.

Our proposed method, MGQ, uses a population-based optimization process that provides truly multi-objective solutions through a Pareto-frontier scheme. Furthermore, in contrast to the former approaches, it employs per-channel quantization for layers, per-layer quantization for activations and asymmetric quantizers - the best symmetry and granularity scenarios for post-training quantization, in terms of model accuracy and compression, according to an empirical analysis of the literature [54].

4.3 Genetic Quantization on Deep Neural Networks

In this work, we introduce Multi-objective Genetic Quantization (MGQ) - a multi-objective Genetic-Algorithm-based method that learns optimized quantization schemes for deep neural networks. Section 4.3.1 defines the employed quantizer and section 4.3.2 describes the optimization process that searches for parameters of the quantizer, i.e., that

determines the bit-width settings to be used in each portion of weights and activations of the network.

4.3.1 Quantizer

Our methodology is designed to produce low-bit deep neural networks according to a quantization that comprise the following design choices:

I **Uniform:** A uniform quantization is a setting whose levels are equally spaced following a uniform distribution (see Section 2.4.1). According to Jain et al. [49], a uniform quantization is more amenable to hardware implementations as it facilitates integer operations. In this work, we aim to produce hardware-friendly models that can be deployed in dedicated hardware as is described in Section 4.1.

II **Asymetric:** An asymmetric quantization employs a range of the high-precision axis which is asymmetric with respect to the zero-point (see Section 2.4.2). Consequently, the quantization range is adjusted to the exacts numeric limits of the data. According to Krishnamoorthi [54], who performed experimental evaluations on several CNN architectures, in post-training scenarios, asymmetric quantization provides superior accuracy than symmetric settings.

III **Deterministic:** A deterministic quantization means that the high-precision values are projected to their nearest discrete levels (see Section 2.4.3). Although in a statistical sense, the stochastic projection is able to achieve better results due to the unbiased estimation, at inference, quantization is deterministic, causing a mismatch with training values. It is observed that due to this mismatch, stochastic quantization underperforms deterministic quantization [54].

According to the above properties we define the following quantizer:

$$Q(x) = round\left(\frac{x}{\Delta}\right) - z \tag{4.1}$$

$$\Delta = \frac{max(x) - min(x)}{2^b - 1} \tag{4.2}$$

$$z = round(min(x) * \Delta) \tag{4.3}$$

where x is a float-point tensor, Δ is the size of the quantization levels, z is the zero-point and b the employed bit-width. As a result, the original 32-bit high-precision values are mapped to unsigned integers within the range of $[0, 2^b - 1]$.

Another important aspect of a quantization scheme is its granularity, which can be uniform, per-layer of per-channel. When the granularity is uniform, a unique quantizer is defined for the whole representation. In per-layer quantization, there is one quantizer - with individual parameters: interval size (Δ) and offset (z) - for an entire tensor, while in per-channel quantization, there is one quantizer defined to each channel within the tensor. According to Krishnamoorthi [54], per-channel quantization in weights provided the best accuracy results in the majority of scenarios. For activations, according to works of the literature [54], per-layer quantization is the most adequate option as a per-channel setting would complicate the inner-product computations at the core of convolution and matrix-multiplication operations. Our method follows these recommendations.

4.3.2 Quantization Search

In order to obtain the most adequate bit-width setting for the weights and activations of a neural network model - which corresponds to the set of parameters b in Equation 4.2 for all quantizers - we employ a multi-objective evolutionary search inspired on the Non-Sorting Genetic Algorithm II [24] (NSGA-II).

4.3.2.1 Quantization Encoding

As individual encoding, we use a concatenation (B_a, B_w) where B_a and B_w are arrays of integers corresponding to the activation bit-widths and weights bit-widths, respectively. According to the adopted quantization granularities, B_a contains one value per layer of the model and B_w contains one value for each neuron.

4.3.2.2 Multi-objective GA-based Search

Algorithm 5 illustrates the proposed GA-based quantization. The population starts with individuals created randomly (line 4). The population evolves generation by generation through genetic operations (line 5). A function is used to assign the fitness value for each individual (lines 6-8), i.e., to assess how well an individual solves the target problem. According to the non-dominating elitism operation, S_e is updated by adding the individuals from P that are not dominated by individuals of S_e (line 9). Then, individuals from P are selected according to the NSGA-II selection operation (line 10). After that, the next generation is formed from the union of the resulting individuals from the operations of mutation and cross-over over the selection and those selected by elitism (line 11). If the stopping condition (discussed on Section 4.4.5) was met, the iterations stop (lines 12-14). At the end, S_e contains a set of the best non-dominating individuals of all generations (line 16).

Our multi-objective fitness function is composed by the metrics Compression Ratio and Classification Accuracy described in Section 4.4.4. For details regarding genetic operators (cross-over, mutation, NSGA-II selection and non-dominating elitism), refer to Section 4.3.2.3 and regarding the employed multi-objective fitness function $multi\_fitness$, refer to Section 4.4.5.

Algorithm 5 Multi-objective quantization search

```
1   Let T be a training set of n_t samples
2   Let B be a set of bit-widths
3   Let P, S_e e S_s be sets of pairs (q, fitness_q), where q and fitness_q are an individual
    and its fitness, respectively
4   P ← Initial random population of individuals (with values of B)
5   For each generation g of N_g generations do
6       For each individual q ∈ P do
7           fitness_q ← multi_fitness(q, T)
8       End For
9       S_e ← non_dominating_elitism(P, S_e)
10      S_s ← nsga2_selection(P)
11      P ← mutation(S_s,B) ∪ crossover(S_s)
12      If stopping condition is met
13          Break outer loop
14      End If
15  End For
16  Return S_e
```

4.3.2.3 Genetic Operators

Algorithm 5 employs the following GA operators:

- **Cross-over**: Our method employs the Two-point cross-over operator, which selects two points of an individual and exchanges the portion in between with another individual. In this operation, B_a and B_w behave as separated individuals. For instance, crossing-over two individuals $(B1_a, B1_w)$ and $(B2_a, B2_w)$,the operation is independently executed between $B1_a$ and $B2_a$ and between $B1_w$ and $B2_w$.

- **Mutation**: Our method employs the Uniform mutation operator, which exchanges the selected positions with a random option of the set of possible bit-widths B. In this operation, B_a and B_w behave as separated individuals. For instance, in the mutation of an individual $(B1_a, B1_w)$, both $B1_a$ and $B1_w$ have values randomly changed.

- **Non-dominating Elitism**: This operation aims to select the best individuals ever generated through the GA search according to the Pareto-dominance criteria. It sorts the set of individuals composed by $P \cup S_e$ and selects all the Pareto non-dominated individuals.

- **NSGA-II Selection**: This operation selects all the individuals of the most advanced Pareto frontiers towards the directions that optimize the objectives in the search space. Consequently, the GA search tends to generate individuals each time closer to the optimal frontier. For a detailed explanation of the NSGA-II selection of individuals, please refer to the description provided in Section 2.3.2.

4.3.3 Selecting Final Solution

As well as NSGA-II, our quantization search employs a Pareto non-dominating strategy and produces a set of solutions as output. It allows that one chooses the most appropriate solution according to his/her needs. For example, it could be by selecting the solution of higher/lower value on a specific most-pertinent metric or by imposing the most appropriate lower/upper bounds for the support of specific hardware.

In this work, we employ a strategy to select the solution of best trade-off among the objectives by using the Hypervolume indicator [98] (Equation 4.4). We measure the Hypervolume Contribution (Equation 4.5) of each solution p of the generated solution

set S. This function quantifies the contribution to the Pareto-frontier, i.e. the solution that most advances it towards the objectives optimization gradient. We, therefore, select the most contributing solution p^* as the final answer (Equation 4.6), using the following equations:

$$p^* = \max_{\forall p \in S} HC(S, p) \tag{4.4}$$

$$HC(S, p) = H(S \cup \{p\}) - H(S \setminus \{p\}) \tag{4.5}$$

$$H(S) = \Lambda \left(\bigcup_{p \in S\,;\, p < r} [p, r] \right) \tag{4.6}$$

where $[p, r] = \{q \in R^d | p \leq q \text{ and } q \leq r\}$ denotes the box delimited below by $p \in S$ and above by a reference point r. The function $\Lambda(\cdot)$ denotes the Lebesgue measure [32].

4.4 Experimental Setup

In this section, we present the adopted experimental setup, which includes the employed image dataset (Section 4.4.1) , the configuration of parameters of the method (Section 4.4.5), the baselines used for comparative analysis (Section 4.4.3) and the metrics used to evaluate the effectiveness and compactness of the produced quantized models (Section 4.4.4).

4.4.1 Dataset

We conducted experiments using one of the most famous datasets of the computer vision literature: the Imagenet dataset [56], an image dataset with more than 1 Million samples grouped in 1000 categories. Its train, validation and test sets contain respectively 1.281.167, 50.000, and 100.000 samples. In our methodology, the quantization process was performed using the validation set and final results were computed using the test set.

4.4.2 Deep Neural Network Models

Table 4.2. Statistics on CNN Architectures employed in the experiments.

CNN Architecture	Torchvision Name	# Layers	# Neurons
ShuffleNetV2 [68]	shufflenet_v2_x0_5	116	4976
MobileNetV2 [96]	mobilenetv3_small_wd2	107	18056
ResNet18 [39]	resnet18	45	5800
ResNet50 [39]	resnet50	111	27560

We evaluated the model quantization on four architectures of convolutional neural networks: ShuffleNetV2 [68], MobileNetV2 [96], ResNet18 [39] and ResNet50 [39], detailed in Table 4.2. Their implementations and pretrained weights were imported from Torchvision framework [71]. These architectures were chosen because they are often employed in the experimental evaluations of works of the model quantization literature (Section 4.2) facilitating comparison with baselines.

4.4.3 Baselines

In order to experimentally evaluate our method, we performed the comparison with all the post-training approaches available in the literature that employ mixed-precision quantization: EvoQ [126], GAQ [65] and ZeroQ [15]. Please refer to Section 4.2.2 for more details. We also present results regarding the full-precision (32-bits) pre-trained models before quantization.

4.4.4 Evaluation Metrics

- **Compression Ratio (CR)**: This ratio measures the size proportion, in bytes, between the quantized model M_q and the original full-precision model M_{fp}. It is defined according to the following equation:

$$Compression\_Ratio(M_q, M_{fp}) = \frac{|M_{fp}|}{|M_q|} \tag{4.7}$$

- **Classification Accuracy**: In a set of classified samples S, this metric measures the proportion of correctly-predicted labels to the total amount of evaluated samples. It is defined according to the following equation:

$$Accuracy(S) = \frac{\#\ of\ correct\ labels}{|S|} \tag{4.8}$$

- **Accuracy Loss (AL)**: This metric is a proportion (%) between the difference of an accuracy rate to the respective reference value. In this work, it is used to quantify the accuracy loss caused by the model quantization process. Consequently, the employed accuracy (A) and reference value (A_{ref}) refer to the quantized model and original model, respectively. It is defined according to the following equation:

$$Accuracy\_Loss(A, A_{ref}) = 1 - \frac{A}{A_{ref}} \tag{4.9}$$

4.4.5 Parameters

Table 4.3 presents the values adopted for the GA-based quantization learning process. The values for population size, cross-over and mutation were chosen empirically. Initially, it was applied a parameter search according to a 2^k Fractional Factorial Design (please refer to item 16.3.3 of [48]) over a portion of the dataset. For the parameters which presented major sensitivities, a binary search was employed for the exploration of different values.

The total number of generations was defined aiming to ensure convergence of the evolutionary algorithm. However, we empirically observed that typically the best fitness values may stop improving at earlier iterations. Thus, one might impose a stopping condition regarding the fitness value as an option to avoid unnecessary iterations.

The set of possible bit-widths B was defined to allow the use of the resulting quantized models in mixed-precision hardware designs. Mixed-precision integer operations with low bit-widths are supported by specialized hardware such as BISMO [111, 112] and BitFusion [99].

The multi-objective fitness function is composed by the metrics Compression Ratio (Eq. 4.7) and Classification Accuracy(Eq. 4.8), which are described in Section 4.4.4. The accuracy values were computed by evaluating one batch of 224x224 image samples of size n_t which assumes the values {1000, 1100, 500, 200} for ShuffleNetV2, MobileNetV2, ResNet18 and Resnet50, respectively. We observed that larger evaluation batches tend to improve the results or at least facilitate the quantization search. With that in mind,

in our experiments, the batch sizes were stipulated considering all the remaining space available in a GPU memory of 10GB after the loading of the respective network model and inference buffers.

Table 4.3. Multi-objective Genetic algorithm parameters. The indicated variables refer to Algorithm 5 and individual encoding described in Section 4.3.2.1.

Parameter	Value
Set of possible bit-widths (B)	{2,3,4,5,6,7,8}
Two-point Cross-over Probability ($B_w \vert B_a$)	60% | 60%
One-point Mutation Probability ($B_w \vert B_a$)	40% | 40%
Number of Generations (N_g)	400
Population Size	200

4.5 Results and Discussion

Table 4.4. Compression Rate (CR), Accuracy and Accuracy Loss (AC) results for ShuffleNetV2 and MobileNetV2 architectures on Imagenet. The best results are indicated in **bold**. At the second column, we indicate the bit-widths for weights (W) and activations (A) or weather the method employ mixed-precision (MP). The Accuracy columns present respectively the quantized and full-precision accuracies.

Method	Bit-widths		ShuffleNetV2			MobileNetV2		
	W	A	CR	Accuracy	AL	CR	Accuracy	AL
EvoQ [126]	MP	MP	**8.00**	66.39 (69.36)	4.28	7.51	68.90 (71.88)	4.15
ZeroQ [15]	MP	6	5.35	62.90 (65.87)	4.51	5.34	72.85 (73.03)	**0.25**
MGQ (ours)	MP	MP	7.82	65.55 (67.71)	**3.19**	**7.70**	71.30 (71.87)	0.79
Full	32	32	9.2MB			13.4MB		

In the performed experiments, we evaluate the quantization performed by our method and compare it with the original full-precision models pre-trained on Imagenet dataset [56] and with the baselines. We apply all these quantization approaches in recent architectures of convolutional neural networks and evaluate the quantized models in terms of classification accuracy and model compression. Table 4.4 present the results for ShuffleNetV2 and MobileNetV2, and Table 4.5 for ResNet18 and ResNet50.

Observing the compression ratios, it is possible to conclude that MGQ and EvoQ offer the best model compression among post-training mixed-precision methods, with MGQ proving the higher ratios in the majority of scenarios. The accuracy losses indicate

Table 4.5. Compression Rate (CR), Accuracy and Accuracy Loss (AC) results for ResNet18 and ResNet50 architectures on Imagenet. The best results are indicated in **bold**. At the second column, we indicate the bit-widths for weights (W) and activations (A) or weather the method employ mixed-precision (MP). The Accuracy columns present respectively the quantized and full-precision accuracies.

Method	Bit-widths		ResNet18			ResNet50		
	W	A	CR	Accuracy	AL	CR	Accuracy	AL
EvoQ [126]	MP	MP	8.00	68.55 (69.76)	1.73	8.00	75.51 (76.15)	0.84
GAQ [65]	MP	8	6.16	68.90 (70.40)	2.13	5.79	74.60 (76.40)	2.36
ZeroQ [15]	MP	6	5.34	71.30 (71.47)	**0.24**	5.33	77.43 (77.72)	**0.37**
MGQ (ours)	MP	MP	**8.24**	69.02 (69.76)	1.06	**9.01**	75.48 (76.15)	0.88
Full	32	32	44.6MB			97.5MB		

that, although ZeroQ usually provides the best accuracy maintenance, MGQ did not stand far behind. Furthermore, its accuracy results achieved significantly-close levels to full-precision accuracy. Comparing to MGQ, EvoQ usually provides higher accuracy losses and ZeroQ achieves consistently lower model compression. Consequently, it is possible to conclude that MGQ holds the best accuracy-compression trade-off among the studied methods.

4.6 Conclusions

As stated in Section 4.1, we propose an approach for DNN quantization combining the following aspects:

- Mixed-Precision Quantization, as it allows custom bit-widths for the each portion of the model improving representability and, consequently, enhancing accuracy and compactness;
- Post-training Quantization, which allows quantization of pre-trained models and requires only a small portion of data, being useful in situations where the dataset is not available or when retraining is not viable;
- Per-channel Quantization, which usually provides superior results than the alternatives and produces accuracies close to floating point across a wide range of networks, due to a higher customization to the model weights;
- Population-based Multi-objective Optimization, which allows the optimization of a

model according to several pertinent metrics and can naturally find multiple Pareto-optimal solutions in one single simulation run.

We proposed a method, called MGQ, that employs a Multi-objective Genetic Algorithm to define the bit-widths for a Uniform, Asymetric and Deterministic quantization scheme. It is the first method of the literature to combine: mixed-precision, assymetric quantization, per-channel quantization for weights, per-layer quantization for activations and multi-objetive optimization. These are the most sucessceful design choices for post-training model quantization evaluated by the literature so far.

We evaluated our method on four CNN architectures using the Imagenet dataset and showed that conclude that MGQ provided the best accuracy-compression trade-off among the literature of post-training mixed-precision quantization.

4.6.1 Future Work

Different Combinations of Objectives. Although MGQ allows you to select the most appropriate pruned model for a given scenario from a set of optimized solutions, one might benefit from models optimized according to other combinations of objectives. A customised configuration of objectives would make possible a search dedicated to each scenario requirements thus presenting greater potential to find even better results. With that in mind, we plan to evaluate MGQ with different combinations of objectives such as: {maximize accuracy, upper bound runtime memory, upper bound latency}, for cases in which there is low limit resources such as mobile environments; {maximize accuracy, upper bound runtime memory, minimize latency} for cases in which it is desirable to have user-friendly response such as smartphone applications; and {maximize accuracy, upper bound latency} for real-time applications.

Combining Approaches of Model Compression. The model optimization provide by MGQ could benefit from a combination of with other approaches of model compression such as network pruning and knowledge distillation. Furthermore, the approaches that can be addressed as combinatorial problems could be solved with the Genetic Algorithm, increasing its integration with our method.

Chapter 5

Final Conclusions

After the conducted experiments, we were able to answer the research questions proposed in Chapter 1:

I **Can we produce improved representations by optimizing those that already exist (e.g. hand-designed representations and pre-trained Deep Learning models) instead of learning them from scratch?**
Yes. In Chapter 3, we proposed two approaches that improved the compactness and effectiveness of hand-designed shallow representations: Unconstrained Approach (UA) and Sized-constrained Approach (SCA). The former consistently improved the precision (P@10 and MAP) in an image retrieval task for all tested scenarios, with 10%-increasing in the majority of them. The latter was capable of compacting the original representations until 50% maintaining the initial precision. Furthermore, in Chapter 4, we studied a class of methods that compact pre-trained deep models by reducing their parameters bit-widths, making them more resource-efficient. For instance, our proposed approach (MGQ) compacted models to at most 13% of their size with drops of less than 2% of the original classification accuracy.

II **How well GA-based Representation Learning performs against the current state-of-the-art, i.e., Deep Learning methods?**
Considering the study of shallow representations presented in Chapter 3, the experiments comparing the representation learning approaches in content-based image-retrieval show considerable superior results of the GA-based approaches (UA and SCA) against the Deep-Learning-based baselines (WTA-AE and Alexnet) in the majority of scenarios. Both UA-BIC and UA-GCH obtained from 10% to 30% more in precision than WTA-AE and Alexnet with 60x smaller representations. SCA-BIC obtained superior precisions than WTA-AE and Alexnet at 96,128,256,384 representation sizes. SCA-GCH had at least 10% superior precision at all sizes (8,16,32,48,64,128,192) showing that these approaches highly suffer at extremely low dimensionality. These results show superior representability of GA-learned representations over deep-learned ones in the context of image retrieval.

III **Is it possible to optimize already existing representations (e.g. hand-designed representations and pre-trained Deep Learning models) to be simultaneously more compact and more effective in their task?**

Yes. SCA-BIC and SCA-GCH, which are incremental representation learning approaches and, consequently, employ optimization over already existing image representations, provide results reducing the representation size and improving retrieval precision. SCA-BIC provided representations of size 96 (75% of the original size) that have superior P@10 and MAP for Groundtruth, Coil-100, ETH-80, Supermarket-Produce and MSRCORID datasets (Figs. 3.23 and 3.25). SCA-GCH provided representations of size 48 (75% of the original size) that have superior P@10 and MAP for Groundtruth and ETH-80 datasets (Figs. 3.24 and 3.26).

Bibliography

[1] (2007). Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3 – 13.

[2] Alizadeh, M., Fernández-Marqués, J., Lane, N. D., and Gal, Y. (2018). An empirical study of binary neural networks' optimisation. In *International Conference on Learning Representations*.

[3] Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[4] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37--49.

[5] Banner, R., Nahshan, Y., and Soudry, D. (2019). Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, pages 7950--7958.

[6] Baskin, C., Schwartz, E., Zheltonozhskii, E., Liss, N., Giryes, R., Bronstein, A. M., and Mendelson, A. (2018). Uniq: Uniform noise injection for non-uniform quantization of neural networks. *arXiv preprint arXiv:1804.10969*.

[7] Bengio, Y., Courville, A., and Vincent, P. (2013a). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[8] Bengio, Y., Léonard, N., and Courville, A. (2013b). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

[9] Bethge, J., Bornstein, M., Loy, A., Yang, H., and Meinel, C. (2018). Training competitive binary neural networks from scratch. *arXiv preprint arXiv:1812.01965*.

[10] Bharti, V., Biswas, B., and Shukla, K. K. (2020). Recent trends in nature inspired computation with applications to deep learning. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 294--299. IEEE.

[11] Bhunia, A. K., Bhattacharyya, A., Banerjee, P., Roy, P. P., and Murala, S. (2019). A novel feature descriptor for image retrieval by combining modified color histogram and

diagonally symmetric co-occurrence texture pattern. *Pattern Analysis and Applications*, pages 1--21.

[12] Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. (2020). What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.

[13] Bo, L., Ren, X., and Fox, D. (2011). Hierarchical matching pursuit for image classification: Architecture and fast algorithms. *Advances in neural information processing systems*, pages 2115--2123.

[14] Cai, H., Zhu, L., and Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

[15] Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. (2020). Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169--13178.

[16] Choudhary, T., Mishra, V., Goswami, A., and Sarangapani, J. (2020). A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, pages 1--43.

[17] Coates, A. and Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921--928.

[18] Criminisi, A. (2004). Microsoft research cambridge object recognition image database. available online[1].

[19] da S. Torres, R. and Falcão, A. X. (2006). Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada (RITA)*, 13(2):161--185.

[20] da S. Torres, R., Falcão, A. X., Gonçalves, M. A., Papa, J. P., Zhang, B., Fan, W., and Fox, W. A. (2009). A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):283 – 292.

[21] Darwish, A., Hassanien, A. E., and Das, S. (2020). A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review*, 53(3):1767--1812.

[22] Davis, L. (1991). *Handbook of Genetic Algorithms*. VNR Computer Library VNR Computer Library. Van Nostrand Reinhold. ISBN 9780442001735.

[1] `https://www.microsoft.com/en-us/research/project/image-understanding/`

[23] Davis, S. M., Landgrebe, D. A., Phillips, T. L., Swain, P. H., Hoffer, R. M., Lindenlaub, J. C., and Silva, L. F. (1978). Remote sensing: the quantitative approach. *New York, McGraw-Hill International Book Co., 1978. 405 p.*

[24] Deb, K. (2002). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-2. *IEEE Transactions in Evolutionary Computation*, 6(2):182--197.

[25] Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485--532.

[26] Dong, Z., Yao, Z., Cai, Y., Arfeen, D., Gholami, A., Mahoney, M. W., and Keutzer, K. (2019a). Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *arXiv preprint arXiv:1911.03852*.

[27] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. (2019b). Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 293--302.

[28] dos Santos, J. A., Penatti, O. A. B., and da Silva Torres, R. (2010). Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification. *VISAPP (2)*, pages 203--208.

[29] Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

[30] Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. (2019). Learned step size quantization. *arXiv preprint arXiv:1902.08153*.

[31] Fan, W., Fox, E. A., Pathak, P., and Wu, H. (2004). The effects of fitness functions on genetic programming-based ranking discovery for web search. *Journal of the American Society for Information Science and Technology*, 55(7):628--636.

[32] Fleischer, M. (2003). The measure of pareto optima applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519--533. Springer.

[33] Fromm, J., Patel, S., and Philipose, M. (2018). Heterogeneous bitwidth binarization in convolutional neural networks. *Advances in Neural Information Processing Systems*, 31:4006--4015.

[34] García-Lamont, F., Cervantes, J., López-Chau, A., and Ruiz-Castilla, S. (2020). Color image segmentation using saturated rgb colors and decoupling the intensity from the hue. *Multimedia Tools and Applications*, 79(1-2):1555--1584.

[35] Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., and Keutzer, K. (2018). Squeezenext: Hardware-aware neural network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1638--1647.

[36] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

[37] Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2020). Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*.

[38] Guo, Y. (2018). A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*.

[39] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770--778.

[40] He, X. and Cheng, J. (2018). Learning compression from limited unlabeled data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 752--769.

[41] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527--1554.

[42] Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3--10.

[43] Hinz, T., Navarro-Guerrero, N., Magg, S., and Wermter, S. (2018). Speeding up the hyperparameter optimization of deep convolutional neural networks. *International Journal of Computational Intelligence and Applications*, 17(02):1850008.

[44] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314--1324.

[45] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869--6898.

[46] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1424–1437.

[47] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

[48] Jain, R. (1991). The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling. In *Wiley professional computing*.

[49] Jain, S. R., Gural, A., Wu, M., and Dick, C. H. (2019). Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *arXiv preprint arXiv:1903.08066*.

[50] Khaldi, B., Aiadi, O., and Kherfi, M. L. (2019). Combining colour and grey-level co-occurrence matrix features: a comparative study. *IET Image Processing*, 13(9):1401–1410.

[51] Kim, T. K. (2015). T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540--546.

[52] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[53] Kravchik, E., Yang, F., Kisilev, P., and Choukroun, Y. (2019). Low-bit quantization of neural networks for efficient inference. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0--0.

[54] Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

[55] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*.

[56] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[57] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436--444.

[58] Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:II--409.

[59] Li, T., Leng, J., Kong, L., Guo, S., Bai, G., and Wang, K. (2019a). Dcnr: deep cube cnn with random forest for hyperspectral image classification. *Multimedia Tools and Applications*, 78(3):3411--3433.

[60] Li, X., Li, D., Peng, L., Zhou, H., Chen, D., Zhang, Y., and Xie, L. (2019b). Color and depth image registration algorithm based on multi-vector-fields constraints. *Multimedia Tools Appl.*, 78(17):24301--24319.

[61] Li, Y. (2005). *Object and Concept Recognition for Content-based Image Retrieval*. PhD dissertation, University of Washington, Seattle, WA, USA.

[62] Li, Y. and Shapiro, L. G. (2002). Consistent line clusters for building recognition in cbir. *Proceedings of the International Conference on Pattern Recognition*.

[63] Lin, D., Talathi, S., and Annapureddy, S. (2016). Fixed point quantization of deep convolutional networks. In *International conference on machine learning*, pages 2849--2858.

[64] Liu, Y., Sun, Y., Xue, B., Zhang, M., and Yen, G. (2020). A survey on evolutionary neural architecture search. *arXiv preprint arXiv:2008.10937*.

[65] Long, Y., Lee, E., Kim, D., and Mukhopadhyay, S. (2020). Q-pim: A genetic algorithm based flexible dnn quantization method and application to processing-in-memory platform. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1--6.

[66] Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823--870.

[67] Luccheseyz, L. and Mitray, S. (2001). Color image segmentation: A state-of-the-art survey. *Proceedings of the Indian National Science Academy (INSA-A)*, 67(2):207--221.

[68] Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116--131.

[69] Makhzani, A. and Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*.

[70] Makhzani, A. and Frey, B. J. (2015). Winner-take-all autoencoders. In *Advances in neural information processing systems*, pages 2791--2799.

[71] Marcel, S. and Rodriguez, Y. (2010). Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485--1488.

[72] Markidis, S., Der Chien, S. W., Laure, E., Peng, I. B., and Vetter, J. S. (2018). Nvidia tensor core programmability, performance & precision. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 522--531.

[73] Miyashita, D., Lee, E. H., and Murmann, B. (2016). Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*.

[74] Mohseni, S. A., Wu, H. R., Thom, J. A., and Bab-Hadiashar, A. (2020). Recognizing induced emotions with only one feature: A novel color histogram-based system. *IEEE Access*, 8:37173–37190.

[75] Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. (2019). Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1325--1334.

[76] Nakamura, R., Fonseca, L., dos Santos, J. A., Torres, R. d. S., Yang, X.-S., and Papa, J. P. (2014). Nature-inspired framework for hyperspectral band selection. *IEEE Transactions on Geoscience and Remote Sensing*, 52(4):2126--2137.

[77] Nayar, S. K., Nene, S. A., and Murase, H. (1996). Real-time 100 object recognition system. *Proceedings of IEEE International Conference on Robotics and Automation*, 3:2321–2325 vol.3.

[78] Neill, J. O. (2020). An overview of neural network compression. *arXiv preprint arXiv:2006.03669*.

[79] Ng, A. (2011). Sparse autoencoder. Available online[2].

[80] Nogueira, K., Penatti, O. A., and dos Santos, J. A. (2017). Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539 – 556.

[81] Nvidia, C. (2018). Nvidia turing gpu architecture. Available online[3].

[82] Omran, M. G., Engelbrecht, A. P., and Salman, A. (2005). A color image quantization algorithm based on particle swarm optimization. *Informatica*, 29(3).

[83] Penatti, O. A., Valle, E., and Torres, R. d. S. (2012). Comparative study of global color and texture descriptors for web image retrieval. *Journal of visual communication and image representation*, 23(2):359--380.

[84] Penatti, O. A. B. and d. S. Torres, R. (2008). Color descriptors for web image retrieval: A comparative study. *2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*, pages 163–170.

[2] `https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf`

[3] `https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf`

[85] Pereira, E. M., Torres, R. d. S., and dos Santos, J. A. (2021). A genetic algorithm approach for image representation learning through color quantization. *Multimedia Tools and Applications*. Available from: <http://dx.doi.org/10.1007/s11042-020-10194-z>.

[86] Pérez-Delgado, M.-L. (2019). The color quantization problem solved by swarm-based operations. *Applied Intelligence*, 49(7):2482--2514.

[87] Ponti, M., Nazaré, T. S., and Thumé, G. S. (2016). Image quantization as a dimensionality reduction procedure in color and texture feature extraction. *Neurocomputing*, 173:385--396.

[88] Ranzato, M., Poultney, C., Chopra, S., and Cun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1137--1144. MIT Press.

[89] Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352--2449.

[90] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. (2020). A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*.

[91] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840. Omnipress.

[92] Rocha, A., Hauagge, D. C., Wainer, J., and Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. *Computers and Electronics in Agriculture*, 70(1):96--104.

[93] Rodriguez-Coayahuitl, L., Morales-Reyes, A., and Escalante, H. J. (2019). Evolving autoencoding structures through genetic programming. *Genetic Programming and Evolvable Machines*, 20(3):413--440.

[94] Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. *Artificial Intelligence and Statistics*, pages 448--455.

[95] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

[96] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510--4520.

[97] Scheunders, P. (1996). A genetic lloyd-max image quantization algorithm. *Pattern Recognition Letters*, 17(5):547--556.

[98] Shang, K., Ishibuchi, H., He, L., and Pang, L. M. (2020). A survey on the hypervolume indicator in evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*.

[99] Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Chandra, V., and Esmaeilzadeh, H. (2018). Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 764--775.

[100] Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., and Aleksic, M. (2018). A quantization-friendly separable convolution for mobilenets. In *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, pages 14--18. IEEE.

[101] Smeulders, A. W., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12):1349--1380.

[102] Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24--35.

[103] Stehling, R. O., Nascimento, M. A., and Falcão, A. X. (2002). A compact and efficient image retrieval approach based on border/interior pixel classification. *International Conference on Information and Knowledge Management*, pages 102--109.

[104] Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

[105] Suganuma, M., Shirakawa, S., and Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497--504.

[106] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International journal of computer vision*, 7(1):11--32.

[107] Taghanaki, S. A., Abhishek, K., Cohen, J. P., Cohen-Adad, J., and Hamarneh, G. (2020). Deep semantic segmentation of natural and medical images: A review. *Artificial Intelligence Review*, pages 1--42.

[108] Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

[109] Tang, W., Hua, G., and Wang, L. (2017). How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 2625--2631.

[110] Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J. A., Tiedemann, S., Kemp, T., and Nakamura, A. (2019). Mixed precision dnns: All you need is a good parametrization. *arXiv preprint arXiv:1905.11452*.

[111] Umuroglu, Y., Conficconi, D., Rasnayake, L., Preusser, T. B., and Själander, M. (2019). Optimizing bit-serial matrix multiplication for reconfigurable computing. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(3):1--24.

[112] Umuroglu, Y., Rasnayake, L., and Sjalander, M. (2018). Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing. In *Field Programmable Logic and Applications (FPL), 2018 28th International Conference on*, FPL '18.

[113] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning*, pages 1096--1103.

[114] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371--3408.

[115] Wang, J. Z., Li, J., and Wiederhold, G. (2001). Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on pattern analysis and machine intelligence*, 23(9):947--963.

[116] Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612--8620.

[117] Wistuba, M., Rawat, A., and Pedapati, T. (2019). A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*.

[118] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37--52.

[119] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. (2019). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734--10742.

[120] Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., and Keutzer, K. (2018). Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*.

[121] Xie, L. and Yuille, A. (2017). Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397.

[122] Yang, Y. and Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270--279.

[123] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM computing surveys (CSUR)*, 38(4):13.

[124] Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., and Xin, J. (2019). Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*.

[125] Yu, K., Lin, Y., and Lafferty, J. (2011). Learning image representations from the pixel level via hierarchical sparse coding. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1713--1720.

[126] Yuan, Y., Chen, C., Hu, X., and Peng, S. (2020). Evoq: Mixed precision quantization of dnns via sensitivity guided evolutionary search. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1--8.

[127] Zeng, S., Huang, R., Wang, H., and Kang, Z. (2016). Image retrieval using spatiograms of colors quantized by gaussian mixture models. *Neurocomputing*, 171:673--684.

[128] Zhang, D., Yang, J., Ye, D., and Hua, G. (2018a). Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365--382.

[129] Zhang, S. and He, F. (2019). Drcdn: learning deep residual convolutional dehazing networks. *The Visual Computer*, pages 1--12.

[130] Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018b). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Computer Vision and Pattern Recognition*, pages 6848--6856.

[131] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212--3232.

[132] Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. (2017). Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*.

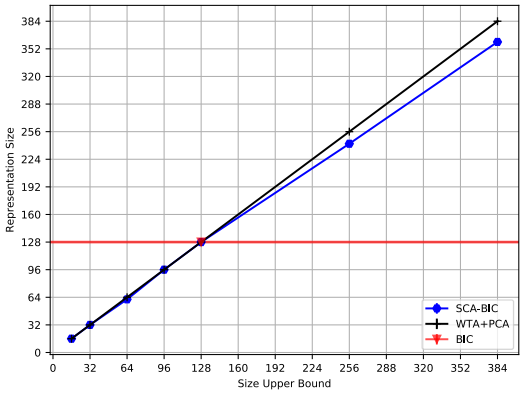
[133] Zhou, Y., Moosavi-Dezfooli, S.-M., Cheung, N.-M., and Frossard, P. (2018). Adaptive quantization for deep neural network. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[134] Zmora, N., Jacob, G., Zlotnik, L., Elharar, B., and Novik, G. (2019). Neural network distiller: a python package for dnn compression research. *arXiv preprint arXiv:1910.12232*.

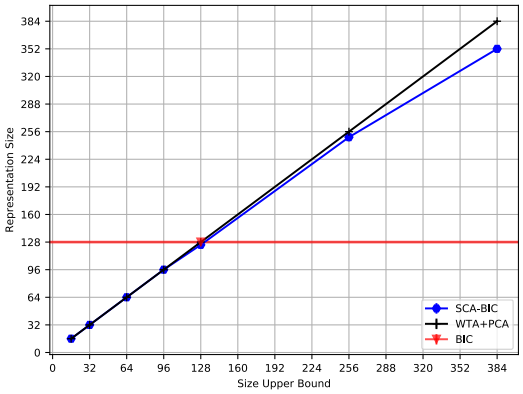
[135] Érico M.D.A. Pereira and dos Santos, J. A. (2017). Image representation learning by color quantization optimization. In *Proceedings of 30th Conference on Graphics, Patterns and Images (SIBGRAPI), 2017, Niterói, RJ.* Available from: <`http://urlib.net/rep/8JMKD3MGPAW/3PJ6MCH`>.

Appendix A

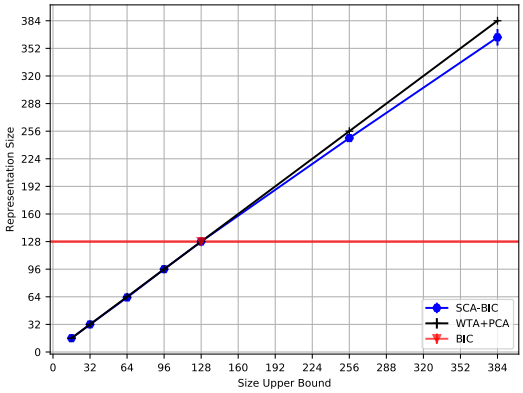
Representation Sizes for Size-Constrained Approach



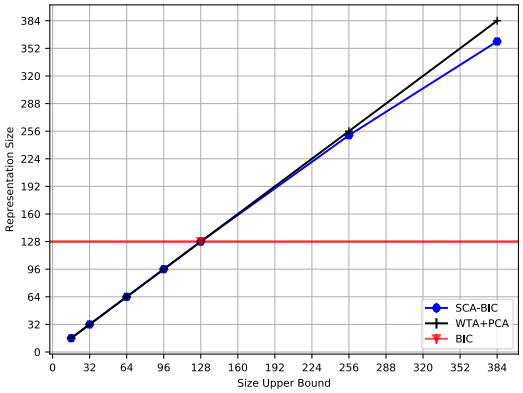
(a) Groundtruth



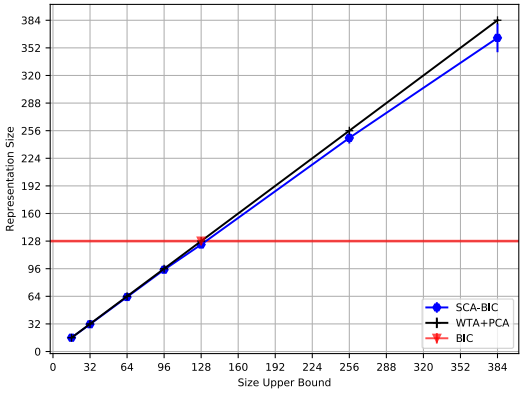
(b) Coil-100



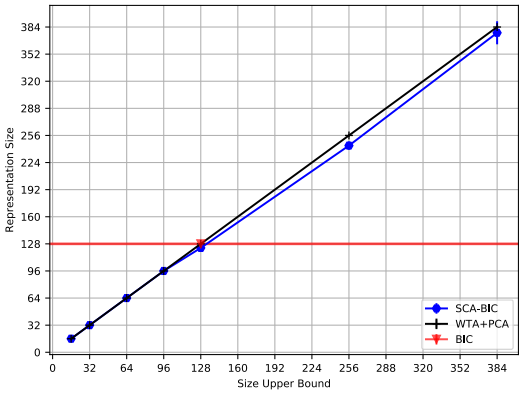
(c) Corel-1566



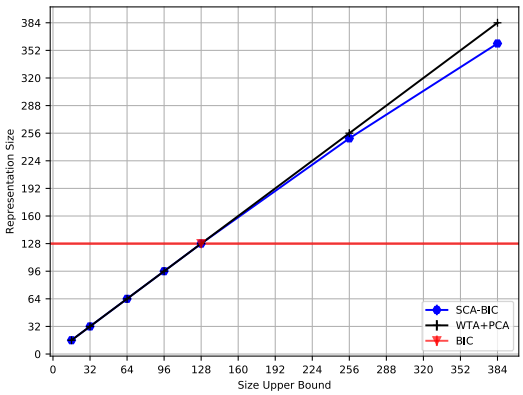
(d) Corel-3906



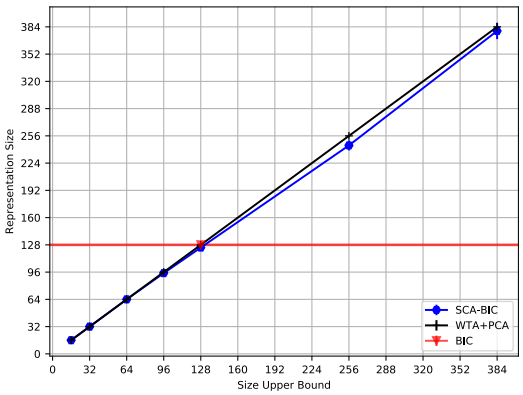
(e) ETH-80



(f) Supermarket Produce

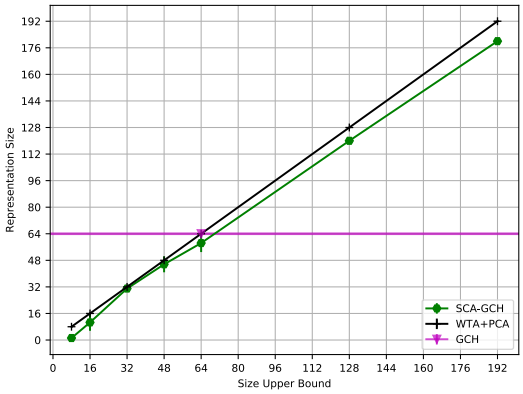


(g) MSRCORID

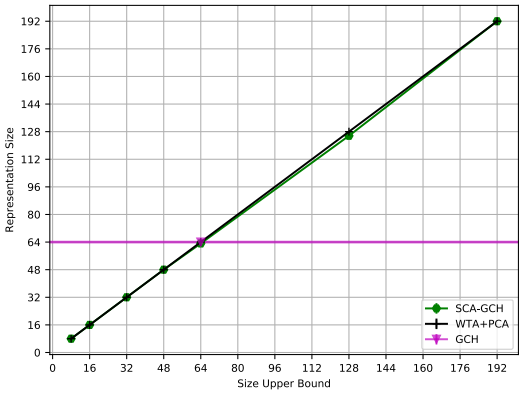


(h) UCMerced Land-use

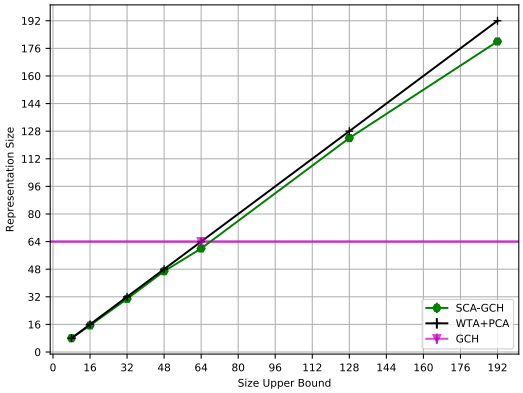
Figure A.1. Comparison between the representation size results of SCA, WTA Autoencoder and BIC feature extractor



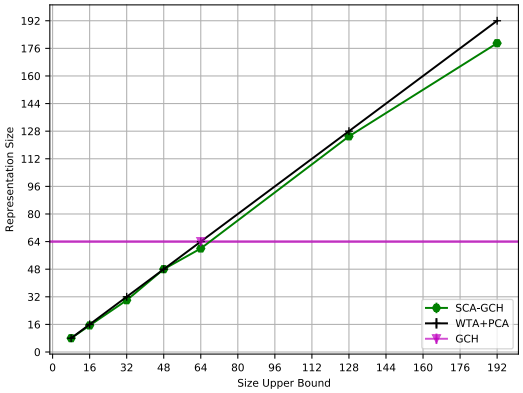
(a) Groundtruth



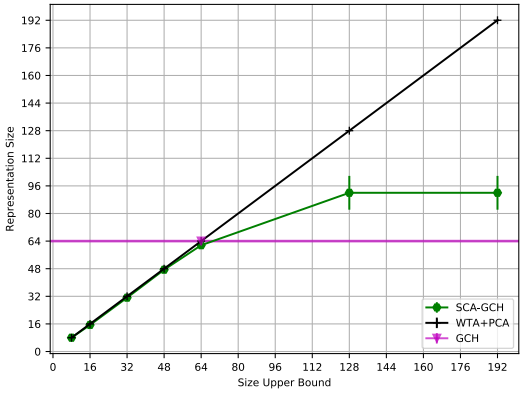
(b) Coil-100



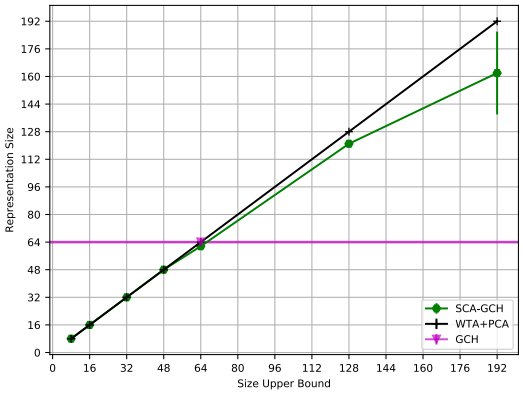
(c) Corel-1566



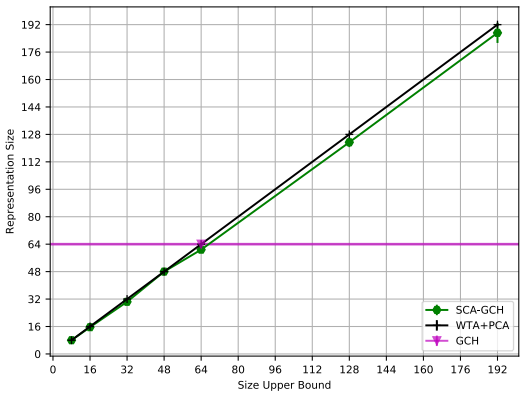
(d) Corel-3906



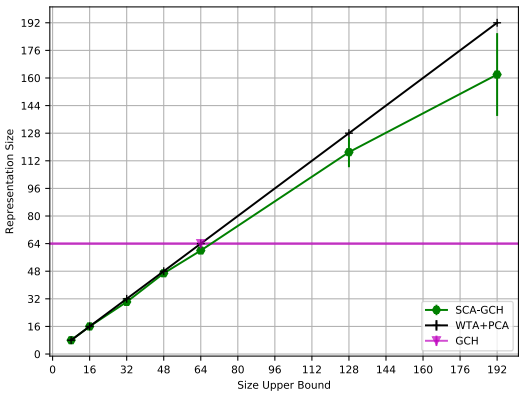
(e) ETH-80



(f) Supermarket Produce



(g) MSRCORID



(h) UCMerced Land-use

Figure A.2. Comparison between the representation size results of SCA, WTA Autoencoder and GCH feature extractor